Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Design of a Payment Network with Fees

Bachelor Thesis

Gerrit Janssen

gjanssen@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Georgia Avarikioti, Yuyi Wang
Prof. Dr. Roger Wattenhofer

August 26, 2018

Acknowledgements

I would like to thank my supervisors Georgia and Yuyi for their feedback and advice during our weekly discussions, which always motivated and encouraged me. I am also grateful for the opportunity and experience of being a co-author of a scientific paper. Finally, I would like to thank my family and friends for their continuous support throughout the whole journey.

Remark: While working on this thesis, my supervisors suggested to bundle the results available at that point in a short paper and to submit it to *CBT'18* [3], the International Workshop on Cryptocurrencies and Blockchain Technology, held in September 2018 in Barcelona. The produced paper [4] was accepted and will be published later in 2018. Naturally, since the paper and this thesis partially report on the same results, there will be some overlap. Parts, such as proofs, that were written entirely by my co-authors while working on the paper are marked as such in this thesis with a citation of the paper. (The parts written entirely by me will not be cited.)

Abstract

A big issue of blockchain systems is scalability. Solutions, such as payment channels, have been suggested to alleviate the problem by allowing transactions to take place off-chain. We study the problem of creating a payment channel network with fees, which are charged by a Payment Service Provider (PSP). Customers, however, only prefer this variant of making transactions, if the total fee of routing the payment through the network costs less than the fee on the blockchain.
We introduce various network structures and algorithms that allow the PSP to maximize its profit by assigning optimal fees to the the network's payment channels. For tree-structured networks, where the PSP wants to cater to all transactions, a linear program can find a solution. In the case of paths, we introduce a polynomial-time dynamic program. Finally, we examine star graphs. We provide an optimal fee assignment and a near-optimal solution, when adding an additional central node.

Contents

Chapter 1

Introduction

Even though cryptocurrencies, most prominently Bitcoin [9] and Ethereum [1], have gained quite some popularity in recent years, they are still far from being commonly used in everyday payments. One of the largest obstacles of the underlying technology used by cryptocurrencies, the blockchain, is scalability. This is why the topic is actively researched. Several solutions to this issue have already been suggested, one of which are payment channels [6, 10, 2]. The aim of these is to enable secure, real-time transactions, while making as little use of the blockchain as possible.
In this thesis, we will study the problem from the perspective of a Payment Service Provider (PSP). A PSP would like to provide its customers with an alternative of making payments in place of the blockchain. In order to do so, the PSP creates a network of payment channels by opening channels between interacting parties. However, as nothing comes for free, the PSP looks to charge fees for the use of its payment channel network to maximize its own profit. Customers, on the other hand, prefer this alternative method only if it costs less than a transaction on the blockchain, the PSP's direct competitor. This thesis provides various solutions to the problem stated above and variations thereof.

1.1 Related Work

A lot of research in the direction of payment channels has already been done. This includes the Raiden Network [2] for Ethereum [1] and the Lightning Network [10] for Bitcoin [9]. Both implement payment channels to allow real-time, secure off-chain transactions. The concept of duplex micropayment channels to enable such payment networks is also studied profoundly in [6] by Decker and Wattenhofer and in [5] by Burchert et al.
Other research is focused more on how to route payments in these networks. Flare, a routing algorithm proposed by Prihodko et al.[11] aims to quickly find payment routes in the Lightning Network by collecting information on the topology of the network in advance. SpeedyMurmurs, presented by Roos et al. [12],

and landmark routing, described by Malavolta et al. [8], are two other designs of routing algorithms proposed in recent papers. The latter is used in the decentralized IOU credit network SilentWhispers, a credit network with many desirable privacy properties. SpeedyMurmurs aims to improve routing algorithms, such as the previously described ones, by utilizing embedding-based routing and by distributing funds before path discovery.
Heilman et al.[7], on the other hand, propose a payment hub for Bitcoin, called TumbleBit. This allows anonymous off-chain transactions, completed in seconds.

Chapter 2

Preliminaries

2.1 Payment Channel Networks

In order to formally define the described problem, we introduce in this section some background and the necessary notation.

2.1.1 Payment Channels

Payment channels provide an alternative of securely carrying out transactions with minimal use of the blockchain. Typically, a payment channel is set up between two parties using a well defined creation protocol, where a funding transaction, locking in the funds for the channel, is committed to the blockchain. In our setting though, we assume that the PSP has the ability to open a channel for its customers. In fact, this is possible with three-party channels funded by the PSP alone. The PSP then distributes the money to the other parties of the channel. Various techniques, described in detail in [6, 10, 2], then allow participants in a network of payment channels to make transactions in a secure and fast manner.

2.1.2 Channel Networks

A channel network is defined as an undirected graph $G = (V, E)$ with a set of vertices V and a set of edges E, where $|V| = n$ and $|E| = m$. A vertex (or node) $v \in V$ denotes one of n participants wishing to use the network, whereas an edge $e \in E$ between two nodes i and j stands for an opened channel C_{ij}. Figure 2.1 shows an example of a very simple channel network.

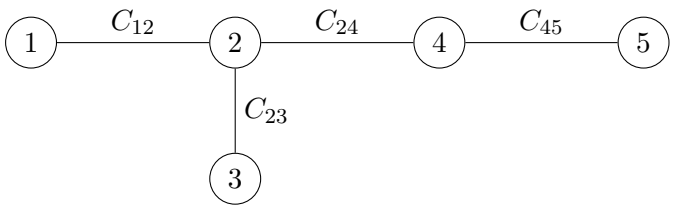
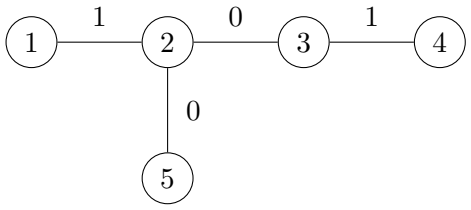


Figure 2.1: A simple network G_1 with 5 participants and 4 channels.

For the sake of simplicity, we make the following two assumptions: The edges of the network graph are undirected and the capacities of the channels are infinite. The latter suggests that the PSP has the means to fund channels with a large enough capital for the transactions performed in the network.
Furthermore, we define the cost of opening a new channel to be 1. This represents the cost of opening a payment channel by submitting a funding transaction to the blockchain.

2.1.3 Transactions

Given a sequence of transactions for n participants, we can define a transaction matrix $T \in \mathbb{N}^{n \times n}$. An entry $T[i, j]$ denotes the number of transactions from the sender i to the receiver j and vice versa. Note that the direction of the transactions is not important and therefore T can be represented by a triangular matrix, as can be seen in an example in Figure 2.2. If there are no transactions for a pair (i, j) of nodes, then the corresponding matrix entry is 0. Since transactions where the sender is identical to the receiver are pointless, the diagonal entries are 0 as well.



$$T = \begin{pmatrix} 0 & 2 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 2.2: A possible assignment of fees f_{E_1} and a transaction matrix T for the previously shown network G_1.

2.1.4 Fees

In order to make profit, the PSP is able to charge fees for the use of the channels in its network. For each edge $e \in E$ a fee $f_e \in \mathbb{R}$ can be assigned and we denote an assignment of fees to the set of edges E by f_E. We require every fee to be positive. Moreover, there is no point in fees that are greater than 1. This is due to the fact that the channel network is in competition with the blockchain, where the cost of a transaction is defined to be 1. In a case where an edge has a fee larger than 1, a rational customer would always prefer to make a transaction via the blockchain instead and effectively render this edge obsolete. Therefore, we also require every fee to be at most 1.
In Figure 2.2 an instance of a possible fee assignment, represented by the labels on the edges, is shown.

As mentioned above, we only consider positive fees in this thesis. As a matter of fact, there exist cases, where, under certain conditions, even negative fees would be viable for maximizing the profit. However, that seems to be far more complex and will not be considered in this thesis.

2.2 Profit Function

To measure the value of a network we introduce a profit function. The profit of a channel network depends on the structure of the underlying graph, the transactions carried out between participants in the network and the fee assignments. We define the profit of a graph $G = (V, E)$, given a transaction matrix T, as follows:

$$p(G, T, f_E) = -m + \sum_{i,j \in V} \sum_{e \in path(i,j)} f_e \cdot X_{ij} \cdot T[i, j],$$

$$\text{where } X_{ij} = \begin{cases} 1, & \text{if the participant chooses to use the network,} \\ & \text{i.e. } \sum_{e \in path(i,j)} f_e \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

By $path(i, j)$ we denote the set of edges on the path from sender i to receiver j, where the overall fees are the lowest (i.e., the cheapest path).
The double sum in the profit function represents the profit generated by all transactions in the network. For every possible pair of nodes (i, j) in the network, we sum all the fees on the path from i to j and multiply by the number of transactions between theses two nodes. However, the profit for a pair of nodes is only taken into account if the fees on their path are at most 1. This indicates that the customer only uses the payment channel network when the cost is not higher than the blockchain fee. Finally, we subtract the cost of opening the payment

channels, which is equal to the number of edges m, because each channel is opened by a transaction with cost 1 on the blockchain.
The profit for the channel network G_1 with transactions T and fees f_{E_1} as shown in Figure 2.2 is equal to $p(G_1, T, f_{E_1}) = -4 + (2+1+0+2+5+6) = 12$. Notice that the single transaction $T[1, 4]$ is not carried out in the network, because there exists no path from node 1 to node 4 with a total fee of at most 1.

2.3 The CNDF Problem

We can now formally define the *Channels Network Design with Fees (CNDF)* problem.

Definition 2.1. (CNDF) Given a transaction matrix $T \in \mathbb{N}^{n \times n}$, return a graph $G = (V, E)$ with $|V| = n$, and fee assignments on edges f_E, such that the profit function $p(G, T, f_E)$ is maximized.

This problem is actually more complex than it seems. Intuitively, one might think that simply connecting every node pair (i, j) that is involved in a transaction by an edge with fee 1, as shown in Figure 2.3, results in optimal profit. However, this is not necessarily the case. For instance, the profit $p(G_2, T, f_{E_2}) = 11$ is actually lower than the profit $p(G_1, T, f_{E_1}) = 12$, where G_1, T and f_{E_1} are the components from Figure 2.2.

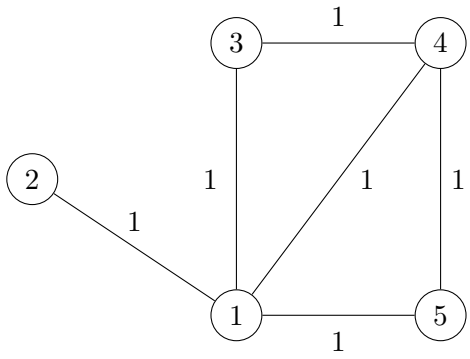


Figure 2.3: A network G_2 where we connect every party of a transaction by an edge with an assigned fee of 1 (f_{E_2}).

In the following chapter we will discuss several solutions to a relaxed version of the *CNDF* problem, where the structure of the channel network is already known. The goal is then to find an optimal assignment of fees, such that the profit is maximized.

Chapter 3

Designing a Network with Fees

3.1 Linear Program for Trees

In a first approach to finding a solution to *CNDF*, we will only consider trees. Assuming a channel network with the structure of a tree, we want to find the best possible assignment of fees, in such a way that every transaction can be carried out within the network, i.e., for every transaction the sum of the fees is at most 1. In other words, every participant will always prefer to use the payment network over the blockchain. This problem can in fact be solved efficiently as described in the following.

Theorem 3.1. *Given any tree and any transaction matrix, there exists a polynomial-time algorithm to optimize the profit if every transaction is performed using the payment network.*

Proof. To solve this variation of the problem, we can use linear programming to find the optimal profit along with an optimal assignment of fees.
In order to do so for some given tree $G = (V, E)$ and a given transaction matrix T, we first need to determine the objective function that we want to maximize. Moreover, we need to specify suitable inequality constraints. We compute the objective function by analyzing how many times each transaction uses each edge in the network. This gives us an objective function

$$f(x) = \sum_{i=0}^{m-1} c_i \cdot x_i.$$

The argument of the objective function, a vector $x = (x_0, \cdots, x_{m-1})$ with $m = |E|$ components represents the fees of the edges, that we wish to maximize, and c_i denotes the number of times the edge i was used by a transaction.

Then, to determine the inequality constraints, which are imposed by the constraint that each transaction must have a total fee of at most 1, we define one inequality for every transaction t as follows:

$$\sum_{i=0}^{m-1} e_i \cdot x_i \leq 1,$$

where $e_i = 1$ if edge i was used for transaction t, and 0 otherwise. It is possible to solve this linear program in polynomial time and it results in an optimal vector of fees x. □

Even though this result works well for some problem instances, it has a few drawbacks: Firstly, the linear programming approach limits the payment network to be a tree. Without this constraint, there might exist multiple paths for a transaction. Then, it is not clear which path should be chosen to route the transaction, as the fees are not yet assigned. The transactions' routing, in turn, has implications on the inequality constraints of the linear program and consequently on the fees and the profit. Moreover, when considering the original definition of *CNDF*, trees are not generally the best network structure for an optimal solution, as one can easily verify with the example given in Figure 3.1.

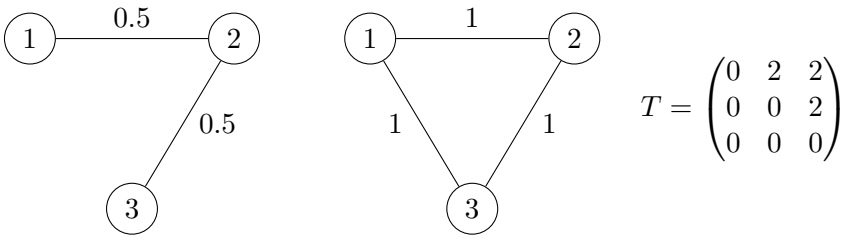
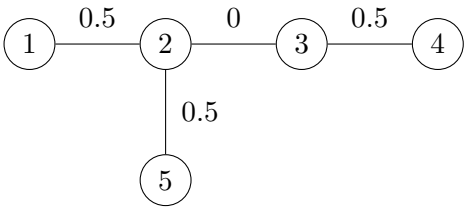


Figure 3.1: Above graphs G_3 (left) and G_4 (middle) show an example, where a tree (graph G_3) is not a favorable network structure, even with an optimal fee assignment f_{E_3}. Compared to the fully connected graph G_4 with profit $p(G_4, T, f_{E_4}) = 3$, the tree has a lower profit $p(G_3, T, f_{E_3}) = 2$ when using the transactions T.

The second shortcoming of this method is the restriction that every transaction has to connect. As a consequence every transaction is given equal weight. However, it is definitely more profitable in certain cases to dismiss a small number of transactions in favor of a larger portion, for which the PSP might then be able to charge a higher fee. Such a scenario is shown in Figure 3.2 and also motivates the next section of this chapter.



$$T = \begin{pmatrix} 0 & 2 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 3.2: The graph G_1 with transactions T as shown before has the fee assignment $f_{E_1'}$ depicted in the diagram when using the linear programming approach. The profit $p(G_1, T, f_{E_1'}) = 9$ is worse than the profit calculated for the fee assignment f_{E_1} in 2.2.

3.2 Dynamic Program for Paths

As we have just seen, linear programming for trees and attempting to cater to every customer's transactions does generally not result in an optimal profit. Therefore, we remove the constraint that every transaction should be able to be carried out within the channel network. The problem is now more complex because we do not know how to select the transactions and applying a linear programming approach is not possible anymore. For a more restricted graph structure, paths, we provide a polynomial-time dynamic programming algorithm (Algorithm 1) that is able to find an assignment of fees, which maximizes the profit.

3.2.1 Algorithm

Algorithm 1: Dynamic Program for Paths

Initialization .....

n = number of nodes, m = number of edges, i.e., $n-1$
Set all entries of $M[m,m,m]$ to 0
Set all entries of $P[m,m]$ to 0

Compute tensor M .....

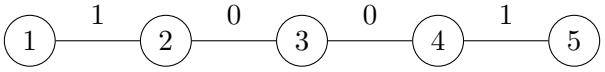
for *every* $1 \leq i \leq j \leq k \leq m$**:**
  $p = 0$
  for *every entry T[u,v] in* T**:**
    if *T[u,v] only uses edges in the interval [i,k]*:
      $p = p + T[u,v]$
  $M[i,j,k] = p$

Compute the dynamic programming table .....

for *every* $1 \leq x \leq y \leq m$**:**
  $P[x,y] = M[1,x,y]$
  for $lastX = 1$ **to** $x-1$**:**
    if *P[lastX,x-1] + M[lastX+1,x,y] >P[x,y]*:
      $P[x,y] = P[lastX, x-1] + M[lastX+1, x, y]$
      Store edges with a fee of 1, i.e., x and edges that have a fee of 1 for $P[lastX, x-1]$

profit = maximum entry in P
fee assignment = edges with a fee of 1 stored for the maximum table entry

First, we compute a tensor M, where $M[i, j, k]$ is the maximal profit when the fee of edge j is 1 and all remaining fees are 0, while considering only transactions in the interval $[i, k]$ of edges (line 6). Next, we compute the dynamic programming table P, initialized with $M[1, x, y]$ (line 8), the maximum entry of which is the optimal profit at the end of the algorithm (line 13). $P[lastX, x-1]$ denotes the best possible profit when setting the edge with index $lastX$ to 1 (it is possible that more edges have a fee of 1 before that, but $lastX$ is the last edge where this is the case) and only using the edges up to (and including) $x-1$. $M[lastX+1, x, y]$ represents the profit in the interval $[lastX+1, y]$ when setting the fee of the x-th edge to 1. Thus, for some fixed x, y we iterate over all possible profits of the preceding part of the graph, add it to the profit of the corresponding current interval and only consider the maximal profit (provided it is greater than only setting the x-th edge's fee to 1)(lines 10/11).
To retrieve the fee assignment which has optimal profit, we can do the following: We define an additional matrix E, where the entries of each row are initialized with the number of the row. Now, every time we update an entry $P[x, y]$, we set $E[x, y] = [x, E[lastX, x-1]]$. These denote the edges that are assigned a fee of 1 to attain the calculated profit. When the algorithm has ended, we read the entry $E[x', y']$, where x', y' are the indices of the maximum value in P, and set the fee of the edge contained in $E[x', y']$ to 1 in the optimal fee assignment.
Figure 3.3 presents an application of the algorithm to an example.



$$T = \begin{pmatrix} 0 & 4 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 8 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 3.3: The path G_5 with a corresponding transaction matrix T has an optimal fee assignment f_{E_5} with 0 and 1 as displayed. f_{E_5} can be calculated using Algorithm 1. One can easily verify that a greedy approach does not work in this case, as it results in a lower profit.

3.2.2 Correctness and Runtime

We now present a proof of correctness for Algorithm 1 and analyze its runtime. In order to do so, we first need to show that a fee assignment where each fee is either 0 or 1 can indeed achieve optimal profit, as is done in the following lemma, first presented in [4].

Lemma 3.2. *[4, Lemma 1 on pp. 5-6] For every given path and for every set of transactions, the optimal profit can always be achieved by assigning edges a fee* 0 *or* 1.

Proof. Assume that we are given some optimal fee assignment $f = (f_1, f_2, \ldots, f_m)$ for the path of length m, but this assignment may use other values, not only 0 or 1. We show that only using 0 and 1 one also can achieve the same (or even more) profit.
Based on the given fee assignment f, we compute the set S of all maximal intervals (i.e., there does not exist a pair of intervals (i, j) and (i', j') such that $i \leq i'$ and $j \geq j'$) where the sum of the fees on the edges in that interval is less or equal to 1. That is, an interval (i, j) is in S if and only if it satisfies that $\sum_{k=i}^{j} f_k \leq 1$ and $\sum_{k=i-1}^{j} f_k > 1$ (or $i = 1$) and $\sum_{k=i}^{j+1} f_k > 1$ (or $j = m$). The optimal profit can be obtained by solving a linear program.
It is well known that every linear program reaches its optimal at the vertex of the feasible region. Hence, we only need to show that every entry of every vertex of the feasible region defined above is either 0 or 1. Equivalently, we show that every feasible solution is a convex combination of vectors with only 0 and 1.
We prove this by induction on the length of the path. For the base case, when the length is 1, i.e., a single edge, it is trivial. Now, assume that this result holds for paths of length smaller than m, and we prove that it also holds for length equals to m. The key observation is that, for any path, there always exists an assignment f' with only 0 and 1 such that $\sum_{k=i}^{j} f'_k = 1$ for every $(i, j) \in S$, as follows:

1. Let $f'_k = 0$ for all k.
2. For k from 1 to m, consider all intervals (i, j) in S such that $i \leq k \leq j$. If all such intervals (i, j) satisfy $\sum_{t=i}^{j} f'_t = 0$, then let $f'_k = 1$.

We define $K := \{k : f'_k = 1\}$ and let $\theta := \min\{f_k : k \in K\}$. Now we write f as a convex combination $f = \theta \cdot f' + (1-\theta) \cdot f''$. Since $\sum_{k=i}^{j} f'_k = 1$ for every $(i, j) \in S$, it follows that $\sum_{k=i}^{j} f''_k \leq 1$ for every $(i, j) \in S$. By the definition of θ, we know that there exists at least one index t such that $f''_t = 0$ ($f_t = \theta$). According to these two facts, f'' can be considered as a feasible solution for the path of length $n-1$, which by the induction hypothesis is also a convex combination of vectors with only 0 and 1. The lemma is proved. □

Unfortunately, the lemma above only allows us to reduce the runtime to $\mathcal{O}(2^m)$ when applying a brute force approach to find an optimal fee assignment. With Algorithm 1 the runtime is further reduced to polynomial time, as is shown by the following theorem.

Theorem 3.3. *Algorithm 1 returns the optimal solution and the time complexity of the algorithm is* $\mathcal{O}(n^5)$.

Proof. Let $OPT(x, y)$ denote the profit of the sub-path from edge 1 up to and including edge y where we set the fee of edge x ($x \leq y$) to 1. We claim that $OPT(x, y)$ fulfills the following recurrence:

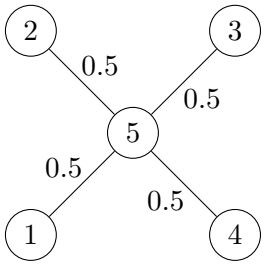
$$OPT(x,y) = max \begin{cases} M[1,x,y] & \textit{(Case 1)} \\ P[lastX, x-1] + M[lastX+1, x, y] & \textit{(Case 2)} \end{cases}$$

If $OPT(x, y)$ is equal to *(Case 1)*, this means that we reach the maximum profit in the subgraph from edge 1 to y by only setting the fee of edge x to 1 in the entire subgraph. Consequently, every transaction, that only uses edges from this subgraph, can generate profit.
Otherwise, if $OPT(x, y)$ happens to be *(Case 2)*, we know that there are at least two edges with a fee of 1 in the subgraph from edge 1 to y, namely on edge x and on edge $lastX$. Therefore, profit is generated by transactions in the first part of the subgraph, i.e. from edge 1 to $x - 1$, and at the same time in the second part, that is from $lastX + 1$ to y. However, no transactions, which use edges in both parts of the subgraph, can generate profit, as such a transaction would then cross both edges with a fee of 1.
Because of this, we can iterate over every possible sum of the profits of $P[lastX, x-1]$ and $M[lastX + 1, x, y]$ and choose the maximum thereof. Note, that we do not necessarily choose the maximum of both terms, but instead pick the maximal sum or otherwise we might only obtain a locally optimal solution. This method can be used, since we were able to split the subgraph from 1 to y in two parts as explained above. Moreover, we have already precomputed both terms: $M[lastX + 1, x, y]$ was computed at the very beginning of the algorithm and $P[lastX, x - 1]$ is always an entry of the table that was the result of a prior computation with exactly the same recurrence.

The tensor M can be computed in time $\mathcal{O}(n^5)$. The computation of the table P can be accomplished in time $\mathcal{O}(n^3)$, since we have 3 loops that iterate over parts of the edge indices. Therefore, the complete algorithm can be implemented with runtime $\mathcal{O}(n^5)$. □

3.3 Star Graphs

In the previous section we have studied a polynomial-time algorithm for paths. However, paths are not really an optimal structure for a payment network, as they are not as highly connected as one would expect. Therefore, we will analyze in this section another network layout, namely star graphs.
A star graph, as for instance in Figure 3.4, has some interesting properties when considering the *CNDF* problem. Firstly, star graphs are trees, which reduces the number of channels necessary while still maintaining the connectivity of the network. Secondly, the diameter (i.e., the longest path) of star graphs equals 2. This suggests that the paths of possible transactions overlap less than it might be the case with other tree-structured networks.



$$T_u = \begin{pmatrix} 0 & 2 & 2 & 2 & 2 \\ 0 & 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 3.4: A star graph S_1 with 5 nodes and uniform transactions T_u.

3.3.1 Optimal fees for star graphs

In the following, we will consider a specific pattern of transactions, where every participant in the network carries out exactly one transaction to every other party (called *uniform transactions* in the following). The reason therefor is that it makes it easier to reason about the optimality of fee assignments. The optimal fee assignment of a star with the described transactions is as follows:

Theorem 3.4. *For any star $S = (V, E)$ with $|V| = n \geq 3$ and uniform transactions T_u, a fee assignment $f_e = 0.5, \forall e \in E$ is optimal.*

Proof. In order to prove the statement, we define the following linear program: For each pair of fees f_i and f_j, where $i \neq j$, we obtain an inequality constraint $f_i + f_j \geq 1$. Furthermore, we have the constraints $0 \leq f_i \leq 1, \forall i$.
Now, if we require that all transactions are facilitated by the network, we treat every inequality as an equation. Solving this system of linear equations results in an optimal fee assignment where every fee is 0.5 (or, for $n = 3$, an assignment with 0 and 1, which has exactly the same profit).

On the other hand, if the requirement that every transaction should be carried out within the network is not given, we only consider those inequality constraints, which correspond to a transaction facilitated by the network, as equations. Depending on this system of equations and the remaining inequality constraints, we obtain a vector of fees f as a solution. We know that the solution of the linear program corresponds to a vertex of the feasible region, a convex polytope. Moreover, each vertex of this linear program's feasible region has coordinates $f_i \in \{0, 0.5, 1\}$. This, in turn, means that we are able to only consider fee assignments which agree with this format. The profit of a star graph with such a fee assignment f'_S can generally be computed as follows:

$$p(S, T_u, f'_S) = x^2 + 2y + 2yz + xz,$$

where x, y and z are the number of edges with a fee of $0.5, 1$ and 0, respectively. Naturally, x, y and z are non-negative and $x + y + z = |E|$ has to hold.
Once again, this problem can be solved using a linear program, which reaches its maximal solution at the point $(x, y, z) = (|E|, 0, 0)$. Thus, the profit of the star graph is definitely maximized, when every edge has a fee $f_i = 0.5$.

□

Corollary 3.5. *For any star $S = (V, E)$, where $|V| = n \geq 3$, and uniform transactions T_u while using the optimal fee assignment $f_e = 0.5, \forall e \in E$, the profit is equal to $(n-1)^2 - (n-1)$.*

Proof. The star graph contains $\binom{n-1}{2}$ pairs of nodes which are connected by a path of length 2. The optimal profit for such a path in the star with uniform transactions equals 4. However, we now included the profit for the paths of length 1 too many times. Hence, we subtract $2\binom{n-1}{2}$, where 2 is the profit using only the paths of length 1, and then add the correct profit of $(n-1) \cdot 0.5 \cdot 2 = (n-1)$. Thus, the profit from transactions amounts to

$$\begin{aligned} 4\binom{n-1}{2} - 2\binom{n-1}{2} + (n-1) &= 2\binom{n-1}{2} + (n-1) \\ &= (n-1)(n-2) + (n-1) \\ &= (n-1)^2 \end{aligned}$$

Subtracting the number of edges, $(n-1)$, of the star yields $(n-1)^2 - (n-1)$ and the statement is proved. □

3.3.2 Payment Hubs

Stars with an optimal fee assignment and uniform transactions as presented seem to achieve quite good profit, only missing out when transactions are routed between the center node and a leaf. However, we can add an additional node acting as a payment hub for the network, like in Figure 3.5, and obtain a situation where every transaction can be carried out within the network with best possible profit.
Assuming the optimal network is connected, we can show with the following theorem from [4] that the star graph using a payment hub is a near-optimal solution.

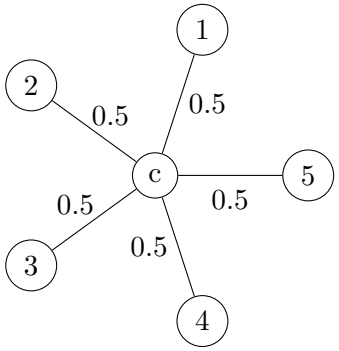


Figure 3.5: A star graph S'_1 with the 5 original nodes and an additional center node c.

Consider an optimal solution for the network G_{opt} with a fee assignment $f_{E_{opt}}$ reaching the profit $opt(G_{opt}, T, f_{E_{opt}})$ for a given transaction matrix T. Further, let $S = (V_S, E_S)$ be a star graph, where $V_S = V \cup \{c\}$. c denotes the additional node acting as the payment hub. To the edges we assign uniform fees, i.e., $f_e = 0.5, \forall e \in E_S$.

Theorem 3.6. *[4, Theorem 3 on p.7] If G_{opt} is connected, then $p(S, T, f_{E_S}) \geq opt(G_{opt}, T, f_{E_{opt}}) - 1$.*

Proof. If G_{opt} is one connected component, then $|E_{opt}| \geq n - 1$. For the star graph S, we have $V_S = V \cup \{c\}$ and $|E_S| = n \leq E_{opt} + 1$. Furthermore, the sum of fees on all shortest paths is equal to 1 due to the uniform fees equal to 0.5 and the star structure. The profit function maximizes its value when all transactions go through the graph G_{opt} with total fee equal to 1, hence

$$opt(G_{opt}, T, f_{E_{opt}}) \leq \sum_{i,j \in V} T[i,j] - |E_{opt}| \leq \sum_{i,j \in V} T[i,j] - |E_S| + 1 = p(S, T, f_{E_S}) + 1$$

The last equality holds since the sum on every shortest path equals to 1. □

An implication of the theorem above is that only networks with the structure of a tree are able to achieve a better profit than the star with a payment hub, which is because trees contain one less edge. However, depending on the transactions, not every tree automatically has a higher profit. The following theorem gives such a condition for the transactions, where the path of one transaction t_0 fully contains two other non-intersecting transactions t_1 and t_2, as displayed in Figure 3.6.

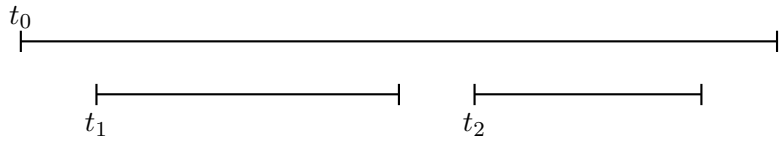


Figure 3.6: The path of t_0 contains the paths of two other transactions t_1 and t_2, which do not overlap.

Theorem 3.7. *For any tree and any transactions, where the path of a transaction t_0 fully contains the paths of two other non-intersecting transactions t_1 and t_2, the star with a payment hub achieves better profit.*

Proof. We can analyze the following three cases:

Case 1: We set the fee of some edge on the path of t_1 to 1 and the remaining fees on the path of t_0 to 0. Then, the profit from these three transactions equals $t_0 + t_1$. Setting the fee of an edge on the path of t_2 to 1 works analogously and results in $t_0 + t_2$ profit.

Case 2: One fee on the paths of each t_1 and t_2 is 1, while the remaining fees on the path of t_0 are 0. This yields the profit $t_1 + t_2$.

Case 3: Setting the fee of a single edge that is only contained in the path of t_0 results in a profit of t_0.
Using only fees of 0 and 1 is possible because all transactions are on a path, which means that Lemma 3.2 can be applied. Due to the fact that every transaction in the star goes through with total fee equal to 1, the profit achieved by these three transaction is $t_0 + t_1 + t_2$ for the star. For the tree with the mentioned criteria, on the other hand, this is not the case, as we have just shown with the case analysis. Thus, the star generates a higher profit and the theorem is proved. □

Chapter 4

Conclusion and Future Work

4.1 Conclusion

In the course of this thesis, we have introduced several methods how a payment channel network with fees could be designed.
We first presented a linear programming algorithm that allows the calculation of optimal fees for given, tree-structured graphs, while facilitating all transactions and maximizing the PSP's profit. The constraint on the transactions, however, is not always beneficial when looking to maximize the profit of a network. Therefore, we also introduced a polynomial-time dynamic program to assign fees in paths without any restriction on the transactions.
Finally, we considered a very promising structure, namely star graphs. Although, we have shown that star graphs with a fee assignment where every fee is 0.5 yields acceptable profits, they can be further improved by adding an extra node. This node acts as payment hub, connecting all parties in the network. We have proven that such a structure is a near-optimal solution when considering connected payment networks. Finally, we have given a condition on the transactions, such that the star achieves the optimal profit.

4.2 Future Work

Even though we managed to provide some algorithms and solutions to variants of the *CNDF* problem, there are still many aspects to explore and prove.
As we have seen throughout this thesis, the *CNDF* problem seems to be hard to solve in its original definition, as well as its variations. However, finding a polynomial-time algorithm or rather, as we believe, a proof of hardness is still required, for instance, for the following conjectures.

Conjecture 1. *The CNDF problem, as stated in Definition 2.1, is an NP-optimization problem.*

Conjecture 2. *Given any graph $G = (V, E)$ (except for a path), where $|V| = n$, and uniform transactions $T_u \in \mathbb{N}^{n \times n}$, return a fee assignment f_E, such that the profit is maximal. This is an NP-optimization problem.*

Doing further research on star graphs, especially with payment hubs, is another interesting direction. We have shown one condition for transactions, where the star results in higher profit than a tree. However, it might be that the star with an extra node is even an optimal solution, or there exist other restrictions one can put on the transactions.

Bibliography

[1] Ethereum white paper https://github.com/ethereum/wiki/wiki/White-Paper

[2] Raiden network (2017), http://raiden.network/

[3] International workshop on cryptocurrencies and blockchain technology - CBT'18 (August 2018), http://deic.uab.cat/conferences/cbt/cbt2018/

[4] Avarikioti, G., Janssen, G., Wang, Y., Wattenhofer, R.: Payment network design with fees, paper for CBT'18; to be published in "Data Privacy Management, Cryptocurrencies and Blockchain Technology" by Springer International Publishing later in 2018; also available at https://www.tik.ee.ethz.ch/file/d19a7a7d45a8a029c086c14102ffabbe/Payment_Network_Design_with_Fees-Full_Version.pdf

[5] Burchert, C., Decker, C., Wattenhofer, R.: Scalable funding of bitcoin micropayment channel networks. In: Spirakis, P., Tsigas, P. (eds.) Stabilization, Safety, and Security of Distributed Systems. SSS 2017. pp. 361–377. Springer, Cham (2017)

[6] Decker, C., Wattenhofer, R.: A fast and scalable payment network with bitcoin duplex micropayment channels. In: Pelc, A., Schwarzmann, A.A. (eds.) Stabilization, Safety, and Security of Distributed Systems. pp. 3–18. Springer International Publishing, Cham (2015)

[7] Heilman, E., Alshenibr, L., Baldimtsi, F., Scafuro, A., Goldberg, S.: Tumblebit: An untrusted bitcoin-compatible anonymous payment hub. In: Network and Distributed Systems Security Symposium 2017 (NDSS), February 2017

[8] Malavolta, G., Moreno-Sanchez, P., Kate, A., Maffei, M.: Silentwhispers: Enforcing security and privacy in decentralized credit networks. In: Network and Distributed Systems Security Symposium 2017 (NDSS)

[9] Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)

[10] Poon, J., Dryja, T.: The bitcoin lightning network: Scalable off-chain instant payments (2015), https://lightning.network

[11] Prihodko, P., Zhigulin, S., Sahno, M., Ostrovskiy, A., Osuntokun, O.: Flare: An approach to routing in lightning network (2016), https://bitfury.com/content/downloads/whitepaper_flare_an_approach_to_routing_in_lightning_network_7_7_2016.pdf

[12] Roos, S., Moreno-Sanchez, P., Kate, A., Goldberg, I.: Settling payments fast and private: Efficient decentralized routing for path-based transactions. In: Network and Distributed Systems Security Symposium 2018 (NDSS)