Brief Announcement: Brokering with Hashed Timelock Contracts is NP-Hard

Eric Chan
University of California, Riverside

Mohsen Lesani
University of California, Riverside

ABSTRACT

In recent years, many different cryptocurrencies have risen in popularity. Since coins vary in fiat value and functionality, it has become important to securely exchange between them. A common exchange method is hashed timelock contracts (HTLC). However, this method did not support brokerage transactions that allow parties to leverage assets they gain during the transaction. We consider HTLC with brokering. The transaction fees for HTLC is a direct function of the size of the leader set. Thus, brokers are interested in finding the minimum leader set of a given transaction graph. We show that finding the minimum leader set on general transaction graphs with brokering is NP-hard. We then introduce flower transaction graphs, a common type of transaction graphs with brokering, and show that finding the minimum leader set of a flower graph is also NP-hard through a reduction from the knapsack problem.

CCS CONCEPTS

• **Computer systems organization** → *Dependable and fault-tolerant systems and networks.*

KEYWORDS

Blockchain

ACM Reference Format:
Eric Chan and Mohsen Lesani. 2021. Brief Announcement: Brokering with Hashed Timelock Contracts is NP-Hard. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing (PODC '21), July 26–30, 2021, Virtual Event, Italy.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3465084.3467952

1 INTRODUCTION

With the rise of hundreds of cryptocurrencies, it has become important to be able to exchange between them. Since coins vary in value and functionality, people may wish to exchange their coins at any given time. However, trading across blockchains is not atomic by default. This has led to the development of the hashed timelock contracts (HTLC), smart contracts that enabled parties to swap assets across blockchains [1]. However, this form of HTLC does not support brokerage transactions that allow parties to leverage assets gained during the transaction. Simple swap transactions require parties to own the assets that they exchange at the outset. This means swaps are unable to accommodate brokerage or arbitrage:

PODC '21, July 26–30, 2021, Virtual Event, Italy

ACM ISBN 978-1-4503-8548-0/21/07.
https://doi.org/10.1145/3465084.3467952

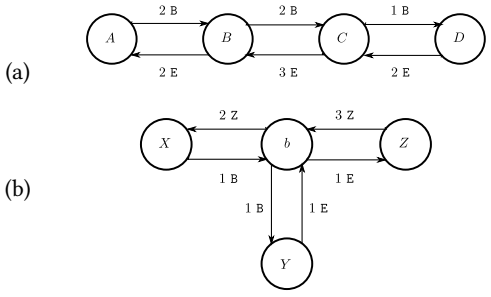

Figure 1: Example (a) Brokering (b) Flower graphs

arranging for an exchange between other parties for a commission, or acquiring an asset and immediately selling it for more.

To address this issue, a previous work [3] transforms the transaction to an equivalent transaction such that each party owns the assets that it exchanges. However, the transformed graph might not be strongly connected; and its safe execution requires certain parties to be trusted. Another work [2] presents a voting protocol that transfers assets tentatively in a central smart contract. However, all parties act as leaders and need to vote to commit the transaction.

An HTLC inputs and hashes the secret of each leader, and triggers only if the calculated hash values match the stored hash values. Therefore, the transaction fees for HTLC is a direct function of the size of the leader set. Hence, it is of interest to find the minimum leader set of a given transaction graphs.

In this short paper, we consider HTLCs that allow brokering, and the problem of finding the minimum leader set. Parties can leverage assets they gain during the transaction instead of owning the assets at the beginning. We discuss the difficulty of finding the minimum leader set on these transaction graphs. We introduce flower graphs, a common type of transaction graphs where a broker mediates between swapping parties. We show that computing the the minimum leader set of flower graphs is also NP-hard by a reduction from the seemingly unrelated knapsack problem.

2 HASHED TIMELOCK CONTRACTS

We briefly review the HTLC mechanism presented in [1]. We refer to the corresponding protocol, the cross-chain swap protocol, as HTLCP. A market clearing service is an untrusted, third-party service that assembles the transaction as a graph G with parties as vertices and asserts as edges. The service also computes a feedback vertex set (i.e., the removal of this set would leave G acyclic), and designates them as the *leaders*. All other vertices are called the *followers*. Each leader l_i generates a secret s_i, then computes the hash of his secret $h_i = \mathsf{hash}(s_i)$. Each contract corresponds to an edge. A contract is *hashlocked* by the hash values h_i of all leaders:

a contract *triggers* (sends the assets escrowed on it to the specified counterparty) only after all preimages s_i have been given to it. Lastly, a contract is *timelocked*: if all secrets are not provided to the contract within the timelock duration, the escrowed assets are returned to the sender. Leaders create all of their outgoing contracts first. A follower creates her outgoing contracts once she verifies all of her incoming contracts are created. When a leader observes that all of his incoming contracts have been created, he releases his secret s_i: he applies s_i to all the hashlocks on his incoming contracts. As these secrets are published on the ledger, other parties can see these secrets. Thus, if a party sees a secret applied to his outgoing contract, he applies it to his incoming contract. This allows all secrets to propagate to all contracts and trigger them.

3 BROKERING

A *broker* is a party that has at least one pair of incoming and outgoing contracts on the same asset. The broker may not own the asset on the outgoing contract and only receive it from an incoming contract during the transaction. Therefore, we represent all the edges on the same blockchain as one master smart contract that keeps track of the tentative owners of the assets in that blockchain during the transaction. Parties that own the asset create edges by escrowing their assets for another party in the master contract. Brokers create an edge by transferring an asset that they tentatively own in the master contract to another party. We say a leader set L is *adequate* for a transaction graph G, if using the HTLCP edge creation rules, the set L can create all edges. We say a set of vertices L is a minimum leader set for a graph G if (1) L is an adequate leader set, and (2) for all adequate leader sets L' of G, $|L| \leq |L'|$.

Any feedback vertex set is an adequate leader set for transaction graphs with no brokering; it can successfully execute the transaction. However, when brokering is allowed, an arbitrary feedback vertex is no longer adequate as a leader set. For example, in Figure 1.(a), the set of parties $\{B, C\}$ compose a feedback vertex set. However, B and C are also brokers. In this case, they do not possess enough B or E assets to create either of their outgoing contracts. Thus, if B and C are chosen as the leaders, the protocol does not progress. However, this is not to say brokers should never be leaders. As there is a cycle between B and C, at least one of them should be a leader. It is always necessary that the leader set forms a feedback vertex set. Clearly, finding the minimum leader set for a transaction graph *with* brokering is at least as hard as finding a minimum leader set on transaction graphs *without* brokering. Since latter is equivalent to finding a feedback vertex set, which is known to be NP-hard, the former is also NP-hard.

We consider a common class of transaction graphs with brokering that we call a flower graph due to its shape. A flower graph is a graph where there is one vertex that is incident to every edge, and has incoming and outgoing edges to every other vertex. As a transaction graph, only the special vertex can be a broker. No other vertex is a broker, i.e., all other parties have the assets to create their outgoing contracts at the beginning of the transaction. This pattern is common as the market clearing service is often the only broker. It has a list of all the offers from its users and can match them to make a flower graph, with itself as the broker for profit. An example flower graph is shown in Figure 1.(b). In this transaction, the party

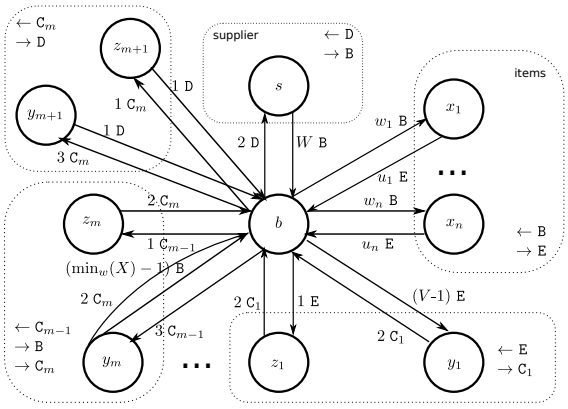

Figure 2: Reduction of Knapsack to Flower graph

X exchanges one B for two Z with the broker b. The broker b passes the B to the party Y in turn. The party Y gives the broker b one E, that he passes to the party Z. The party Z gives the broker b three Z. The broker passes two Z back to X and keeps one Z as profit.

4 FLOWER GRAPHS ARE NP-HARD

We now show that finding the minimum leader set for the flower graph is also NP-hard by a reduction from the knapsack problem. Let the Knapsack problem be given as $K = (X, V, W)$, where X is a set of items $\{x_1, .., x_n\}$, such that each item $x_i = (v_i, w_i)$ is a pair of a value v_i and a weight w_i. We say a knapsack instance K is satisfiable iff there exists a subset R of X such that the total value of all items in R is at least V while the total weight of all items in R is at most W. More precisely, $\sum_{(v_i,w_i)\in R} v_i \geq V$ and $\sum_{(v_i,w_i)\in R} w_i \leq W$. It is well known that the Knapsack problem is NP-hard.

Theorem 4.1. *Finding a minimum leader set for a flower graph is NP-Hard.*

We show a polynomial time reduction from the Knapsack problem to finding a minimum leader set in a flower graph. For a knapsack instance $K = (X, V, W)$, we generate a flower graph instance $F = (\mathcal{V}, \mathcal{E})$ such that K is satisfiable if and only if F has an adequate leader set of size two. Given $K = (X, V, W)$, we generate the flower graph $F = (\mathcal{V}, \mathcal{E})$ shown in Figure 2 in the following steps.

(1) Let b be the broker in F. We create a single party s with an edge to b that transfers W bitcoins B, which we write as $(s, b) = W$ B. In addition, we add the edge $(b, s) = 2$ D. We will refer to s as the *supplier*. (2) For each item $x_i = (v_i, w_i) \in X$, we create a party x_i such that $(b, x_i) = w_i$ B and $(x_i, b) = v_i$ E. We will refer to this set of parties as the *items*. (3) We compute $m = \lceil \sum_w(X)/\min_w(X) - 1 \rceil + 1$, where $\sum_w(X)$ and $\min_w(X)$ are the sum and minimum of the weights of all objects in X respectively. We use m different cryptocurrencies, which we will refer to as $C_1 ... C_m$. (3.a) We create a party y_1 such that $(b, y_1) = (V-1)$ E and $(y_1, b) = 2\ C_1$. We create another party z_1 such that $(b, z_1) = 1$ E and $(z_1, b) = 2\ C_1$. (3.b) For $i = 2$ to m, we create two parties y_i and z_i. The party y_i receives a single contract $(b, y_i) = 3\ C_{i-1}$ from b. We add two edges from y_i to b, $(y_i, b) = 2\ C_i$ and $(y_i, b) = (\min_w(X) - 1)$ B. We add two edges for z_i, $(b, z_i) = 1\ C_{i-1}$ and $(z_i, b) = 2\ C_i$. (3.c) We create a party y_{m+1}

such that $(b, y_{m+1}) = 3\ \mathtt{C}_m$ and $(y_{m+1}, b) = 1\ \mathtt{D}$. Finally, we create the party z_{m+1} such that $(b, z_{m+1}) = 1\ \mathtt{C}_m$ and $(z_{m+1}, b) = 1\ \mathtt{D}$.

We construct the flower graph such that the sequence of contracts that the broker b creates yields a solution to the knapsack problem. We want the parties s and b to be the only leaders. The *supplier* s supplies b with W B to create contracts. The broker b can then use these bitcoins to create contracts to the *item* parties. The item parties are constructed such that if there is a knapsack solution, then b will tentatively acquire at least V E. The rest of the graph is constructed such that b can finish the transaction with V E. To finish the transaction, all contracts must be created.

In particular, in order to create contracts to *item* parties that b had not chosen in the beginning, the other parties should send more bitcoins to b. However, we should construct the graph such that the other parties do not send these bitcoins to b before it solves the knapsack problem. For example, if there were parties that would trade their Bs for Es, then b could subvert solving the knapsack problem by interleaving exchanges between item parties and these parties. Therefore, we set them to release additional bitcoins only after b passes V E to them (y_1 and z_1). Further, to force b solve the knapsack problem, we should prevent any other party from replacing s in the leader set. Choosing any other single party as the leader should not release enough bitcoins to create a contract to any item party. To achieve this, the construction creates two sets of parties $\{y_2, .., y_{m+1}\}$ and $\{z_2, .., z_{m+1}\}$ for a sequence of exchanges. We say that a party is activated if it creates its outgoing contracts. A party y_i is activated only if all the y and z parties before it are activated. A party z_i, on the other hand, is activated if either y_{i-1} or z_{i-1} are activated. Therefore, choosing any single party as a leader cannot activate any y party (except itself). A y party releases bitcoins but does not release enough bitcoins to activate an item party. Therefore, no party can replace s as a leader.

Lemma 4.2. *If K is satisfiable, then F has an adequate leader set of size two.*

Proof. Let R be the set of items that satisfy K. Let $L = \{s, b\}$ be the leader set of F. We show that L is adequate by execution. Since s is a leader, s creates the contract for W B to b. Since b is also a leader, b can securely create contracts using the assets that it receives during the transaction. Thus, b can create outgoing contracts to every party $x_i \in R$. Since R satisfies K, we know the total weight of items in R is no more than W. Since b can leverage W B, he can create the contracts to all $x_i \in R$. Since all *item* parties $x_i \in R$ receive their incoming contracts, they create their outgoing contracts. The sum of all of these contracts is at least V E.

Thus, the broker b can give $(V-1)$ E to y_1 and 1 E to z_1. Thus, b gets 4 $\mathtt{C}_1$, which he can give to y_2 and z_2 in exchange for $(\min_w(X)-1)$ B and 4 $\mathtt{C}_2$. Then, b repeatedly uses the 4 $\mathtt{C}_i$ received from y_i and z_i to create the contracts to y_{i+1} and z_{i+1}, each time gaining $(\min_w(X) - 1)$ B in return. After repeating this process m times, by the definition of m, the broker b is left with (at least) $\sum_w(X)$ B , i.e. the total weight of all X items in bitcoin. The broker b can then use these bitcoins towards creating contracts for *item* parties that have not yet been created, namely, $X \setminus R$. Additionally, after b creates the contracts for y_{m+1} and z_{m+1}, b receives 2 D in return. Then b can create the final contract (b, s). Thus, the set $L = \{s, b\}$ that is of size two is adequate. □

Lemma 4.3. *An adequate leader set for F is at least size two, and if L is an adequate leader set for F of size two, then L is $\{s, b\}$.*

Proof. We show, by the process of elimination, that the leader set is $L = \{s, b\}$. Firstly, we show that $b \in L$. Since every party that is not b only has b in its neighborhood, then either b is in the leader set, or every other party is in the leader set. This latter implies $|\mathcal{V}| - 1 > 2$ leaders; thus, b must be a leader. Further, since b is a broker, at least one other party should be a leader.

Secondly, s must be in the leader set L. It is assumed that no single item $x_i \in X$ has value greater than or equal to V. Otherwise, the Knapsack problem is trivial. Thus, for each $x_i \in X$, if $\{b, x_i\}$ is the leader set, b can gain a maximum of $(V-1)$ E . The only parties that desire E are y_1 and z_1. Thus, b can either create a contract to y_1, or z_1, but not both. Regardless of b's choice, b will receive 2 $\mathtt{C}_1$.

The only parties that desire $\mathtt{C}_1$ are y_2 and z_2, but b can only create the contract to z_2. Thus, b gets 2 $\mathtt{C}_2$, and this process continues for z_3 to z_{m+1}, leaving b with 1 D . However, the only party that desires D is s. Yet, b needs 2 D coins to create the contract (b, s). At this point, b cannot create any contracts with any other party, and there are still contracts that have not yet been created. Thus, for each $x_i \in X$, the set $\{b, x_i\}$ is an inadequate leader set.

Notice that the above applies to each z_i as well. Specifically, if $\{b, z_i\}$ is chosen as the leader set, b can only move assets from z_i to z_{i+1}, until it reaches z_{m+1}. However, this leaves b with only one D, resulting in the same state.

The same also holds for each y_i. If $\{b, y_i\}$ is the leader set, then b will receive $(\min_w(X) - 1)$ B and 2 $\mathtt{C}_i$. By definition of $\min_w(X)$, the broker b cannot create any contracts to any item party x_i. This leaves the 2 $\mathtt{C}_i$. This results in the same scenario as choosing z_i, eventually leading to a stalemate with one D. Thus, for each x_i, y_i and z_i, the sets $\{b, x_i\}$, $\{b, y_i\}$ and $\{b, z_i\}$ are all inadequate. □

Lemma 4.4. *The knapsack instance K is satisfiable if and only if F has an adequate leader set of size two.*

Proof. First, the forward direction follows directly from Lemma 4.2. By construction, we can take $\{b, s\}$ to be our leader set and create all contracts. Next, we show the backward direction: if F has an adequate leader set of size two, then K is satisfiable. By Lemma 4.3, if F has an adequate leader set of size two, then the leader set must be $\{b, s\}$. F is constructed in a manner such that b only receives W B from s, and only the parties $x_1, .., x_n$ desire bitcoins. Thus, b must select a set of parties R such that $\sum_{(v_i, w_i) \in R} w_i$ does not exceed W. Additionally, the incoming contracts to both parties y_1 and z_1 should be created. Otherwise, even if b manages to create the contract to only one of them, it can receive 2 $\mathtt{C}_1$, which cannot generate any more bitcoins. Since V E are required to create the contracts for y_1 and z_1, it must hold that $\sum_{(v_i, w_i) \in R} v_i$ is at least V. Thus, b must have created contracts to a subset R of X spending at most W B and gaining at least V E. This set R is a solution to K. □

As shown by Lemma 4.3, the minimum leader set for F is at least size two. Thus, Theorem 4.1 is immediate from Lemma 4.4.

Lastly, we note that m can be exponential in the size of the knapsack in the general case, but it is also known that the knapsack problem remains NP-hard even when the ratio of weights with respect to each other is polynomially bounded.

REFERENCES

[1] Maurice Herlihy. 2018. Atomic cross-chain swaps. In *Proceedings of the 2018 ACM symposium on principles of distributed computing*. 245–254.

[2] Maurice Herlihy, Barbara Liskov, and Liuba Shrira. 2019. Cross-Chain Deals and Adversarial Commerce. *Proc. VLDB Endow.* 13, 2 (Oct. 2019), 100–113.

[3] Narges Shadab, Farzin Houshmand, and Mohsen Lesani. [n.d.]. Cross-chain Transactions. In *2020 IEEE International Conference on Blockchain and Cryptocurrency*.