Assessing the Efficacy of Heuristic-Based Address Clustering for Bitcoin

Hugo Schnoering[*1,2], Pierre Porthaux[†1], and Michalis Vazirgiannis[‡2]

[1]Coinshares
[2]Ecole Polytechnique

March 4, 2024

Abstract

Exploring transactions within the Bitcoin blockchain entails examining the transfer of bitcoins among several hundred million entities. However, it is often impractical and resource-consuming to study such a vast number of entities. Consequently, entity clustering serves as an initial step in most analytical studies. This process often employs heuristics grounded in the practices and behaviors of these entities. In this research, we delve into the examination of two widely used heuristics, alongside the introduction of four novel ones. Our contribution includes the introduction of the *clustering ratio*, a metric designed to quantify the reduction in the number of entities achieved by a given heuristic. The assessment of this reduction ratio plays an important role in justifying the selection of a specific heuristic for analytical purposes. Given the dynamic nature of the Bitcoin system, characterized by a continuous increase in the number of entities on the blockchain, and the evolving behaviors of these entities, we extend our study to explore the temporal evolution of the clustering ratio for each heuristic. This temporal analysis enhances our understanding of the effectiveness of these heuristics over time.

Introduction

Bitcoin [1] is a peer-to-peer network designed to transfer value in the form of bitcoins between participants. These value transfers are encoded in transactions, and the entire sequence of transactions since the network's inception is chronicled on a public and decentralized ledger known as the

[*]hschnoering@coinshares.com
[†]pporthaux@coinshares.com
[‡]mvazirg@lix.polytechnique.fr

Bitcoin blockchain. Each participant, or *user*, can choose to create an arbitrarily large number of identities, or *addresses*, to engage with the network. As a result, studying Bitcoin transactions from its inception to date involves examining transfers of bitcoins between several hundred million different addresses. Several heuristics have been developed to detect addresses that may belong to the same user. Aggregating large numbers of addresses reduces the total number of entities, and will thus facilitate the analysis of the Bitcoin blockchain.

Related Works The explosion in the number of Bitcoin addresses has been observed in several studies. The issue of reducing the number of entities has thus been raised multiple times [2, 3]. In order to address this concern and make the analysis of the Bitcoin network and blockchain feasible, several authors have employed an initial clustering step, i.e., creating super-entities by merging entities based on specific rules called heuristics. These heuristics rely on the micro-structure of transactions, key management patterns, and Unspent Transaction Outputs (UTXO), aiming to identify identities/pseudonyms that could reasonably belong to the same user.

The most commonly used heuristics are the *common-input-ownership heuristic* and the *change heuristic*. The common-input-ownership heuristic was first mentioned in the Bitcoin white paper authored by the pseudonym Nakamoto. This heuristic was further developed in the works of [4], [5], and [6], all agreeing on the same definition of this heuristic. On the other hand, the change heuristic exists in various forms, with over twenty identified according to Moser *et al.* [7]. The initial versions emerged in 2013 in the works of [4] and [6]. The heuristic described in the former paper focuses on transactions with two recipients, whereas the latter defines it potentially involving more than two recipients. These change heuristics have since undergone modifications or refinements [8, 9]. Other less common heuristics have been developed, specifically targeting Coinbase transactions or transactions involving mining pools [10].

Several studies initially employ these heuristics to reduce the number of entities. Subsequently, traditional machine learning algorithms are applied to further decrease the entity count. Androulaki et al. [4] utilized K-means and Hierarchical Clustering algorithms on a node network obtained through the application of specific heuristics. Remy et al. [11] constructed a network where nodes represent clusters discovered by the common-input-ownership heuristic, adding edges from senders to recipients under specific conditions. The Louvain algorithm is subsequently applied to this constructed network to identify the final entities. He et al. [10] also use the Louvain algorithm on a cluster graph obtained through the application of four different heuristics. Chang et al. [12] used common heuristics and continue clustering by detecting certain predefined transaction patterns and applying specific clustering rules. Moser et al. [7] trained a Random Forest classifier that, based on conclusions from various heuristics and transaction features, detects the change address in a transaction.

The assessment of the relevance of heuristics, i.e., their actual ability to group identities belonging to the same user, has been addressed in several studies [6, 12, 7]. However, the absence of a ground truth dataset representing a significant number of users of different types makes the task challenging. Hence, the utilization of a particular heuristic primarily relies on conviction. Conversely, measuring the effectiveness of reducing entities through a heuristic, i.e., its ability to decrease the number of entities, is more direct. Zhang et al. [13] introduced in their work the address reduction ratio and evaluated its value for various combinations of heuristics. They also highlighted the additional contribution to address reduction by two change heuristics compared to the common-input-ownership heuristic. Finally, several studies [9, 7] have focused on the dynamics of cluster formation over time, as well as the issue of cluster collapse.

Contributions Our contributions are manifold. In section 1, we introduce various concepts related to Bitcoin and define notations that allow for the precise definition of the clustering heuristics. In section 2, we examine six clustering heuristics, including the widely used common-input-ownership heuristic, and our own version of the change address heuristic. We also design four novel heuristics:

1. A refined version of the common-input-ownership heuristic that excludes entities involved in CoinJoin transactions from being merged.
2. A first alternative change address heuristic that detects change amounts, leveraging a known human bias for round amounts.
3. A second alternative change address heuristic that identifies change addresses by assuming input UTXOs are chosen to minimize links between the addresses belonging to the same user.
4. A third version of the common-input-ownership heuristic aimed at identifying consolidation transactions conducted by businesses or services using Bitcoin.

Finally, in section 3, we introduce the clustering ratio as a measure of a heuristic's clustering effectiveness. We proceed to evaluate the clustering effectiveness of the designed heuristics and also assess the effectiveness of a combination of four heuristics. To measure the temporal effectiveness of these heuristics, we calculate these ratios across different block indices, extending our analysis up to block index 700,000.

1 Address Clustering

1.1 Users and Keys

Bitcoin relies on the principles of public-key cryptography. A user u can possess or control a set of one or multiple private keys, represented collectively as $\mathcal{K}_u$. Each private key $k \in \mathcal{K}_u$ secures a portion of u's wealth. Thus, u must keep its private keys secret, as their knowledge allows to access and control the associated funds. Instead of exposing the private

keys to the network, Bitcoin employs a range of derived identifiers, such as public keys, public key hashes, and various other quantities. All of these identifiers are computed from private keys through one-way cryptographic / hash functions. These pseudonymous representations of u, collectively referred to as *addresses*, can thus be safely shared with others, enabling the other participants to identify u. The generation of a private key by a user is a relatively straightforward process, involving the selection of a number from the set $\{0,1\}^{256}$ without the need for intermediaries. This allows a user to potentially use a very large number of keys, and therefore have as many identities on the network. However, managing multiple keys can be complex, especially concerning their storage or the creation of cryptographic signatures required for transactions. The introduction of Hierarchical Deterministic (HD) wallets, as outlined in Bitcoin Improvement Proposal 32 [14], has made private key management simpler. Hierarchical deterministic wallets rely on a master key from which new private keys can be deterministically derived. This improvement simplifies the creation and the storage of private keys, enabling a user u to store his funds in different keys that cannot be easily linked by observers. This enhances the overall privacy and confidentiality of his Bitcoin transactions.

1.2 TXO and Transactions

Transaction Output (TXO) The unit of value of the Bitcoin network, the bitcoin, exists in the form of *transaction output*. A *transaction output* (TXO) τ is defined by two components: a value $v \in \mathbb{N}$ in satoshis (1 satoshi = 10^{-8} bitcoin) and $|p\rangle$, a part of a computer program written in Bitcoin script. $|p\rangle$ is called a *locking script* because it specifies the conditions under which τ, and by consequent the associated value v, can be spent. In most cases, $|p\rangle$ specifies one or several identifiers derived from private keys $k_1, ..., k_i$ allowing only users who knows $k_1, ..., k_i$ to 'unlock' $|p\rangle$ and spend the output τ. The group of users $u_1, ..., u_j$ that have the knowledge of these keys are considered the owners of τ. Since $|p\rangle$ is derived from private keys, it is also referred to as an address.

Transaction A transaction Δ is defined by two subsets of TXOs: a set of input TXOs, Δ_{in} that will be fully consumed during the transaction, and a set of output TXOs, Δ_{out} that will be created during the transaction. A transaction can thus be seen as a transformation of TXOs that allows a change in the distribution of value among users represented by their scripts. A transaction output that has been consumed in a transaction does not exist anymore and thus cannot be double-spent in another transaction. A transaction output that has not been spent yet is called *unspent transaction output* or UTXO. Let $v_{\text{in}}(\Delta) \triangleq \sum_{(|p\rangle, v) \in \Delta_{\text{in}}} v$ be the total value of the input TXOs and $v_{\text{out}}(\Delta) \triangleq \sum_{(|p\rangle, v) \in \Delta_{\text{out}}} v$ be the total value of the output TXOs. For a transaction to be valid, it is imperative that $v_{\text{in}}(\Delta) - v_{\text{out}}(\Delta) \geq 0$ [1]. We can see in Figure 1 a schematic representation of a transaction. The difference $v_{\text{in}}(\Delta) - v_{\text{out}}(\Delta) \geq 0$ is

[1] This is not true for Coinbase transactions, but we will not consider them in this work.

implicitly given to the miner as a form of remuneration for integrating the transaction into the chain.

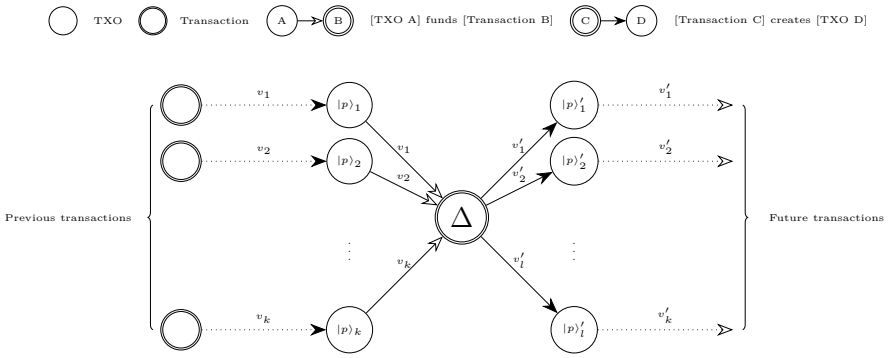

Figure 1: Schematic of a transaction Δ. Nodes with a single (resp. double) border symbolize TXOs (resp. transactions). TXOs consumed by Δ originate from prior transactions, while those created in Δ may serve as input TXOs in subsequent transactions.

Example of Payment Transaction An illustrative example of a transaction is the transfer of payment from one user to another. Let us consider two users, u and u', where u intends to transfer a total value v' to u'. Assuming that u has control over a set of TXOs, denoted as $\tau_1, \tau_2, ..., \tau_k$ with a cumulative value exceeding v. The corresponding Δ takes the following form: $\Delta_{\text{in}} = \{\tau_1, ..., \tau_k\}$ et $\Delta_{\text{out}} = \{(|p\rangle', v')\}$, where $|p\rangle'$ is a script belonging to u'. This transaction is depicted in Figure 2.

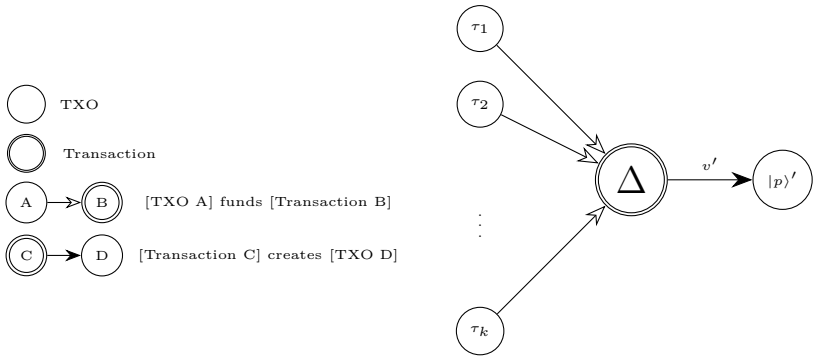

Figure 2: Schematic of a payment transaction Δ.

Change TXO In the previous example, the portion of the input value that exceeds the payment value v is given to the miner. This residual leftover can greatly exceed the standard miner reward, resulting in a loss for u. In order to receive back a part of the input value that is not used to fund the payment and to cover the usual miner fees, an additional output

TXO is added to Δ_{out} to account for the change. Therefore, we have $\Delta_{\text{out}} = \left\{ \left(|p\rangle', v' \right), \left(|p\rangle, v \right) \right\}$ where $|p\rangle$ is an address of u. The miner will now receive $v_{\text{in}}(\Delta) - v - v' \geq 0$. This transaction is depicted in Figure 3.

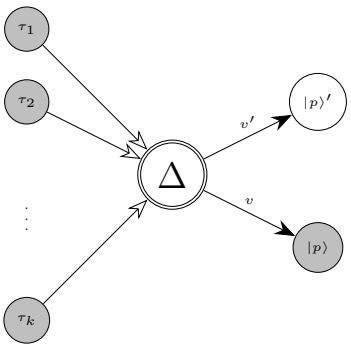

Figure 3: Schematic of a payment transaction Δ with change. Gray (resp. white) TXOs belong to user u (resp. u').

The change script is often referred to as the *change script* or *change address*. A script is said to be *reused* if the script has been used to protect multiple TXOs. This is the case, for instance, if u chooses to protect its change TXO with a script used to protect one of the input TXOs. However, to enhance privacy in Bitcoin, it is commonly recommended to use a new address for each change TXO to break the link between the change and the inputs.

1.3 Script Clustering

Let $\mathcal{S}$ be a set of locking scripts. A *clustering* of $\mathcal{S}$ is a partition of $\mathcal{S}$ consisting of non-empty and mutually exclusive subsets, such that their union equals $\mathcal{S}$. Each of these subsets is referred to as a *cluster*. The *number of clusters* is the number of subsets in the partition. Given a clustering $\mathcal{C}$ and a set of clusters $\mathcal{A} \subset \mathcal{C}$, we define the operation of *merging* $\mathcal{A}$ as the process of replacing the current clustering $\mathcal{C}$ with a new clustering $\mathcal{C}'$ such that $\mathcal{C}' = (\mathcal{C} - \mathcal{A}) \cup \left\{ \bigcup_{c \in \mathcal{A}} c \right\}$. This operation consolidates all clusters of $\mathcal{A}$ into a new, larger cluster. In the following, merging a subset of scripts $\mathcal{B}$ will be equivalent to merging all clusters in $\mathcal{C}$ that contain at least one script of $\mathcal{B}$, i.e.

$$\{c \in \mathcal{C} \mid c \cap \ \mathcal{B} \neq \emptyset\} \tag{1}$$

The *atomic clustering* is the clustering consisting of all singletons, where each script forms its own cluster.

2 Heuristics

We reiterate that this paper aims to identify to identify scripts likely belonging to the same user or entity. Focusing solely on the entities rather

than individual scripts will facilitate future analytical works. Uncovering entities equates to discerning a clustering among the scripts, where the resultant clusters will denote the final entities. To this end, we have to identify the sets of scripts to be merged. In this section, we will develop several heuristics that propose groups of scripts, which could reasonably belong to the same entity. These groups will thus be potential candidates for merging. These heuristics are based on information contained in the transaction's microstructure, particularly the management of TXOs and addresses. Subsequently, we will heavily employ two notations: $n_{\text{in}}(\Delta)$ and $n_{\text{out}}(\Delta)$, denoting the respective number of distinct input scripts and output scripts in Δ.

2.1 Common-Input-Ownership Heuristics

A proposed transaction can be integrated into the chain if validated by the network. For a transaction Δ to be considered valid, each input TXO $(|p\rangle, v) \in \Delta_{\text{in}}$ is accompanied by an *unlocking script* $\langle q|$. This script, also written in Bitcoin script language, proves to the network that the owner of the mentioned TXO agrees to spend their TXO to fund Δ. $\langle q|$ usually contains one or more cryptographic signatures. These signatures encipher information using the private keys from which the identifiers in $|p\rangle$ are derived. These signatures are easily verifiable and tamper-proof. Finding a valid signature without knowledge of the private keys within a reasonable timeframe is highly unlikely. This way, the network can ensure that the owners of $(|p\rangle, v)$ indeed agree to spend their funds in Δ. The responsibility of constructing a transaction, especially gathering all valid signatures, lies with the transaction initiator. If a single user owns all the input TXOs, the task is straightforward as he can calculate all the signatures using his private keys. Conversely, if the input funds belong to multiple owners, each owner will need to compute the signatures. This scenario requires collaboration and greater coordination efforts to construct the final transaction. Heuristic 1 often referred to as the *common-input-ownership heuristic* states that it is more reasonable to assume that all scripts in Δ_{in} belong to the same user.

Heuristic 1 (Common-Input-Ownership Heuristic) *Let Δ be a transaction, if:*

a. $n_{in}(\Delta) \geq 2$,

then the input scripts of Δ are merged.

However, it is not uncommon for users to coordinate in constructing a transaction, sometimes by engaging a third-party coordinator. This is precisely the case with CoinJoin transactions. These transactions pool funds from multiple users to mix them and challenge the heuristic 1. In a prior study [15], the authors developed several heuristics to detect transactions that could reasonably be identified as CoinJoin. Utilizing these heuristics, we constructed a heuristic closely resembling heuristic 1, excluding transactions related to CoinJoin.

Heuristic 2 (CoinJoin-resistant C-I-O Heuristic) *Let Δ be a transaction, if:*

a. $n_{in}(\Delta) \geq 2$,

b. Δ *is not a CoinJoin transaction (according to [15]),*

then the input scripts of Δ *are merged.*

2.2 Change Address Heuristic

Fresh addresses / scripts are often created in order to protect the new (output) TXOs created by a transaction. Consider the simple payment transaction illustrated in figure 3 as an example. Both users u and u' can create a new address to receive the output TXOs. By generating a new script $|p\rangle'$ to receive the payment of u, u' protects his privacy in two ways: hiding to u his other addresses, and thus his other holdings, and preventing external users to track his funds. u can also create a new address $|p\rangle$ to receive his change. By doing so, external observers will not be able to discern with certitude which address received the change and which address received the payment. If u generates a new address whenever he makes a payment, his remaining funds are transferred to a new change address after each transaction. This creates a chain-like structure of scripts as depicted in figure 4. A *script-chain* is defined as a sequence of scripts $|p\rangle_1 \rightarrow |p\rangle_2 \rightarrow ... \rightarrow |p\rangle_k$ satifying:

- all scripts belong to the same user u,
- each script is used only once,
- $|p\rangle_{i+1}$ protects the change TXO of a transaction funded by $|p\rangle_i$

Only the most recent script (the leaf) in the chain holds the funds belonging to u.

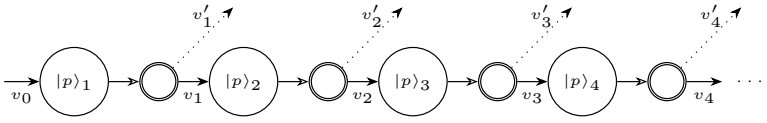

Figure 4: Schematic of a script chain. Full edges represent the transfer of change. Dotted edges represent payments.

If u uses a HD wallet, he is very likely to use fresh change addresses after each transaction. If the recipient of u's payment uses a fresh payment address then it is not possible to distinguish the change address among the recipient scripts. However, if the recipient reuses an address, then it is possible to infer that the other output address is the change address. Heuristic 3 analyses single-input payment transaction (conditions a. and b.) to detect if - the input script and one output script are likely to be derived from a HD wallet (conditions c. and d.), and - the other script has been reused (condition e.).

Heuristic 3 (Detection of Change Addresses) *Let Δ be a transaction, if:*

a. $|\Delta_{in}| = 1$*, the input script will be denoted* $|p\rangle_{in}$*,*

b. $|\Delta_{out}| = n_{out}(\Delta) = 2$*,*

c. $|p\rangle_{in}$ *has not been reused (according to available information),*

d. there exists exactly one output script $|p\rangle_{change}$ *that has not been reused (according to available information),*

d. the other output script $|p\rangle_{pay}$ *has been reused (according to available information),*

then $|p\rangle_{in}$ *and* $|p\rangle_{change}$ *are merged.*

It is specified 'according to available information' in certain conditions, as determining whether a script has been reused or not depends on the other transactions known to us. For example, if an address $|p\rangle$ is reused for the first time at date t' but we are only studying transactions before date $t < t'$, then this knowledge is unavailable to us. This heuristic can be compared to other widely used 'change heuristics' [4, 6, 13] (see appendix A).

2.3 Round Output Value Heuristic

It is a well-known fact that humans exhibit a preference for round numbers. This phenomenon has been studied through the analysis of transaction amounts within a large database of mobile transactions [16]. If we assume that the same psychological bias influences Bitcoin transactions, payments transactions designed by humans are likely to exhibit rounded amounts. Consequently, the payment output is expected to have a rounded value (in satoshis), specifically a multiple of 10^i. Conversely, the mining fee $(v_{\text{in}}(\Delta) - v_{\text{out}}(\Delta))$ is dependent on the transaction size and network congestion, and is therefore calculated algorithmically. As the mining fees are deducted from the change output, the value of the change output is likewise not subject to the rounding bias. Since users frequently think in terms of dollars, the rounding exponent i is likely to fluctuate with the price of a satoshi in dollars. Let p be the price of a satoshi in dollars, and let x be a 'small' value in dollars in comparison to typical payment value. We choose i as the largest integer satisfying $10^i \times p \leq x$, i.e. the integer part of $\log_{10}\left(\frac{x}{p}\right)$. Let $y \gg x$ represent the amount sent, y is likely to be a multiple of 10^i satoshis due to the rounding bias. Heuristic 4 analyzes payment transaction (conditions a. and b.) to detect if - an output has a round amount and is not reused (condition d.), and - the other output amount is rounded to a lesser precision (condition e).

Heuristic 4 (Detection of Round Output Value) *Let Δ be a transaction, p be the price of a satoshi at the time of the transaction, and x be a 'small' amount in dollars. If:*

a. $|\Delta_{in}| = 1$, *the input script will be denoted* $|p\rangle_{in}$,

b. $|\Delta_{out}| = n_{out}(\Delta) = 2$,

c. $|p\rangle_{in}$ *has not been reused (according to available information),*

d. there exists one output script $|p\rangle_{pay}$ *that has not been reused (according to available information), and whose value is a multiple of* 10^i, *with* i *the integer part of* $\log_{10}\left(\frac{x}{p}\right)$,

e. the value of the other output script is not a multiple of 10^{i-j}, *with* $0 \leq j < i$,

then $|p\rangle_{in}$ *and* $|p\rangle_{change}$ *are merged.*

We plotted on Figure 5 the evolution of the rounding exponent as a function of the price of a satoshi and a size of $x = 1$ dollar. To calculate the price of a satoshi, we retrieved the Bitcoin price in dollars from `blockchain.com`; our historical data goes back to January 1, 2012.

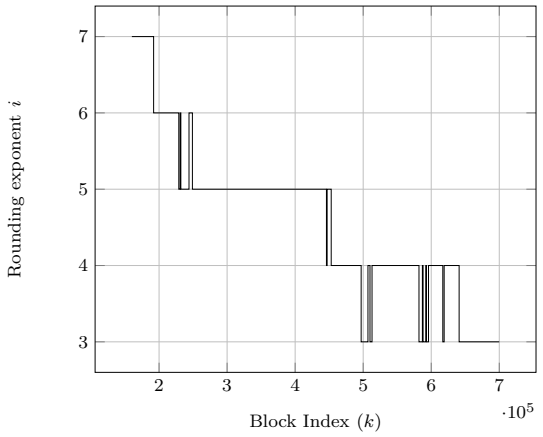

Figure 5: Evolution of the rounding exponent i for $x = 1$ dollar w.r.t. the block index.

2.4 Force Merge of Inputs Heuristic

If the user u uses a HD wallet, he is very likely to generate a new script whenever he receives a payment. Consequently, each new reception address is possibly the root of a new script-chain. Thus, u may have several script-chains under his control. If u intends to transfer funds, denoted as v, to another user u', he utilizes the unspent transaction outputs (UTXOs) protected by the leaves of his script-chains. To prioritize the preservation of his privacy, he attempts to spend funds from a single leaf /script, thereby avoiding having his scripts appear together in a same transaction. However, if no individual chain possesses enough funds to cover the transaction cost, u is compelled to consolidate UTXOs from multiple chains to fund the transaction. Consequently, this consolidation of transaction outputs establishes a link between several of scripts belonging to u. We

leverage this observation to develop heuristic 5 for clustering scripts likely to be owned by the same user. Heuristic 5 analyzes multi-input payment transactions (conditions a. and b.) to detect if - the input scripts and the supposed change script are likely to be derived from a HD wallet (conditions c. and d.), and - the privacy leakage is minimized, i.e. the set of input TXOs merged to reach v is minimal (condition e.).

Heuristic 5 (Detection of Forced Merge of Inputs) *Let Δ be a transaction, if:*

a. $|\Delta_{in}| = n_{in}(\Delta) \geq 2$,

b. $|\Delta_{out}| = n_{out}(\Delta) = 2$, *we suppose that the output script associated with the higher value v_{max}, denoted $|p\rangle_{max}$ is the payment output while the other script $|p\rangle_{min}$ is the change script.*

c. *No input script has been reused (according to available information)*

d. *The change script has not been reused (according to available information)*

e. $v_{in}(\Delta) - \min_{(v,|p\rangle)\in\Delta_{in}} v \ < \ v_{max}$

then the input scripts and $|p\rangle_{min}$ are merged.

2.5 Service Deposit Address Heuristic

Since the emergence of Bitcoin, a wide range of services has been developed, encompassing exchanges, gambling platforms, marketplaces, and more. To utilize these services, users are required to deposit their funds into addresses associated with the specific service. Due to the ease of creating new addresses, these services often have the capacity to generate dedicated deposit addresses for individual customers. Consequently, tracking the deposited funds and determining which funds are available for each customer becomes relatively straightforward. A typical service may exercise control over thousands of addresses, with the largest ones even managing millions. Subsequently, the funds deposited into these addresses are periodically transferred to specialized wallets known as hot or cold wallets, where they are more securely stored. Hot wallets refer to wallets that are connected to the network, enabling easy access and quick transactions, these addresses are in general used to honor customer withdrawals. Cold wallets are offline storage devices, designed to enhance security by keeping funds of the service disconnected from online threats. To save transaction fees, the funds from a large set of deposit addresses are moved together in the same transaction, we use the heuristic 6 to detect such consolidation transactions (many inputs to one output). The heuristic is a special case of the heuristic 1.

Heuristic 6 (Detection of Service Deposit Addresses) *Let Δ be a transaction, if:*

a. $n_{in}(\Delta) \geq a$,

b. $n_{out}(\Delta) = 1$,

then the input scripts of Δ *belong to the same user.* a *is an input parameter.*

3 Results

This section aims to assess the effectiveness of the clustering heuristics developed previously. For each heuristic h, we iteratively construct clusters by applying h to each transaction and merging scripts associated with the same entity as determined by h. On the blockchain, transactions are grouped by blocks, and these blocks form a chain to which a new block of transactions is added approximately every 10 minutes. The term 'block index' refers to a block's position within the chain. Let k be a block index, we define as $\mathcal{S}_k$ the set of scripts present in transactions up to the block indexed by k. Using a clustering heuristic h, an iterative clustering construction can be performed as described below:

- Start from the trivial clustering.
- For every block k, analyze every transaction and consolidate scripts identified as belonging to the same entity as per heuristic h. Continue processing subsequent blocks with the updated clustering.

We denote by $\mathcal{C}_k^h$ the clustering obtained with heuristic h by processing every block up to index k.

Clustering ratio In order to evaluate the clustering effectiveness of h, we compare the final number of entities to the initial number of entities. In our case, the initial and final number of entities are respectively the number of scripts $|\mathcal{S}_k|$ and the number of clusters $|\mathcal{C}_k^h|$. We define the *clustering ratio* as follows:

$$r_k^h = \frac{|\mathcal{C}_k^h|}{|\mathcal{S}_k|} \in (0, 1] \tag{2}$$

The closer this number is to 0, the higher the clustering effectiveness of h.

Data To extract data from the Bitcoin blockchain, we have set up a Bitcoin Core full node. This required downloading and syncing the complete transaction ledger from a network of peers. After installing the latest Bitcoin Core software and configuring the node, the entire transaction history was saved in the local blockchain data directory, specifically in the 'blkXXXXX.dat' files in the '/.blocks' folder. We then delved into the Bitcoin protocol and file structure, using parsing techniques to extract transaction details. This process ensured accurate data for our analysis. We have reported in table 1 the total number of scripts in $\mathcal{S}_k$ for different values of k. We have also plotted in figure 6 the evolution of the total number of scripts w.r.t the block index. It can be observed that the number of scripts has exponentially increased over the years.

Block Index (k)	Number of Scripts Observed ($\mathcal{S}_k$)
100000	174K
200000	6.6M
300000	35.6M
400000	129.3M
500000	346.8M
600000	569.1M
700000	874.6M

Table 1: Total number of scripts.

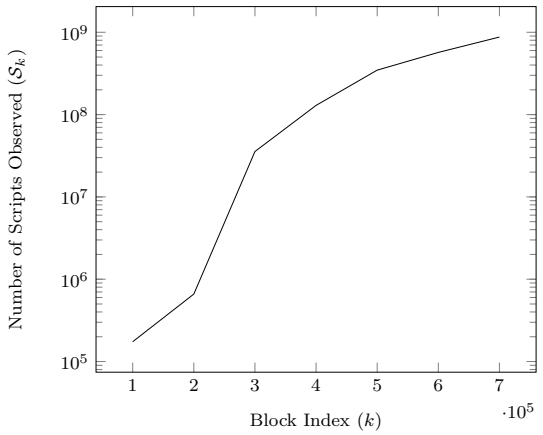

Figure 6: Evolution of the number of scripts w.r.t. the block index.

Results The clustering ratios for the different heuristics developed in the preceding section were computed across various block indices. In implementing the Change Address and Force Merge of Inputs heuristics, we considered transaction information up to block index 700,000. When applying the Round Output Value heuristic, we fixed x at 1 dollar and j at 1. For the Service Deposit Address heuristic, we set a to 25. The evolution of clustering ratios relative to block index for various heuristics is illustrated in Figures 7 and 8. Excluding Bitcoin's first years of existence, we observe relatively stable clustering ratios from block 400,000 onwards. The C-I-O-type heuristics demonstrate the highest efficiency, effectively halving the number of entities that need to be studied. At block index 700,000, the discrepancy in clustering ratios between the two C-I-O heuristics is approximately 0.7%, resulting in a difference of 6 million clusters between their respective final clusterings. The Change Address heuristic is the second most efficient, diminishing the number of scripts by approximately 15%. Finally, the three remaining heuristics (Round Output Value, Force Merge of Inputs, and Deposit Addresses) each decrease the number of scripts by 5% to 10%.

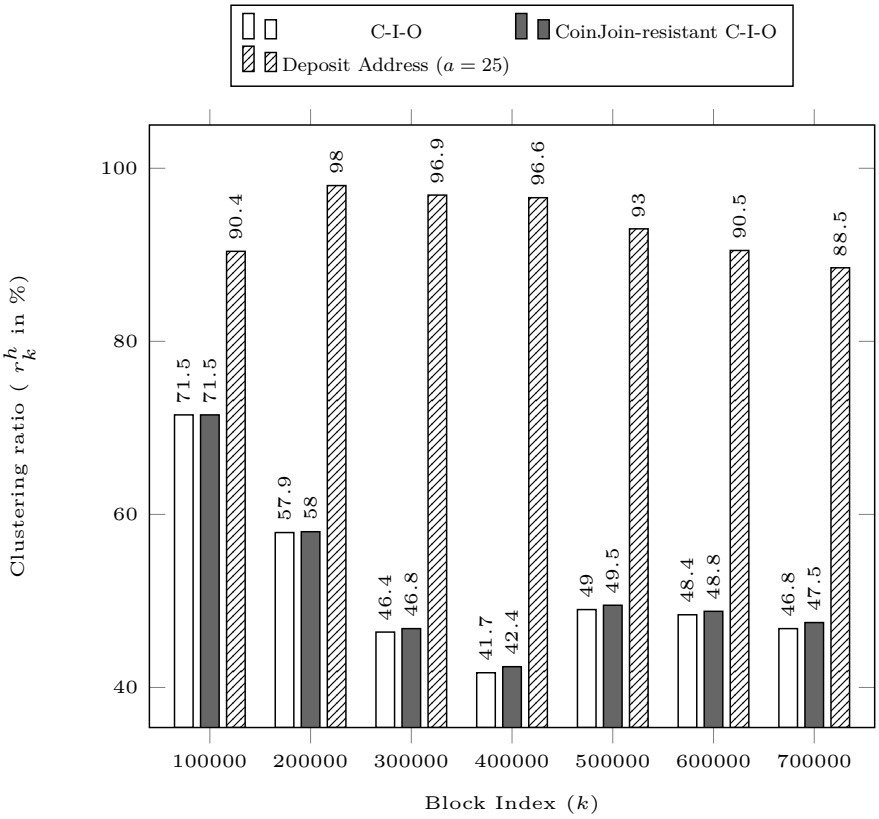

Figure 7: Evolution of the clustering ratio for the C-I-O heuristic and its derivatives.

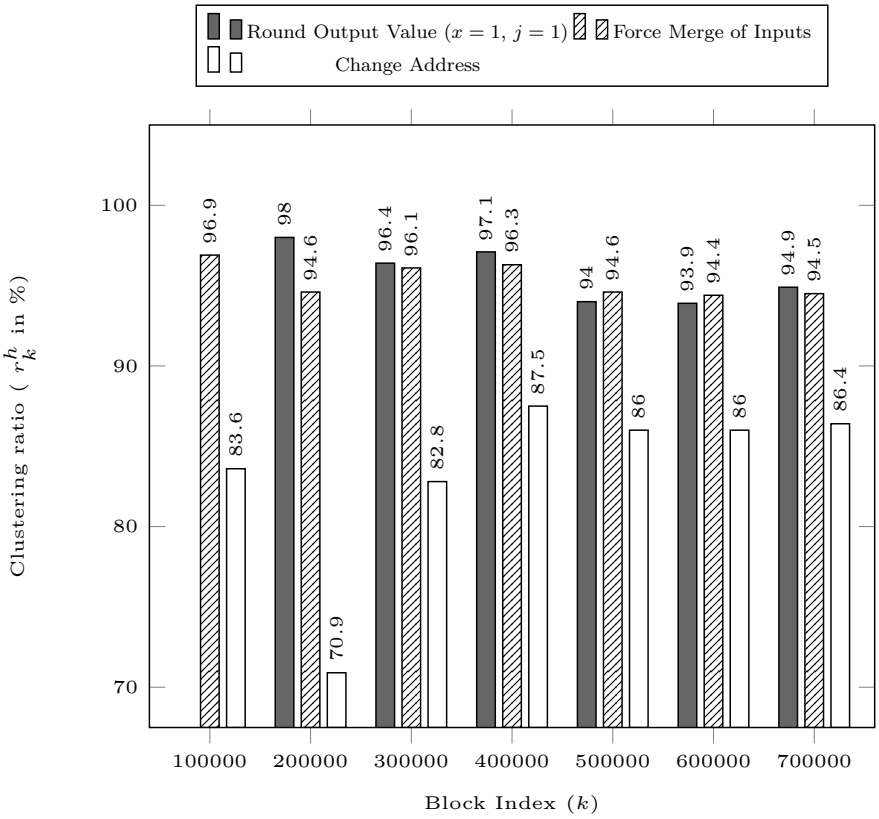

Figure 8: Evolution of the clustering ratio for the change heuristic and its derivatives.

Discussion The impressive performance of the C-I-O heuristic largely explains why most studies utilize and rely on it to reduce the number of entities under scrutiny. For studies that do not want to merge entities involved in CoinJoin transactions, the CoinJoin-resistant C-I-O achieves nearly comparable performance. The Service Deposit Address heuristic is a specific case of the C-I-O heuristic, explaining its lower performance. However, the increasing clustering power of the Service Deposit Address may indicate widespread adoption of deposit addresses by services. The Change Address heuristic and its derivatives exhibit lesser reduction performance. The observed clustering stability with the Change Address and Force Merge of Inputs heuristics may suggest widespread adoption of fresh change addresses, driven by the increasing use of HD wallets since February 2012 (around block index 170,000). In addition to lower performance, these heuristics rely on stronger assumptions and require more resources, especially to determine address reuse. It is not surprising, therefore, that these heuristics are generally less utilized. However, they do identify addresses belonging to the same entity under conditions other than the C-I-O heuristic. If the reduction performance of the C-I-O heuristic is insufficient, combining these heuristics with a C-I-O-type heuristic should further decrease the number of entities and facilitate analysis. We thus combined four heuristics: CoinJoin-resistant C-I-O, Change Address, Round Output Value, and Force Merge of Inputs, aiming to capitalize on the individual clustering power of each heuristic. The evolution of the clustering ratio achieved through this Combined Heuristic is illustrated in Figure 9. This ratio remains relatively stable over blocks, reaching a 70% reduction. This implies that the initial study of 874 million entities at block 700,000 is reduced to approximately 250 million clusters, representing a reduction of 624 million entities.

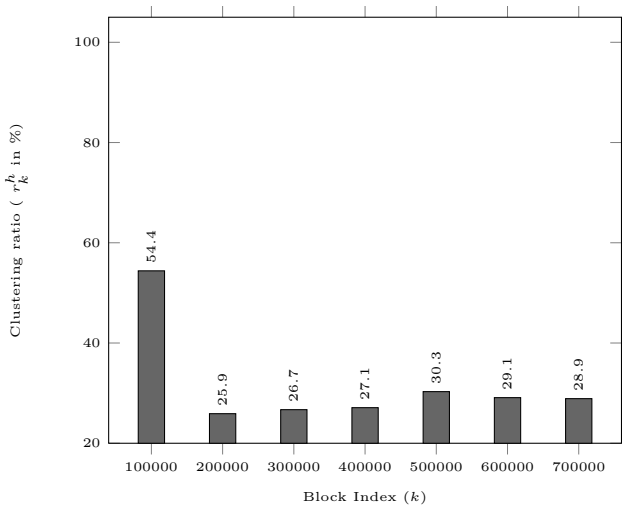

Figure 9: Evolution of the clustering ratio for the Combined Heuristic w.r.t. the block index.

Conclusion

In this study, we explored various heuristics aimed at inferring groups of addresses likely belonging to the same entity. The application of one or more heuristics allows for clustering addresses on the Bitcoin blockchain into groups belonging to the same user. Consequently, an analytical study can reasonably choose to investigate these groups of entities rather than the initial set of addresses. Moreover, this clustering step reduces the number of entities to study, thus saving resources and facilitating the use of algorithms with high computational complexity. We introduced six heuristics, providing motivation by explaining their underlying assumptions and the behaviors they aim to detect. We then measured their reduction power by comparing their initial number of entities to the final number of clusters through the clustering ratio. For each heuristic, we also examined the evolution of this ratio over time to assess the efficiency of these heuristics. The common-input heuristic, almost universally employed in analytical work, halves the number of addresses. Other heuristics exhibit lower reduction powers, ranging from 5% to 15%. However, combinations of these heuristics with the common-input-ownership heuristic further enhance their effectiveness. This is explained by the fact that these heuristics were designed to detect different patterns of behavior or transactions. A strategic combination of a common-input heuristic with a change address heuristic, along with two other heuristics, achieves a 70% reduction in the number of addresses. This translates to the study of 250 million clusters instead of 874 million addresses at block 700,000.

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, 2008.

[2] Erwin Filtz, Axel Polleres, Roman Karl, and Bernhard Haslhofer. Evolution of the bitcoin address graph: An exploratory longitudinal study. In *Data Science–Analytics and Applications: Proceedings of the 1st International Data Science Conference–iDSC2017*, pages 77–82. Springer, 2017.

[3] Mikkel Alexander Harlev, Haohua Sun Yin, Klaus Christian Langenheldt, Raghava Mukkamala, and Ravi Vatrapu. Breaking bad: De-anonymising entity types on the bitcoin blockchain using supervised machine learning. 2018.

[4] Elli Androulaki, Ghassan O Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating user privacy in bitcoin. In *Financial Cryptography and Data Security: 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers 17*, pages 34–51. Springer, 2013.

[5] Fergal Reid and Martin Harrigan. *An analysis of anonymity in the bitcoin system*. Springer, 2013.

[6] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M Voelker, and Stefan Savage. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140, 2013.

[7] Malte Möser and Arvind Narayanan. Resurrecting address clustering in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 386–403. Springer, 2022.

[8] Dmitry Ermilov, Maxim Panov, and Yury Yanovich. Automatic bitcoin address clustering. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 461–466. IEEE, 2017.

[9] Martin Harrigan and Christoph Fretter. The unreasonable effectiveness of address clustering. In *2016 intl ieee conferences on ubiquitous intelligence & computing, advanced and trusted computing, scalable computing and communications, cloud and big data computing, internet of people, and smart world congress (uic/atc/scalcom/cbdcom/iop/smartworld)*, pages 368–373. IEEE, 2016.

[10] Xi He, Ketai He, Shenwen Lin, Jinglin Yang, and Hongliang Mao. Bitcoin address clustering method based on multiple heuristic conditions. *IET Blockchain*, 2(2):44–56, 2022.

[11] Cazabet Remy, Baccour Rym, and Latapy Matthieu. Tracking bitcoin users activity using community detection on a network of weak signals. In *Complex Networks & Their Applications VI: Proceedings of Complex Networks 2017 (The Sixth International Conference on Complex Networks and Their Applications)*, pages 166–177. Springer, 2018.

[12] Tao-Hung Chang and Davor Svetinovic. Improving bitcoin ownership identification using transaction patterns analysis. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(1):9–20, 2018.

[13] Yuhang Zhang, Jun Wang, and Jie Luo. Heuristic-based address clustering in bitcoin. *IEEE Access*, 8:210582–210591, 2020.

[14] Pieter Wuille. Bip 32: Hierarchical deterministic wallets. `https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki`. Accessed on 30/10/2023.

[15] Hugo Schnoering and Michalis Vazirgiannis. Heuristics for detecting coinjoin transactions on the bitcoin blockchain. *arXiv preprint arXiv:2311.12491*, 2023.

[16] Hai Wang, Tian Lu, Yingjie Zhang, Yue Wu, Yiheng Sun, Jingran Dong, and Wen Huang. Last digit tendency: Lucky numbers and psychological rounding in mobile transactions. *Fundamental Research*, 2023.

A Heuristics from Related Works

Heuristic 7 (Shadow Address [4]) *Let Δ be transaction, if:*

a. *$|\Delta_{out}| = n_{out}(\Delta) = 2$ (two output scripts / addresses),*

b. *there exists one output script $|p\rangle_{change}$ that has not been used before,*

c. *the other output script $|p\rangle_{pay}$ has already been used before*

then $|p\rangle_{change}$ is the shadow (change) address. Thus, the scripts of Δ_{in} and $|p\rangle_{change}$ are merged.

Heuristic 8 (One-time Change Addresse [6]) *Let Δ be transaction, if:*

a. *the set of input scripts (i.e. present in Δ_{in}) does not intersect the set of output scrips (i.e. present in Δ_{out}) (no self-change),*

b. *there exists exactly one output script $|p\rangle_{change}$ that has not been used before,*

then $|p\rangle_{change}$ is the change address. Thus, the scripts of Δ_{in} and $|p\rangle_{change}$ are merged.

Heuristic 9 (Address reused-based Change Address [13]) *Let Δ be transaction, if:*

a. *the set of input scripts (i.e. present in Δ_{in}) does not intersect the set of output scrips (i.e. present in Δ_{out}) (no self-change),*

b. *there exists exactly one output script $|p\rangle_{change}$ that has not been used before and that is not reused after,*

then $|p\rangle_{change}$ is the change address. Thus, the scripts of Δ_{in} and $|p\rangle_{change}$ are merged.