The impact of reactionary behavior in channel creation games

How actions influence transaction routing in the bitcoin lightning network

by

D.D.M. Moonen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday August 22, 2023 at 11:00 AM.

Student number:	4608062	
Project duration:	December 14, 2021 – August 22, 2023	
Thesis committee:	Prof. Dr. L. Chen,	TU Delft
	Dr. S. Roos,	TU Delft, supervisor
	Dr. T. Durieux,	TU Delft

Preface

Preface

This thesis analyzes the impact of a multi-actor model on the rewards a party is able to obtain by operating in the Bitcoin lightning network. This project was performed at the Distributed Systems Group [1] at Delft University of Technology, and is part of the master of the Cyber Security master degree at the faculty of computer science. The responsible professor is Lydia Chen, an Associate Professor of the Distributed Systems Group. The supervising professor is Stefanie Roos, an Assistant Professor of the Distributed Systems Group. The third and final committee member is Thomax Derieux, an Assistant Professor of the Software Engineering Research Group.

The source code of this project has been made available[2].

[1] `https://www.tudelft.nl/en/eemcs/the-faculty/departments/software-technology/distributed-systems`

[2] `https://github.com/DMoonen/reactionary-games-impact-analysis`

Acknowledgement

“Energy and persistence conquer all things.” —Benjamin Franklin

This thesis was not a walk in the park. It has often pushed me beyond what I had previously thought my capabilities were. It has made me learn and improve myself, but the journey has been worth it. As it allowed me to achieve something that I am proud of. Those that supported me during this time have had a great impact on me and that is why I want to thank those who have motivated me, and supported me in overcoming this challenge. I want to thank my family and friend for supporting me and providing a place to recharge and vent my frustrations. I want to show my gratitude to Annamarie Struive and Lilly Umans for their help in proofreading this thesis. And lastly, I want to thank Oğuzhan Ersoy and Stefanie Roos for their guidance and their expertise as my supervisors during this thesis.

D.D.M. Moonen
Delft, August 2023

Abstract

Payment channels allow parties to utilize the blockchain to send transactions for a cheaper fee. Previous work has analyzed to which degree a party can profit by facilitating the transaction process. The aim is to increase the usability of the network and to be rewarded for providing this service. However, previous work focuses on maximizing the reward of the individual player in isolation, a model that we aim to expand. That is why in this work we extend the action space to allow other parties to act and react, and observe the impact this has on the rewards of the player that would otherwise act in isolation. Testing existing placement strategies by performing channel placement games, we can assess the difference in the reward that indicates the potential loss that competition may cause when operating in the Bitcoin Lightning Network. Furthermore, we have developed a new strategy that is able to improve the performance in the multi-actor model.

Contents

1 Introduction

Ever since its invention in 2008, Bitcoin [Nak08] has impacted the transaction market by providing an alternative method for transferring funds. In doing so it gives actors with the ability to send funds to anyone, anywhere across the globe without the need for a centralized authority to process the transaction. Making it a promising alternative technology in today's day and age where transactions occur frequently.

However, Bitcoin's blockchain does suffer from issues that arise from scaling. It takes roughly 10 minutes to process a transaction and processes an average of 7 transactions per second (tps) [Cro+16] [Li+18]. If we were to take Paypal as an example has a tps of close to 200, or Visa which has a tps of over 1500 [MDP18], it becomes clear that Bitcoin has a significantly lower tps rate than the traditional central authority competitors.

When a large number of transactions need to be processed using Bitcoin's technical limitation of 7 tps [Li+18], it leads to congestion. This congestion ultimately manifests itself in high transaction fees, as parties compete to have their transactions processed on the blockchain.

This is why various researchers have proposed the solution of payment channels [DW15] [BDW18] [McC+16] [PD16]. When a payment channel is created, two parties lock funds on-chain (on the blockchain) to be used for direct transactions between them. This requires only a limited number of transactions (namely two) to lock the value, after which many transactions can be performed without the need for them to be on the chain. By sending signed messages the two parties keep track of who owns which part of the locked funds, which are also used as collateral when a party tries to cheat the other. This channel is not limited to two parties. By creating a network of channels it is possible to create a payment channel network (PCN), which reduce the number of on-chain transaction even further.

Bitcoin Lightning Network [PD16] or Raiden network [Net20] on the Ethereum blockchain [Woo+14] are such PCN's and have the following advantages. First, they create a network where the throughput is seemingly limitless. If the channels are well-funded, and the transactions are balanced, then no channel will run out of funds. This means that the number of transactions that can occur on the network would only be limited by the latency between devices. The second advantage is that the transactions are near instantaneous, requiring only the latency between the two parties. Since the transactions are performed between two parties, the transaction is processed when the signed messages are exchanged. No longer is it required to wait until the majority of the chain has verified that the transaction is indeed legitimate for it to be processed. This feat can be achieved whilst still being able to rely on the security that the blockchain provides. It would require the owner of the channel to look for and dispute fraudulent transactions, but it would be able to maintain a high level of security through the blockchain.

However, there are costs involved with channel creation. The implicit cost is that the funds are locked and can not be spent elsewhere, where returns might be higher. The explicit costs are the transaction fees that are paid for the transactions, to lock the funds. These disincentivize the maintenance of payment channels. And therein lies the problem, because the network requires a large number of reliable, well-funded channels to operate effectively [Gud+19].

For there to be a network with a large number of reliable, well-funded channels to exist, the benefits need to outweigh the costs. These benefits come in 2 forms, saved transaction fees and routing rewards. When one is part of the network, it means that transactions can be processed off the chain. For every processed transaction one saves the transaction fees that would normally be paid on the chain. If one performs frequent transactions, the money saved can be quite a large sum. The routing rewards are obtained by facilitating a transaction that makes use of an established channel that charges a fee. In this sense, one is rewarded for the connectivity of the channel, and the locking of the funds. Since the saved transaction fees are determined by the demand on the blockchain, this is not a variable that can be changed to provide an incentive. This means that the incentive to lock funds and create a channel largely comes from transaction fees.

This work focuses on the routing incentives within the BitcoinLightning Network, which has been investigated in Ersoy et al [ERE20]. This closest related work by Ersoy et al proposes a non-default fee selection algorithm that allows for a higher return on newly created channels. They show that fees have the potential to motivate rational actors to fund payment channels. However, in their model, they assume a single actor creates new channels and fixed routing fees by other parties, whereas in reality there would be the possibility of fee changes and additional channel creations. Furthermore, the model currently does not include capacity information and contains same-value transactions, whereas in reality transactions would have different payment values. In general, it would be interesting to see if the actions of the other parties in the network that follow the channel creation influence or otherwise impact the ”optimal” choice.

For this reason, the aim of this thesis is to add to this existing work and extend the model found in [ERE20]. By modeling multiple actors that interact consecutively in the channel creation game, the fee strategy from [ERE20] can be analyzed in a more realistic scenario. By answering ”How is the optimal channel creation choice impacted by a multi-actor model?” a better understanding of rewards and incentives can be obtained. This provides a more realistic approximation of what the benefits will be for actors that lock funds for a payment channel in the Bitcoin Lighting Network. These incentives will ultimately guide the decision for an actor to either lock or abstain from locking funds. To answer the research question **”How the rewards of an optimal channel creation choice are impacted by a multi-actor model?”**, we will consider 2 factors. Namely the impact of the existing actors **”How are rewards impacted when parties are allowed to change their channel fees?”**, and the impact from additional channel creations **”How are rewards impacted when there is an additional party participating in the network creation game?”**.

To analyze the impact of these two factors we conduct multiple experiments with slight variations in action space. This allows us to isolate and observe the specific effects these factors have on the rewards obtained by routing transactions within the network. The first experiment serves as a baseline where the action space remains unchanged. In this scenario, we examine the rewards without any additional parties or changes in channel fees. The second experiment introduces another party to the network. This new party will react to changes made by the leading party, providing insight into how their presence influences the rewards. In the third experiment, we allow parties already present in the network to alter their channel fees. This explores the impact that fee adjustments have on the rewards. Lastly, the fourth experiment combines the action spaces from the previous two experiments. This allows us to examine the combined impact of the action space on the obtained rewards.

Our experiments conclude that both the ability of the existing network to alter existing channels and the inclusion of additional parties have a negative influence on the rewards obtained within the network. We found this negative effect on rewards to be increased when the existing network was given the ability to change channel fees. Our results of this thesis indicate that actors within the network tend to undercut each other, leading to lower rewards. These findings suggest that the optimal channel creation choice is negatively affected in a multi-actor model.

Based on these conclusions, we aim to mitigate the impact of other actors on the rewards. To address this, we analyze strategies that take into account the behavior of the other actors during the decision-making process. Namely what the competitive advantage will be for the leading party when the reaction to their actions is known beforehand. Our findings indicate that whilst this provides a significant advantage, it is not enough to counterbalance the reduction in rewards. Our contributions are:

- Extending the model to allow other actors to perform actions.
- Analyzing the impact of reactionary games.
- Implementation to remove rewards that would originate from the routing party.
- Analyzing two game-theoretical strategies that aim to reduce the influence of other actors.

2 Background Knowledge

This thesis builds upon common concepts found in various fields related to computer science. While we assume that these concepts are familiar to the reader, this chapter aims to provide background knowledge for those who may not have obtained it yet. Section 2.1 covers the fundamentals of graph theory, which is essential for understanding the blockchain technology discussed in 2.2. In section 2.3, methods for creating graphs suitable for analysis are described. Section 2.4 delves into the fundamental principles of game theory, while section 2.5 explores related work. By studying these sections, the reader will gain the foundation to comprehend the content that will follow in this thesis.

2.1. Graph Theory

In this section, we will highlight some of the fundamental principles underlying graph theory, which will play a crucial role in comprehending Section 2.2 while providing a solid foundation upon which this body of scientific literature is built. Graph theory, a branch of mathematics that focuses on constructing and studying networks that capture relationships, takes center stage. One notable relationship is that of topological structures and the diverse ways in which one can interact with them. The model comprises connections, also known as vertices or nodes. The connections between two vertices are called edges and are defined as (u, v), where u and v represent vertices. The topological structure, known as the graph, can be described as a collection of vertices and edges, denoted as $G = (V, E)$.

Graphs can possess the property of being either **directed** or **undirected**, a characteristic determined by the edges within the graph. Specifically, it refers to whether the edges allow traversal in both directions or restrict traversal to a single direction. An edge that permits traversal in only one direction is termed a directed edge, whereas an edge enabling traversal in both directions is an undirected edge. A graph comprising solely directed edges is referred to as a directed graph, while a graph consisting solely of undirected edges is an undirected graph. It is not possible for a graph to have both undirected and directed edges simultaneously. However, in the event of an anomalous graph with a mixture of directed and undirected edges, it can be transformed into a directed graph. To achieve this, for every undirected edge (a, b), one can introduce directed edges (a, b) and (b, a), thereby creating a directed graph. The reverse transformation will not be viable. If a directed edge (a, b) exists, converting it into a directed edge (a, b) would necessitate the addition of an extra undirected edge (b, a), which might not have been present initially. Hence, a graph containing both undirected and directed edges cannot be converted into a directed graph.

A path is a sequence of edges that joins a sequence of vertices, where all vertices, as well as edges, are distinct [BW10]. If we intend to join vertices a and b, we can achieve this by utilizing the edge (a, b). In the event that such an edge does not exist, an alternative sequence could involve (a, c) followed by (c, b), provided that both of these edges exist within the network. This sequence can be represented as the path (a, c, b). Multiple paths exist within a graph, and they play a vital role in analyzing the optimal traversal options. When edges in a graph are assigned weights or requires a fee to traverse, it follows that the paths within the graph also obtain a weight. This path weight corresponds to the sum of the weights of all individual edges along the path. The weight of a path can provide valuable information for decision-making processes. Since the concept of weights is frequently employed in graphs, it is implicit that weighted paths are also widely utilized. Technically, there is no restriction on a path returning to a vertex that has been visited earlier. When this happens, it is known as a **cycle**. Cycles can have practical applications in certain problems. However, in this specific context, they are generally undesirable since they increase the path weight, which represents the cost of traversal. In our case, minimizing this cost is the desired outcome for every party within the network. Hence, our objective will be to eliminate cycles whenever possible. One such method of eliminating cycles is by using the concept of the **shortest paths**, which represents the path with the fewest number of edges required to reach the destination. In the event that this shortest path contains a cycle, we can simply remove the cycle by recognizing that it revisits the same vertex twice. By doing so, we create an even shorter path. This process guarantees the removal of all existing cycles, ensuring that the shortest path remains free of any cycles. A similar situation arises when we utilize **weighted shortest paths**. However, instead of limiting the number of edges, our goal is to minimize the path weight. This approach allows us to include more edges into the path, but only if it results in a lower overall cost. In other words, we choose to traverse additional edges only if they lead to a reduction in fees. However, this approach can only be employed when the graph does not contain any edges with negative weights. Negatively weighted edges can decrease the path weight and potentially incentivize paths to include cycles. Since this goes against the purposes of this work, we will only consider graphs with positive weights.

Another optional parameter is the fee. The fee is what one has to pay in order to make use of a certain edge. The edge charges a fee to traverse it. The inclusion of this concept exists to allow the model to more accurately describe real-world scenarios. This ensures that the results derived from the computations will be better applicable to the real-world scenario as well. For this reason, the use of fees in graphs is a common practice. In this work, we will use the terms edge weight and fee interchangeably. Both indicate the cost of using an edge, and the reward obtained when someone makes use of an edge is directly correlated with the cost spent to make use of said edge. When one does not want to incorporate fee into their model, one should set all of their edges to weight 1, as to still allow for path and shortest paths to be computed.

While our intuitive perception of a graph often involves a vast interconnected network, not every graph guarantees that every vertex can reach or be reached by others. Consider the following directed graph: $G = (V = a, b, E = (a, b))$. In this graph, vertex $a \in V$ can reach vertex $b \in V$ through the directional edge $(a, b) \in E$. However, vertex b cannot reach a since there is no edge $(b, a) \in E$. This example demonstrates the importance of considering the connectivity between vertices. This concept is referred to as the level of **connectivity**. A graph is considered connected if there exists a path between every pair of vertices in the graph. Some definitions state that connectivity requires every vertex to be able to reach every other vertex, excluding the trivial path to itself. These two definitions are essentially equivalent, and this work will treat them as such. When we extend the concept of connectivity further, it implies that not only should there be a path between every pair of vertices, but that path must have a length of one (or zero in the trivial case). In other words, if every possible edge that can exist is present in the (directed) graph, we refer to it as being **fully connected**. This implies that for each existing vertex, there is an edge directly connecting it to every other vertex. Consequently, a directed graph can have a maximum of $\#V(\#V - 1)$ edges, where $\#V$ represents the number of vertices $v \in V$. A directed graph is said to be **strongly connected** when every vertex is able to reach every node in the network. However, this path does not have to be of length 1. A directed graph is said to be **weakly connected** when there exists a pair of vertices that cannot reach each other within the directed graph, but all pairs of vertices can reach each other in the underlying directed graph [BG08]. When there exists a vertex that can neither be reached by nor can reach other vertices it is called **disconnected**. A graph that contains a disconnected vertex can be expressed in groups that are connected to each other, these groups are called **connected components** of the graph. Because of the problems that arise from working with non-connected graphs, this work will only make use of connected graphs.

Vertices within a graph possess a property known as **degree**. This property is determined by the number of connections that a vertex has. Specifically, the degree of a vertex refers to the count of edges directly connecting it to distinct destinations. The notation for a vertex a with x unique connections is represented as $Deg(a) = x$. Computing the degree of a vertex differs between directed and undirected graphs. In the case of a directed graph, the number of connections leading to vertex a may differ from the number of connections leaving a. To describe this concept, we use the terms **in-degree** for connections going to vertex a and **out-degree** for connections leaving vertex a. However, in this work, where only symmetric edges are utilized, the out-degree and in-degree will always be the same as the degree. Therefore, in the context of this work, we will exclusively use the term ”degree” to refer to this value, as it will always represent the same number.

Not all edges in a graph are equal in terms of their usage. Some edges may experience higher traffic or usage compared to others. This discrepancy could arise from factors such as their position within the graph, the degree of the vertices they connect, or the fee they charge relative to other edges in the network. Regardless of the underlying reasons, it is important to distinguish and compare the usage of edges. One metric that aids in this analysis is the **betweenness centrality**.

The betweenness centrality metric operates on the assumption that each vertex in the network sends one transaction to every other vertex through the shortest (weighted) paths. By considering all these paths, it calculates how frequently an edge appears in any of the shortest paths and assigns a corresponding score. Performing this computation for all edges enables a comparison of their scores. Mathematically, the betweenness centrality of a vertex v is given by the equation: $C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$ where σ_{st} represents the set of shortest paths from vertex s to vertex t [Fre77; RKJ14].

2.2. Bitcoin Lightning Network

This section will dedicate itself to the fundamental understanding of the solution to the congestion problem present in the blockchain space. The congestion problem occurs when the theoretical transaction limit has been reached, and transactions are required to participate in a bidding war to be validated (processed). Fortunately, there exists a solution that leverages the properties of the blockchain while introducing functionality off the blockchain, which is known as a **layer 2 solution**. In this thesis, we will explore how such a layer 2 solution addresses the congestion problem.

Now consider the following scenario: we have two individuals who frequently exchange transactions with each other. Instead of engaging in the bidding war and paying high transaction fees they decide to perform one transaction where they **lock value**. From that point onward, instead of performing a transaction, they simply shift the distribution of the locked value. This scenario describes a **channel** being created by two **parties** and we will continue to use this terminology throughout the rest of this thesis. While section 2.1 has introduced these concepts as nodes and edges, in the context of channels they are often referred to as channels and parties. Now consider the scenario where parties have the ability to send a theoretically infinite number of transactions to each other. They continue sending transactions until one of the parties decides to discontinue the channel. This process is called **Channel closure**, where the current distribution is used to unlock the value and provide to each party their share. Although the solution of channels effectively solves the congestion problem, there are still some concerns that need to be addressed.

Firstly we want to address the security concerns that arise when funds are locked in a channel. There exists the possibility that someone might attempt to forge a fraudulent transaction in order to manipulate the balance of locked funds in their favor. If successful they will have effectively stolen funds from the other party, which is not preferable. To address this risk, the concept of **signing** messages was introduced, which if we assume that the act of signing is secure would be able to prevent it.

For a transaction to be accepted as valid, both parties need to sign before the transaction is accepted. This process involves the following steps. First, one of the parties makes a new distribution of funds according to the transaction and performs a cryptographic operation on the document using a secret value that only they possess. Then this transaction proposal is sent to the other party, who can confirm the origin due to the cryptographic signing that has been performed. Once the other party agrees on the transaction they sign it using their secret value and send the resulting transaction proposal back. Having received a signature on the transaction proposal the initiating party knows that the transaction has been accepted, and concludes the transaction. It is important for both parties to record the updated state of the channel, as the consequences of not doing so are explained below.

In the event that a party attempts to deceive the other in any step of the process, the affected party has the ability to contest or **dispute** the fraudulent transaction. By providing the last transaction accepted by both parties and the security of the blockchain the dispute can be resolved. However, as it currently stands, disputing fraudulent transactions requires parties to actively monitor the blockchain in order to contest them. For parties that do not have the means to actively monitor all transactions the concept of a ”watchtower” has been developed, where this task can be outsourced to other nodes to monitor for fraud [Ren18].

Secondly, we will address the issue of available balance within the channel. The locked value within the channel is finite. When one party uses the channel more frequently or with higher transaction amounts, over time the distribution will become heavily skewed towards one side. When this occurs a transaction will attempt to go over the **capacity** of the channel. Since this is not allowed, it makes the channel unable to be used in one direction. One possible solution would be for the party to wait until transactions are made in the reverse direction to redistribute the funds, however, this is not always feasible. Another solution is to perform a process called **rebalancing**. A transaction is used to add more value to the total value locked in the channel thus changing the distribution of funds. By increasing the value locked in the channel, it ensures that the funds are not fully owned by one party anymore, enabling transactions to be performed in both directions again.

However, it is worth noting that if the distribution of transactions is not evenly balanced in both directions of the channel, it is only a matter of time before the channel's capacity is reached again. While there exists work that attempts to create channels that are balanced. The asymmetric nature of transactions makes it difficult to perfectly balance a channel without continuous monitoring [LMZ20] [KG17] [PN20]. Achieving a balanced distribution of funds in a channel is a difficult task. For the scope of this work, we assume that this problem has been solved, and thus that channels do not require rebalancing.

Now that we have a better understanding of how channels work and how they are created, we can explore how this knowledge can be further developed. In the following scenario, we will consider three parties named a, b, c, and two channels named (a,b), and (a,c). Party b intends to send a transaction to party c without using the blockchain. Although they have set up a channel with party a, they lack a direct channel with party c. For providing access to their channels while not participating in the transaction they are awarded monetary compensation in terms of a **fee**. **Locktime** refers to the duration a channel is willing to reserve funds to facilitate a transaction. When a transaction request is received, the funds are locked for the specified lock time period before allowing other parties to access them.

Now instead of a network with two parties and three channels, a network can be scaled to any size. This is what the **(Bitcoin) lightning network** is. It serves as a Layer 2 solution, which is a second layer of functionality built on top of the Bitcoin blockchain. The lightning network is a network of channels that has grown significantly in size, allowing transactions to be performed. The lightning network offers several advantages. Firstly it addresses the throughput issue. The network's throughput has no inherent limit. meaning that it can handle any number of transactions regardless of how large the network will grow in size. Secondly, it provides faster transaction settlement speed. Transactions within the lightning network can be settled in under a minute and occur in milliseconds [Ant17]. This is an order of magnitude faster than the estimated 10 minutes it takes to confirm a transaction on the blockchain. Although the transaction throughput does not have a fundamental limit, latency serves as an upper limit. However, it is uncertain what kind of advancements will be made to further reduce latency and thus increase transaction speed. Thirdly the network enhances transaction privacy. Since transactions are not published on the blockchain, they can be hidden from actors that only monitor on-chain transactions, since only the locking of value is published on the blockchain. Within the lighting network routing a transaction through multiple channels makes it possible to hide the source and destination when the parties are non-adjacent [Ant17]. Lastly, channels have the added benefit of being able to perform transactions smaller than one **satoshi**, which is the smallest unit of a Bitcoin [Ant17]. However, there is a constraint that a channel initiation request is only valid for 24 hours. If the request is not responded to within this time, it becomes void. In the Lightning Network, the source party is required to specify the route their transaction takes to reach its destination. This concept known as **source routing** allows the initiating party to determine the path of the transaction. Within the network, each party publishes its maximum lock time and fees, which are stored by every party in the network.

This information facilitates source routing by providing source nodes within the network to construct a path that best suits their interest. Typically this comes down to reducing the fee spend to perform their transaction, but it might also take into account transaction time or chance of success. What matters is that it grants parties the freedom to determine what path their transaction should follow.

The general formula that represents the fee associated with using a channel between endpoints u and v is as follows:

$$fee[u,v] = bf[u,v] + fr[u,v] * amt[u,v]$$

The fee of using the channel is determined by the base fee $bf[u,v]$ in addition to the variable fee $fr[u,v]$ multiplied by the desired amount to be transferred over the channel $amt[u,v]$.

The path from party a to b is determined using either Dijkstra's shortest path algorithm [Dij22] or Yen's algorithm [Yen70]. The choice of algorithm is left to the source, and the cost function used within the algorithm can be freely selected. The selection of cost functions impacts which path will be chosen, which in turn impacts the fees and success rates. Next, we will describe three such cost functions.

The first cost function is **LND**[1] and aims to search for paths that have low timeouts, and low fees. It is defined as follows:

$$cost[u,v] = amt[u,v] * lt[u,v] * rf + fee[u,v] + bias[u,v]$$

In this equation, the transaction amount is multiplied by the lock time $lt[u,v]$ and rf (a risk factor set to $15 * 10^{-9}$ at default). Additionally, there is a bias term that rewards channels with a successful transaction history The bias $bias[u,v]$ is calculated by dividing a penalty by the estimated chance of success based on the channel's track record.

The second cost function, known as **c-Lightning**[2], also prioritizes low timeouts and low fees. However, this cost function multiplies these factors instead of adding them. This results in the following cost function:

$$cost[u,v] = (amt[u,v] + scale * fee[u,v]) * lt[u,v] * rf + bias$$

In this equation, $amt[u,v]$, $fee[u,v]$, $lt[u,v]$, rf have the same meaning as they do in LND, with default values 10 and 1 for rf and the $bias$ respectively. The introduction of scale introduces a degree of randomness by selecting a value in the interval [0.995, 1.05].

The third and final cost function we will discuss is **Eclair**[3]. Eclair's cost function considers the channel capacity, the time the channel has been open, as well as the fee, and lock time.

$$cost[u,v] = fee[u,v] * (n_{lt}[u,v] * lt_{ratio} + (1 - n_{cap}[u,v]) * cap_{ratio} + n_{age}[u,v] * age_{ratio})$$

Here n_{lt}, n_{cap}, and n_{age} represent the normalized lock time, capacity and age respectively. These values are in between the minimum and maximum of each category. The ratio factors in the equation are set values. $lt_{ratio} = 0.15$, $cap_{ratio} = 0.5$, $age_{ratio} = 0.35$.

Out of these 3 candidates LND is seen as the best option. Compared to the other algorithms, it compromises less in terms of fee ratio and lock time, yet performs well [KR21]. Furthermore, previous work shows that 90% of the lightning network uses LND [TSZ19][MZ21][Zab+22], which, further indicates that it is the preferred option.

2.3. Network creation

Researching graphs involves selecting the appropriate graph to use. When a network is readily available, this step is straightforward. However, in situations where a suitable graph is not provided, this task becomes challenging. The lightning network, for example, undergoes constant real-time changes, presenting a unique challenge. This subsection explores two commonly used approaches; working with a snapshot (2.3.1), and using the Barabási–Albert algorithm (2.3.2) to create graphs fit for research.

2.3.1. Resizeing from snapshot

Analyzing the properties of the ever-changing lightning network can be difficult. To allow analysis to be performed, the network is observed and stored as a **snapshot**, which represents the state the network was in at a specific moment in time. This snapshot can be used to analyze network properties, whilst the real network changes over time. By analyzing the network properties based on the snapshot while the real network continues to evolve, we increase the level of reproducibility of the research results. However, as the lightning network grows in size, analyzing the snapshot becomes more computationally expensive.

2.3.2. Barabási–Albert

Another method of network creation is random graph creation. Albert-László Barabási and Réka Albert have come up with the Barabási–Albert algorithm, which is designed for generating scale-free networks. [BA99]. The network starts with two values: the desired size of the network, and the number of connections that each node should at least have.

[1]https://github.com/lightningnetwork/lnd
[2]https://github.com/ElementsProject/lightning
[3]https://github.com/ACINQ/eclair

The network starts with an arbitrary number of nodes, all of which are connected to each other. It is important to ensure that the initial number of nodes is greater than the required number of connections for a newly joined node. This ensures that every node will have at least the specified number of connections. Additional nodes are then added to increase the transaction size until the desired size of the network is reached. The connections for these nodes are created based on probability. The probability of connecting to a particular node increases the higher the number of connections that node has. Mathematically, the probability of creating a connection is determined by the equation $p_i = \frac{k_i}{\sum_j k_j}$, where k_i represents the degree of node i, and k_j represents the degree of all preexisting nodes [AB02]. The resulting network is a scale-free network, which follows the power law. This means that the distribution of node degrees follows a fixed distribution. Therefore, adding or removing nodes from the network will not change this distribution.

2.4. Game Theory

After gaining an understanding of how topological structures can be modeled, it is important to introduce the branch of mathematics that analyzes decision-making processes. This branch is called game theory. In game theory, an acting party is also known as **player** or **agent**. A player has the ability to make decisions called **moves**. It is assumed that all players act rationally, making moves that maximize their own benefits. Game theory serves as a tool for simulating different actions players can take and analyzing their effectiveness in achieving a desired outcome. It also enables us to analyze how agents can be influenced into taking actions that lead to desired outcomes. The scope of game theory goes beyond that of economics and has its applications in disciplines such as diplomacy, military, psychology, biology, political science, computer science, sociology, and more [MS+16].

To determine the optimal move in game theory, we require a **reward function** that enables calculations for all the agents involved. This function determines what it is that an agent values, and in turn will influence what the best action to perform is. Differences in reward functions might change which action is best for one to take. For instance, the choice between valuing time or money, or immediate rewards versus future gains, can be influenced by the reward function.

In game theory, the level of satisfaction of a player is often expressed using the concept of **utility** [Sto68]. Utility represents the perceived value or desirability of a player. The higher the utility, the higher the level of satisfaction for the player. Therefore, the higher the utility, the stronger the player's motivation to pursue the outcome. While there are multiple methods in which utility can change as the reward grows, within the scope of this thesis we assume that the utility is linear $y = x$. Reward in terms of satoshi's is directly correlated with utility, and thus maximizing the reward maximizes the utility of the player. With this linear utility function, we do not differentiate between rewards obtained in the future or present.

Now that a player can determine what it is that they value, they require a framework that determines their best move. This structure is named a **strategy**. Take a game of rock-paper-scissors as an example. One strategy could be to always play rock. However, another strategy could be to roll a 3-sided dice and pick either rock, paper, or scissors based on the results. Yet another strategy would be to play the move that another party has played during the last round.

Naturally, it follows that different strategies will have different levels of success, depending on what strategy your opponents play. Therefore, it is not guaranteed that the strategy played will yield optimal rewards. However, it is in a player's interest to pick or create a strategy that will yield them the highest level of utility. For this reason, it is often required for a player to adjust their strategy in reaction to their opponent.

The concept of a strategy is similar in definition to a concept in computer science called an algorithm. These two terms will both be used throughout the remainder of this Thesis. However, In this work, they will be used with a different meaning. Even though both an algorithm and a strategy are a series of steps that can result in obtaining a move, we reserve the term strategy for the act of determining a move. The term algorithm shall be reserved for the steps taken to compute an output. It is important to note that the output of a strategy is solely used for determining a move, whereas the output of an algorithm does not have to be used to perform a move.

This work will also include a specific kind of game within the game theory which is called a Stackleberg game. In Stackleberg games, there is a leading player that observes the state of the game and makes their move. Subsequently, all other players are allowed to simultaneously make their move reacting to the leading player's move. This kind of setting is often encountered in the business context where companies are given the choice between investing in innovation that sets them apart from the competition or spending their resource elsewhere. Innovation is frequently associated with higher costs compared to copying the innovation of competitors that have already been proven effective. Therefore the balance of innovation and imitation often revolves around maximizing profit derived from the innovation before competitors have caught up, thereby negating the differentiating factor. This work focuses on the lightning network, as highlighted in section 2.2. While the player order is determined by the principles of a Stackleberg game, the specific game that is being played is that of a network creation game. A leading player places a channel with a certain fee value representing their move. Their goal is to route transactions within the network via their channels and obtain a fee as a reward for providing this service. After the leading player has made their move other players within the network alter the fee of their existing channels to optimize the total fee reward the player would obtain by routing transactions through their channels. Once this step is performed, we observe the reward obtained by routing transactions and play another round. It is up to the leading player to determine which combination of channel destination, and channel fee will yield the highest reward, and thus yield them the highest utility.

We have yet to describe the different strategies available to the player(s). However since this is an integral part of this work, we will do so in section 3.2.1

2.5. Related work

The related work of this thesis relates due to its analysis of placement strategies or the application of game theory. Because these two topics are often closely intertwined, it is difficult to separate the two.

The first of this category paper we would like to discuss in the thesis is the work of Lange et al. [LRT21]. This work focuses on different placement strategies. In it, it compares the uniform random, highest degree, betweenness centrality, k-center, k-means, and MBI placement strategies in a simulated setting. The work analyses the effect of different placement strategies and observes the reward as the network expands in size. Here one party creates their channel after which 5000 parties sequentially join the network creating 10 channels each using default settings. The analysis is able to compare the performance of different placement strategies, however, does not take into account the existing network.

The second paper in this category is the work of Avarikiot et al. [Ava+20] that observes the network topology over time. It does so by analyzing actors that play a network creation game to determine which parameters will create a state where no actor can create a benefit by changing their strategy. The following paper follows up on this. Wang et al. [Wan+22] provide an algorithm that may be used to obtain a network that is in Nash equilibrium. The fourth paper from Bai et al. [BXW22] presents a model by which fair prices for each of the channels can be calculated. By utilizing the Shapley value, each channel is assigned a score that represents how "valuable" they are to the overall network. This value then influences the acceptable fee of a channel, which in turn determines the profit. Lastly, we would like to discuss the work of the closest related work from Ersoy et al. [ERE20] that studied how to motivate participants to willingly lock funds in the Bitcoins layer 2 solution. The work shows that routing fees can serve as a strong incentive to motivate rational agents to lock. This incentive is optimized by a greedy algorithm that allows a party to increase the fee, when routing through the party, in turn maximizing the total value of the obtained fee. The formula used to do so is as follows

$$\sum_{\forall S,R\in V|S\neq R\neq A} Pr(M=(S,R)) \sum_{j=1}^{T_{max}} Pr(T=j) \sum_{Ch_i\in C} f(Ch_i,j) * Pr[X_i(j,S,R)]$$

$Pr(M = (S,R))$ represents the probability for S to send a message to R. $\sum_{j=1}^{T_{max}} Pr(T=j)$ represents the probability $f(Ch_i,j)$ represents the fee that would be obtained when a certain channel is used to route a transaction. $Pr[X_i(j,S,R)]$ represents the probability of the channel being chosen, which comes down to the betweenness centrality of the channel. The authors of the paper condition the experiment the be able to simplify the equation. First, by having every node send a transaction to every other node we make the chance of any given sender-receiver pair to be the same. That is how the authors were able to simplify the formula and remove $\sum_{\forall S,R\in V|S\neq R\neq A} Pr(M=(S,R))$ from the equation. Second, by analyzing fixed transaction amounts $\sum_{j=1}^{T_{max}} Pr(T=j)$ can effectively be taken out of the equation as well, reducing it further. Thus after these reductions the maximization of the fee can be seen as a fee weighted betweenness centrally formula, or $\sum_{Ch_i \in C} f(Ch_i, j) * Pr[X_i(j, S, R)]$ formally.

3 Model

The methods used to construct a model have a direct influence on the subsequent analysis conducted on that model. This section aims to provide the reader with insights into the model space. Section 3.1 outlines the assumptions made regarding the network and the interacting parties. These assumptions serve as the foundation and the constraints to our model space. In section 3.2 we provide a more comprehensive description of the model, describing its structure and choices made during its creation. This will offer readers with a deeper understanding of the model and its capabilities.

3.1. Assumptions

Firstly, we make the assumption that individual parties prioritize optimizing their rewards and act rationally. As is commonly the case in game theory, a party will adjust their action if it leads to a higher reward. Consequently, we assume that parties will not hesitate to perform selfish actions, even if it may negatively impact the level of decentralization within the network. From this, it follows that there are no parties willing to lower their reward by charging a negative or zero fee. As a result, all channel weights in the network will be positive.

Secondly, the cost of opening and closing a channel remains constant over time. This allows us to perform a comparative analysis that is not impacted by the changing transaction fees caused by the blockchain. By removing this variable that could interfere with the analyses, we can better isolate the impact that choices have on the rewards, rather than the influence of changing transaction fees on the rewards.

Thirdly, we assume that participants within the network have access to the topology of the whole network. Since parties keep track of the nodes present within the lightning network this would be improving the realism of our model.

Fourthly we assume that other parties will always be willing to accept channel creation requests. In this thesis, we study the best-case scenario for a party creating a channel. This means that we leave the game-theoretical aspect of deciding when to deny a channel creation request for personal gain, as it falls outside of the scope of this thesis.

Fifthly, we assume that every party will have an infinitely large amount of funds at their disposal to use on channels. Normally a party would have to limit the number of channels based on how many satoshi's they have to lock. However, for the scope of this thesis, we assume that satoshi's required to create and to initially fund a channel is not a constraint In this work the number of channels a party may place are fixed, which will serve as a limit.

Sixthly, we assume that channels will always be balanced. Similarly to the second assumption, we constrain our model to remove a variable. Maintaining a channel as to reduce the number of channel resets is ongoing research. For the sake of this work we will assume that this problem has already been solved.

Seventhly, the reward must stem from routing a transaction. Transactions have the ability to reward the party from which it originates. However, the party who performs the transactions has to pay the transaction fees. When a party sends a transaction it is not possible for them to earn fee from it, as any earned fee would come out of their own pocket. Therefore unlike previous work, we remove the ability for parties to increase their rewards from transactions that originate from them.

Lastly, it is assumed source routing is in place. This is another constraint that we have placed to improve the realism of our model, as this is the case in the lightning network. Section 3.2.3 details which specific cost function is used.

3.2. Model description

Our experiments allow parties to perform actions within the model. Section 3.2.1 details the different actions that are allowed to be performed as well as who can perform them under which conditions. Section 3.2.2 described the placement strategies by which channels may be created. It details different methods a party may utilize to perform the action. Section 3.2.3 details which parameters are available to us to create our model. In section 3.2.4 we describe a small experiment to determine the values of some of those parameters, whereas in section 3.2.4 we interpret the results and determine the parameters to be used.

3.2.1. Action space

The desired goal of the experiments is to assess the impact that other parties have on the reward fee that a party can obtain within the lightning network. Our aim is to replicate previous research and enhance the model to enable a larger number of parties to actively respond to changes within the network.

The highlighted actions in our study are as follows:
1. A party may create a new channel.
2. A party may modify the fee of their existing channel.

In previous work, the only action permitted during the experiment was the creation of channels by an acting party. However, in our extended model, parties are now able to alter the fees of their existing channels. Creating new channels costs two transactions on the blockchain and the value of the funds that are locked. To modify an existing channel it costs 3 transactions in the worst case. One to close the channel, and 2 more to create a channel. In the best case, you announce your fee change within the network. It is important to note that certain conditions are in place that determine whether a certain party is permitted to perform a certain action. We make the distinction between two types of parties: those already present in the network, and those joining the network. While all parties have the ability to modify their channel fees, only the parties joining the network are allowed to create new channels.

3.2.2. Placement strategies

As discussed in 3.2.1, some parties are allowed to create new channels within the network. However section 3.2.1 does not detail the method used to determine who to make a connection to. Therefore we highlight the 6 strategies that a (leading) party may use to place their channels in this section. The first five placement strategies are implemented based on [LRT21] whereas the sixth placement strategy has been described in [ERE20].

The first placement strategy is that of **uniform randomness**. As the name suggests, we take a list of all the parties we are not connected to already and pick one based on a uniform random distribution. After we have made our choice, we connect to said party.

The second strategy is named **highest degree**. This strategy ranks parties based on how many channels they have already established. As explained in 2.1 the out-degree and in-degree are the same in this work, and we therefore use use the degree to determine which party has the most channels. This party is then chosen and a connection is made with them. The rationale behind this strategy is that the better connected the neighbors are, the higher the chance that a transaction can be routed through us. While generally an effective strategy, this method has been shown to induce a “rich-gets-richer” effect [BA99]. This effect ensures that routing through the most important nodes becomes even more likely, and the rewards are skewed towards the parties with the most channels.

The third placement strategy is called **betweenness centrality**. This is a metric to determine how centrally a party is located in the network., It is computed as follows: every party in the network sends a simulated transaction to every other party within the network using the shortest path algorithm. After these transactions are sent, it is counted in how many shortest paths a channel is present. Dividing this number by the total number of transactions gives a score, which is the betweenness centrality. Being centrally located increases the chance that a transaction is routed through one of your channels, and is, therefore, a benefit. Rohrere et al. observed in their analysis that the single most centrally located party was present in 37% of the shortest path of all transactions in the network [RT20]. This further proves that creating channels that increase your centrality is an effective strategy.

K-center is the fourth placement strategy. This strategy analyzes the longest paths within the network. Then it establishes a channel that makes that breaks one of these shortest paths. The act of breaking the longest path means creating a new route via our party that is the shortest in distance. The rationale behind this strategy is that breaking an existing longest path will place you within the shortest path of some parties. This in turn will yield fees when the transactions are routed by your established channel(s).

The fifth placement strategy is that of **k-means** (referred to as k-median in [LRT21]). The rationale behind this strategy is similar to that of k-center. However, rather than breaking one of the longest paths within the network, this strategy creates a connection that minimizes the average shortest path within the network.

The sixth strategy is an improvement on betweenness centrality that in previous work has been called ”greedy”. However, in this work, we have decided to call it **fee weighted centrality**, for we deem it more intuitive. Here the betweenness centrality score is multiplied by the (hypothetical) fee this channel would charge. The score that follows from this process is the reward the channel would bring in. This process maximizes the expected fee rather than the betweenness centrality. In cases where a lower betweenness would lead to a higher result, it allows one to still pick the most optimal candidate.

3.2.3. Model Parameters

There are several parameters that determine the properties of the model.

The first parameter is network size, which influences the level of realism in the model. If the model is too small, it might not accurately represent the actual network. Conversely, if the network size is too large, it could become computationally infeasible to perform calculations. Ideally, we would use the most up-to-date snapshot of the lightning network to achieve the highest level of accuracy. However, due to computational limitations, we are unable to use snapshots for the scope of this thesis. Therefore, our approach is to simulate a smaller yet representative network. Section 4.2 details the steps taken to create such a representative network.

The second parameter to consider is the choice of the path selection algorithm. The available options for the path selection algorithm are LND, C-Lightning, and Eclair. In this work, we aim to use only one path selection algorithm to eliminate potential bias introduced by mapping these different algorithms onto different parts within the network. Among these three algorithms, LND is the preferred choice due to its high usage rate and its ability to compromise less than the other two other algorithms, as discussed in section 2.2.

The third parameter to consider is that of placement strategy. This work will utilize the placement strategies discussed in 3.2.2. We have chosen to employ a variety of different placement strategies, as different placement strategies may react differently to the network’s ability to respond.

3.2.4. Initialization problem

Once the model has been created, our aim is to create a new party to join the network.

However, for this joining the occur, we are required to create channel(s) for parties already present in the network. Determining how many channels we should create and to which parties they should be connected to, is not without its issues. Therefore, in this section, we detail our analysis by which we have determined the method by which we initialize a party.

The first issue that one comes across when initializing a party within the network, is that some of the placement algorithms in this work are dependent on the existence of channels in order to perform their calculation. When the party is not connected to the network these placement strategies will yield incorrect results in the form of an error, or by returning an empty list of channels to create.

When we initialize a party, we create channels for said party. These channels have the possibility to bias the results of the placement strategies. Even when the same channels are created for each of the placement strategies analyzed in this work. The inherent differences between the placements strategies result in them reacting differently to the number of channels created. Depending on which initial connections are chosen certain channels might yield lower results due to their competitiveness with the initially created channels, impacting the choice.

Furthermore, there exists the problem of network size. If the network is sufficiently small that creating the initial connections does not leave the choice for the placement strategy at any point during the algorithm, then it has impacted the results. While this case should realistically only happen when the number of initial channels and the number of placed channels are greater than the number of parties in a network, it is something that we should verify.

The question becomes: How do we initialize a newly introduced party to the network without biasing or limiting the placement strategy?

The solution to this initialization problem must meet a number of criteria which we describe the importance of later in this subsection. The criteria are as follows:

1. The initialization should have at least 2 channels.
2. The initialization should be deterministic.
3. The initialization should limit the bias it has on the result of the placement strategies.
4. The initialization should leave options for the placement strategies.

Whilst not being an exhaustive search, nor conclusively solving the initialization problem, we aim to find an initialization that fits these criteria and thus can be used as the initialization for our analysis.

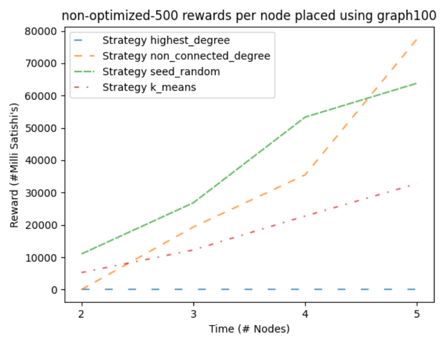
In order to do this we look at three aspects to determine our initialization. These aspects are as follows: Which placement strategy will be used to determine the initial channels? How many channels will each newly added party get as part of its initialization? What fee will be asked when routing a transaction over the channels created during the initialization? By analyzing these three aspects within the network we can analyze which combination yields an initialization that satisfies the criteria.

For this analysis, one network will be analyzed. This network was created using the Barabási–Albert with party size 200 and transformed into a bidirectional weighted graph. The weights for this graph were obtained by sampling weights from a list created from the snapshot used in Ersoy Et Al. [ERE20]. This list containing duplicates is created by simulating the fee formulate in which transactions of 100 satoshi's are sent between the parties. However, these candidate weights were constrained by $0.1 * most\_freq <= weight <= 10 * most\_freq$ with most_freq being 1000 in the case of this network. Since this is one of the three graphs that are used further in this work, more can be read in section 4.2. Because this work analyses multiple parameters to find a sufficient starting condition we opted to only analyze one network. That is why the network that simulates transactions of 100 satoshi's was chosen

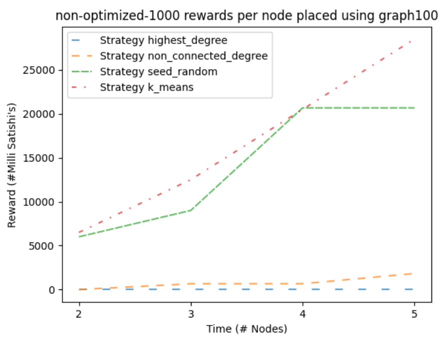
In order to determine which placement strategy would yield the best results the plan was to test the list of placement strategies that we wanted to observe for the main experiment. However as stated previously, some of them do not function when the party is not already connected to the network. For this reason, betweenness centrality, k-center, and fee weighted centrality were dropped as initialization candidates. Leaving uniform randomness, highest degree, and k-means as initialization candidates. However, as an addition, we wanted to add one more placement strategy namely the non-connected highest degree. This placement strategy follows the same placement strategy as the highest degree with the criteria that none of the initially chosen parties are already connected to any of the others. This alteration to the placement strategy allows it to be useful in cases where a leading party is placing channels.

In order to determine the number of channels needed for the initialization we simply compute a range of values. This allows us to later compare which number of channels is best. Since we need at least two channels, the lower bound is set at two. The upper bound was to not exceed the lowest number of channels a party may have in the existing network. In our case, this is five.

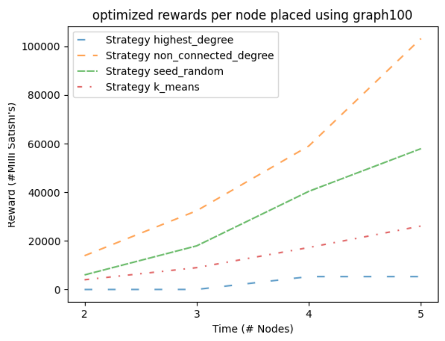
In order to determine the fee for the initialized channels, this analysis will look at three different cases. First, the case where the fee is set to 1000 which is the default. This case observes what occurs when we join in as the most frequently picked value. The second case we observe is when the fee is set to 500, which will be at 50% of the default value. This case will analyze the potential to undercut some of the channels that run default values. The third case is when we allow the newly



(a) Figure depicting the reward obtained using the flat fee of 500.



(b) Figure depicting the reward obtained using the flat fee of 1000.



(c) Figure depicting the reward obtained using the optimization algorithm

Figure 3.1: Rewards graphs of the three different analyzed algorithms

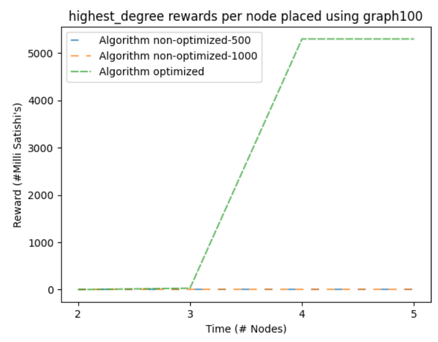
created channel to infer its fee based on the current state of the network. First, a connection is made where the fee is set to the default value, after which the optimum fee is inferred. This analyses the fee in its optimized form.

It is important to note that within this network we have only control over one side of the bidirectional channel. As we have assumed in 3.1 the fee can only be changed when the newly joined party is the source of the channel. The other part of the bidirectional channel, where the destination of the channel is the newly joined party will always be set as the default value of the graph.

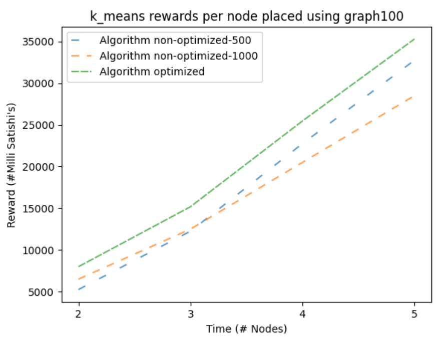
3.2.5. Initialization Analysis

Figure 3.1 depicts the performance of the three algorithms. Figure 3.1a shows the scenario where connections charge a fee of 500 for re-routing a transaction. The figure shows that the four strategies perform relatively similarly when two connections are placed. However, when the number of placed connections increases, the differences in reward become more apparent. Seeded random and non-connected degrees seem to be competing for the best performance, with k-means coming in third with roughly half the reward. Highest degree is stuck at 0 rewards. Figure 3.1b depicts a change in the fee calculation algorithm. In this case, the charged fee is set at a fixed 1000 satoshi's. Similarly to 3.1a, seeded random and non-connected degrees are performing well, with k-means and highest degree performing poorly. Again highest degree is not able to obtain a reward higher than s0. Figure 3.1c allows us to observe what occurs when a party is allowed to determine its own fee. When a connection is made, a party analyses fee values from 1 to 10000 to set the fee to the value that maximizes the highest theoretical reward the channel can receive. This case shows a clear distinction between the four strategies. Non-connected degree performs the best in this scenario, with seeded random taking the second spot. Third place is taken by k-means, and highest degree is fourth. This is the only scenario where highest degree does not end with a reward of 0, despite having 5 connections made within the network.

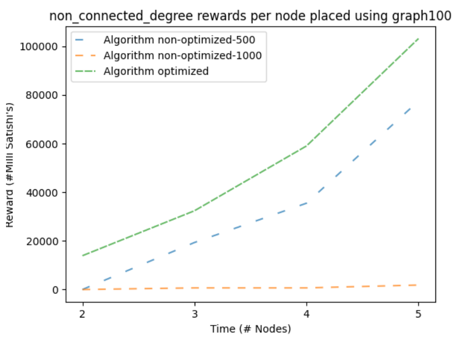
Figure 3.2 shows the same reward data as figure 3.1. Where figure 3.1 groups the data based by



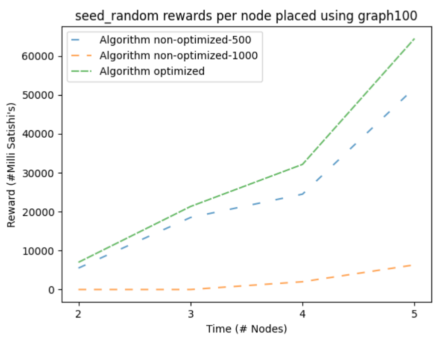
(a) Figure depicting the reward obtained when applying the highest degree strategy



(b) Figure depicting the reward obtained when applying the k-means strategy



(c) Figure depicting the reward obtained when applying the non-connected degree strategy



(d) Figure depicting the reward obtained when applying the random seed strategy

Figure 3.2: Rewards graphs of the four different analyzed strategies

algorithm, allowing for the strategies to be compared, groups figure 3.2 the reward data by strategy. This makes it possible to compare the performance of the three algorithms. Figure 3.2a depicts the highest degree strategy. This graph shows that the optimized strategy is the only strategy that results in a score that is above 0. Both the non-optimized 500 fee strategy as well as the non-optimized 1000 strategy remain with a reward of 0. It must be noted that this strategy yields the lowest rewards out of the four strategies analyzed Figure 3.2b shows the k-means strategy. This strategy has one unique feature, namely that its best and worst-performing algorithms are very close in scores compared to the other three figures. Again the optimized strategy performs best, the non-optimized 500 fee comes in second, and the non-optimized 1000 fee performs the worst in this scenario. The non-connected degree strategy can be seen in figure 3.2c. In this figure, the optimized algorithm also yields the highest rewards among the three. Non-optimized 500 fee takes the second spot, and non-optimized comes in last. This strategy yields by far the best maximum rewards among the four strategies. Figure 3.2d shows the reward data when a random seed is used as the strategy. In this case, the optimized algorithm is again the best-performing one. Non-optimized 500 fee performs the second best, and the non-optimized 1000 fee performs the worst. The reason behind this is due to an assumption made about the model. Section 5.3.1 details why this assumption leads to an inability to obtain a reward for the placement strategy.

The next step is to determine which fee will be charged to route transactions. As can be seen in figure 3.2, the case where the fee is optimized outperforms other algorithms regardless of which placement strategy is used. This makes it the best candidate for determining the best fee for a given connection.

We will be looking at 3.1 to determine which placement strategy is best used to initialize a party. Non-connected degree performs best out of the four strategies. A case could be made for k-means, which performs similarly to non-connected degree in two out of the four strategies. However, k-means seems to start with a lower fee, or is overtaken in performance. Furthermore, if we compare the best performance between non-connected degree and k-means, we can see that non-connected degree has a better reward at any point in the analysis regardless of how many connections are placed. The aim of this work is to obtain an indication of the maximum amount of reward fee possible, therefore we want our initialization to bring in as much reward as possible. This makes non-connected degree the best placement strategy to use in our initialization.

That leaves us with the third and final question, how many channels will each new party place during the initialization process? Both figures 3.2 and 3.1 show that the reward only goes up as the connection count increases. Therefore the added benefit of making more initial connections is increasing the initial reward. However since the focus of the experiment that follows this initialization is to show the change in rewards as more connections are placed using other strategies, this parameter has little impact. Therefore picking the required amount of two initial connections seems appropriate to reduce some of the computational overhead that comes with this initialization.

To conclude, the initialization will be placing 2 connections for every new party that joins the network. We shall use non-connected degree strategy to determine who the other end of said 2 connections will be, and use the optimized algorithm to determine what the fee of said connection will be.

The last step we took was to verify that these parameters meet the criteria we set for the initialization process. Firstly, we need to ensure that the number of channels criteria is met. It is easy to verify that this criteria is met. We require at least two channels, and the chosen solution will place 2. Furthermore, since all of the four strategy candidates can place channels without already being connected to the network, we can guarantee that placing two will be possible unless we observe the trivial network that only has 1 party in it.

Secondly, is the initialization deterministic? To show that this is the case we need to look at two parts of the initialization. That the placement strategy and the fee optimization function are deterministic. Since none of the placement strategies make use of randomness, we know that the chosen strategy is deterministic. Similarly, since the fee optimization function does not make use of randomness it is also deterministic.

Thirdly, the initialization should limit the introduced bias for an experiment that will be following the initialization. As some of the placement strategies are not able to be used if there does not exist a connection to the network, adding this start state will always add some level of bias. The intended purpose of the initialization is to change the result in a way that non of the placement strategies crash. However, by choosing the candidate that outperforms in terms of reward we provide a baseline of reward. The following experiment then measures how well the placement strategies can increase that reward. When we add to this that the chosen placement strategy is not a placement strategy tested in the follow-up experiment, there is no clear bias that one would expect to occur when the placement strategy used in the initialization matches the placement strategy used in the following experiment. We would like to reiterate that we do not claim to eliminate bias. Rather than the initialization used aims to minimize the bias that occurs. We feel confident that the steps taken to reduce the bias introduced by the chosen initialization method are enough to minimize the bias.

Lastly, we need to show that the connections made in the initialization do not limit the choices available to the placement strategies during the following experiment. Since in the worst case there will be a total of 50 parties to choose from, if we were to take 2 options away during the initialization process it leaves the strategy with 48 choices. Since the following experiment will place fewer than 48 connections we choice pool will never be reduced to zero. Therefore the chosen initialization will always leave sufficient options for the placement strategy.

4 Methodology

This chapter details the methodology of this thesis. Section 4.1 reiterates the goal of this thesis. Section 4.2 details the process used to create the graphs for the experiments. Section 4.3 explains the different experiments that will be run, whereas 4.4 highlights the code used to perform them. In section 4.5 the testing of our code is highlighted and in section 4.6 we detail the server architecture. In section 4.7 we detail the setup of the experiments.

4.1. Research question

Our goal is to extend the existing literature by placing it in a more realistic setting. By examining the impact of a multi-actor model on the optimal channel creation choice, we aim to answer the question, ”How the rewards of an optimal channel creation choice are impacted by a multi-actor model?”. To answer this question, we have divided it into two sub-questions that focus on different aspects of the influence of the multi-actor model. These sub-questions are as follows:

- ”How are rewards impacted when there is an additional party participating in the network creation game?”
- ”How are rewards impacted when parties are allowed to change their channel fees?”

As discussed in section 3.2.1 we analyze two ways in which the action space can be expanded: by allowing an additional party to join the network, and by enabling the network to update its fee values after channels have been created.

4.2. Graph

Due to the time complexity of the algorithms used in this experiment, it is not feasible to make use of existing snapshots of the Bitcoin Lightning Network. The number of channels to analyze when using a snapshot would make the runtime of the experiment exceed the time available to run it. Therefore we opted to create our own graph for our experiments.

One option would be to take an existing snapshot and scale it down to the required size. The aim would be to turn a network with x nodes and y channels into a graph that is z times smaller. Resulting in a graph that contains x/z nodes and y/z channels. However, to perform this down-scaling, the model would be altered in some way. When scaling down the number of nodes we run into the problem of mapping the channels between the nodes. If we were to keep the distribution in k-degree the same then we would choose which node will be connected to other high-degree nodes and thus alter the betweenness centrality of the nodes. If we aim to keep the general betweenness the same, whilst reducing the network in size we run the risk of altering the distribution of degree of the original network. While the goal would be to create a representative graph, the resulting graph would always be lacking in one area or another. Furthermore, this method has the possibility to produce a weakly connected graph, which would not suffice. The following steps would have to be taken to convert it into a strongly connected graph, which would then influence the degree and betweenness distribution. This is why it is difficult to rely on this method to generate a representative graph from a snapshot.

Another option would be to use Barabási–Albert to generate a graph. This method builds up a graph by adding nodes and creating randomly generated channels until the graph size limit is reached. This method disregards the distribution of degree as well as the betweenness centrality of the graph we aim to simulate. However, this approach would be computationally inexpensive to perform as the Barabási–Albert is clearly defined. Furthermore, this is guaranteed to create a strongly connected graph. However, the resulting graph would contain a range of different degrees on its nodes. While fulfilling the requirement of having a graph to perform operations on, the drawback of this method is that it makes modeling the channel weights difficult.

Whichever method selected the resulting graph would need to meet the experiment's needs, as it would serve as part of the model used to assess the different cases. We opted to combine the two methods described above to account for the drawbacks. The steps are as follows: First, we create a graph using Barabási–Albert with a certain node size, which will then randomly pick five other nodes it's not already connected to, and create two directed channels with the target. One from the source to the target, one from the target to the source. The second step then is to provide the newly created channels with their fee. This is done using data from the snapshot. We use a collection of all the different fee values that exist in the snapshot. Obtain the most frequent value that appears within the collection and filter the outliers by only keeping the value if it fits and passes the following test: $10 \cdot \#MostFrequentValue \geq fee \geq 0.1 \cdot \#MostFrequentValue$ and $fee < 10000$.

The upper limit for the fee is set to 10000 for this experiment because this limit was used in previous work. However in practice this fee value is almost never hit, therefore the values with fees higher than 10000 are highly unlikely to re-route transactions. Channels that have a fee higher than that of 10000 will most definitely act as noise within the network which is what we aimed to avoid. The other criterion is that the fee candidate value must be within 0.1 and 10 times the most frequent value within the network. The upper bound is again related to the limit placed on the fee optimization algorithm. Here it ensures that the upper bound is unlikely to exceed 10000 in the first place. With a fee of 1000 being the default value, ten times the default value gives us a limit close to the 10000 set on the fee optimization algorithm. The lower bound was chosen to ensure that freedom of fee choices exists within the network. Preliminary tests showed that small channel weights that connect nodes with high degrees would obtain an unusually high amount of traffic that way. Removing the values that charge an extremely small fee compared to the default value made this less of a problem. Thus by filtering these fee weights, we mitigated the effects of anomalies impacting the graph.

To even further limit this impact we wanted to introduce randomness to the fee values. We opted for adding a random value in the interval [-30,30] to enforce a difference between the previously filtered fee values. By adding randomness to the graph we reduce the chance of there being multiple paths within the network that share a source destination and path fee. This lessens the computational requirements for the experiment. But the positive effects of the randomization are not only experienced by the computation. It also further improves the freedom of choice of the acting parties within the experiment. Regardless of the ability of the network or other parties to react, spreading out the fee values makes the choices of the acting agent more thorough. Taking these steps results in 3 graphs with weighted channels, that will be used to simulate transactions that are small, medium, and large. A small transaction would route 100 satoshi's, a medium-sized transaction will transfer 10000 satoshi's, whereas a large transaction will transfer 1000000 satoshi's to another party within the network.

However, for some of the experiments, the network has the ability to react to the actions taken by the actors. It would make sense that this is a continuous process that also occurred before our experiment starts. Therefore we also require graphs that have already been optimized to keep the starting condition consistent throughout all of the experiments.

In order to save time we have pre-computed graphs that have the network (all channels) optimized. These graphs were created as follows: Transforming a non-optimized graph used for case 1 of the impact analysis into an optimized graph is a simple step. The only thing that needs to be performed is a network update. A network update entails that we optimize all the channels within the graph concurrently. As for case 2, the graph used would already have to be optimized, therefore there is no difference between a non-optimized graph and an optimized graph. Therefore performing the network update as in case 1 would suffice. For case 3, we use the optimized graph from case 1 as a baseline, connect one more party to the graph, and perform another network optimization. The reason why we use the graph from case 1 is that it makes the computation less expensive. Since our aim in this case is to have 2 nodes start with the exact same channels, we can simply clone the channels used for the party in case 1, to connect the starting channels for the extra party that is to be added. Since the 4th case is a combination of the 2nd and the 3rd, turning the graph of the 4th case into an optimized graph is the same as combining the steps we have taken for the previous two cases. Since the optimized graph of the 3rd case already contains the 2nd optimized graph of the third case already fulfills the requirements of the 4th case.

4.3. The experiments

The following chapter provides a comprehensive overview of the setup of the experiments of this thesis. Section 4.3.1 highlights the process of replicating previous work and compares it to ours. Section 4.3.2 details the steps taken to increase the number of path choices within the graph. Whereas section 4.3.3 determines the impact of differences in action space, section 4.3.4 deals with the impact of routing rewards when the reaction that follows the leading party's action is known during the decision-making process.

4.3.1. Replication

One distinguishing aspect of previous work is the optimization of the graph prior to the experiment. By configuring the model to achieve maximum gain, the subsequent experiment reflects a more realistic lower bound on the long-term earnings achievable through routing. Inspired by previous work we have opted to also perform this optimization step at each of our experiments, however, it is our aim to closely resemble previous work. Therefore we have replicated previous work and compared it to ours. So that we may highlight the potential difference in starting state that is bound to be noticeable in the results of the experiments to follow.

However, when replicating previous work, we ran into one problem, namely that our additional code highlighted an error. When performing the betweenness centrality reduction on the snapshot, the resulting score would have the possibility to end up negative. Our analysis highlighted that this only occurred if the snapshot was used in combination with the library code, but could not be replicated by manual computations nor changes in the network. To account for this we have placed extra constraints on the reduction to prevent it from going into the negatives.

```
1  def remove_own_betweenness_score(graph, src_id, bet_scores):
2      node_ids = graph.nodes()
3
4      for dst in node_ids:
5          if src_id != dst:
6              paths = nx.all_shortest_paths(graph, src_id, dst, weight=’weight’)
7              path_list = list(paths)
8              path_weight = 1 / len(path_list)
9              for path in path_list:
10                 if round(bet_scores[(path[0], path[1])]) - 1 >= 0.0:
11                     bet_scores[(path[0], path[1])] -= path_weight
12
13     for key in bet_scores.keys():
14         value = bet_scores[key]
15         if np.abs(value) < 0.0001:
16             bet_scores[key] = 0.0
17         if value < 0.0:
18             bet_scores[key] = 0.0
19     return bet_scores
```

Lines 10, 17, and 18 add extra constraints that prevent negative values, which enables us to perform the comparison. It must therefore be taken into account that the reward values of the snapshot will be influenced slightly by the change in code. We detail the results obtained in the graphs below:

4.3.2. Fee Noise Analysis

Our work deviates from previous work in its distribution of fee values. Rather than relying on the default value that occurs frequently, we introduced slight variations to the channel fees during the graph generation process. This approach aims to increase the range of the available options for fee values during path selection. By adding a bit of noise to each channel fee, we set out to assess the potential impact this might have on the resulting rewards. This experiment allows us to explore the influence of fee distribution on the overall rewards within the network.

4.3.3. The Cases

This section details the four cases used to analyze the impact that a multi-action model will have on the rewards of an individual present in the network.

To study the impact of the different additions in action space, we first create observe the behavior and rewards of the leading party in isolation. In this case, action space is not extended, and therefore only the leading party is able to create channels. By creating channels and setting a preferred fee they will increase their profits from routing which leads to a reward for the party. After each channel it is measured the rewards the leading place would have obtained, and stored for analysis. This first scenario of playing in isolation will serve as a baseline to which we measure rewards from the experiments where we have extended the action space. For this reason, we will refer to this scenario as the baseline.

After the baseline has been established, our next case will expand the action space to allow the network to react to changes that occur. The second case is then tasked with analyzing the impact these reactions will have on the reward. After the leading party has placed a channel, the state of the network is saved. This saved state is then used by every party in the network to change the fee values of every channel. It is important to note that this is done in parallel, thus fee updates are not known to other channels whilst the network reaction takes place. After this network reaction, we determine the reward that every party would obtain and play another round until the predetermined number of channels have been placed. This second case will also be referred to as the network reaction case.

The third case will remove the ability of the network to react which we introduced in the second case, but will allow an additional party to join the network. This additional party will react to the leading party within the network creation game. The leading party will be placing a number of channels iterative within the network similar to the first case, the baseline. However, after each channel that the leading party creates, the additional party will be tasked to place a channel in reaction to it. After placing this channel, we determine the rewards that every party obtains within the network as we would in the baseline and play another round placing more channels. Once both parties have placed the predetermined amount of channels, this case is concluded. The case will also be referred to as the additional party case.

In the fourth and final case of our action space analysis, we will have combined the action spaces from the additional party case and the network reaction case, to test the combined impact of the two additions in action space. After the leading party places their channel, an additional party is allowed to react. Once the additional party has placed their channel in reaction to the leading party, we again task every channel in the network to update their fee in parallel. Once the network update has concluded, we determine the party rewards as we would normally.

4.3.4. Stackleberg Games

Since we aim to analyze the impact on rewards when a party has knowledge about future moves will have, we again have separated this into two parts that we can analyze.

The first Stackleberg case examines the impact on rewards when the fee values that result from a network update are known during the channel creation process. Since these fee values determine the leading party's rewards after the network update, we essentially know the reward we shall obtain in the future given any channel creation. Rather than optimizing the rewards of a channel now, we optimize for channel rewards obtained in the future. Doing so allows us to study our ability to minimize the impact of the network update, and this is the impact that knowledge of future moves shall have.

The second Stackleberg case follows a similar approach to the first Stackleberg case. However in this case, while analyzing a channel candidate, we simulate the reaction of the additional party and their future move. Similar to the first case we optimize the future rewards of channel candidates rather than current rewards. This case would then attempt to minimize the impact that the additional party would have on the rewards and thus show the impact of having knowledge on future actions.

4.4. Code

The codebase is available on github[1]. However, in this section, we set out to highlight some of the fundamental parts of the code.

The first part of the essential code is the algorithm created in [ERE20], which has the ability to determine the maximum reward given a party and the fee space. It divides the search space into

[1] `https://github.com/DMoonen/reactionary-games-impact-analysis`

different intervals and searches them in order to find the maximum rewards. By taking into account the maximum and minimum fee values in the interval as well as the betweenness centrality the optimization steps from 4.7.1 can be performed.

```python
"""Function that maximizes channel rewards, by efficiently searching different fee values.
By calculating the maximum theoretical reward for an interval, intervals can be discarded
    aiding in the search.

:param graph: The graph object.
:param min_fee: Lower bound of the search space.
:param max_fee: Upper bound of the search space.
:returns: Void. Return is stored in a global variable.
"""
def maximize_channel_reward(graph, min_fee, max_fee):
    global max_rew_fee
    global max_rew
    global global_max_rew
    global global_rewards
    global edge_global_rew

    er = np.zeros(div + 1)
    er_max = np.zeros(div)

    # If the different fee values present, are less then the amount of divisions.
    # Enter the base
    if max_fee - min_fee <= div:
        # For all the fee candidates, calculate the optimal fee
        for fee in np.arange(min_fee, max_fee + 1):
            if global_rewards[fee] == 0:
                e_rew, r_rew = compute_node_rew_init(fee, graph)
                if e_rew == 0.0:
                    # Therefore in stead of computing them, we'll set them here
                    for index_2 in np.arange(0, div + 1):
                        fee_2 = ((max_fee - min_fee) * index_2 // div) + min_fee
                        if fee_2 > fee:
                            global_rewards[fee_2] = e_rew + r_rew
                            edge_global_rew[fee_2] = e_rew

                global_rewards[fee] = e_rew + r_rew
                edge_global_rew[fee] = e_rew

            er_local = global_rewards[fee]
            # If the calculated fee yields a better reward than the current best, replace it.
            if er_local > global_max_rew:
                max_rew_fee = fee
                global_max_rew = er_local
                max_rew = global_max_rew
        return
    # Else/ Recursion
    else:
        # Separate the fee values into divisions
        for index in np.arange(0, div + 1):
            # Set current div fee
            fee = ((max_fee - min_fee) * index // div) + min_fee
            # Compute fee yield
            if global_rewards[fee] == 0:
                e_rew, r_rew = compute_node_rew_init(fee, graph)

                if e_rew == 0.0: # if e_rew is 0.0, we will obtain the same value for all
    divs greater then current.
                    # Therefore in stead of computing them, we'll set them here
                    for index_2 in np.arange(0, div + 1):
                        fee_2 = ((max_fee - min_fee) * index_2 // div) + min_fee
                        if fee_2 > fee:
                            global_rewards[fee_2] = e_rew + r_rew
                            edge_global_rew[fee_2] = e_rew

                global_rewards[fee] = e_rew + r_rew
                edge_global_rew[fee] = e_rew

            er[index] = global_rewards[fee]
            # If fee yield is better than current best update
            if er[index] > global_max_rew:
                max_rew_fee = fee
                global_max_rew = er[index]
                max_rew = global_max_rew
            if er[index] == 0:
                break

        # Compute the maximum possible reward for the div
        for index in np.arange(0, div):
            # f_i
            fee = ((max_fee - min_fee) * index // div) + min_fee
            # f_i+1
            fee_next = ((max_fee - min_fee) * (index + 1) // div) + min_fee
            # (r_i * f_i+1) // f_i + (R_i+1 - r_i+1) => (r_i * f_i+1) // f_i + r'_i+1
            er_max[index] = (edge_global_rew[fee] * fee_next) // fee + (er[index + 1] -
    edge_global_rew[fee_next])

        # Recursively call the interval that contains the highest reward
        for index in np.arange(0, div):
            if er_max[index] > global_max_rew:
                rec_min_fee = ((max_fee - min_fee) * index // div) + min_fee
                rec_max_fee = ((max_fee - min_fee) * (index + 1) // div) + min_fee
                maximize_channel_reward(graph, rec_min_fee, rec_max_fee)
```

The second part of essential code is the code used to run the experiments. There are many cases and placement strategies that will be tested therefore we will detail the high-level abstraction of the experiment loop. Algorithm 1 describes the iterative channel placement game in terms of pseudocode.

Algorithm 1 Iterative Channel Placement

```
1: $channel\_amount \leftarrow 5$
2: for $placement\_algorithm = uniform, highest\ degree, \ldots, fee\ weighted\ centrality$ do
3:     $N \leftarrow \{all\ parties\ in\ the\ network \mid n = (lt, fee)\}$
4:     $E \leftarrow \{all\ channels\ in\ the\ network \mid e = (u, v, fee)\}$
5:     $Calculaterewards$
6:     for $channel = 1, 2, \ldots, channel\_amount$ do
7:         $n' \leftarrow Party\_Choice(N, E, placement\_algorithm)$
8:         $e' \leftarrow Create\_Channel(n')$
9:         $E \leftarrow E \cup e'$
10:        Calculate rewards
11:    end for
12: end for
```

The iterative channel placement algorithm analyses all the different placement strategies, initializing a clean starting state every run. After this step, #channel_amount number of channels are created using the action space, ensuring the calculation rewards at the appropriate times.

Note that this pseudocode details the experiment loop for the cases where the leading party operates in isolation. The cases in which the action space is extended differ slightly in the steps present. Furthermore, in our experiment we analyze different transaction amounts, this would imply we need to analyze 3 different graphs.

The third part of the essential code is an alteration we have made. We have expressed the need for us to punish parties that receive rewards from ”routing” transactions that originate from themselves. The code shown below is the additional code, that subtracts the reward that stems from routing one’s own transactions.

```
def remove_own_betweenness_score(graph, src_id, bet_scores):
    node_ids = graph.nodes()

    for dst in node_ids:
        if src_id != dst:
            paths = nx.all_shortest_paths(graph, src_id, dst, weight='weight')
            path_list = list(paths)
            path_weight = 1 / len(path_list)
            for path in path_list:
                bet_scores[(path[0], path[1])] -= path_weight

    for key in bet_scores.keys():
        value = bet_scores[key]
        if np.abs(value) < 0.0001:
            bet_scores[key] = 0.0
    return bet_scores
```

The code above will adjust the betweenness centrality scores stored in bet_scores. It will do so, by recreating all shortest paths originating from the src_id and subtracting the weighted score from the appropriate channel used in the shortest path(s). Because Python suffers from floating point errors, we remove any residual value if it is smaller than 1 millionth of a score point.

4.5. Testing

As part of this thesis, we have tested the correctness of our code and enhanced the reliability of our results. However, testing large networks presents us with a challenge. The main challenge we faced was related to the size of our graph. Running tests on a large network is time-consuming, and given the constraints of the overall test framework, we were unable to perform them on our large graphs. To mitigate this challenge, we used small representative graphs to perform unit and integration tests. However, it is important to note that these results are not guaranteed to be transferable to the larger network. Unfortunately, addressing this inherent problem of working with large networks was beyond the scope of this thesis.

Regardless of the transferability problem, we were able to perform checks on the experiments. One such check was to analyze the return data generated by the experiment and compare it with our expected outcomes. Any discrepancies between the output and the expected results indicated the possibility of a mistake. These discrepancies would either need to be explained as unexpected but valid results, or analyzed to fix faulty behavior. This method proved to be useful in identifying and addressing bugs.

In addition, we used another fruitful check in the Stackleberg cases. We compared the reward values obtained during the decision-making process, with the actual rewards obtained. Verifying that the rewards obtained are indeed higher or equal to the expected rewards. This test then verifies the expected behavior of the code even further.

4.6. Servers

As part of our experiment, we have made use of two different servers to run our experiments in parallel. The first server we have made use of is the DAS6 server which provides clustered computing power. Each cluster consists of multiple nodes which each have access to a dual 16-core CPU with a clock speed of 2.8 GHz. For this experiment, we made use of 6 of these nodes.

The architecture of the DAS6 server, with its ability to run multiple parallel programs for a relatively short duration, was an ideal setup for conducting our baseline, and additional party cases. These cases did not involve the computationally expensive task of updating fees, which allowed us to complete all of the baseline and additional party experiments within the span of 24 hours.

Please note that the use of the DAS6 server was specifically chosen for these cases due to their time constraints, as other experiments exceeded the server's time limitations.

For the remaining experiments that analyzed the impact of channel fee updates, we used a private machine with 16 cores, and a clock speed of 3.2 GHz. Running the remaining scripts would take approximately 18 days, with an average runtime of 1.29 days per script.

Additionally, we used this machine to perform the Stackleberg experiments and to perform the comparison to previous work that uses the snapshot as the network. All of these experiments exceeded one week in runtime, highlighting their computationally intensive nature. The use of a private machine allowed us to perform the experiment with longer runtime requirements.

4.7. Setup

The setup of the experiment involves combining the graphs that we have created as described in section 4.2. These networks that simulate small, medium, and large transactions were all tested in the different cases detailed in section 4.3. The comparison to the lightning snapshot, the optimization analysis, the impact of the 4 different action spaces, and the two game-theoretical experiments were run on the server architecture described in section 4.6. This resulted in the data described in section 5.

4.7.1. Performance

After taking advantage of the performance increase that parallelization allows for, the time it would take to run the computationally expensive code was still too long. Therefore time was taken to investigate where the performance can be further increased. Three such increases were found and are detailed below. When calculating the optimal fee to charge for making use of a certain channel we can reduce the search space based on information we find during the search. The first improvement as found in [ERE20] makes use of intervals. If the search space is divided into different intervals, it allows us to compute the maximum hypothetical reward that this interval can yield. If this highest maximum reward is lower than the current best score, we can simply skip the interval, as searching it will not yield a better result. The second improvement improves the search even further. If the channel reward is zero, we do not have to compute fee values higher than the one currently calculated. The following example shows why this is the case. Take the case of $R_i = r_i + r_i'$, where R_i represents the total reward of a party, where r_i and R_i' represent the observed channel and the other channels of a certain party, and where $r_i = 0.0$. Since $r_i = 0.0$, we conclude that the channel given its current fee value, the channel performs no routing and therefore has a reward of 0.0. If we combine this knowledge with the fact that increasing the fee value of a channel will never increase the usage (or betweenness centrality) of the channel we can conclude that the channel reward will also be 0.0 for the fee values higher than we are observing at state i. This allows us to simply assign the value of R_i we have calculated which allows us to reduce computation. This information can then later potentially be used to discard intervals and reduce the search space reducing the computation even further. Note that this only works in the case of searching from low values to high values. If we were to take the case $R_i = r_i + r_i'$, where i is the current fee observed, and where $r_i' = 0.0$. Here we know that the observed channel undercuts all other outgoing channels, making it solely responsible for the reward R_i. However, if we were to lower the fee for the betweenness centrality we cannot ensure that this increased edge betweenness will always lead to a lower reward when multiplied by the lower fee.

Similar to the second improvement, we can potentially reduce the search space by precomputing fee values. Analysis shows that there exists a fee value that is most common. We will take the fee value 1000 as an example, and therefore we precompute the fee value of 999 (most common occurrence - 1) and 1001 (most common occurrence + 1) updating the best reward found. Having potentially found a higher reward value, the hypothetical reward to be found in an interval needs to be higher for the algorithm to not disregard the interval. Probing an interval around the value that occurs the most, allows us to start our analysis with a hypothetical reward that is similar to the most occurring hypothetical reward value. If we then encounter intervals with a lower hypothetical reward value than we have precomputed, we can skip these intervals and thereby reduce our search space.

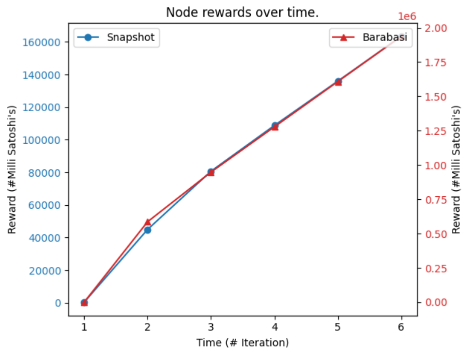
4.7.2. Result normalization

During the action space impact analysis, it is our aim to compare the results of the different action spaces to each other. However, we are faced with a problem, namely that there is a difference in the number of participants between the experiments. This leads to a different overall reward pool from which parties gain. In the baseline, and network update analysis, there are 201 parties that perform transactions within the network. However, in the cases that contain an additional party, there are 202 parties, increasing the total transactions being sent. In the case of 201 parties, there would be $201 \cdot 200 = 40,200$ transactions, whereas with 202 parties $202 \cdot 201 = 40,602$ transactions, which is 402 paths more. This number might seem low but can have a large impact. A path can have multiple hops, which means it is routed multiple times. If we assume that a transaction has 3 routes on average, and all channels use the default fee, then that means that there are $3 \cdot 1000 \cdot 402 = 1.206.000$ satoshi's more in the reward pool. To account for this, we have normalized the rewards that stem from networks with 202 parties by a factor of $40,200/40,602 \approx 0.990$, which allowed us to compare the results of the different action space experiments to each other.

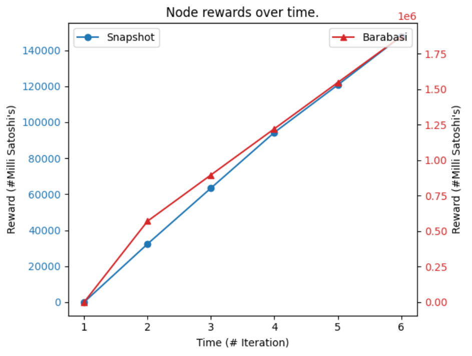
5 Results

This chapter covers the results of the experiments described in section 4. Section 5.1 covers the results of the replication, whereas section 5.2 covers the results of the fee distribution analysis. Section 5.3 shows the results of the impact analysis, whereas section 5.4 covers the results of the game-theoretical analysis. It must be noted that these sections cover some discussion and conclusions where it makes sense to introduce them.

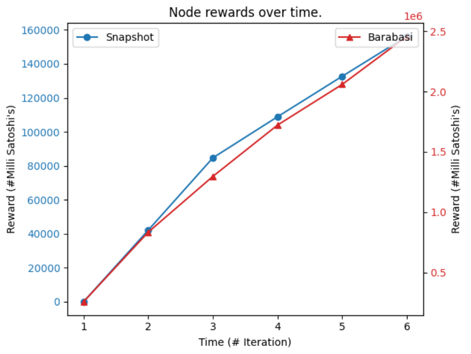
5.1. Replication



(a) Figure depicting the reward obtained in the snapshot, and the Barabasi-Albert graph using placement strategy: fee weighted centrality and transaction amount: 100



(b) Figure depicting the reward obtained in the snapshot, and the Barabasi-Albert graph using placement strategy: fee weighted centrality and transaction amount: 10000



(c) Figure depicting the reward obtained in the snapshot, and the Barabasi-Albert graph using placement strategy: fee weighted centrality and transaction amount: 1000000

Figure 5.1: Figures depicting reward trends when comparing a snapshot of the Bitcoin Lightning Network to the graphs created by using the Barabasi-Albert algorithm

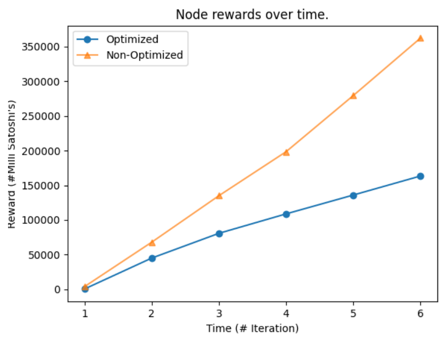
As can be seen in figure 5.1, the reward line of the snapshot is similar in pattern to that of the

Barabasi reward line where the difference lies in the y-axis. After normalizing the reward of the snapshot (by a factor $(201 \cdot 200)/(1111 \cdot 1110) \approxeq 3.26\%)$ such that we can compare them as found in 4.7.2 the rewards obtained from the snapshot are still around 2 million satoshis in figures 5.1a (1938011.84) and 5.1b (1871425.22). This reaches even higher numbers in figure 5.1c (2457294.52) where the leading party is able to obtain 2.5 million satoshis. The rewards for the Barabasi network differ in size and range around 150 thousand satoshis. 163376 when observing small transactions, 147745 for medium-sized transactions, and 156251 for transactions large in size.

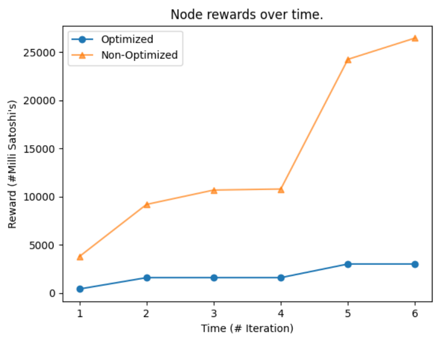
From these findings, we conclude that the behavior observed by the leading party in the snapshot is comparable to the behavior of the leading party in the Barabasi network. However, it must be noted that there is a measurable difference between the reward number the two networks display. If one wanted to translate obtained rewards from the Barabasi network to the snapshot, the reward would need to be increased by at least a factor of 10 to obtain a reward that is comparable to those obtained from a snapshot.

5.2. Fee Noise Analysis

This section details the results of the experiment described in section 4.3.2. In this section, we detail some representative graphs that follow from our analysis. The complete list of graphs can be found in the appendix.



(a) Figure depicting the channel placement game rewards with placement strategy: fee weighted centrality and transaction amount: 100



(b) Figure depicting the channel placement game rewards with placement strategy: highest degree and transaction amount: 100

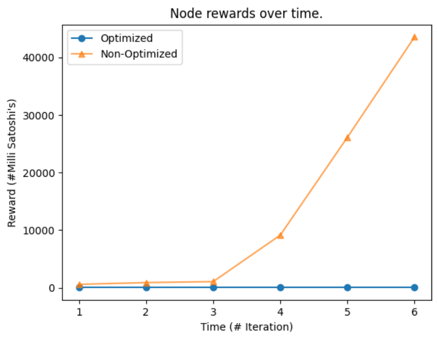
Figure 5.2: Figures depicting reward trends when comparing a network with and without fee noise in a channel placement game.

Figure 5.2a denotes the reward that can be obtained by an actor when they operate within a network that was not optimized before performing case 1 as an experiment. The baseline as an overlay in the figure shows the rewards an actor can obtain when they perform the experiment on a graph that has been optimized using fee weighted centrality on each channel in the network. Most of the figures show this behavior, although some present this behavior more than others. The average impact seems to be around 50%. Figure 5.2b is an extreme case, which shows a discrepancy between the rewards at iteration 1. This indicates that even a small difference in fees across the network has a large impact on this placement strategy since the first channel already starts with over 9 times the rewards than the network where no fee noise has been added.

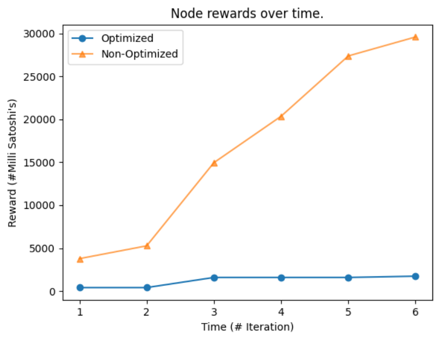
Figure 5.3a depicts unexpected behavior only present when the highest degree placement algorithm is used and the transaction amount is 1000000. Here the graph optimization process impacts the results of the experiment in such a way that the baseline does not seem to trend upwards at all. The reward of the graph where the graph optimization is not performed also has difficulty trending upwards. However, after placing multiple connections it does obtain the ability to route transactions and thus profit.

Figure 5.3b depicts another case of unexpected behavior, present in all three cases where betweenness centrality is used as placement algorithm. Here the placement strategy also starts off with a discrepancy between the network with and without noise from the first channel onwards.

Besides betweenness centrality, the rewards of all placement strategies are negatively impacted when optimizing the experiment graph beforehand. The degree of impact varies based on placement strategy but is sufficiently large that it needs to be taken into account whether noise has been added



(a) Figure depicting the channel placement game rewards with placement strategy: highest degree and transaction amount: 1000000



(b) Figure depicting the channel placement game rewards with placement strategy: betweenness centrality and transaction amount: 100

Figure 5.3: Figures depicting unexpected behavior when comparing a network with and without fee noise in a channel placement game.

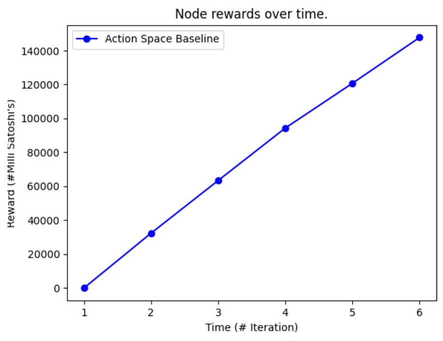
to the graph during the experiment.

5.3. Action Space Impact Analysis

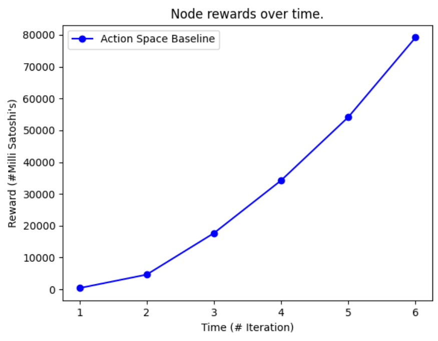
As previously mentioned in section 4.1 we have divided the goal of taking the existing work out of its vacuum into different cases. This section explores the obtained results from the experiments on these cases. Subsection 5.3.1 highlights the results of the baseline which is needed to provide context to cases 2 and 3. Subsection 5.3.2 shows the results obtained when the model allows for the existing network to react to the connections made. Subsection 5.3.3 shows the impact on the result when a new actor not previously present starts competing for rewards within the network. Subsection 5.3.4 combines the model space of 5.3.2 and 5.3.3 and allows for both the existing network and a new party to react on connections created within the network. Because it combines the model spaces, it will also be relying on cases 2 and 3 to serve as a baseline.

5.3.1. Case 1: Baseline

In the first case of the impact analysis, it is only the leading party that is allowed to interact with the network and plays a channel creation game in isolation. Below we detail and interpret the results that will be used as a baseline to compare the changes in action space that will follow.



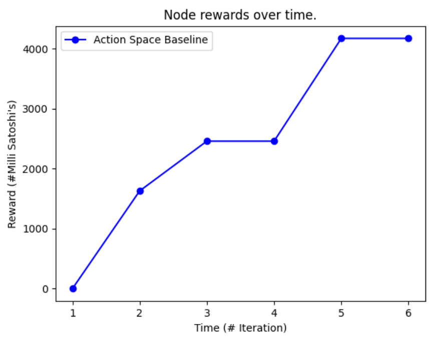
(a) Figure depicting the channel placement reward obtained using placement strategy: fee weighted centrality and transaction amount: 10000



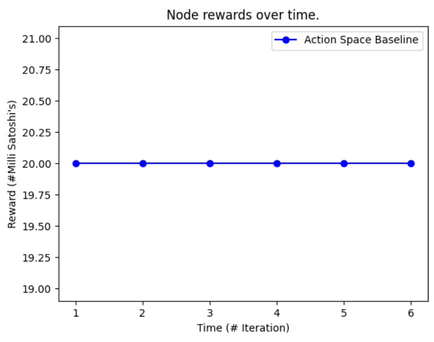
(b) Figure depicting the channel placement reward obtained using placement strategy: k-center and transaction amount: 100

Figure 5.4: Figures depicting reward trends when playing a channel placement game with a limited action space.

If we observe all 18 graphs that depict reward data, we observe a general pattern that as the number of created channels increases, so does the reward a party is able to obtain. This pattern is displayed in two trends. The first trend is that of an exponential increase, whereas the second trend displays a logarithmic trend. Figures 5.4a and 5.4b are examples of this general trend, where 5.4a shows the logarithmic trend and 5.4b displays the exponential trend. Both graphs are representative of their respective trend and are caused by the placement strategy used. Placement strategies that are able to leverage every new channel to its maximum value will find it difficult to keep up the growth in rewards. As a result, they display a logarithmic trend where every new channel increases the total reward by a large amount, but the additional reward that a new channel brings slightly decreases with every channel created. We can take 5.4a as an example, we take a look at its partial rewards: [0.0, 32342.0, 63401.0]. Where the first additional channel adds 32342.0 to the reward pool, the second additional channel adds 31059.0. Whilst adding 31059.0 to the rewards is noteworthy, it is 1282 less than what the first additional channel provides. Something similar happens for the placement strategies that display an exponential pattern. 5.4b shows the reverse. Here every new additional channel has diminishing returns.



(a) Figure depicting the channel placement reward obtained using placement strategy: highest degree and transaction amount: 10000



(b) Figure depicting the channel placement reward obtained using placement strategy: highest degree and transaction amount: 1000000

Figure 5.5: Figures depicting unexpected behavior when playing a channel placement game with a limited action space.

Figure 5.5 depicts unexpected behavior in the placement algorithms. These figures deviate from the patterns found and described earlier. Figure 5.5a displays a deviation from the logarithmic trend we observed. At iterations 4 and 6 an additional channel created does not contribute to an increase in rewards for the leading party. Figure 5.5b displays this behavior to an even larger extent, where none of the channels are able to increase the reward.

Since the highest degree placement strategy has already been shown to have difficulty in obtaining reward during the initialization (as seen in figure 3.1), the comparison performed in section 5.2 (as seen in figure 5.3b), this behavior while not expected, can be explained. Poor performance is a side effect that occurs when making use of the highest degree in combination with one of the assumptions made about the model. The assumption in 3.1 that we do not have control over the other side of the bidirectional channel is the assumption that explains this behavior.

In our model, we work with optimized channels which means that channel fees are optimized. Furthermore, parties with the highest degree are likely to be connected to each other. When a new route is created between the leading party and the two parties with the highest degree the chance occurs that the existing optimized channel will be less expensive than the route that makes use of the leading party. If the existing channel has a fee of less than the default then no matter what the leading party charges as a fee, the part of the channel that the leading party does not control will cost the default fee and will outprice the route. When it is never beneficial to make use of a route, then it follows that this route will not be performing any routing. Furthermore, in this case of the impact analysis, it is not possible to change the fee value after the channel has been created. In cases 2 and 4 it is possible for this value to be changed when a network optimization occurs. Therefore in these cases, it might be possible for the channels to provide their parties with rewards from routing.
Table 5.1 allows us to compare the reward of the different placement strategies and transaction sizes to each other. When comparing the different transaction sizes we observe that the average reward value is similar. Where the differences lie is in which placement strategy was used. We observe that placement strategies betweenness centrality, highest degree both perform worse than a uniform random choice. The fee weighted centrality, k-center, and k-means placements strategies were able to outperform a

	Small	Medium	Large	Average
Betweenness Centrality	1732.0	4133.0	1037.0	2300.67
Fee Weighted Centrality	163376.0	147745.0	156251.0	155790.67
Highest Degree	2999.0	4171.0	20.0	2396.67
K-center	79193.0	85680.0	48226.0	71033.0
K-means	105508.0	117751.0	141822.0	121693.67
Uniform Random	53415.5	33962.65	31013.0	39463.72
Average	67703.92	65573.78	63061.5	

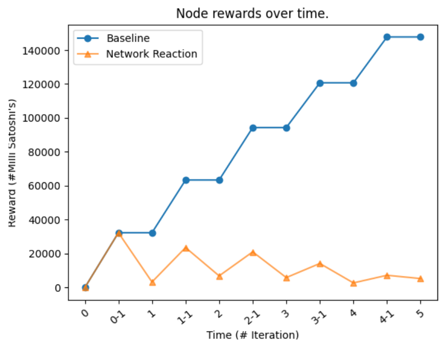
Table 5.1: Table depicting the rewards of a channel placement game.

uniform random choice.
From our results we conclude that with one exception, all placement strategies are able to increase their reward without competition, regardless of transaction size. However, the level at which the different placement strategies are able to obtain rewards differs. Furthermore, we conclude that our results are sufficient to serve as a baseline to measure the impact on rewards that other action spaces will have. We conclude that fee weighted centrality performs the best of the different placement strategies while k-means place as second best. Strategies betweenness centrality and highest degree perform the worst in this setting, even worse than picking at random. The fact that the average network reward is similar to each other is to be expected.

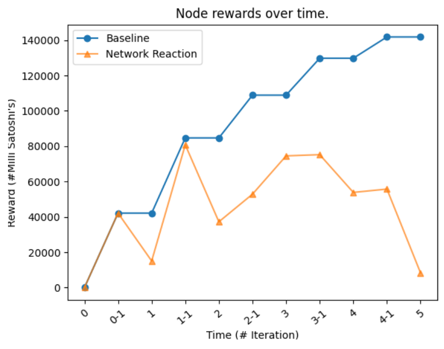
Our networks are created using the Barabasi-Albert algorithm which is scale free. Therefore an increase in network size should not lead to a difference in network structure. What we did not expect was for our results to indicate network preference for some of the placement strategies. Placement strategies k-center seems to prefer small and medium-sized networks over large ones. Whereas the uniform random placement strategy seems to prefer the small-size network. However, our results are not conclusive enough to conclude with confidence whether or not this preference is inherent to transaction size.

5.3.2. Case 2: Network Reaction

In the second case of the impact analysis, we extend the action space to allow the network to react to every change made by the leading party. Every party is given the ability to change the fee on all of their channels. Below we detail and interpret the results from the experiment and compare the results to the baseline.



(a) Figure depicting the channel placement rewards obtained using placement strategy: fee weighted centrality and transaction amount: 10000



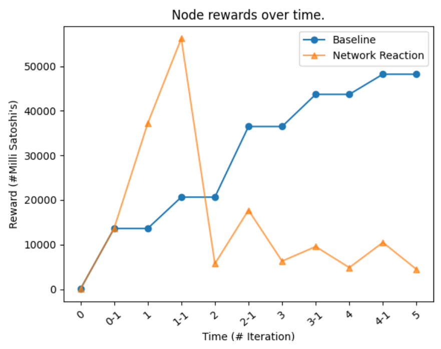
(b) Figure depicting the channel placement rewards obtained using placement strategy: betweenness centrality and transaction amount: 10000

Figure 5.6: Figures depicting reward trends when playing a channel placement game with an action space extended to allow the network to react.

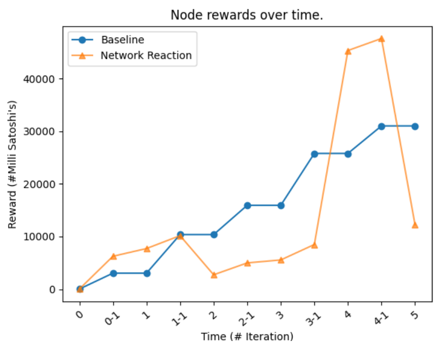
Similar to 5.3.1 the results of the 18 reward lines display patterns. Figure 5.6 shows us a common pattern that are easy to observe. In figure 5.6a we observe that the reward that the leading party is able to obtain differs from the rewards gained by the baseline. It shows a setting where the leading party is not able to keep up its reward growth and over the duration of the game loses its ability to profit. The leading party ends the experiment with 3.60% of the rewards it would have obtained in the baseline experiment.

Figure 5.6b shows the case where the leading party is able to maintain its profits for a couple of iterations longer before eventually having its rewards taken by other parties present in the network.

At steps 0,1,2,...,5 it is up to the leading party to place a channel, that is why on steps 0-1,1-1,2-1,...4-1 we observe the temporary increase in rewards. The reason that these are only temporary is that during steps 0-1,...,4-1 the network reaction takes place. This has the effect of lowering the reward of the leading party. The leading party ends the experiment with 0% of the rewards it would have obtained in the baseline.



(a) Figure depicting the channel placement rewards obtained using placement strategy: k-center and transaction amount: 1000000



(b) Figure depicting the channel placement rewards obtained using placement strategy: uniform random and transaction amount: 1000000

Figure 5.7: Figures depicting unexpected behavior when playing a channel placement game with an action space extended to allow the network to react.

Figure 5.7 covers figures that displayed unexpected behavior in the reward lines. Specifically, the fact that the reward of the network reaction temporarily exceeds the reward the leading party could obtain when playing in the baseline experiment. In figure 5.7a at steps 1 and 1-1 the reward goes to 37164.0 and 56244.0 after which the reward goes down to 4441.5 which is 9.20% of the baseline. Figure 5.7b displays the same pattern. However, the rewards go down in the next network reaction, and the experiment ends with 12204.75, which is 39.35% of the baseline experiment.

	Small	**Change(%)**	**Medium**	**Change(%)**	**Large**	**Change(%)**
Betweenness Centrality	270.4	−84.39	0.0	−100	1021.33	−22.29
Fee Weighted Centrality	2198.12	−98.65	5311.83	−96.40	6491.83	−95.85
Highest Degree	267.33	−91.09	0.0	−100	352	1760
K-center	486.07	−99.38	1370.83	−98.40	4441.5	−90.79
K-means	2356.33	−97.77	4430.5	−96.24	8330.42	−94.13
Uniform Random	15084.1	−71.76	10762.75	−68.31	12204.75	−60.65
Average	3443.73	−90.51	3645.99	−93.23	5473.64	232.72

Table 5.2: Table depicting the rewards of a channel placement game.

The data in table 5.2 affirms the behaviour observed in the graph results. Observing the changes in rewards compared to case 1: the baseline, we observe a severe impact on the ability to profit. From the results in the table betweenness centrality and highest degree display unexpected behavior. In the case of highest degree this is due to it being the only reward that has increased and in the case of betweenness centrality because its reward has not gone down by more than 60%.

In both cases, this behavior can be explained by the fact that these placement strategies have difficulty obtaining rewards in the baseline. Therefore it is more difficult for the network reactions to impact these already low reward scores. The average change without these 2 scores would be -85.35% which falls in line with the results obtained in the experiments using small and medium transaction sizes. Furthermore, without these cases, there exists no placement strategy that has outperformed uniform random, which indicates that all of the placement strategies are negatively impacted by the ability of the network to react.

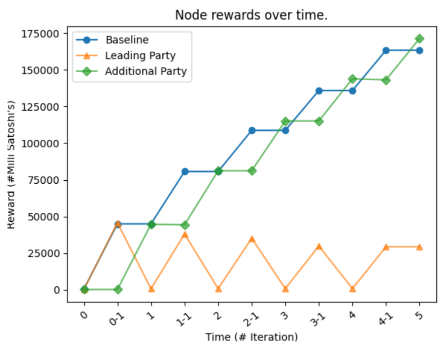
The rewards show that the highest degree placement strategy is able to increase its rewards when channel fees are allowed to be changed. This then confirms the explanation stated in 5.3.1, which hypothesized that the highest degree rewards would increase.

We conclude that the rewards are negatively impacted by extending the action space to allow for network reaction. In some cases, a leading party is able to briefly keep up with or outperform the rewards obtained when playing in isolation. However, given five iterations of placement its reward will decrease. A leading party that uses a placement strategy that does utilize randomness is bound to lose 85-100% of their rewards.

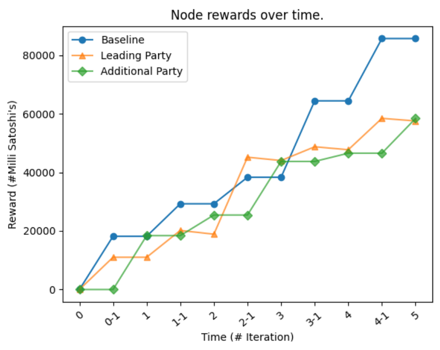
The general pattern remains. The ability for the network to adjust its fees, combined with the goal of each party to maximize its own rewards leads to the leading party being outperformed due to the channel charging lower fees, undercutting parties throughout the network. This behavior was to be expected. We conclude that the negative impact is regardless of transaction size or placement strategy and conclude that it is related to the number of parties in the network.

5.3.3. Case 3: An Additional Party

In the third case of the impact analysis, we remove the ability for network reactions but extend the action space to allow an additional party to join. This additional party will react to the leading party by placing a channel of their own. The most representative figures have been shown below:



(a) Figure depicting the channel placement rewards obtained using placement strategy: fee weighted centrality and transaction amount: 100



(b) Figure depicting the channel placement rewards obtained using placement strategy: k-center and transaction amount: 10000

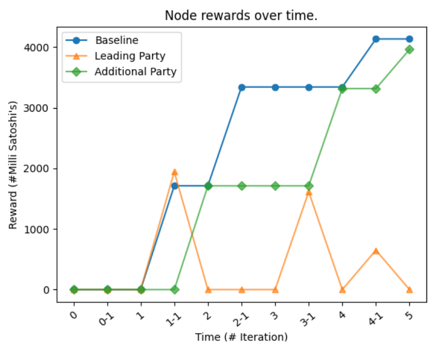
Figure 5.8: Figures depicting reward trends when playing a channel placement game with an action space extended to allow an additional party to react.

Figure 5.8 shows the general trend for the reward lines present in the data. This trend can be divided into two patterns of which we have chosen two representative examples. Figure 5.8a represent the cases where the reward of the leading party is lowered, whereas the additional party performs on the same level as the baseline. In steps 1 to 5 we again observe the actions of the leading party which increase its rewards by creating a channel. Again in steps 0-1,...,4-1 we observe that these rewards are lowered whilst the additional party increases their reward.

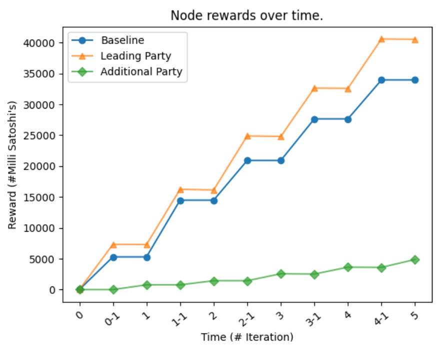
Figure 5.8b represents the pattern where the leading player is able to compete with the additional party. Here both the leading party and the additional party generally perform slightly worse than the baseline, but the reduction in reward is less for the leading party. Similar to 5.3.2 we observe undercutting to obtain rewards from competitors. However, some placement strategies like k-center operate in a way that makes it more difficult for the additional party to immediately undercut. Its strategy lowers the chance that the additional party will choose the same destination, which ensures that direct competition presents itself less often, allowing the leading party to avoid being undercut.

Figure 5.9 displays unexpected behavior observed in the results. Figure 5.9a shows that the competition of the additional party can even reduce the obtained rewards to 0 for the leading party. Regardless of the ability of the placement strategy to obtain rewards, the competition that one additional party poses can be a threat to the ability to obtain rewards.

Figure 5.9b shows that it is even possible for the leading party to outperform the baseline after the



(a) Figure depicting the channel placement rewards obtained using placement strategy: betweenness centrality and transaction amount: 10000



(b) Figure depicting the channel placement rewards obtained using placement strategy: k-center and transaction amount: 1000000

Figure 5.9: Figures depicting unexpected behavior when playing a channel placement game with an action space extended to allow an additional party to react.

reward normalization from 4.7.2 has been performed. Similar to 5.8b, the placement strategy can be utilized to perform well and reduce competition with the additional party. However, in this setting, it further increases the ability of the leading party to limit competition and increase its rewards.

	Small	**Change(%)**	**Medium**	**Change(%)**	**Large**	**Change(%)**
Betweenness Centrality	196.04	−88.68	0.0	−100	8.91	−99.14
Fee Weighted Centrality	29340.59	−82.04	31370.3	−78.77	22.28	−99.98
Highest Degree	196.04	−93.46	0.0	−100	8.91	−55.45
K-center	78555.45	00.80	57545.54	−32.84	50157.43	104
K-means	136876.24	129.73	112612.87	−4.36	97828.71	−31.02
Uniform Random	49411.53	−7.50	40547.77	119.39	29762.18	−4.03
Average	49095.98	−23.53	40346.08	−32.76	29631.40	−30.94

Table 5.3: Table depicting the experiment rewards. Performance is relative to the performance of Case 1: Baseline as found in Table 5.1

Table 5.3 compares the rewards that the leading party obtains to the baseline. As can be seen in the table the addition of an additional party may have a negative influence on the ability of the leading party to obtain rewards. What is interesting about this data is that the addition of the party impacts the different placement strategies in a different way. Where some placement strategies perform poorly across the board, others seem to perform better under some circumstances. K-center seems to perform well in the network with large transaction sizes, whereas k-means seems to perform better when small transaction sizes are sent. Both k-means and k-center shows limited negative impact by the additional party, however perform not as well as random uniform selection. The average impact on rewards is around -30%.

Table 5.4 displays the difference in rewards between the leading and the additional party. As can be seen, it is often the case that the additional party ends the experiments with a higher reward than the leading party. The additional party is able to obtain 65.6 times the rewards of the leading party when the betweenness centrality placement strategy is used. When fee weighted centrality is used the additional party outperforms by 7 times and highest degree 34 times. However in the cases of k-center, k-means, and uniform random the additional party only obtains 0.89, 0.87, 0.14 times the reward that the leading party is able to obtain on average.

There is also a difference in the average rewards the different placement strategies are able to obtain. On average the network with small transactions has the highest rewards whereas the network with large transactions has the lowest average reward.

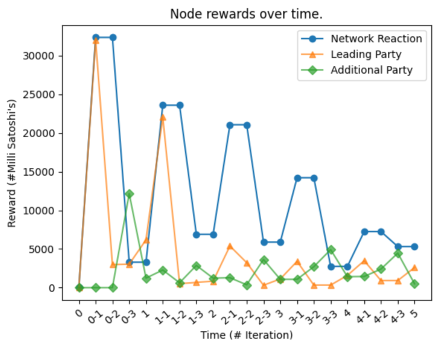
We conclude that in general the addition of a party negatively influences the ability of the leading party to obtain rewards. Competing channels that share the same destination are predetermined to undercut each other and reduce the rewards of one of the channels.

	Small	Medium	Large	Average
Betweenness Centrality - Leading	196.04	0.0	8.91	68.32
Betweenness Centrality - Additional	8483.17	3960.4	1006.93	4483.5
Fee Weighted Centrality - Leading	29340.59	31370.3	22.28	20244.39
Fee Weighted Centrality - Additional	171361.39	113430.69	151248.02	145346.7
Highest Degree - Leading	196.04	0.0	8.91	68.31
Highest Degree - Additional	2744.55	4131.68	8.91	2295.05
K-center - Leading	78555.45	57545.54	50157.43	62086.14
K-center - Additional	78139.6	58380.2	28350.5	54956.77
K-means - Leading	136876.24	112612.87	97828.71	115772.61
K-means - Additional	112365.35	80236.63	109213.86	100605.28
Uniform Random - Leading	49411.53	40547.77	29762.18	39907.16
Uniform Random - Additional	3134.65	4876.04	8743.27	5584.65
Average	55900.38	42257.68	39696.66	

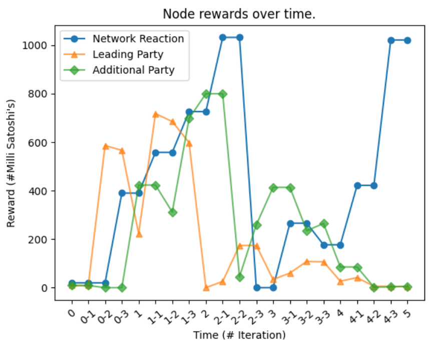
Table 5.4: Table depicting the rewards of a channel placement game.

5.3.4. Case 4: Combined Action Space

In the third case of the impact analysis, the action space will be extended to allow both an additional party to join the network (as in case 3), and we will allow existing channels to be updated (as in case 2). Having studied the two cases independently, combining them allows us to study their full impact.



(a) Figure depicting the reward obtained when case 4 is run using placement strategy: fee weighted centrality and transaction amount: 10000



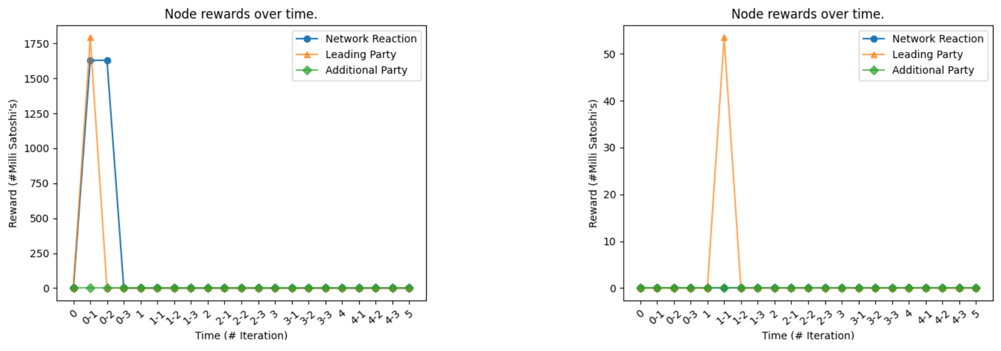
(b) Figure depicting the reward obtained when case 4 is run using placement strategy: betweenness centrality and transaction amount: 10000

Figure 5.10: Figures depicting reward trends when playing a channel placement game with an action space extended to allow an additional party and the network to react.

Similar to the other cases, we have selected figures to represent the trends present in the reward lines. In figure 5.10, we will be looking at the reward data of case 4. At steps n the leading party is given the choice to place a channel. In steps n-1 the network is given the ability to change the fee on channels. At steps n-2 the additional party places their channel after which the network reacts again in steps n-3. Since the rewards of Case 2: Network Reaction impact the rewards of the leading party more than Case 3: An Additional Party we interpret the rewards of Case 4 by using the rewards of Case 2 as a baseline so that we may more easily observe the impact of Case 3.

While the rewards of figure 5.10b are small in size, it does show even more clearly the behavior of the reward line. At step 1 the leading party places a connection to which the resulting network reaction lowers it slightly in 1-1. The additional party that places a connection after this network reaction is 1-2 which even further lowers the leading party's reward score. The network reaction that follows this step then makes the reward drop the most in 2. Having profitable channels within the network rewards the leading party, whereas being undercut by an additional party or the whole network decreases the rewards.

When a placement strategy is able to perform well in obtaining rewards the same thing occurs, however, the impact of undercutting has a larger impact on rewards. Figure 5.10a is a representative example that displays this behavior. As can be observed the decrease in rewards occurs the first time the network reacts as can be seen in steps 0-2.



(a) Figure depicting the reward obtained when case 4 is run using placement strategy: highest degree and transaction amount: 10000

(b) Figure depicting the reward obtained when case 4 is run using placement strategy: betweenness centrality and transaction amount: 10000

Figure 5.11: Figures depicting unexpected behavior when playing a channel placement game with an action space extended to allow an additional party and the network to react.

Figure 5.11 shows the unexpected behavior of two of the placement strategies. Figure 5.11a, as well as figure 5.11b, show the placement strategies that performed poor, both in case 2 and in case 4. While these placement strategies have performed poorly in previous experiments they usually were able to perform decently in the experiments where the transaction size was medium. However, these placement strategies have performed poorly regardless of transaction size, which is consistent with expectations.

	Small	**Change(%)**	**Medium**	**Change(%)**	**Large**	**Change(%)**
Betweenness Centrality	0.0	−100	0.0	−100	4.35	−99.57
Fee Weighted Centrality	3345.56	152.20	2605.35	−50.95	5161.01	−20.50
Highest Degree	0.8	−99.85	0.0	−100	5.42	−98.46
K-center	548.27	112.80	190.88	−86.08	466.44	−89.50
K-means	3741.16	−58.77	46.21	−98.96	9563.63	114.80
Uniform Random	6839.41	−54.66	4389.28	−59.22	26747.65	219.16
Average	2412.53	−8.0	1205.29	−82.54	6991.42	4.32

Table 5.5: Table depicting the experiment rewards. Performance is relative to the performance of Case 2: Network Reaction as found in Table 5.2

Table 5.5 displays the rewards the leading party obtains in case 4. The data is compared to the performance in case 2 and the difference is given.

As we can see from the table there are some placement strategies that perform well. This means that after the decrease that occurs from the network reactions, they are able to limit the decrease in rewards that stems from the additional party participating. We note that fee weighted centrality is able to increase its reward compared to case 2, to lose only 50% of its reward using medium-sized transactions, and lose only 20% using large transaction sizes which if we compare this to the other placement strategies is only a small decrease in rewards. Only k-center with small transaction sizes and k-means with large transaction sizes are able to perform at this level. Furthermore, when large transaction sizes are used, the uniform random placement strategy performs rather well. It is able to double the rewards when compared to case 2.

When observing the average performances we notice that the leading party performs well compared to the additional party. In the case of medium-sized transactions, the placement strategies k-center and k-means allow for the additional party to obtain a higher reward than the leading party. Placement strategies fee weighted centrality and k-means both allow the leading party to obtain the largest reward on average. Placement strategies betweenness centrality and highest degree perform poorly. The uniform random placement strategy has the best performance, outperforming fee weighted centrality

	Small	Medium	Large	Average
Betweenness Centrality - Leading	0.0	0.0	4.35	1.45
Betweenness Centrality - Additional	0.0	0.0	5.58	1.86
Fee Weighted Centrality - Leading	3345.56	2605.35	5161.01	3703.97
Fee Weighted Centrality - Additional	1438.51	498.5	272.42	736.48
Highest Degree - Leading	0.8	0.0	5.42	2.07
Highest Degree - Additional	0.0	0.0	5.42	1.81
K-center - Leading	548.27	190.88	466.44	401.86
K-center - Additional	287.04	325.91	205.11	272.69
K-means - Leading	3741.16	46.21	9563.63	4450.33
K-means - Additional	1874.78	400.7	7902.44	3392.64
Uniform Random - Leading	6839.41	4389.28	26747.65	12658.78
Uniform Random - Additional	176.93	4761.76	37683.24	14207.31
Average	1506.29	1101.55	7335.23	

Table 5.6: Table depicting the rewards of a channel placement game.

and k-means by a factor of 3.

The data shows that the third party performs just as well as the leading party when the betweenness centrality placement strategy is used, the additional party performs 0.20 times the reward of the leading party when fee-weighted centrality is used, 0.87 the performance of the highest degree placement strategy, 0.68 times the performance using k-center, 0.76 times the performance using k-means, and 1.12 times the performance when uniform random is used.

From this, it is concluded, similar to cases 2 and 3 that the inclusion of network reactions as well as the presence of a third party has a negative effect on the rewards of the leading party. The addition to the network reaction, by introducing the third party reduces the rewards of the leading party by 60%-90% in many of the cases. This reduction in rewards is on top of the negative effect that the network reaction already has on rewards.

It must be noted that there exist some cases in which this reduction does not occur. Fee weighted centrality is able to increase its rewards compared to case 2 when using small transaction sizes and is able to limit the reduction to -20% in the case of large transaction sizes. K-center is able to increase its reward using small transaction sizes compared to case 2, and k-means is able to increase its rewards using large transactions.

In contrast to case 2 (5.3.2) and case 3 (5.3.3) the k-center and k-means placement strategy perform well using different transaction sizes. From the performance of the uniform random placement strategy, it is concluded that it performs well in a network with large transaction sizes.

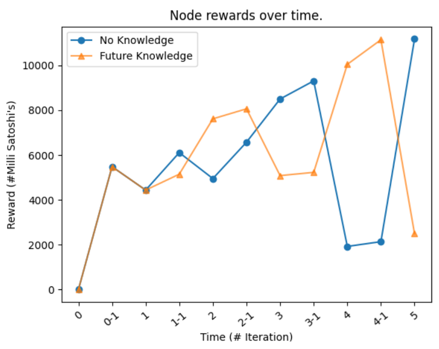
5.4. Game Theory

Based on the data in 5.3 it was concluded that the ability of parties to react severely impacts the reward the leading party is able to obtain. However, the question remains whether this negative effect can be mitigated by changing the placement strategy of the party. This subsection details the result that follows from an experiment where the leading party is given knowledge of the reaction that other parties will make to the leading parties channel placement. Subsection 5.4.1 details how rewards are obtained when the future actions that stem from network reactions are known, whereas section 5.4.2 details the obtained rewards when the future actions of an additional party are known.

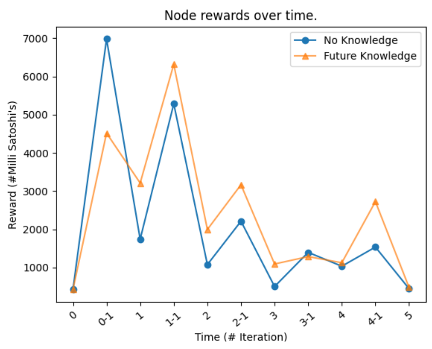
5.4.1. Network

This subsection details the experiments in which we analyze whether the negative impact found in 5.3.2 can be negated when the leading party has knowledge about future reactions that will take place in reaction to their moves.

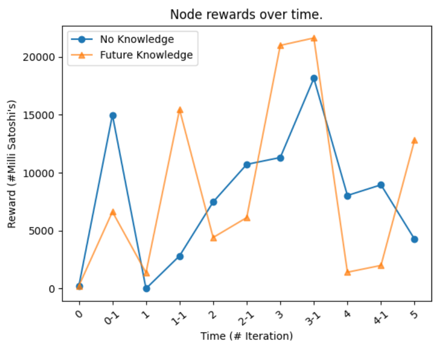
An interesting pattern that can be observed in the graphs 5.12 is that the baseline and the game theory reward are closely competing for the spot of the highest reward. They seem to overtake each other for the top spot every other step in the graph. But we can see a pattern in the data. When we observe steps 0-1, 1-1, 2-1, ... in the graph, we notice that the game theory reward is lower than the reward of the



(a) Reward graph with transaction amount: 100



(b) Reward graph with transaction amount: 10000



(c) Reward graph with transaction amount: 1000000

Figure 5.12: Figures depicting reward trends when playing a channel placement game with knowledge on 1 iteration of future network reactions.

baseline. The game theory data line overtakes the baseline on the steps that follow the n trend. This has to do with the difference between the two algorithms. The baseline uses the fee weighted centrality which optimizes their reward for the n-1 tick in the graph. In contrast, the game theory algorithm creates a connection that will see its reward increased once the network is allowed to react at step n. This pattern is clearly seen in 5.12a and 5.12c, but not so much in 5.12b. Here after showing the pattern briefly the game theory algorithm takes on a higher reward even in the n-1 steps before eventually being slightly overtaken by the baseline according to the pattern.

However in graphs 5.12a and 5.12c we observe unexpected behavior in steps 4 to 5. The expectation was that knowledge of the reaction would always lead to an advantage. The graphs indicate that a party could temporarily be worse off. The steps in 5.12a indicate that while the pattern seems to hold, the baseline is not able to obtain high rewards at step 4 when the network reacts. During this step, the game theory algorithm performs well, and can even increase its reward during steps 4-1. However, step 5 shows the "reverse pattern". When the network reacts the greedy algorithm is able to obtain high rewards whereas the game theory algorithm can not. This "reverse pattern" is something that can be seen in 5.12a already in steps 2-1 to 4. Therefore it is likely that in steps 5-1 to 6 the game theory rewards will again overtake the reward of the baseline. In 5.12b we observe a similar pattern as we find in 5.12a. Both in steps 1-1 to 3 and 3-1 to 5 we observe the game theory rewards performing worse than the greedy algorithm, after which the game theory algorithm overtakes the greedy algorithm.

	Small	**Medium**	**Large**	**Average**
Leading - Knowledge	2492.0	482.5	12813.0	5262.5
Leading - No Knowledge	11177.66	449.0	4265.0	5297.22
Change	22.29%	107.46%	300.42%	

Table 5.7: Table depicting the rewards of figure 5.12. Performance is relative to the performance of the additional party.

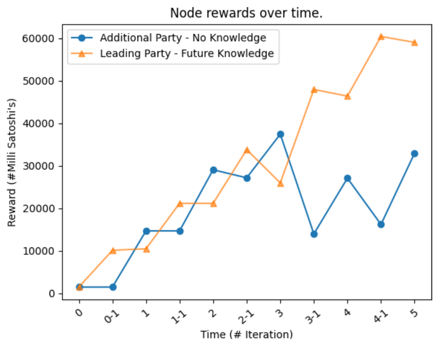
Table 5.7 shows that on average there is no difference between the rewards of the party with future knowledge and the party without future knowledge, however, this average is influenced by the network that uses small transactions. The pattern present in figure 5.12 indicates that the two lines will change places as the iterations continue. If we observe the medium and large-sized transactions we observe that the effect of future knowledge is impactful. While the rewards are low in the medium-sized transaction network a 7.5 % increase is quite the feat. The large transaction network allows future knowledge to be leveraged for a 300% reward.

While performing better than the greedy algorithm in its ability to minimize the reward reduction it does not outperform it in all situations. In the case of small and large transactions, we observe the reward lines oscillating. From this case, it is difficult to conclude which of the two lines performs better, and whether the additional information is useful. When one leverages future knowledge, one opens themselves up to being undercut. With the roughly 50 nodes attempting to undercut your channel, the rewards will be lost. However, from figure 5.12b, it can be concluded that the knowledge can be successfully leveraged to obtain an advantage.

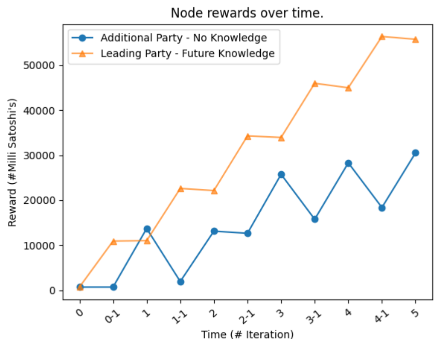
In general, the game theory algorithm performances show that they can minimize the negative impacts on rewards network updates have. However, it must be noted that this is not sufficient to fully negate the negative impact. The impact of the roughly 50 nodes is simply too much for the knowledge to fully negate.

5.4.2. Party

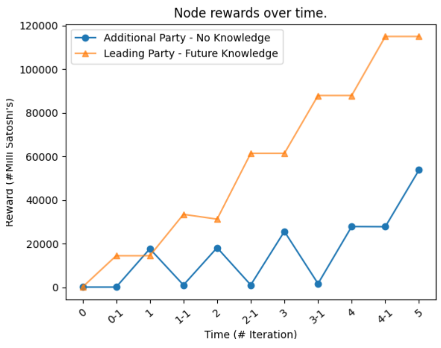
This subsection details the experiments in which we analyze whether the negative impact found in 5.3.3 can be negated when the leading party has knowledge about future reactions that will take place in reaction to their moves.



(a) Reward graph with transaction amount: 100



(b) Reward graph with transaction amount: 10000



(c) Reward graph with transaction amount: 1000000

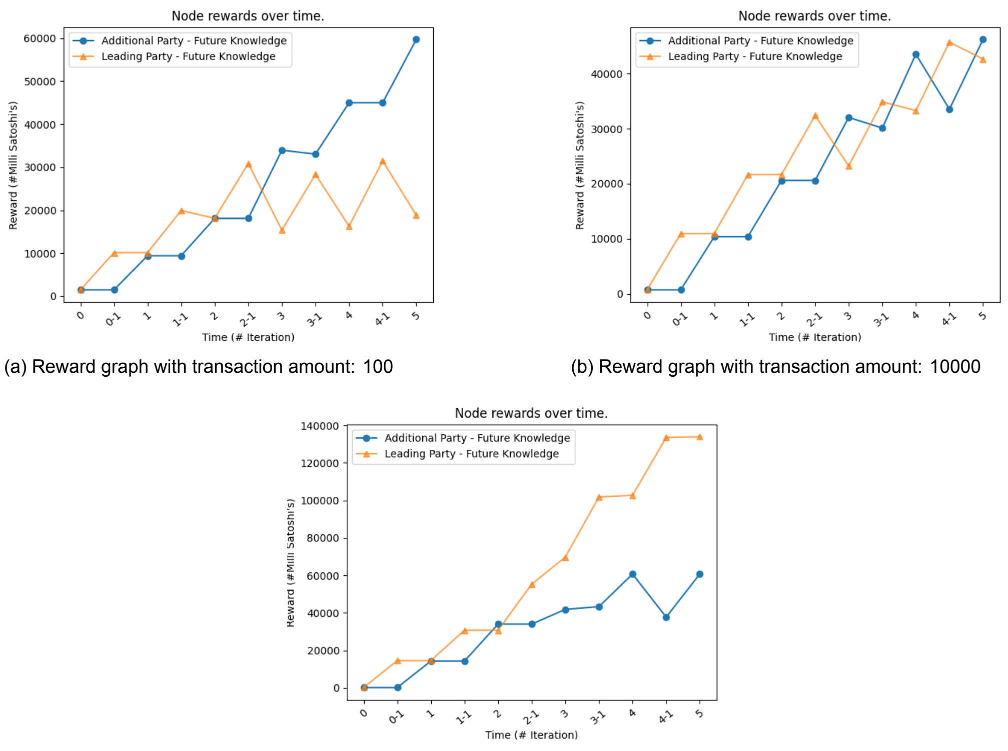
Figure 5.13: Figures depicting reward trends when playing a channel placement game with knowledge on 1 iteration of future reactions of an additional party.

Figure 5.13 shows the rewards of the leading party when they have knowledge of 1 reactionary move that the other party will make. From it, we can see that the leading party is able to leverage their future knowledge to obtain an advantage, represented by their rewards increasing. In all three of the figures, we can observe this trend. In figure 5.13a we observe that the reward lines are rather similar in score for the first 3 iterations of the placement game. However, after this, the leading party is able to further increase their rewards whereas the rewards decrease if the leading party would not have had access to future knowledge. In figure 5.13b the difference between the two reward lines occurs even earlier in the experiment. While at iteration 1 the leading party is slightly worse off, in all the other experiments the leading party is better off, indicating that they are able to leverage the knowledge to increase their reward. In figure 5.13c the pattern is also visible, similar to figure 5.13b it is only at iteration 1 that the leading is temporarily worse off. Again the leading party is able to perform better in all other iterations than if it would not have access to future knowledge. This figure is also the setting in which the leading party is able to obtain the best reward for themselves.

	Small	**Medium**	**Large**	**Average**
Leading - Knowledge	59021.0	55707.5	115045.5	76591.17
Additional - No Knowledge	32951.0	30515.5	53812.5	39092.83
Performance	179.12%	182.55%	213.79%	

Table 5.8: Table depicting the rewards of figure 5.13. Performance is relative to the performance of the additional party.

If we take a look at the rewards at the end of the experiment, we observe the following: We notice that the rewards of the leading party severely exceed what they would have received otherwise. It is able to increase its rewards by 80%, 83%, and 114% respectively, which is no small feat. The average of the leading party is close to doubled which further indicates that the leading party is able to properly leverage its future knowledge to obtain reward.



(a) Reward graph with transaction amount: 100

(b) Reward graph with transaction amount: 10000

(c) Reward graph with transaction amount: 1000000

Figure 5.14: Figures depicting reward trends when playing a channel placement game where both the leading and additional party have knowledge on 1 iteration of future reactions each other.

In figure 5.13 we analyzed what would occur when the leading party would have access to future knowledge during their decision-making. The rewards indicate that the leading party is able to leverage this knowledge to increase their reward. Figure 5.14c displays what would occur if the additional party would also have access to the future knowledge during their decision-making process like the leading party has. Figure 5.14a shows that the additional party is able to reverse the outcome. Whereas in figure 5.13a the leading party is able to obtain a higher reward from iteration 3 onwards, in figure 5.14a it is the additional party that is able to obtain the highest rewards from iteration 3 onwards. This indicates that the advantage disappears when the additional party also has access to future knowledge. Figure 5.14b shows that when the two parties both have access to future knowledge it is also a possibility for them to tie in rewards. In this case, the rewards of both the leading party and the additional party are around 40 thousand, whereas in figure 5.13b the leading party would have a reward of around 50 thousand, and the additional party a reward around the 30 thousand. Both the leading and the additional party attempt to leverage future knowledge, lowering the rewards of their competition in the process, but their competition has the same effect on them. Figure 5.14c shows that it is still possible for the leading party the be ahead of the additional party in terms of rewards. While the leading party is still competing with the additional party by iteration 2 it is able to increase its reward more than the additional party is able to reduce them winning the competition. While it takes an iteration longer compared to figure 5.13c, the outcome is still preferable for the leading party, as they have obtained an advantage.

	Small	**Medium**	**Large**	**Average**
Leading - Knowledge	18845.5	42620.0	133949.5	65138.33
Additional - Knowledge	59723.5	46182.0	60766.5	55557.33
Performance	31.6%	92.29%	220.43%	

Table 5.9: Table depicting the rewards of figure 5.14. Performance is relative to the performance of the additional party.

If we take a look at the rewards at the end of the experiment, we observe the following: We notice that while the average reward of the leading party is higher than that of the additional party, the difference is not nearly as large as in table 5.8. Where the increase nearly doubled the rewards, now the leading party has a 20% higher reward. This is also reflected in the performance where the changes in rewards are -68.4%, -7.71%, and 120.43% which differs greatly from the performance in table 5.8.

The results found in figure 5.12 and table 5.8 conclude that knowledge about future reactions during the decision-making process has a positive influence on rewards, and is able to limit the negative impact that an additional party may have on rewards. We conclude that knowledge of future moves is able to be leveraged against the additional party to increase the rewards of the leading party. Given future knowledge the leading payer is able to outperform the results of section 5.3.3 lowering the impact that the additional party has, however, this is not enough to fully mitigate the effect. Where the leading party was able to obtain rewards of 150 thousand without an additional party, with this mitigation a leading party is able to obtain half of those rewards. In our experiments, the presence of the additional party ensures competition which will negatively impact rewards.

We conclude that when the additional party is given access to knowledge about the future the effect diminishes. While the effect seems to be influenced by transaction size, the ability of the leading party to obtain a reward is made more difficult. This is to be expected because by giving the additional party future knowledge, we have leveled the playing field, essentially removing the advantage that the leading party had.

6 Conclusion, Limitation & Future Work

Section 6.1 discusses the results of Chapter 5. Furthermore, it details the conclusions drawn in this work and answers the research questions. In section 6.2 we highlight the limitations whereas section 6.3 details the future work.

6.1. Conclusion

We have observed differences in the performance between different placement strategies. Our results show that the placement strategies highest degree, and betweenness centrality do not perform well. Not only do they both perform poorly in isolation, the situations where other actors are allowed to act reduce their performance to an even lower level.

It was already known that these placement strategies perform poorly. In the case of highest degree, this placement strategy has been shown to offer low rewards due to low fee costs. This is likely caused by short average paths and high efficiency [LRT21] compare [SZ20; Ava+20]. Furthermore, we have found that many of the parties which use the highest degree placement strategy are already connected to each other, which makes it difficult for the highest degree strategy to perform well. As for the poor performance of the betweenness centrality placement strategy, the fee weighted centrality strategy was specifically created to optimize its poor performance [ERE20]. However, it was unknown what effect the extension of the action space would have on the rewards. We have shown that this extension may have a positive effect on the rewards. The highest degree and betweenness centrality placement strategies show improved rewards when routing large transactions, but these improvements do not allow them to compete on the same level as the other placement strategies analyzed in this work.

The fee weighted centrality, k-means, k-center, and uniform random placement strategies have been shown to perform well in isolation. However allowing the existing channels to change their fee, or adding an additional party to the network has a negative effect on rewards. While k-means, k-center, and the uniform random placement strategies are able to limit the negative impact of one additional party, they are unable to mitigate the negative impact when the entire network is allowed to react. From this, it follows that when we combine the action spaces, the combined negative effects would also not allow for any of the placement strategies to perform well.

Competition lowers rewards, and therefore a placement strategy that obtains high yields will experience the largest negative effects from being undercut. This can be seen in the case fee weighted centrality. That k-means, k-center, and the uniform random placement strategy are able to reduce the negative impacts can be explained when we observe the level of competition that they encounter. When one other party is present in the channel creation game, the two parties are not required to be in direct competition. In these placement strategies, it is often the case that the action of one party removes an option of the other. This explains why the effect of competition is not seen as clearly in the rewards. However, in the case of fee weighted centrality, the two parties are nearly always forced to compete with each other, which enforces undercutting to obtain rewards, lowering the rewards of one party.

As to **how the rewards are impacted when parties are allowed to change their channel fee**, we concluded that this addition to the action space negatively impacts the reward of a party. Our results show that a party is bound to lose 85%-100% on average of their rewards to the competition which is a substantial size. As to **how the rewards are impacted when there is an additional party present in the network**, we concluded that the reward of the leading party is negatively impacted by adding an additional player to the action space. Similar to the network reaction, the rewards are substantially decreased by the competition that takes place. However, since there exist placement strategies that limit the impact of competition the average loss is 30%. When the action spaces are combined, the negative impact increases. The negative impact of the additional party has been shown to stack with the negative impact of the network reactions, which results in a loss of 95%-100% of the reward a strategy might be able to obtain when performing in isolation. Therefore the **rewards of an optimal channel creation choice are negatively impacted by 95%-100%**

That competition lowers rewards was to be expected, however, the levels at which the rewards are impacted were not. A relatively small network of 200 participants is enough to outcompete a party and reduce their rewards by 95%-100%. We, therefore, conclude that a network where every participant competes for rewards is detrimental to the reward of the leading party in a network creation game.

Since the impact of competition is so large we aimed to find a strategy that mitigates this effect. We have created two new placement strategies that allow a leading party to use future knowledge during their decision-making process to their advantage. The first strategy that aimed to reduce the impact of the network updates was able to increase the rewards by -78%, 7%, and 200% depending on transaction size compared to when no access to future knowledge was given. The second strategy that aimed to reduce the impact of the additional party changed the rewards by 80%, 83%, and 114% depending on transaction size. While the placement strategies are able to leverage future knowledge, the strategies are not sufficient to sufficiently negate the negative impact that arises from competition.

From the results of one of the experiments, we must conclude that the strategy is not able to successfully leverage its future knowledge, but this is not technically the truth. In the graph, it can be seen that the reward lines oscillate and at the end of the experiment it so happened to be the case that the reward line that does not have access to future knowledge is performing well. However, from the data, we can conclude that future knowledge is able to be leveraged. Because, from the data, we can infer that the line will very likely be overtaken if another iteration of the game would have been played. This pattern is likely due to competition, when a party has positioned themselves where they may benefit, the reaction by the network is to undercut. This lowers the reward of the leading party.

With competition having a large negative impact on results, there is one main implication it will have on anyone that wants to participate in Bitcoins layer 2 network. They will have to accept that competition is to follow which will negatively impact their rewards. It is left up to them to decide if the reward is worth the investment. For the research field, there are multiple things that may still be researched to aid in this decision. Firstly it can be determined how many parties would be willing to undercut to compete for rewards. Secondly, different methods of negating the negative impact that competition has on rewards could be explored, Thirdly, placement strategies that are less impacted by competition could be explored.

6.2. Limitations

The first limitation is time. Had there been more time, it would have been possible for us to run the experiment on up-to-date snapshots of the lightning network. This would have provided us with results that are more representative of the current state of the network, allowing us to better measure the impact of our work. Although we have taken steps to make our analysis similar to that of the lightning network by sampling weights from it, we would like to see how our model performs on an up-to-date snapshot. The analysis of network reactions is expensive on its own. However, paired with a large network, it is even more expensive to compute.

The second limitation is the effect of our code on the snapshot. During this work, we have made a comparison to previous work. However, the code that removes rewards from the reward pool would need to be changed so as to not crash when performing this action on the snapshot. While this allowed us to obtain an indication of how previous work would perform given our assumption and codebase it was not able to make a perfect comparison.

The third limitation is that of fee sampling. The fee values we sampled to construct our graph were chosen using randomness. In order for this outcome to be fully analyzed we should have recreated it multiple times and compared the impact of the initial sampling. However, the distribution of fee values is heavily skewed towards the default fee of 1000 satoshi's, we therefore feel confident that sampling without replacement would yield similar results.

The fourth limitation is that of the utility function. For the context of this work, we have assumed a utility of x=y. However, it is not a given that this utility function is representative of all the parties operating in the lightning network. Just as there exist parties that are willing to charge low fees for the network to function well, there might be parties whose utility function would be logarithmic. The reverse is also true. The parties that aim to optimize their reward might have a different utility function than the one used in this thesis.

The fifth limitation stems from assumptions made in our reward distribution. In our work rewards are distributed to every party sending a message to every other party within the network. However, in practice, this is not the case. Parties will vary in transaction frequency and destination, which impacts the rewards. Our reward metric does not take into account when and to whom parties send transactions.

6.3. Future Work

Future work can roughly be grouped into three categories: Extensions of the work described in this thesis, further changes in action space that can be analyzed, and further changes in game-theoretical aspects.

There exist extensions to this work which we have left up to future work. The first extension would be to test our impact on a lightning snapshot directly. Due to our time constraints, we were not able to run our experiments on the lightning snapshot. It would be useful to see the results of all of our experiments on a lightning network snapshot. This would also include the Stackleberg games. We estimate that the time it would take to run our current games on a lightning snapshot, to be roughly a year, or 2 months per placement strategy. This is expected to be even longer for the Stackleberg games, of which we do not have a clear estimation. We were not able to run these Stackleberg games on our 200-node graph and instead had to run this on our 50-node graph.

The second extension would be to combine our Stackleberg games. We have observed the impact that the two experiments have in isolation, but we have yet to observe the effects of combining them. By studying the combined effects of the Stackleberg games we can determine if their combination would be able to mitigate the negative effects of competition.

Next are the changes we can make to the action space described in 3.2.1. The first change would be to alter the network reaction. Instead of every party in the network being required to update the fee values on their channels, it would be interesting to see what would happen if only part of the network updated their fee. By only forcing part of the network to update it would make the model more realistic, as fee changes are not common. Whereas our work has shown the upper bound on the negative impact that network interactions could have, this would provide a more realistic impact. However, determining which party gets to update their channel fees would be a problem. Logic could be used to determine for each individual party whether they would react, but picking parties at random, or from a predetermined list would also suffice.

The second change would be to increase the number of additional parties within the network. This work has limited the number of additional parties to one, but this is not required. Increasing the number of additional parties would provide a more complex case in which decisions carry more weight. The 2 or more additional parties would act in parallel to each other after the leading party has made their move. They are likely to come up with the same channel destination and fee, thus sharing rewards. This might end up producing even worse rewards than if they would have picked another party to create a channel with. The implications of game theory would increase the usefulness of this case. What would happen if the additional parties knew they were acting in parallel and thus that they would be sharing results? Should they preemptively undercut the other additional parties, or should they seek out another channel?

The third change to action space we would like to see is the ability to close channels. In our experiments, there exists the possibility for a channel to get undercut. Whilst we observed what happens when parties are allowed to change the fees to remain profitable, another option would be to close the channel and create one with another party. Would this new channel undercut an existing channel for optimal results? Would this provide incentives to placement strategies that provide unique shorter paths, rather than positioning themselves to be the most centrally located node within the network?

The first game-theoretical aspect of this work we would like to see changed would be to break one of the assumptions we have made. We would like to see what happens when we model the effects of accepting or denying channel creation requests. In a realistic setting for a channel to provide a reward, it would have to be well-funded. The party that received the channel creation request would have to decide if the funds the channel would bring in are worth the cost to create it.

The second change we would like to analyze is the effect of strategically refraining from placing a channel. Our results indicate that the addition of network updates can be detrimental to the rewards of a party. But our work does not cover when to refrain from creating more channels once a hypothetical ”sweet spot” has been reached. A party could decide that creating further channels would leave them open to being undercut, and decide to not create any more channels.

The third of the game-theoretical changes we would make is to change the utility function. In our work, we have defined it as linear (x=y). However, realistically there should be a limit to the amount of greed a party has. Studying the different utility functions has the possibility to impact the behavior of parties, potentially resulting in different actions being selected.

Bibliography

[AB02] Réka Albert and Albert-László Barabási. “Statistical mechanics of complex networks”. In: *Reviews of modern physics* 74.1 (2002), p. 47.

[Ant17] Andreas M Antonopoulos. *Mastering Bitcoin: Programming the open blockchain*. ” O’Reilly Media, Inc.”, 2017.

[Ava+20] Zeta Avarikioti et al. “Ride the lightning: The game theory of payment channels”. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2020, pp. 264–283.

[BA99] Albert-László Barabási and Réka Albert. “Emergence of scaling in random networks”. In: *science* 286.5439 (1999), pp. 509–512.

[BDW18] Conrad Burchert, Christian Decker, and Roger Wattenhofer. “Scalable funding of bitcoin micropayment channel networks”. In: *Royal Society open science* 5.8 (2018), p. 180089.

[BG08] Jørgen Bang-Jensen and Gregory Z Gutin. *Digraphs: theory, algorithms and applications*. Springer Science & Business Media, 2008.

[BW10] Edward A Bender and S Gill Williamson. *Lists, decisions and graphs*. S. Gill Williamson, 2010.

[BXW22] Qianlan Bai, Yuedong Xu, and Xin Wang. “Understanding the Benefit of Being Patient in Payment Channel Networks”. In: *IEEE Transactions on Network Science and Engineering* 9.3 (2022), pp. 1895–1908.

[Cro+16] Kyle Croman et al. “On scaling decentralized blockchains”. In: *International conference on financial cryptography and data security*. Springer. 2016, pp. 106–125.

[Dij22] Edsger W Dijkstra. “A note on two problems in connexion with graphs”. In: *Edsger Wybe Dijkstra: His Life, Work, and Legacy*. 2022, pp. 287–290.

[DW15] Christian Decker and Roger Wattenhofer. “A fast and scalable payment network with bitcoin duplex micropayment channels”. In: *Symposium on Self-Stabilizing Systems*. Springer. 2015, pp. 3–18.

[ERE20] Oğuzhan Ersoy, Stefanie Roos, and Zekeriya Erkin. “How to profit from payments channels”. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2020, pp. 284–303.

[Fre77] Linton C Freeman. “A set of measures of centrality based on betweenness”. In: *Sociometry* (1977), pp. 35–41.

[Gud+19] Lewis Gudgeon et al. “SoK: Off The Chain Transactions.” In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 360.

[KG17] Rami Khalil and Arthur Gervais. “Revive: Rebalancing off-blockchain payment networks”. In: *Proceedings of the 2017 acm sigsac conference on computer and communications security*. 2017, pp. 439–453.

[KR21] Satwik Prabhu Kumble and Stefanie Roos. “Comparative Analysis of Lightning’s Routing Clients”. In: *2021 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*. IEEE. 2021, pp. 79–84.

[Li+18] Chenxing Li et al. “Scaling nakamoto consensus to thousands of transactions per second”. In: *arXiv preprint arXiv:1805.03870* (2018).

[LMZ20] Peng Li, Toshiaki Miyazaki, and Wanlei Zhou. “Secure balance planning of off-blockchain payment channel networks”. In: *IEEE INFOCOM 2020-IEEE conference on computer communications*. IEEE. 2020, pp. 1728–1737.

[LRT21] Kimberly Lange, Elias Rohrer, and Florian Tschorsch. “On the Impact of Attachment Strategies for Payment Channel Networks”. In: *arXiv preprint arXiv:2102.09256* (2021).

[McC+16] Patrick McCorry et al. “Towards bitcoin payment networks”. In: *Australasian Conference on Information Security and Privacy*. Springer. 2016, pp. 57–76.

[MDP18] Daniela Mechkaroska, Vesna Dimitrova, and Aleksandra Popovska-Mitrovikj. “Analysis of the possibilities for improvement of blockchain technology”. In: *2018 26th Telecommunications Forum (TELFOR)*. IEEE. 2018, pp. 1–4.

[MS+16] Akio Matsumoto, Ferenc Szidarovszky, et al. *Game theory and its applications*. Springer, 2016.

[MZ21] Ayelet Mizrahi and Aviv Zohar. “Congestion attacks in payment channel networks”. In: *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part II*. Springer. 2021, pp. 170–188.

[Nak08] Satoshi Nakamoto. “Bitcoin: A peer-to-peer electronic cash system”. In: *Decentralized Business Review* (2008), p. 21260.

[Net20] Raiden Network. *Fast, cheap, Scalabletoken transfers for Ethereum*. 2020. URL: `https://raiden.network/`.

[PD16] Joseph Poon and Thaddeus Dryja. *The bitcoin lightning network: Scalable off-chain instant payments*. 2016.

[PN20] Rene Pickhardt and Mariusz Nowostawski. “Imbalance measure and proactive channel rebalancing algorithm for the lightning network”. In: *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE. 2020, pp. 1–5.

[Ren18] Willem Rens. “DoS on a Bitcoin Lightning Network channel.” In: (2018).

[RKJ14] Sunil Kumar Raghavan Unnithan, Balakrishnan Kannan, and Madambi Jathavedan. “Betweenness centrality in some classes of graphs”. In: *International Journal of Combinatorics* 2014 (2014).

[RT20] Elias Rohrer and Florian Tschorsch. “Counting down thunder: Timing attacks on privacy in payment channel networks”. In: *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*. 2020, pp. 214–227.

[Sto68] M Stone. *Mathematical Statistics: A Decision Theoretic Approach*. 1968.

[SZ20] Yotam Sali and Aviv Zohar. “Optimizing off-chain payment networks in cryptocurrencies”. In: *arXiv preprint arXiv:2007.09410* (2020).

[TSZ19] Saar Tochner, Stefan Schmid, and Aviv Zohar. “Hijacking routes in payment channel networks: A predictability tradeoff”. In: *arXiv preprint arXiv:1909.06890* (2019).

[Wan+22] Xiaojian Wang et al. “Why Riding the Lightning? Equilibrium Analysis for Payment Hub Pricing”. In: *ICC 2022-IEEE International Conference on Communications*. IEEE. 2022, pp. 5409–5414.

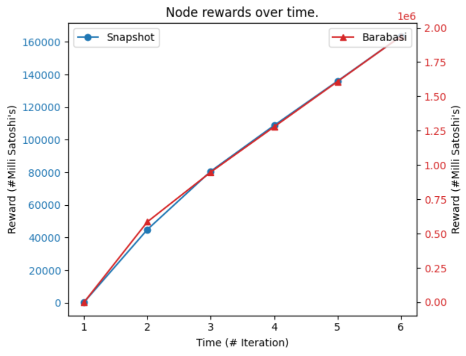
[Woo+14] Gavin Wood et al. “Ethereum: A secure decentralised generalised transaction ledger”. In: *Ethereum project yellow paper* 151.2014 (2014), pp. 1–32.

[Yen70] Jin Y Yen. “An algorithm for finding shortest routes from all source nodes to a given destination in general networks”. In: *Quarterly of Applied Mathematics* 27.4 (1970), pp. 526–530.

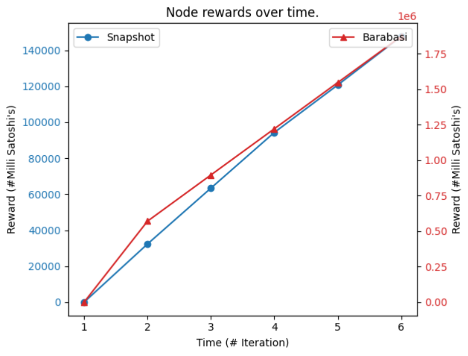
[Zab+22] Philipp Zabka et al. “Empirical evaluation of nodes and channels of the lightning network”. In: *Pervasive and Mobile Computing* 83 (2022), p. 101584.

A
Figures

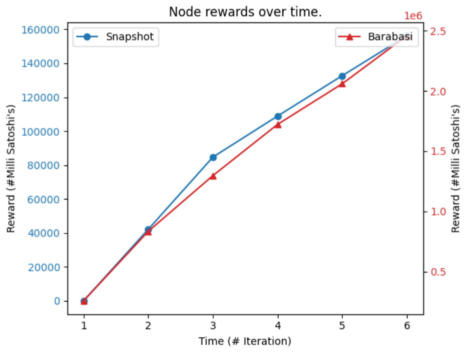
A.1. Replication figures



(a) Figure depicting the reward obtained in the snapshot, and the Barabasi-Albert graph using placement strategy: Fee weighted centrality and transaction amount: 100

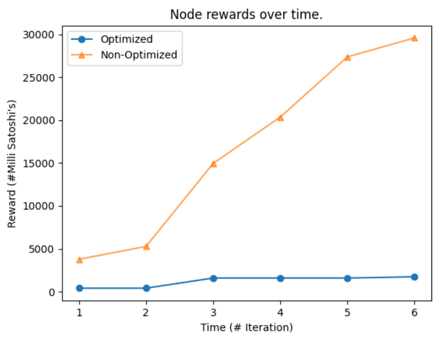


(b) Figure depicting the reward obtained in the snapshot, and the Barabasi-Albert graph using placement strategy: Fee weighted centrality and transaction amount: 10000

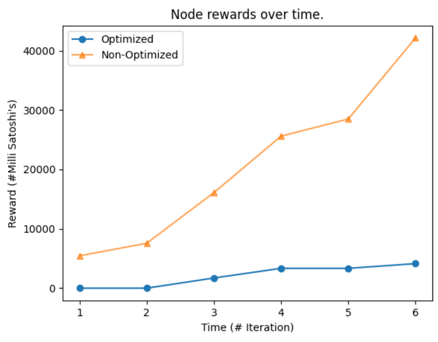


(c) Figure depicting the reward obtained in the snapshot, and the Barabasi-Albert graph using placement strategy: Fee weighted centrality and transaction amount: 1000000

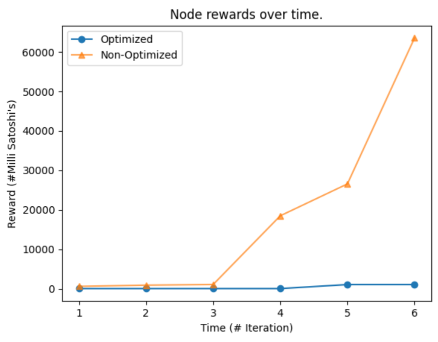
A.2. Fee Noise Analysis



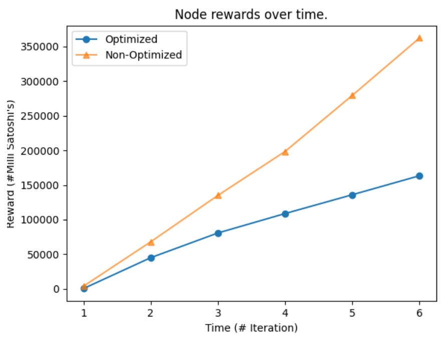
(a) Figure depicting the reward obtained when network optimization is not performed and is performed (baseline) using placement strategy: betweenness centrality and transaction amount: 100



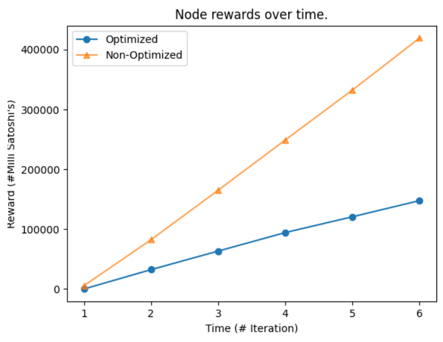
(b) Figure depicting the reward obtained when network optimization is not performed and is performed (baseline) using placement strategy: betweenness centrality and transaction amount: 10000



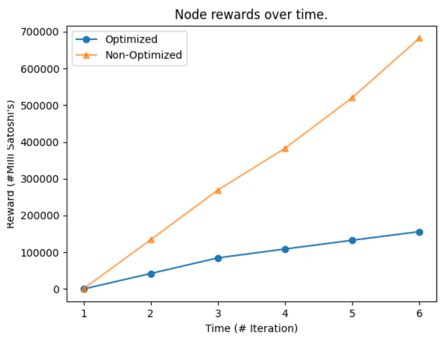
(c) Figure depicting the reward obtained when network optimization is not performed and is performed (baseline) using placement strategy: betweenness centrality and transaction amount: 1000000



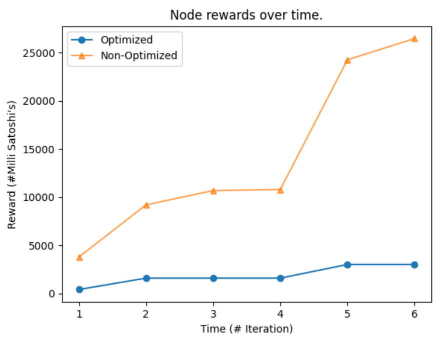
(d) Figure depicting the reward obtained when network optimization is not performed and is performed (baseline) using placement strategy: fee weighted centrality and transaction amount: 100



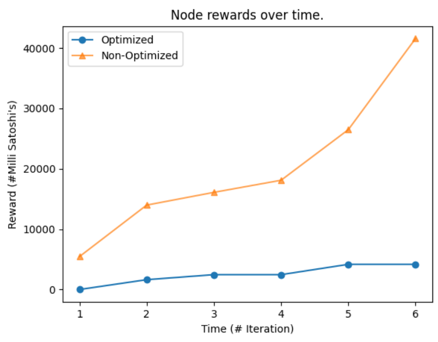
(e) Figure depicting the reward obtained when network optimization is not performed and is performed (baseline) using placement strategy: fee weighted centrality and transaction amount: 10000



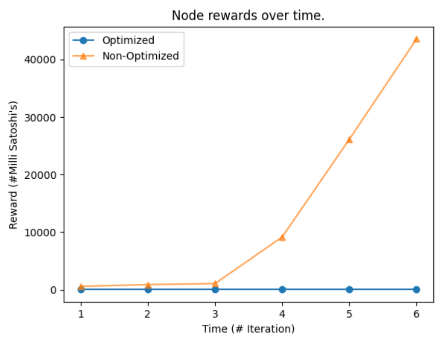
(f) Figure depicting the reward obtained when network optimization is not performed and is performed (baseline) using placement strategy: fee weighted centrality and transaction amount: 1000000



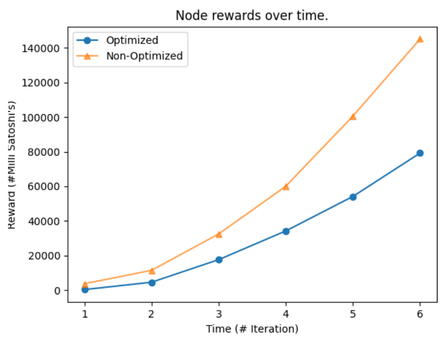
(g) Figure depicting the reward obtained when network optimization is not performed and is performed (baseline) using placement strategy: highest degree and transaction amount: 100



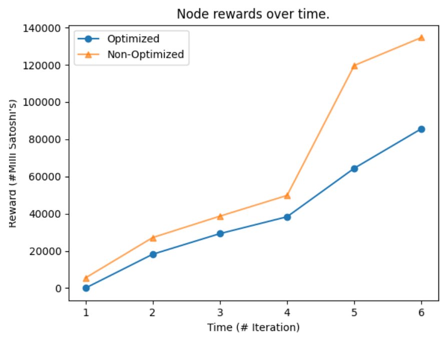
(h) Figure depicting the reward obtained when network optimization is not performed and is performed (baseline) using placement strategy: highest degree and transaction amount: 10000



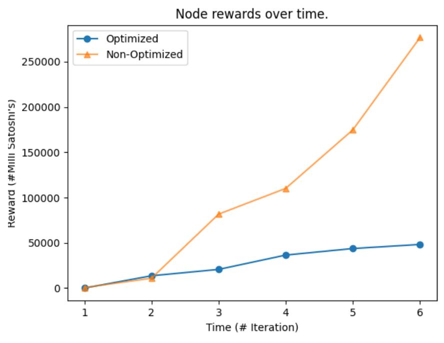
(a) Figure depicting the reward obtained when network optimization is not performed and is performed (baseline) using placement strategy: highest degree and transaction amount: 1000000



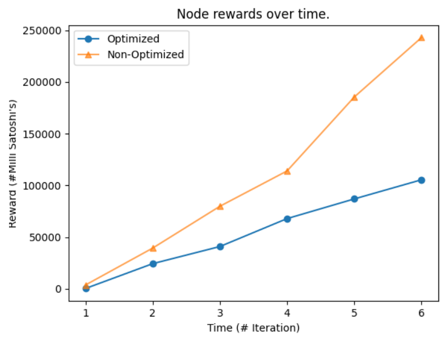
(b) Figure depicting the reward obtained when network optimization is not performed and is performed (baseline) using placement strategy: k center and transaction amount: 100



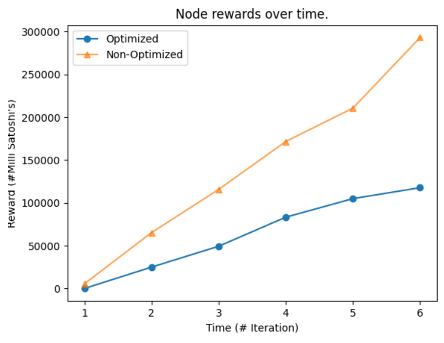
(c) Figure depicting the reward obtained when network optimization is not performed and is performed (baseline) using placement strategy: k center and transaction amount: 10000



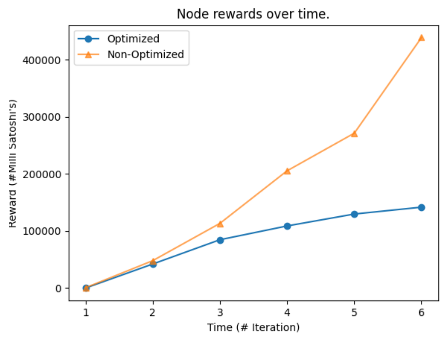
(d) Figure depicting the reward obtained when network optimization is not performed and is performed (baseline) using placement strategy: k center and transaction amount: 1000000



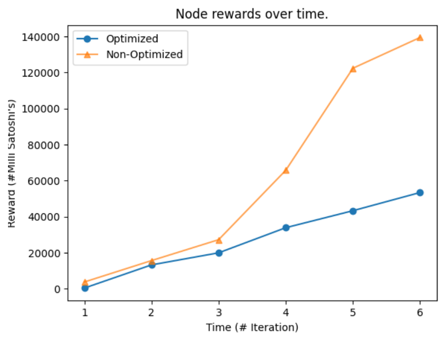
(e) Figure depicting the reward obtained when network optimization is not performed and is performed (baseline) using placement strategy: k means and transaction amount: 100



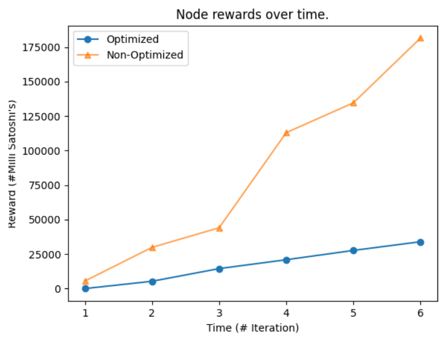
(f) Figure depicting the reward obtained when network optimization is not performed and is performed (baseline) using placement strategy: k means and transaction amount: 10000



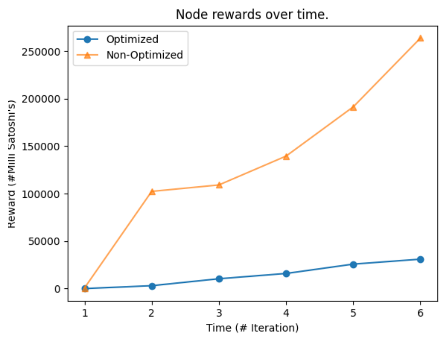
(g) Figure depicting the reward obtained when network optimization is not performed and is performed (baseline) using placement strategy: k means and transaction amount: 1000000



(h) Figure depicting the reward obtained when network optimization is not performed and is performed (baseline) using placement strategy: uniform random and transaction amount: 100

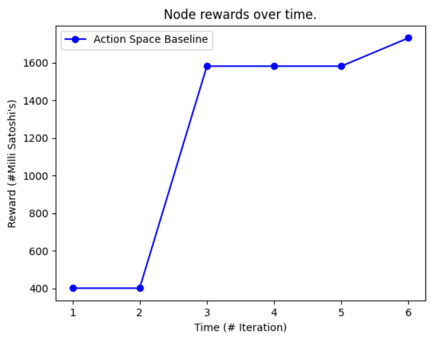


(a) Figure depicting the reward obtained when network optimization is not performed and is performed (baseline) using placement strategy: uniform random and transaction amount: 10000

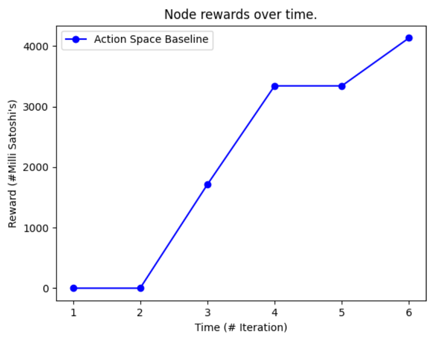


(b) Figure depicting the reward obtained when network optimization is not performed and is performed (baseline) using placement strategy: uniform random and transaction amount: 1000000

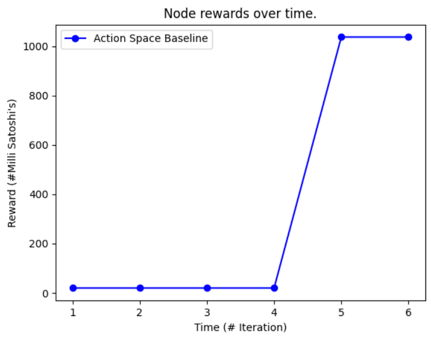
A.3. Case 1 figures



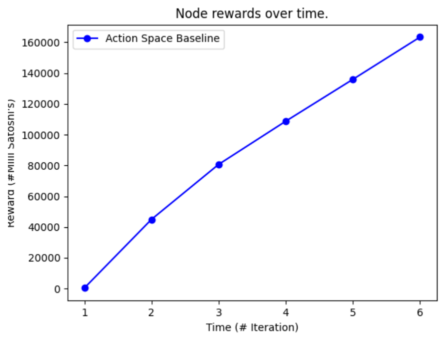
(a) Figure depicting the reward obtained when scenario 1 is run using placement strategy: betweenness centrality and transaction amount: 100



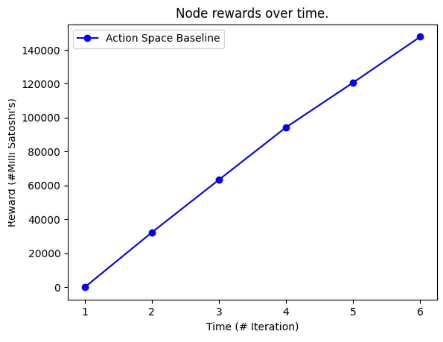
(b) Figure depicting the reward obtained when scenario 1 is run using placement strategy: betweenness centrality and transaction amount: 10000



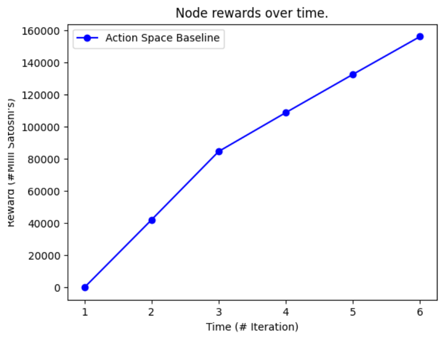
(c) Figure depicting the reward obtained when scenario 1 is run using placement strategy: betweenness centrality and transaction amount: 1000000



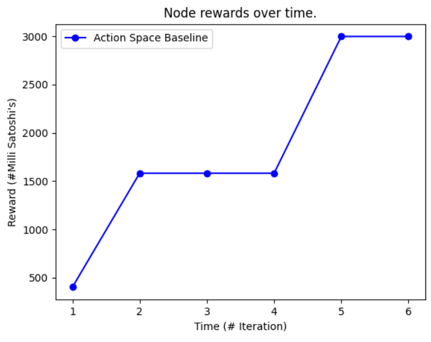
(d) Figure depicting the reward obtained when scenario 1 is run using placement strategy: fee weighted centrality and transaction amount: 100



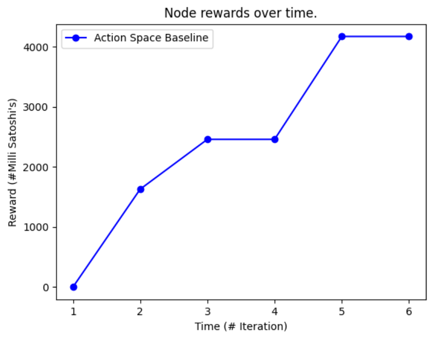
(e) Figure depicting the reward obtained when scenario 1 is run using placement strategy: fee weighted centrality and transaction amount: 10000



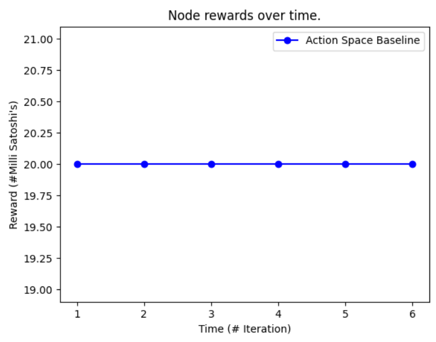
(f) Figure depicting the reward obtained when scenario 1 is run using placement strategy: fee weighted centrality and transaction amount: 1000000



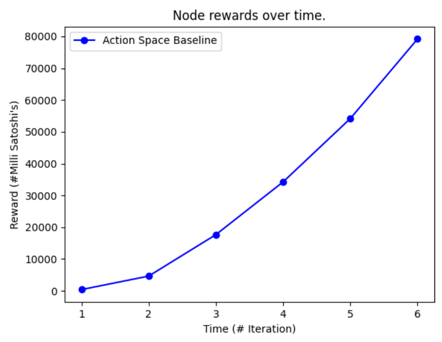
(g) Figure depicting the reward obtained when scenario 1 is run using placement strategy: highest degree and transaction amount: 100



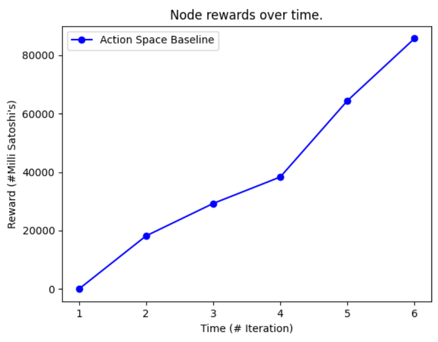
(h) Figure depicting the reward obtained when scenario 1 is run using placement strategy: highest degree and transaction amount: 10000



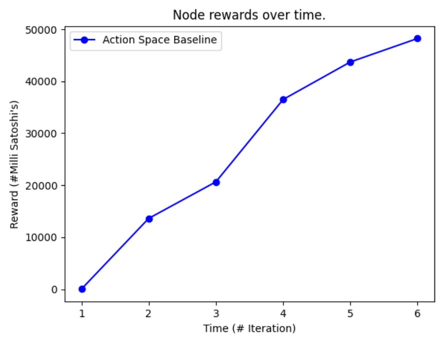
(a) Figure depicting the reward obtained when scenario 1 is run using placement strategy: highest degree and transaction amount: 1000000



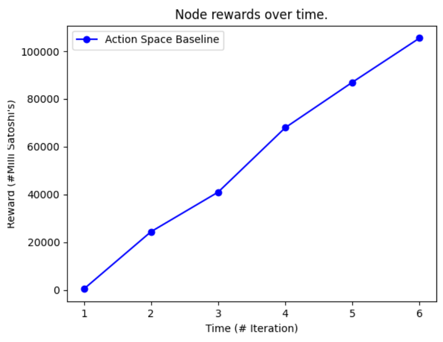
(b) Figure depicting the reward obtained when scenario 1 is run using placement strategy: k center and transaction amount: 100



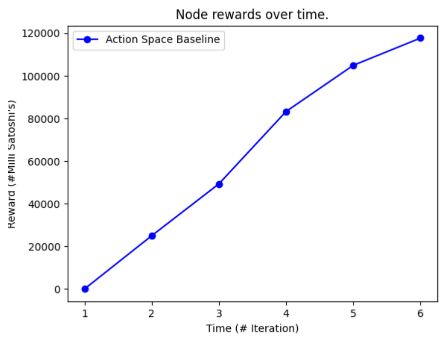
(c) Figure depicting the reward obtained when scenario 1 is run using placement strategy: k center and transaction amount: 10000



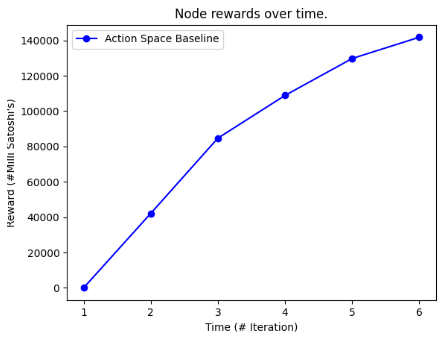
(d) Figure depicting the reward obtained when scenario 1 is run using placement strategy: k center and transaction amount: 1000000



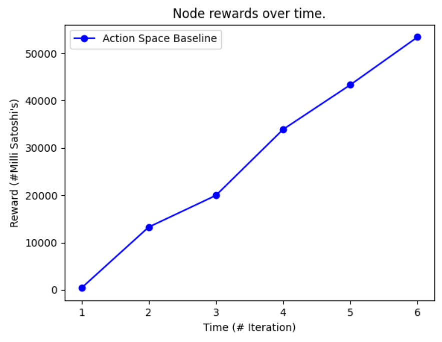
(e) Figure depicting the reward obtained when scenario 1 is run using placement strategy: k means and transaction amount: 100



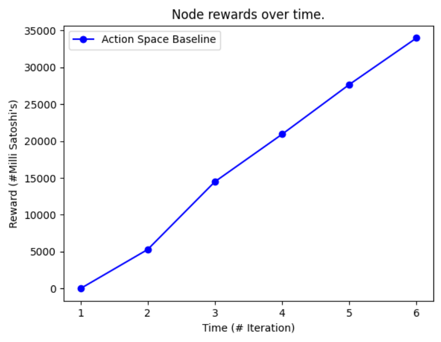
(f) Figure depicting the reward obtained when scenario 1 is run using placement strategy: k means and transaction amount: 10000



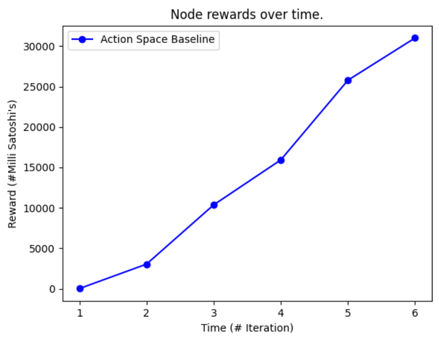
(g) Figure depicting the reward obtained when scenario 1 is run using placement strategy: k means and transaction amount: 1000000



(h) Figure depicting the reward obtained when scenario 1 is run using placement strategy: uniform randomness and transaction amount: 100

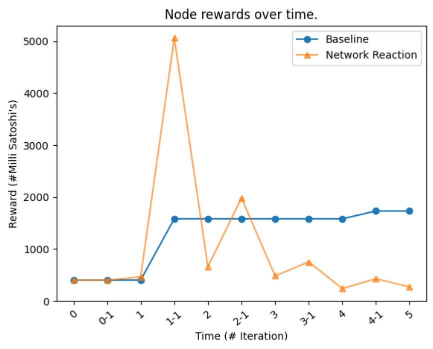


(a) Figure depicting the reward obtained when scenario 1 is run using placement strategy: uniform randomness and transaction amount: 10000

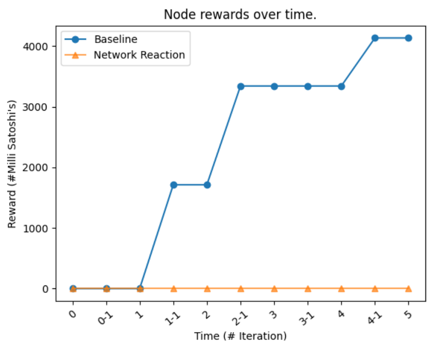


(b) Figure depicting the reward obtained when scenario 1 is run using placement strategy: uniform randomness and transaction amount: 1000000

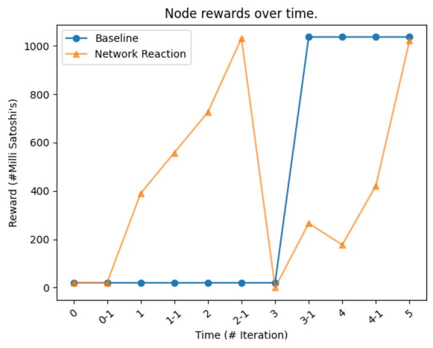
A.4. Case 2 figures



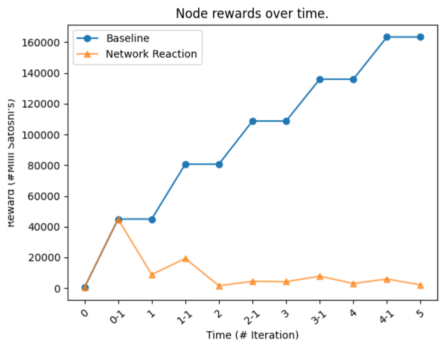
(a) Figure depicting the reward obtained when scenario 2 is run using placement strategy: betweenness centrality and transaction amount: 100



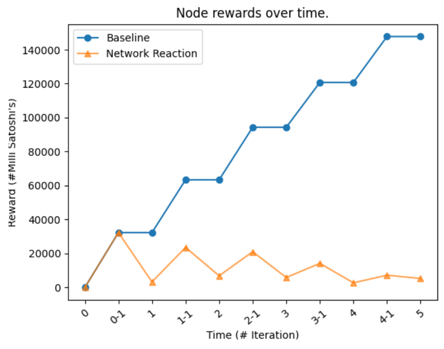
(b) Figure depicting the reward obtained when scenario 2 is run using placement strategy: betweenness centrality and transaction amount: 10000



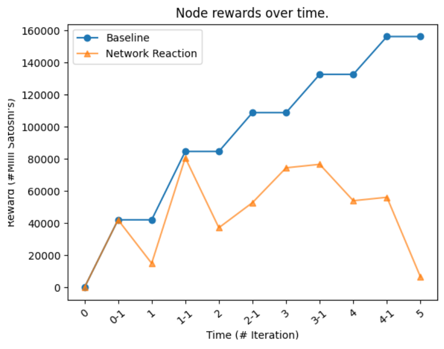
(c) Figure depicting the reward obtained when scenario 2 is run using placement strategy: betweenness centrality and transaction amount: 1000000



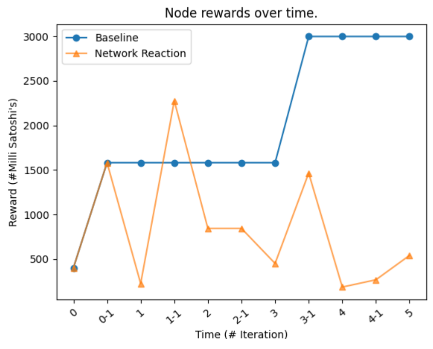
(d) Figure depicting the reward obtained when scenario 2 is run using placement strategy: fee weighted centrality and transaction amount: 100



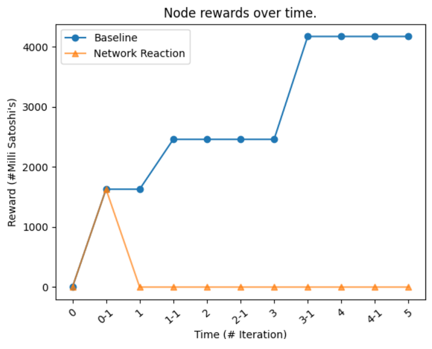
(e) Figure depicting the reward obtained when scenario 2 is run using placement strategy: fee weighted centrality and transaction amount: 10000



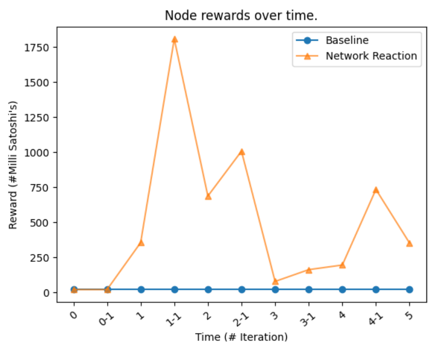
(f) Figure depicting the reward obtained when scenario 2 is run using placement strategy: fee weighted centrality and transaction amount: 1000000



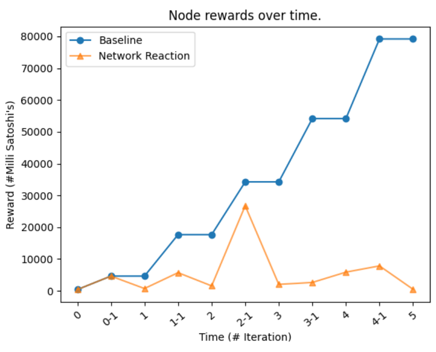
(g) Figure depicting the reward obtained when scenario 2 is run using placement strategy: highest degree and transaction amount: 100



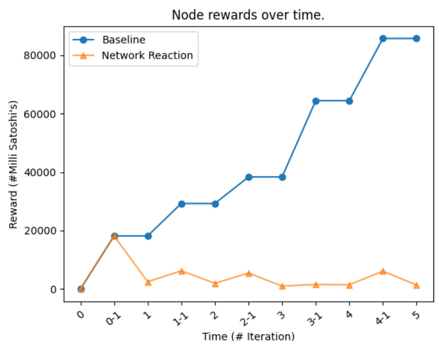
(h) Figure depicting the reward obtained when scenario 2 is run using placement strategy: highest degree and transaction amount: 10000



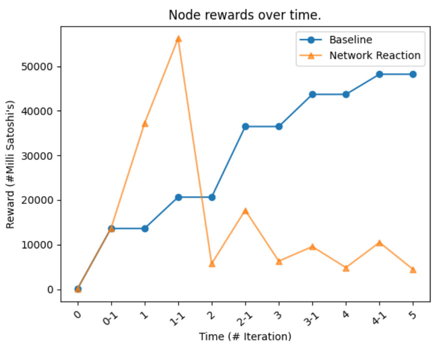
(a) Figure depicting the reward obtained when scenario 2 is run using placement strategy: highest degree and transaction amount: 1000000



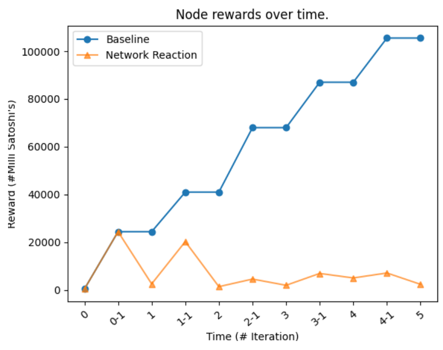
(b) Figure depicting the reward obtained when scenario 2 is run using placement strategy: k center and transaction amount: 100



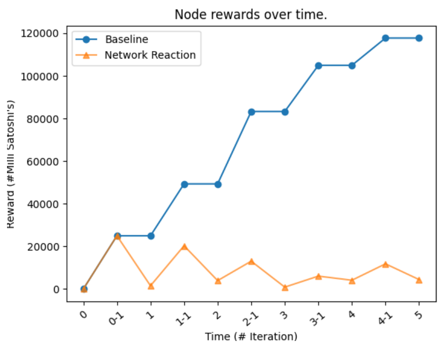
(c) Figure depicting the reward obtained when scenario 2 is run using placement strategy: k center and transaction amount: 10000



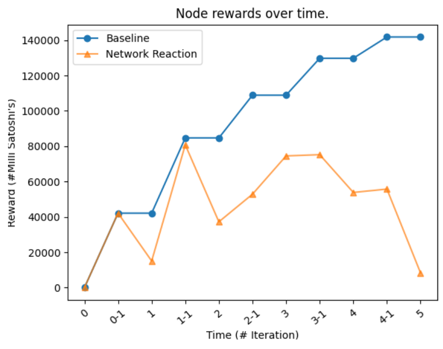
(d) Figure depicting the reward obtained when scenario 2 is run using placement strategy: k center and transaction amount: 1000000



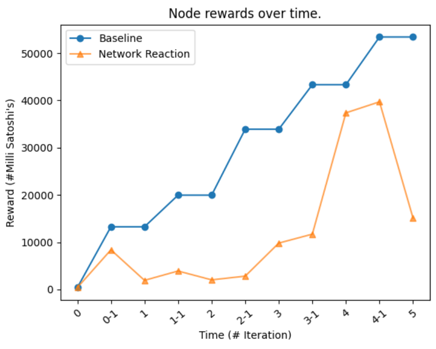
(e) Figure depicting the reward obtained when scenario 2 is run using placement strategy: k means and transaction amount: 100



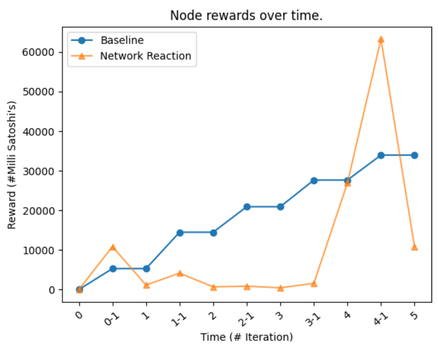
(f) Figure depicting the reward obtained when scenario 2 is run using placement strategy: k means and transaction amount: 10000



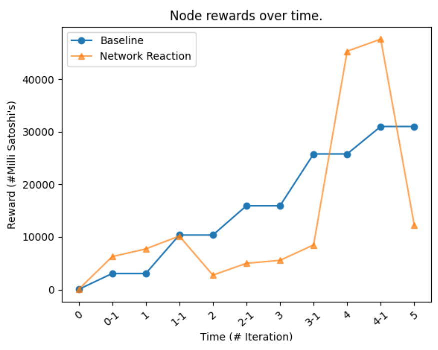
(g) Figure depicting the reward obtained when scenario 2 is run using placement strategy: k means and transaction amount: 1000000



(h) Figure depicting the reward obtained when scenario 2 is run using placement strategy: uniform randomness and transaction amount: 100

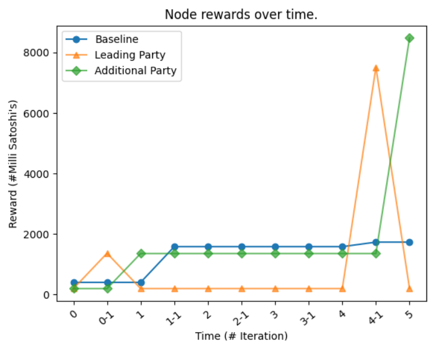


(a) Figure depicting the reward obtained when scenario 2 is run using placement strategy: uniform randomness and transaction amount: 10000

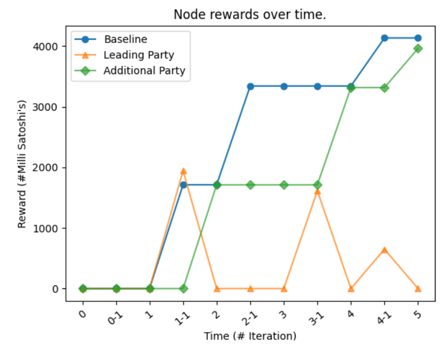


(b) Figure depicting the reward obtained when scenario 2 is run using placement strategy: uniform randomness and transaction amount: 1000000

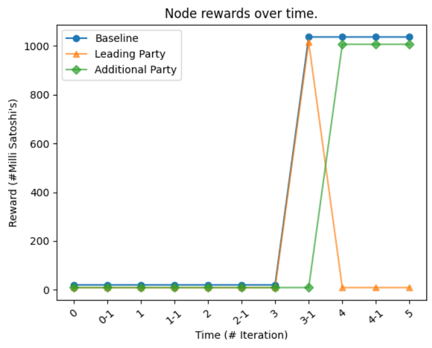
A.5. Case 3 figures



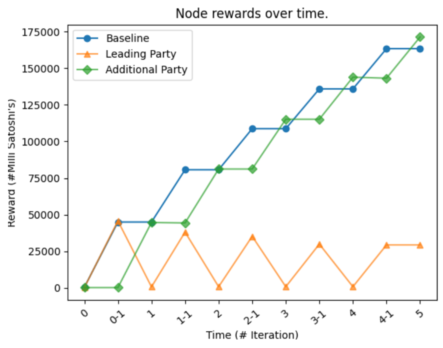
(a) Figure depicting the reward obtained when scenario 3 is run using placement strategy: betweenness centrality and transaction amount: 100



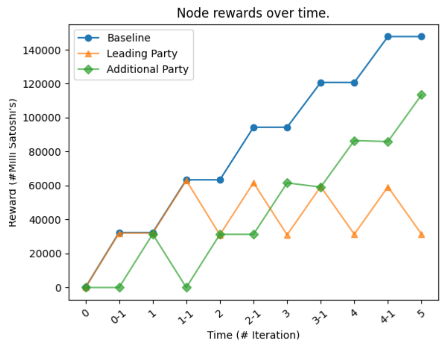
(b) Figure depicting the reward obtained when scenario 3 is run using placement strategy: betweenness centrality and transaction amount: 10000



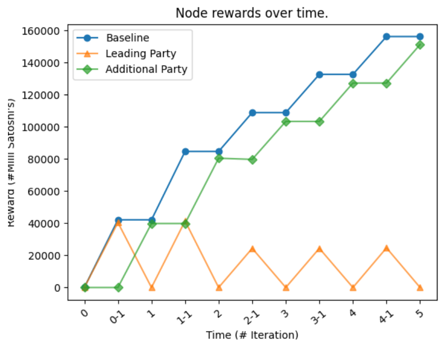
(c) Figure depicting the reward obtained when scenario 3 is run using placement strategy: betweenness centrality and transaction amount: 1000000



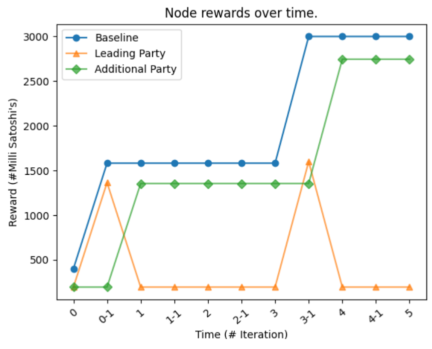
(d) Figure depicting the reward obtained when scenario 3 is run using placement strategy: fee weighted centrality and transaction amount: 100



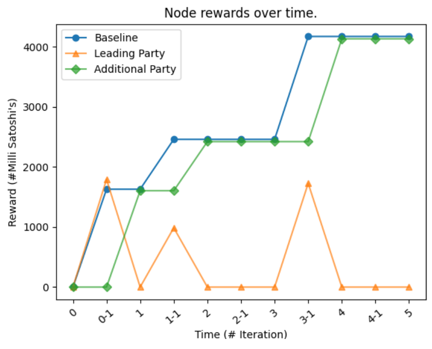
(e) Figure depicting the reward obtained when scenario 3 is run using placement strategy: fee weighted centrality and transaction amount: 10000



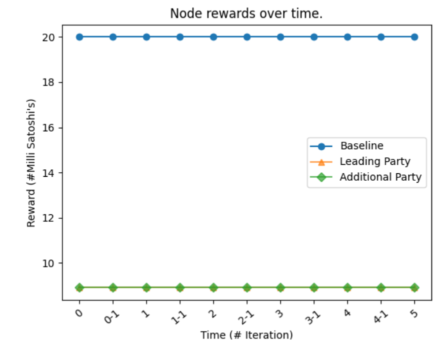
(f) Figure depicting the reward obtained when scenario 3 is run using placement strategy: fee weighted centrality and transaction amount: 1000000



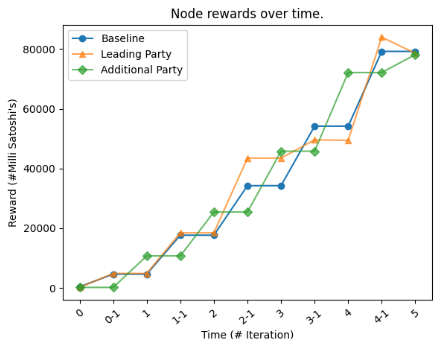
(g) Figure depicting the reward obtained when scenario 3 is run using placement strategy: highest degree and transaction amount: 100



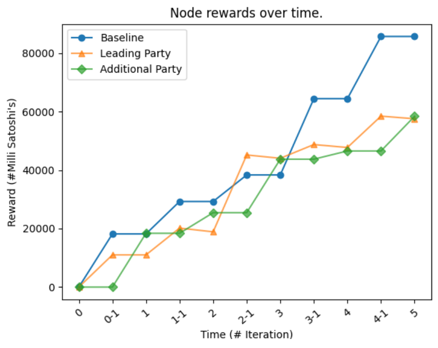
(h) Figure depicting the reward obtained when scenario 3 is run using placement strategy: highest degree and transaction amount: 10000



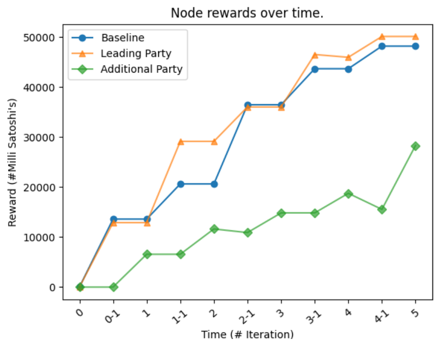
(a) Figure depicting the reward obtained when scenario 3 is run using placement strategy: highest degree and transaction amount: 1000000



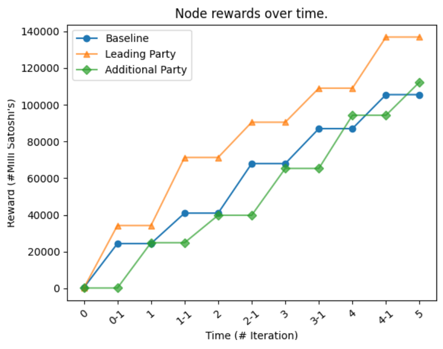
(b) Figure depicting the reward obtained when scenario 3 is run using placement strategy: k center and transaction amount: 100



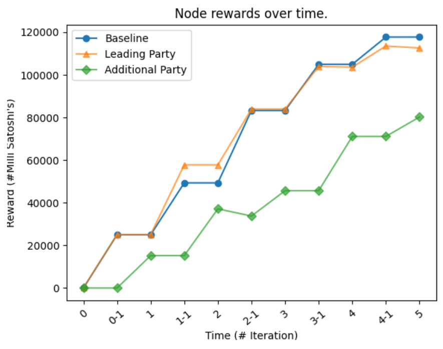
(c) Figure depicting the reward obtained when scenario 3 is run using placement strategy: k center and transaction amount: 10000



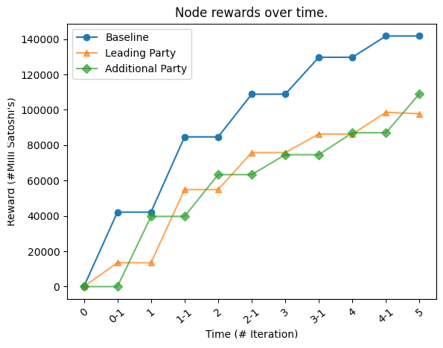
(d) Figure depicting the reward obtained when scenario 3 is run using placement strategy: k center and transaction amount: 1000000



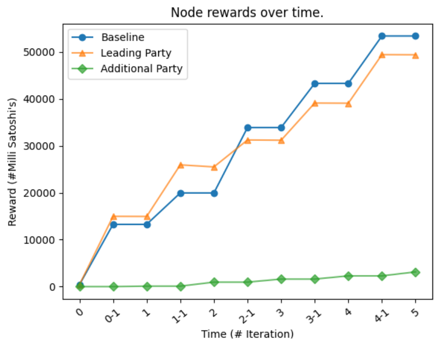
(e) Figure depicting the reward obtained when scenario 3 is run using placement strategy: k means and transaction amount: 100



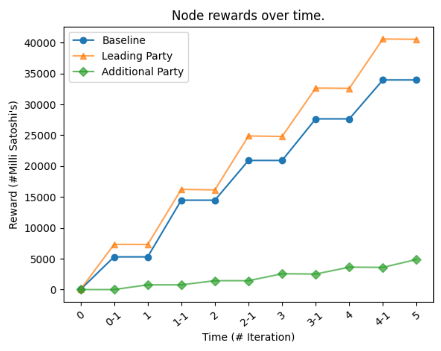
(f) Figure depicting the reward obtained when scenario 3 is run using placement strategy: k means and transaction amount: 10000



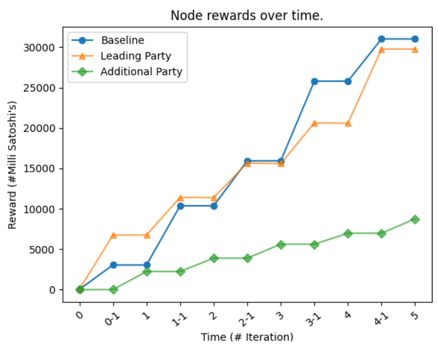
(g) Figure depicting the reward obtained when scenario 3 is run using placement strategy: k means and transaction amount: 1000000



(h) Figure depicting the reward obtained when scenario 3 is run using placement strategy: uniform randomness and transaction amount: 100

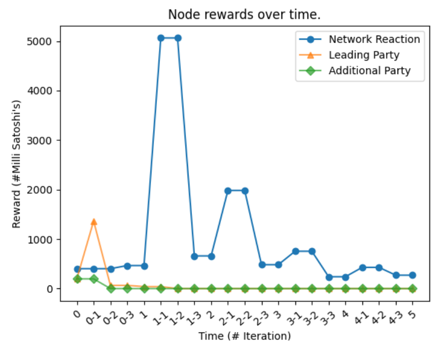


(a) Figure depicting the reward obtained when scenario 3 is run using placement strategy: uniform randomness and transaction amount: 10000

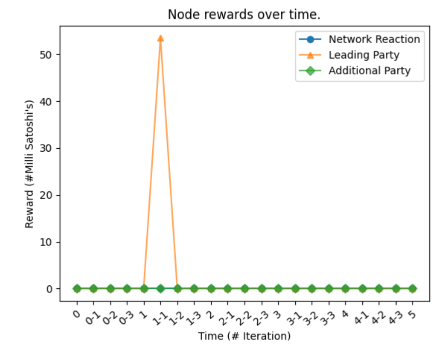


(b) Figure depicting the reward obtained when scenario 3 is run using placement strategy: uniform randomness and transaction amount: 1000000

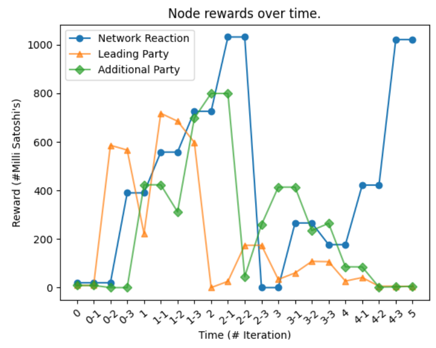
A.6. Case 4 figures



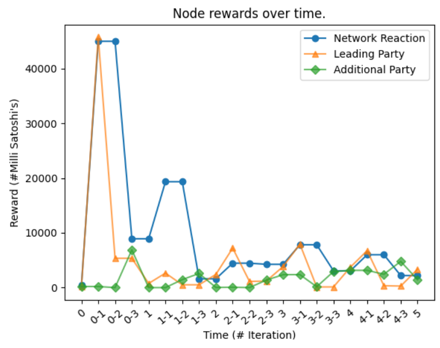
(a) Figure depicting the reward obtained when scenario 4 is run using placement strategy: betweenness centrality and transaction amount: 100



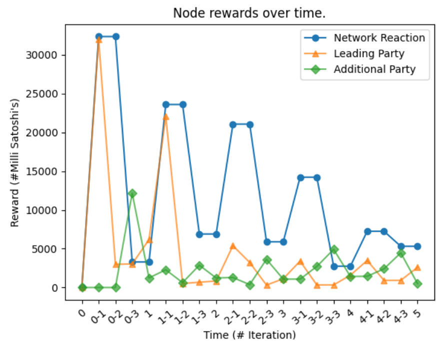
(b) Figure depicting the reward obtained when scenario 4 is run using placement strategy: betweenness centrality and transaction amount: 10000



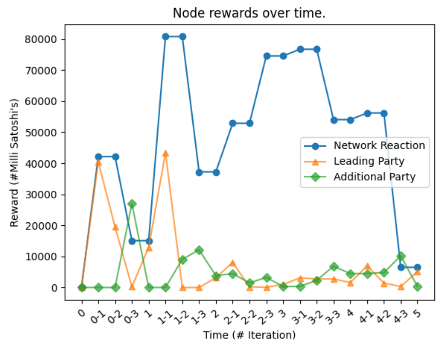
(c) Figure depicting the reward obtained when scenario 4 is run using placement strategy: betweenness centrality and transaction amount: 1000000



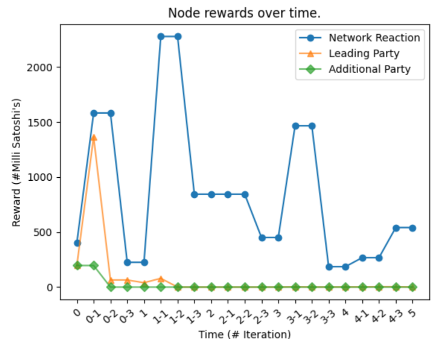
(d) Figure depicting the reward obtained when scenario 4 is run using placement strategy: fee weighted centrality and transaction amount: 100



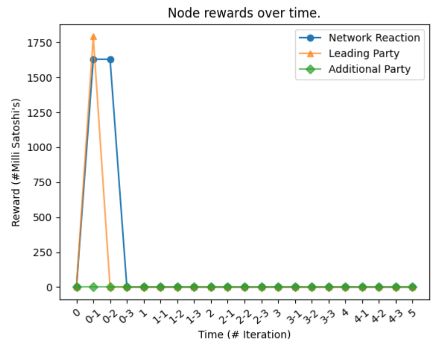
(e) Figure depicting the reward obtained when scenario 4 is run using placement strategy: fee weighted centrality and transaction amount: 10000



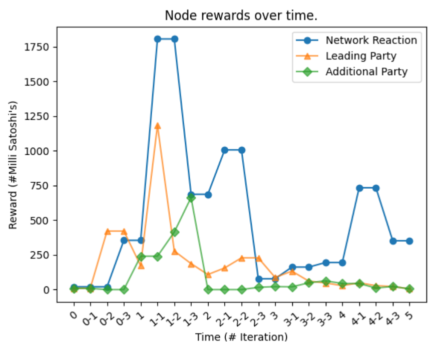
(f) Figure depicting the reward obtained when scenario 4 is run using placement strategy: fee weighted centrality and transaction amount: 1000000



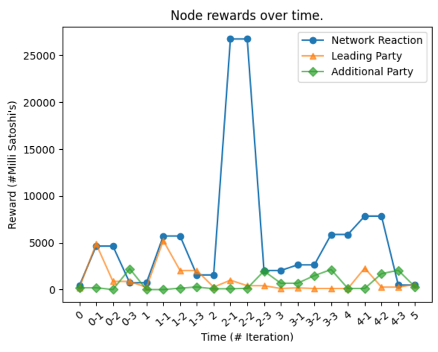
(g) Figure depicting the reward obtained when scenario 4 is run using placement strategy: highest degree and transaction amount: 100



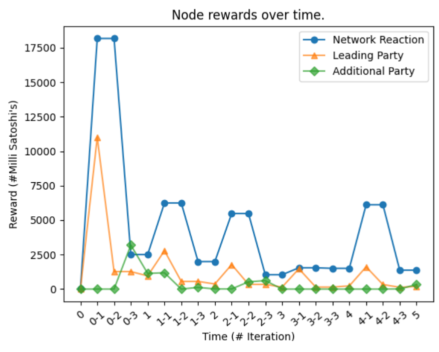
(h) Figure depicting the reward obtained when scenario 4 is run using placement strategy: highest degree and transaction amount: 10000



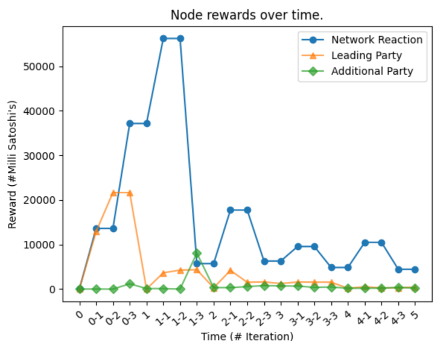
(a) Figure depicting the reward obtained when scenario 4 is run using placement strategy: highest degree and transaction amount: 1000000



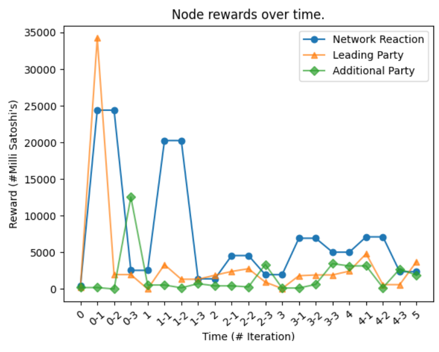
(b) Figure depicting the reward obtained when scenario 4 is run using placement strategy: k center and transaction amount: 100



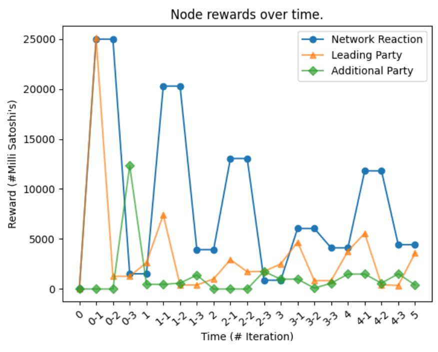
(c) Figure depicting the reward obtained when scenario 4 is run using placement strategy: k center and transaction amount: 10000



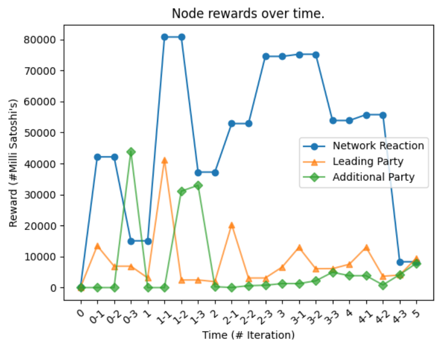
(d) Figure depicting the reward obtained when scenario 4 is run using placement strategy: k center and transaction amount: 1000000



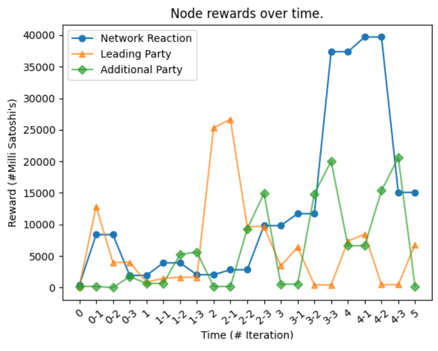
(e) Figure depicting the reward obtained when scenario 4 is run using placement strategy: k means and transaction amount: 100



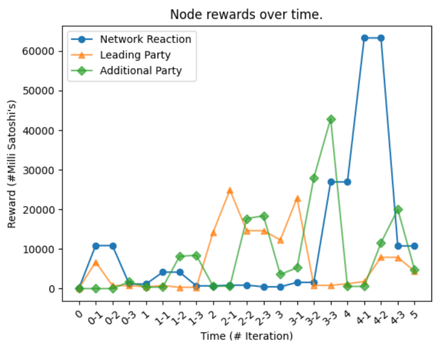
(f) Figure depicting the reward obtained when scenario 4 is run using placement strategy: k means and transaction amount: 10000



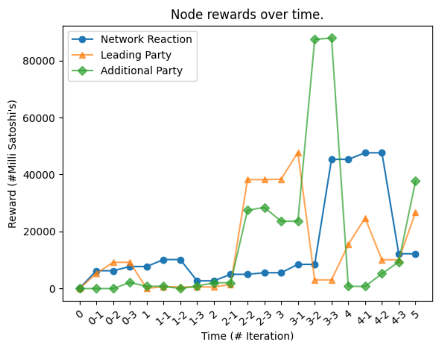
(g) Figure depicting the reward obtained when scenario 4 is run using placement strategy: k means and transaction amount: 1000000



(h) Figure depicting the reward obtained when scenario 4 is run using placement strategy: uniform randomness and transaction amount: 100

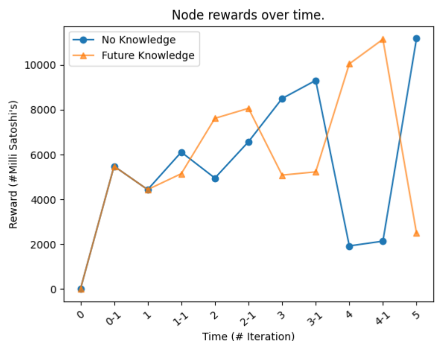


(a) Figure depicting the reward obtained when scenario 4 is run using placement strategy: uniform randomness and transaction amount: 10000

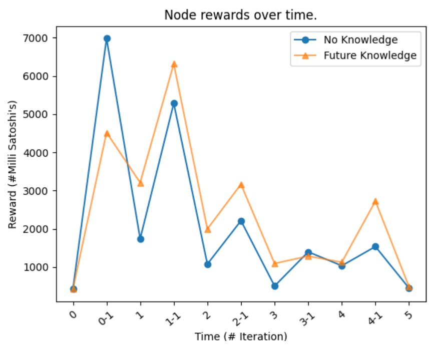


(b) Figure depicting the reward obtained when scenario 4 is run using placement strategy: uniform randomness and transaction amount: 1000000

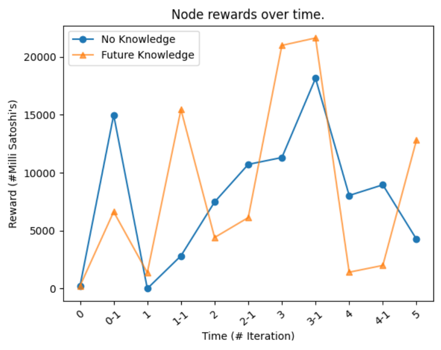
A.7. Game Theory Network figures



(a) Figure depicting the reward obtained when game theory strategy is used to minimize reward reduction from network updates with transaction amount: 100

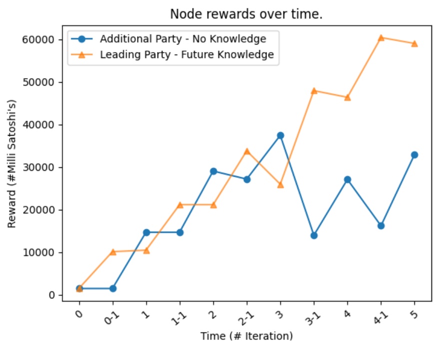


(b) Figure depicting the reward obtained when game theory strategy is used to minimize reward reduction from network updates with transaction amount: 10000

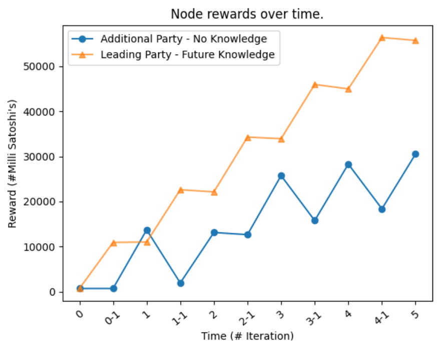


(c) Figure depicting the reward obtained when game theory strategy is used to minimize reward reduction from network updates with transaction amount: 1000000

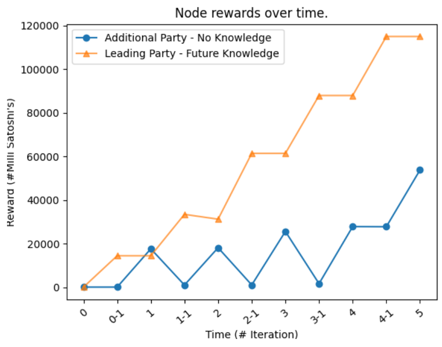
A.8. Game Theory Party figures



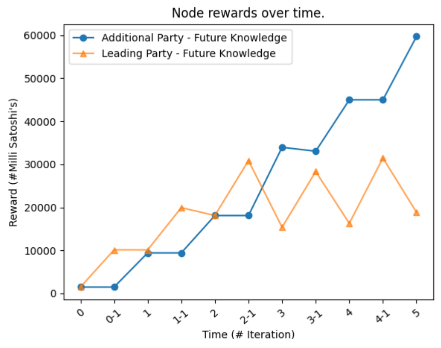
(a) Figure depicting the reward obtained when game theory strategy is used to minimize reward reduction from another party with transaction amount: 100



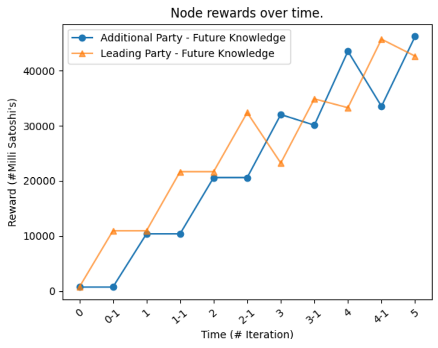
(b) Figure depicting the reward obtained when game theory strategy is used to minimize reward reduction from another party with transaction amount: 10000



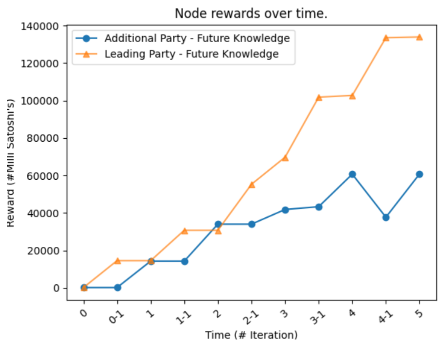
(c) Figure depicting the reward obtained when game theory strategy is used to minimize reward reduction from another party with transaction amount: 1000000



(a) Figure depicting the reward obtained when game theory strategy is used to minimize reward reduction from another party with transaction amount: 100



(b) Figure depicting the reward obtained when game theory strategy is used to minimize reward reduction from another party with transaction amount: 10000



(c) Figure depicting the reward obtained when game theory strategy is used to minimize reward reduction from another party with transaction amount: 1000000