Digital Communications and Networks 8 (2022) 680–686

Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.keaipublishing.com/dcan

Multi-input address incremental clustering for the Bitcoin blockchain based on Petri net model analysis

Fangchi Qin [a], Yan Wu [a,**], Fang Tao [b], Lu Liu [c,*], Leilei Shi [a], Anthony J. Miller [c]

[a] Jiangsu Key Laboratory of Security Technology for Industrial Cyberspace, School of Computer Science and Telecommunication Engineering, Jiangsu University, China
[b] Birmingham Business School, University of Birmingham, UK
[c] School of Informatics, University of Leicester, UK

ARTICLE INFO

Keywords:
Bitcoin
Blockchain
Petri net
Incremental clustering

ABSTRACT

Bitcoin is a cryptocurrency based on blockchain. All historical Bitcoin transactions are stored in the Bitcoin blockchain, but Bitcoin owners are generally unknown. This is the reason for Bitcoin's pseudo-anonymity, therefore it is often used for illegal transactions. Bitcoin addresses are related to Bitcoin users' identities. Some Bitcoin addresses have the potential to be analyzed due to the behavior patterns of Bitcoin transactions. However, existing Bitcoin analysis methods do not consider the fusion of new blocks' data, resulting in low efficiency of Bitcoin address analysis. In order to address this problem, this paper proposes an incremental Bitcoin address cluster method to avoid re-clustering when new block data is added. Besides, a heuristic Bitcoin address clustering algorithm is developed to improve clustering accuracy for the Bitcoin Blockchain. Experimental results show that the proposed method increases Bitcoin address cluster efficiency and accuracy.

1. Introduction

As the first application of blockchain, Bitcoin is currently the most popular digital encryption currency in the world. It is a P2P payment system based on Proof of Work (PoW) consensus mechanism [1,2]. It was proposed by Satoshi Nakamoto [3] in 2008 and released as open-source software in 2009. Users can hold, send and receive Bitcoins through Bitcoin addresses, and each address is associated with a private key. These addresses are generated by the users themselves and are not associated with any private information, such as usernames and residential addresses. Users can create and use addresses without a third party, thus allowing them to freely trade Bitcoin without relying on a centrally regulated payment system. Therefore, compared with traditional accounts, such as bank card numbers, Bitcoin addresses have superior anonymity. The openness of Bitcoin transactions makes it difficult for governments to effectively manage them, so Bitcoin is widely used in certain illegal activities and black market transactions. In addition, although Bitcoin is not a national currency, it is legal to use ubiquitously. Germany became the first country in the world to recognize Bitcoin as a “private currency,” alongside the Japanese government claiming that Bitcoin is a legitimate payment method. Darknet websites (Silk Road, Alpha Bay, etc.) use Bitcoin as a payment method, with most of the items sold on these platforms being contraband (drugs, guns, organs, etc.).

On the other hand, by the end of 2017, the overall capital value of Bitcoin had reached a record high of approximately 326.5 billion USD. This huge capital market has attracted the participation of large numbers of investors, ultimately arousing the interest of numerous hackers especially interested in the trading platform, which has gathered a large number of BTC. In 2014, the Mt.Gox [4,5] was declared bankrupt with the theft of 850,000 Bitcoins. In 2015, about 120,000 Bitcoins were stolen from Bitfinex [6].

Research shows that we can analyze the transaction rules of Bitcoin through the transaction rules of Bitcoin, which can help us identify Bitcoin and the real user entities who speculate about owning Bitcoin. Thus, Bitcoin is pseudo-anonymous. In order to identify these Bitcoin addresses used for illegal transactions, we need to cluster Bitcoin addresses. Although Bitcoin cannot correspond to the true identity of the object in real events, every Bitcoin transaction will be recorded on the Bitcoin blockchain, with the information on the Bitcoin blockchain being open and transparent. We can know that a user can have multiple Bitcoin addresses. Through relational clustering, we can find an address cluster controlled by the same user or group to help us identify Bitcoin ownership.

The purpose of this article is to provide an incremental Bitcoin

* Corresponding author.
** Corresponding author.
E-mail addresses: wuyan04418@ujs.edu.cn (Y. Wu), l.liu@leicester.ac.uk (L. Liu).

https://doi.org/10.1016/j.dcan.2022.09.003
Received 21 January 2021; Received in revised form 1 September 2022; Accepted 4 September 2022
Available online 21 September 2022

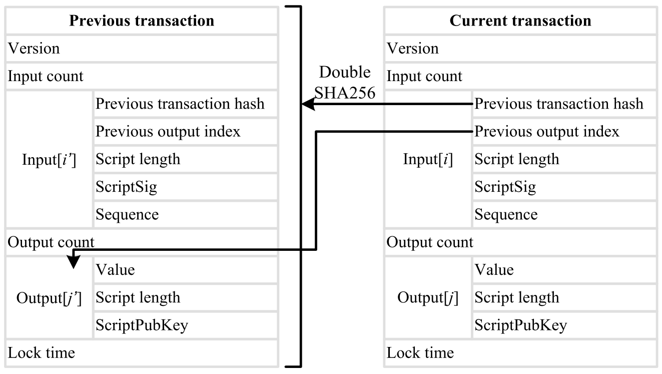

Fig. 1. Structure of Bitcoin transactions.

address clustering method using a heuristic algorithm to improve clustering performance. Initially, a Bitcoin Transaction Net (BTN) based on Petri net is established. It is a three-level complex network containing Bitcoin transactions, coins, and addresses. Then, the proposed incremental clustering algorithm read the previous clustering information and cluster the new addresses. Finally, the updated clustering result is saved.

2. Related works

Bitcoin cluster analysis mainly uses the rules and characteristics of Bitcoin transactions on the blockchain to divide addresses into clusters of users or entities. Reid and Harrigan [7] used the addresses entered as transactions to divide them into clusters. For example, if the addresses *a* and *b* are used as inputs to transaction t_1, *a* and *b* are clustered into the same cluster. If the addresses *b* and *c* are used as the input of transaction t_2, then they are clustered with *a* and *b* into the same cluster. Many studies have used the multi-input address clustering method. Ron and Shamir [8] analyzed the attributes of Bitcoin transactions through this clustering algorithm and studied typical transaction behaviors of users. In addition, they separated some large transactions from the blockchain to observe the main flow of transactions. Fleder et al. [9] annotated the public transaction graph by linking Bitcoin public keys to real people and running the annotated graph through their graph-analysis framework to find and summarize the activity of both known and unknown users. Another address clustering method is called the changed address clustering method. Change of address is used to collect “changes” caused by any transaction (input address) made by the user. Androulaki et al. [10] used the multiple-input address clustering method and the change address clustering method to cluster addresses. Meiklejohn et al. [11] associated the change addresses with some input addresses of the same user, verified the accuracy and comprehensiveness of the algorithm, and analyzed the impact of the number of experimental iterations on the clustering efficiency. Harrigan et al. [12] cited and analyzed the main reasons behind the effectiveness of the Bitcoin blockchain for address clustering. These are the high levels of address reuse and avoidable, merging the existence of super-clusters with high centrality, alongside the incremental growth of address clusters. Although their analysis is only based on the heuristic method of multi-input address transactions, they can be extended to any combination of other heuristic methods. The last clustering method is the coinbase transaction heuristic clustering algorithm (H3). Mao et al. [13] used three heuristic methods to cluster addresses simultaneously and confirmed the accuracy of the method. Using public Bitcoin addresses and their unique labels on social networks can more accurately divide clusters. Meiklejohn [11] and others also collected Bitcoin addresses published on Bitcoin forums, and then used dye analysis and clustering methods to discover the identity information of many Bitcoin addresses. Ermilov et al. [14] proposed an automatic Bitcoin address clustering algorithm, which is not only used to cluster blockchain information but also cluster off-chain information from the Internet. The latter uses off-chain data tags which are used as the reconfirmation of address clusters. This method can avoid the merger of a large number of false clusters generated by the heuristic clustering algorithm based on the Bitcoin blockchain. Neudecker and Hartenstein [15] associated the resulting clusters with IP address information extracted from observing the message flooding process of the Bitcoin network to improve the heuristic algorithm. Spagnuolo et al. [16] proposed a modular framework BitIodine, which parses the blockchain, clusters addresses that may belong to the same user or a group of users, and visually extracts complex information from the Bitcoin network. BitIodine also uses two clustering methods to cluster addresses. BitScope, proposed by Zhang et al. [17], and BitConduite, proposed by Christoph Kinkeldey [18], also make full use of off-chain information to utilize clustering. They usually use a list of recognized addresses from external sources to tag addresses or entities that are linked to those addresses. The above method will re-cluster all the previous Bitcoin addresses each time, with our method utilizing incremental clustering based on the multi-input address clustering algorithm to retain the address clusters generated by the clustering in order to improve the efficiency of the clustering process.

Another research method of the Bitcoin blockchain network is to analyze Bitcoin as Bitcoin traffic. They regard the Bitcoin clusters generated by clustering as vertices with the transaction relationships between vertices as directed edges. The classic Bitcoin traffic analysis method [19,20] initially clusters Bitcoin addresses, then connects the clusters through transaction relationships, and finally analyzes the graphs through visualization and statistical methods. Maesa et al. [21] used user transaction graphs to analyze and verify the “the rich get richer” hypothesis, as well as to analyze and verify that central nodes act as a privileged bridge among different parts of the network. Maesa et al. [22] used graphs to analyze the statistical characteristics of classical complex networks, such as densification, distance analysis, degree and degree distribution, clustering coefficient and several centrality measures. Jourdan et al. [23] used the graph theory to analyze the information revealed by transaction patterns, defining new features related to entities, and studied their effectiveness in practice. Monaco [24] used the graph theory to observe user behavior patterns over time. Petri nets are also used to analyze the flow of Bitcoin transactions. In the Petri net model, the Bitcoin address is the location of the Petri net, and the Bitcoin transaction is the transition of the Petri net. Pinna et al. [25] used the input address clustering method in the Petri net model to cluster addresses, and found some behavioral characteristics, such as using only one address Andrea Pinna used Petri nets to analyze one-time addresses, meaning that addresses can only be used once. A conclusion discovered by Andrea Pinna [26] states transactions form chains whose length is characterized by a power-law distribution. These two models use Bitcoin addresses as the Petri network places/inputs for Bitcoin transactions. In our Bitcoin transaction network, inputs of Bitcoin transactions are not addresses but coins. Therefore, their model cannot easily analyze quantitative trading characteristics. Although our method also uses Petri nets, our method is dissimilar to theirs. Their method advocates trying to find some behavioral characteristics behind Bitcoin transactions, while our method utilizes the transaction characteristics proposed by Wu et al. [27–29] to define the transaction mode, formally models the Bitcoin transactions, and then performs incremental clustering to identify Bitcoin addresses.

3. Formal modeling of Bitcoin transactions

3.1. Bitcoin transaction

The structure of a Bitcoin transaction is shown in Fig. 1. It includes a version number, an input count, an input list (denoted as *Input*), an output count, output list (denoted as *Output*) and lock time. The lock time refers to the timestamp at which the transaction can be included in the block. The input count represents the number of inputs (i.e., the size of

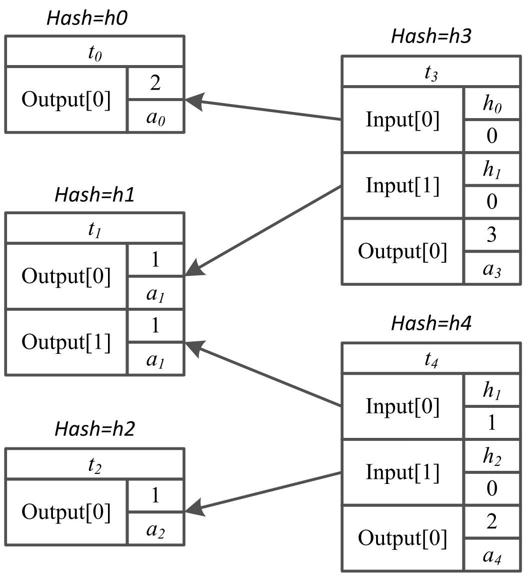

Fig. 2. Sample Bitcoin transactions.

Input). Each *Input[i]* in the *input* list consists of the previous transaction hash, the previous output index, the script length, ScriptSig and the Sequence. By applying SHA256 twice to the previous transaction, the previous transaction hash can be obtained. The previous output index refers to the output of the previous transaction, which is used as the input of the current transaction. The script length specifies the size of the ScriptSig, which unlocks the Bitcoin in the corresponding output of the previous transaction. The sequence was originally designed to allow multiple signatories to agree to update the transaction before it is included in the block. The output count represents the number of outputs (i.e., the size of *Output*). Each *Output[j]* in *output* list consists of value, script length and ScriptPubKey. The value is the number of Bitcoins (unit is satoshi) locked in the *Output[j]*. Script length refers to the length of ScriptPubKey, which is a script used to lock the Bitcoin in the *Output[j]*.

The Bitcoin blockchain provides all Bitcoin transaction information. The information contains the sates of all the coins. It is called Unspent Transaction Outputs (UTXO). Each UTXO specifies the relevant value and conditions of the utilization of Bitcoin. Each transaction has one or more inputs which is the source of funds. Each input points to a valid UTXO and contains authentication information, allowing the use of a corresponding Bitcoin. Similarly, each transaction has one or more outputs, also for the corresponding UTXO. After the transaction completion, UTXOs referenced by their inputs are removed from the UTXO set maintained by every Bitcoin node, and their outputs are added to the UTXO set. A Bitcoin transaction can have multiple inputs and outputs. The multiple inputs may be locked by the same address. We will assume that:

- Each Bitcoin address is controlled by a single real-world entity.
- A single entity may control more than one address.

Bitcoin transactions are divided into two categories: coinbase transactions and regular transactions. Coinbase transactions generate new Bitcoins, which consist of zero input and one output at least. Regular transactions transferring coins between addresses have at least one input and one output. For example, Fig. 2 shows five transactions, where t_0, t_1, and t_2 are coinbase transactions, and t_2 and t_3 are regular transactions. The two inputs of t_3 correspond to *Output[0]* of t_0 and *Output[0]* of t_1. The two inputs of t_4 correspond to *Output* [1] of t_1 and *Output[0]* of t_2. For simplicity, Fig. 2 only shows the value of each output and the Bitcoin address obtained from the ScriptPubKey of each output, the hash value of the previous transaction, and the output index of the previous transaction. For example, the value of *Output[0]* of t_3 is 3, and the address is a_3, which is extracted from ScriptPubKey. The *Input[0]* of t_3 includes h_0 (the hash of transaction t_0) and 0 (the output index of t_0 corresponds to the *Input[0]* of t_3).

This article synchronizes the blockchain data locally by configuring the blockchain environment and building the Bitcoin client (Bitcoin Core). When a new block is generated, the new block data is synchronized to the local. Through a further analysis of the data structure of several blocks, information is extracted from the blockchain data according to the Bitcoin block field structure. Bitcoin transactions use ScriptPubKey (lock script) to identify Bitcoin recipients. Generally, a hash value is contained in the ScripPubKey. An address is obtained by hashing the hash value using Double-SHA256 and then encoding it by Base58. For convenience, our discussion will focus on the previous transaction hash of each input, the output index of the previous transaction and the Bitcoin address obtained from the ScriptPubKey of each output.

3.2. *Construction of Bitcoin transaction Petri net*

Bitcoin transaction Petri net is a formal model of Bitcoin transaction, which is based on the secure Petri net. Petri net is essentially a directed graph. A Petri net is mainly composed of two kinds of nodes: place and transition, so it is also called P/T net. Each node can only connect to other types of nodes, and the directed line segment connecting the place and the transition is called the directed arc. One of the advantages of using Petri nets is that it can be well described systematically using the algebraic form. This formal model provides a set that defines nodes and arcs. A Petri net N is a triple-tuple defined as described below.

Definition 3.1.

$$N=(P, T, F) \tag{1}$$

$P=\{p1, p2, \ldots, pm\}$ is a finite set of m places,
$T=\{t1, t2, \ldots, tn\}$ is a finite set of n transitions,
$F\subseteq(P \times T)\cup(T \times P)$ is a finite set of directed arcs.

On the other hand, Petri nets are very suitable for expressing concurrent and asynchronous computer system models. It can rigorously and accurately reflect the execution process of the system. These characteristics are in full compliance with Bitcoin transactions. Finally, the formal model of the Petri net itself includes the sequence of transactions, and the transaction sequence is also included in the Bitcoin blocks, which is also very consistent.

Given a list of Bitcoin transactions, we map Bitcoin transactions to transitions in the network; the flows of Bitcoin transactions is mapped to the arcs $F\subseteq(P \times T)\cup(T \times P)$ between the places and transitions; the relationshipse between the addresses and the transaction outputs are mapped to the connections between the addresses and the places.

Fig. 3 shows the Bitcoin transaction Petri net that corresponds to the sample Bitcoin transactions in Fig. 2. The transitions t_0, t_1, t_2, t_3 and t_4 represent the five transactions, respectively. Place p_0 represents *Output [0]* of transaction t_0 and *Input[0]* of transaction t_3. Place p_1 represents *Output[0]* of transaction t_1 and *Input* [1] of transaction t_3. Place p_2 represents *Output* [1] of transaction t_1 and *Input[0]* of transaction t_4. Place p_3 represents *Output[0]* of transaction t_2 and *Input* [1] of transaction t_4. Place p_4 is *Output[0]* of transaction t_3, and Place p_5 is *Output[0]* of transaction t_4. For each coinbase transition, we add an input place so that Bitcoin transactions can be analyzed with standard Petri net semantics. Place p_6 (or p_7 or p_8) is an added input for coinbase transaction t_0 (or t_1 or t_2). Each place is annotated with a Bitcoin address. If it represents a transaction's output, the address is extracted from the output's ScriptPubKey. If it is an added input of a coinbase transaction, the address is a unique random number.

The formal definition of the Bitcoin blockchain Petri network is as follows: denoted as a five-tuple $G = (A, P, T, F, \gamma)$, where $A = \{a_1, a_2, \ldots,$

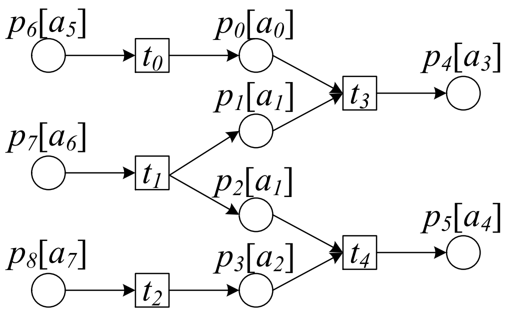

Fig. 3. Bitcoin transaction net for the transactions in Fig. 2.

Table 1
Notations of Bitcoin transaction net.

No	Notation	Meaning
1	$t^{\square} = \{p \mid (p,t) \in F\}$	The set of input places of transition t
2	$a^{\square} = \{p \mid \gamma(p) \in a\}$	The set of out places associated with the same address a
3	$p^{\square} = \{t \mid (p,t) \in F\}$	The set of output transitions of place p
4	$a^{\square} = \{t \mid \gamma(p) = a \wedge t \in p^{\square}\}$	The set of transitions whose input places are mapped to address a
5	$t^{\square} = \{\gamma(p) \mid p \in t^{\square}\}$	The set of addresses associated with all the input places of t

$a_m\}$ is the set of Bitcoin input addresses in the Bitcoin blockchain, and each element in A is an input address in the blockchain network. $P = \{p_1, p_2, \ldots, p_n\}$ is a set of input places of the Bitcoin transactions, and each element in P is an input place in the transaction. $T = \{t_1, t_2, \ldots, t_j\}$ is a collection of Bitcoin transactions, and each element in T is a transaction in the blockchain. $F = \{f_1, f_2, \ldots, f_k\}$ is the set of edges from the input to the transaction, and each element $f \subseteq (p \times t)$ in F is denoted as $f_k < p_n, t_j >$, which is called an arc from p_n to t_j in the network. $\gamma(p)$ denotes an address mapped to place p.

For analysis convenience, Table 1 summarizes the notations of Bitcoin transaction net used in this paper. $^{\circ}$, $^{\square}$ and * denote sets of places, transitions and addresses. For example, in Fig. 3, $t_0^{\square} = \{p_0\}$ $a_0^{\square} = \{p_0\}$, $p_0^{\square} = \{t_3\}$, $a_0^{\square} = \{t_3\}$, $t_0^{\square} = \{a_5\}$.

4. Heuristic clustering algorithm

The clustering procedure computes a partition $\Psi = \{C_1, C_2, \ldots, C_n\}$ of the set of all addresses A with $C_1, \ldots, C_n$ denoting the resulting clusters. For this, it processes all transactions in their temporal sequence. For each transaction t, all selected heuristics compute a partition $\Psi_t = \{\Psi_t^1, \ldots, \Psi_t^m\}$ of all input and output addresses of $t(outputs(t) \cup inputs(t))$. All addresses of transaction t in the partition Ψ_t^i are controlled by one user. Ψ_t is used to update Ψ: First, all clusters Ψ_t^i are added to P. Then, each added cluster Ψ_t^i is merged with all existing clusters in Ψ that contains any of the addresses in Ψ_t^i. This allows all addresses controlled by a single user to be linked according to various heuristic clustering algorithms.

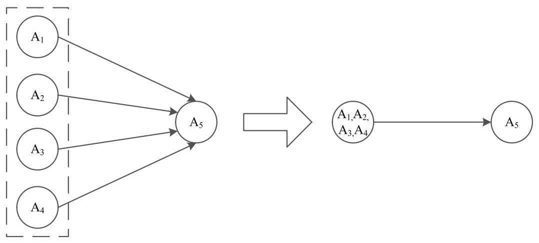

Fig. 4. Multiple input address clustering.

4.1. *Multi-input address heuristic algorithm*

When a user pays for a transaction, the amount of payment often exceeds the amount of Bitcoin in each available address in the user's wallet. In order to avoid the continuous loss of transaction fees caused by the execution of multiple transactions to complete the payment, users will select multiple Bitcoin addresses from the wallet, use the total amount of all Bitcoin credits to complete the payment of the transaction, and then realize multi-input transactions. Since Bitcoin transactions require a separate signature for each address, it is usually believed that all input addresses in a multi-input transaction are controlled by one entity. In other words, if the same transaction has two or more input addresses, they are then considered to be controlled by the same user, which means, for any transaction t, all addresses $a_i \in inputs(t)$ are controlled by the same user. Fig. 4 shows that in a transaction, addresses a_1, a_2, a_3 and a_4 transfer to address a_5 at the same time, which means that addresses a_1, a_2, a_3 and a_4 belong to the same user, who transfers the Bitcoin amount of the sum of addresses a_1, a_2, a_3 and a_4 to address a_5.

Without considering the case where users deliberately use the "CoinJoin" [30] to avoid cluster analysis, the accuracy rate of multi-input address clustering can reach 100%. For a transaction t, the partition judged by this heuristic can be obtained by formula (2).

$$\Psi_t = \left\{inputs(t), \{o_1(t)\}, \ldots, \left\{o_{|outputs(t)|}(t)\right\}\right\} \tag{2}$$

5. Address increment clustering based on multi-input transactions

5.1. *Initial state of Bitcoin transaction net*

Algorithm 1 below describes how to transform a Bitcoin blockchain into a Bitcoin transaction net.

Algorithm 1. Construction of Bitcoin transact net

Input: A block chain (a list of blocks).

Output: $G = (A, P, T, F, \gamma)$.

```
1.  For each block
2.    For each Bitcoin transaction $t$ in the current block
3.      $T = \{t\} \cup T$;
4.      For each output $p$ of transaction $t$
5.        $P = \{p\} \cup P$;
6.        $F = \{(t, p) \cup F\}$;
7.        $\gamma(p)$ = address $a$ extracted from the ScriptPubKey
          field of output $p$;
8.        $A = \{a\} \cup A$;
9.      End For
10.   End For
11. End For
```

Lines 3–4 take out the Bitcoin transactions in each block in turn, and add them to the limited set of transactions. Lines 4–9 process the output of the transaction. Line 5 adds every input in the transaction to the limited set of the input place. Line 6 adds the input directed arc to the limited set of directed arcs between the place and the transition. Line 7 sets up the mapped address of γ (p), which can be obtained from the ScriptPubKey field of output in blockchain. Line 8 adds the address a to the limited set of input addresses.

5.2. *Incremental clustering*

Algorithm 2 below describes the process of incremental clustering.

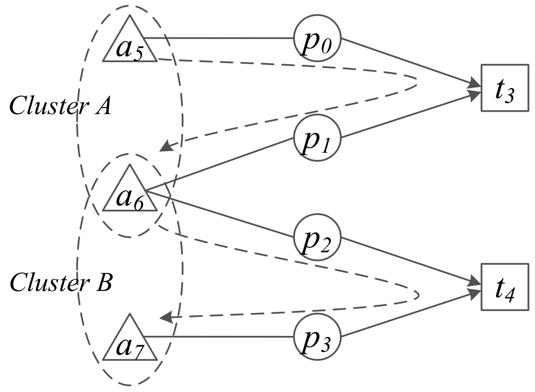

Fig. 5. Before updating the cluster.

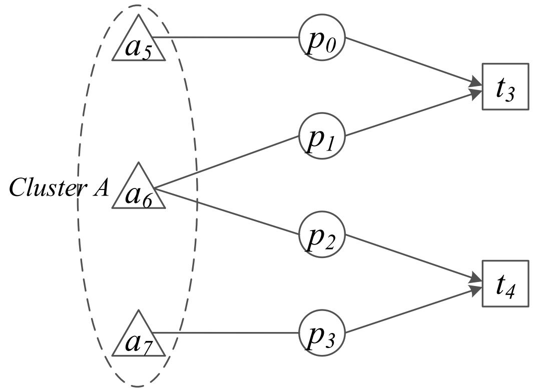

Fig. 6. After updating the cluster.

Algorithm 2. Address incremental clustering

Input: A set of unclustered address.
Output: Address with the cluster *ID* and the cluster *ID* change information.

```
1.  For each address
2.    If address is unclustered
3.      Find the cluster of the address;
4.      Find cluster ID of the cluster;
5.      Assign the cluster ID to the address whose cluster ID
        does not equal to the cluster ID and sign the address
        whose cluster ID is changed;
6.  End For
```

Line 1 traverses the collection and removes the addresses contained in the set sequentially. Line 2 judges whether the retrieved address has been clustered. Lines 3–5 find and confirm the cluster *ID*, assign it to the address in the cluster *ID* and mark the address where the cluster *ID* has changed. If the address is not clustered, we call the address cluster discovery algorithm to find the class of the address.

Algorithm 3 below describes the process of address cluster discovery in Address Increment Clustering based on Multi-input Transactions (AICMT). The principle of this algorithm is similar to the Breadth-First-Search (BFS). It also starts from one vertex (address), finds the vertices mapped by other places through the relationship between places and transitions, pushes them into the end of the queue, and processes the vertices in the queue in the same way. Flagging the clustered places can avoid repeated queueing and improve efficiency.

Algorithm 3. Address cluster discovery

Input: An address *a*.
Output: A cluster of addresses

```
1.  Setup an address set $S$;
2.  Setup an address queue $Q$;
3.  Put the address into $S$;
4.  $Q \rightarrow push(a)$;
5.  While $Q \neq \varnothing$
6.    $Q \rightarrow pop$;
7.    For each ${}^{\circ}a$
8.      If $p_i \in {}^{\circ}a$ is not clustered
9.        Find $t \in p_i^{\square}$
10.       If $t \neq \varnothing$
11.         For each ${}^{\circ}t$
12.           If $p_j \in {}^{\circ}t$ is not clustered
13.             Find $a' = \gamma(p_j)$;
14.             If $a' \in S$
15.               Add the address into $S$;
16.               $Q \rightarrow push(a')$;
17.             End If
18.           End If
19.         End For
20.         Setup $p_j$ as clustered;
21.       End If
22.     End If
23.   End For
24. End While
```

Lines 1–4 set up an address set *S* and an address queue *Q*, put the address *a* into the set and push the address *a* into the queue. Lines 5–7 judge whether the address queue is empty, and if the address queue is empty, then output the address cluster. Otherwise, dequeue the address, then take out the output point of the address in turn. Lines 8–9 judge whether the output point *p* is clustered, and if the output point *p* is not clustered, then find the transaction *t* that uses the output point *p* as the input. Lines 10–13 judge whether the transaction *t* can be found, and if the transaction *t* can be found, then take it out in turn as the input of the transaction *t*. If the output point corresponding to the input is not clustered, then find the address of the output point mapping. Line 14–16 judge whether the address set includes address *a'*, and if the address set does not include *a'*, then add this address to the address set, and put *a* ' into the queue Q.

From Algorithm 2 and Algorithm 3, the total process of AICMT is: read the previously clustered address cluster information, then enter the address set to be clustered, and then take out the addresses in the set

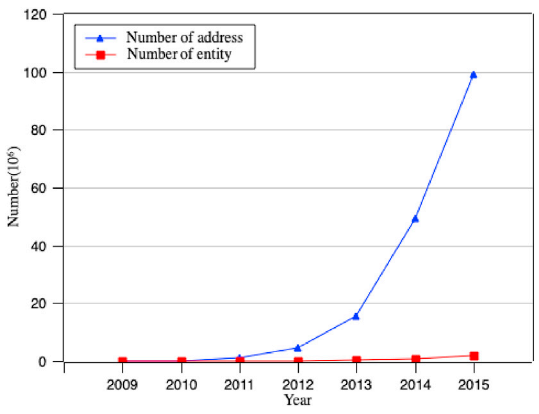

Fig. 7. Number change of entity and address.

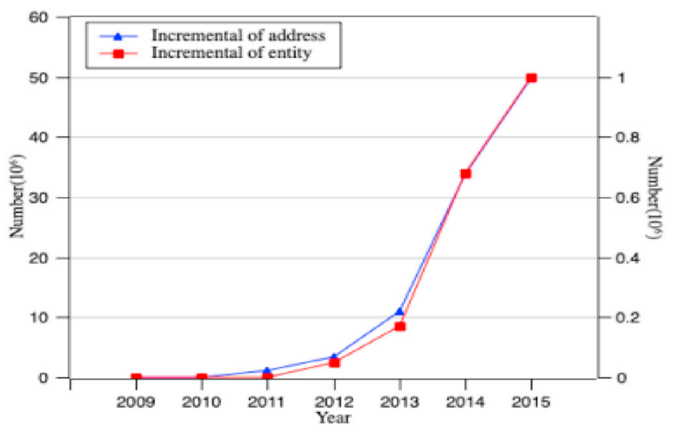

Fig. 8. Incremental change of address and entity.

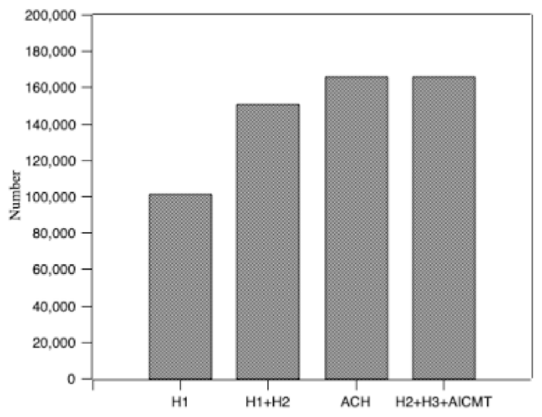

Fig. 9. The results of the four cluster methods.

sequentially from the set. When the address set is empty, the address with cluster *ID* and cluster *ID* change information will be output, otherwise the addresses are taken out from the set sequentially. If the address has been clustered, then continue to read the previous address cluster information; otherwise, set up an address set and an address queue, put the retrieved address a' into the set, and enqueue the address. When the address queue is empty, the address cluster is output, otherwise dequeue the address, and then the output point p' of the address is taken out sequentially. If the output points are not clustered, then look for transaction t' with p' as input. If the transaction can be found, then take the input as t' in turn. If the output point corresponding to the input is not clustered, then look for the address a'' mapped to the output point. If the address set does not include a'', then add this address to the address set, put a'' in the queue, and set the output point as clustered. Then, find and confirm the *ID* of the cluster, assign the *ID* to the address in the cluster, mark the address where the cluster *ID* has changed, and finally save the newly changed cluster

Table 2
Correctness analysis.

Test entity instance	WalletExplorer data/ number	Method of this article/ number
SilkRoadMarketplace	165764	165764
Satoshidice	9961	9961
AlphaBayMarket	79882	79882
Huobi.com	2132	2132
BitPay.com-old	40326	40326
AnxPro.com	55062	55062
BitcoinFog	54439	54439
SilkRoad2Market	15657	15657
MyBitcoin.com	4824	4824
Bitfinex.com-old	1417	1417

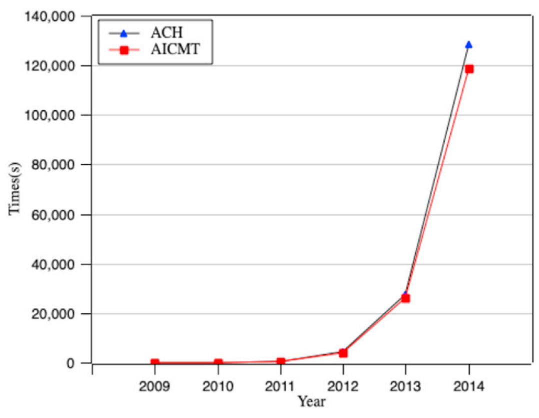

Fig. 10. Time comparison of clustering.

information.

According to AICMT, we can simulate the incremental clustering process as shown in Fig. 3. In Fig. 5, the places p_0 and p_1 are the input places of transaction t_3. Then, the addresses a_5 and a_6 mapped by the places p_0 and p_1 respectively belong to *Cluster A*. Similarly, the places of p_2 and p_3 are the input places of transaction t_4. Then the addresses a_6 and a_7 mapped by places p_2 and p_3 respectively belong to *Cluster B*. In Fig. 6 , assign the *ID* of *Cluster A* to the address in *Cluster B*, and then save the newly changed cluster information, so the addresses a_5, a_6 and a_7 belong to *Cluster A*.

For changing addresses that are input to other transactions, incremental clustering algorithms based on multi-input address clustering can also searched. Therefore, in this case, the results of AICMT in this paper will be consistent with the results of both the multi-input address clustering algorithm and the changing address clustering algorithm.

6. Experiment

6.1. Dataset

The address data of the clustering in this paper is mainly obtained by analyzing the P2PKH address and the P2SH address. The P2WPKH address and P2WSH address appeared at the end of 2017; therefore, this article used the data of the first 391,134 blocks in the Bitcoin network as the experimental data source, and the data covered the period from 2009 to 2015. The specific collection time period is from 2009-01-04 02:15:05 to 2015-12-31 23:59:26.

6.2. Analysis of clustering results

The experiment deals with the clustering transactions that meet the conditions of the data set. Fig. 7 shows the changes in the numbers of transaction entities and address over time. From Fig. 7, it can be seen that the number of addresses increased relatively slowly from 2009 to 2012, and the number of addresses increased significantly from 2012 to 2014. One of the most obvious phenomena that can be observed in Fig. 8 is that the growth rate of Bitcoin addresses is roughly the same as that of Bitcoin entities. We discovered that the growth rate in 2009 and 2012 was relatively slow, and the growth rate increased significantly from 2013 to 2014. The possible reasons for this situation are: a large number of online transaction services make it more convenient for individuals to contact Bitcoin, which determines rapid growth of Bitcoin users and the rapidly increasing number of Bitcoin addresses. The growth rate became relatively slow from 2014 to 2015. This is because in 2013, the main exchange Mt.Gox was stolen and declared bankrupt, which became the biggest crisis of confidence since the birth of Bitcoin, and Bitcoin investors became conservative.

6.3. Analysis of performance

Taking the address "113rNQk7QQnQdAKxkh5CjrSfxHD
qANohE1" in SilkRoadMarketplace as an example, this experiment uses four address clustering methods to cluster its address data up to the end of 2013. Fig. 9 shows the numbers of Bitcoin addresses in the obtained address cluster are 101121, 150852, 165764 and 165764 respectively. Compared with ACH (address clustering based on three heuristic algorithms) [9], address clustering with AICMT has the same accuracy.

By comparing with the results of Walletexplorer.com website clustering, the method in this paper can achieve 100% accuracy in entity clustering. The following Table 2 is a comparison of the numbers of three physical addresses selected as of 2013.

Fig. 10 shows the comparison between the traditional method and AICMT within the whole process of clustering.

In the early stage of Bitcoin Blockchain, the traditional method costed the same time as the proposed method. However, as the numbers of blocks and addresses increase over time, the proposed AICMT shows its advantages with lower time consumption.

7. Conclusions

According to the characteristics of Bitcoin transactions, this paper uses Petri nets to model the Bitcoin transaction network. Based on this model, a Bitcoin addresses incremental clustering algorithm based on multiple input transactions is proposed. The method first describes the Bitcoin transaction network formally using Petri nets, and then the Bitcoin addresses in the transaction network are clustered incrementally. Compared with the existing clustering methods, the proposed address incremental clustering algorithm improves clustering efficiency. The next step in our research is to combine off-chain information for more accurate address clustering.

Declaration of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The work reported in this paper has been partially supported by the National Key Research and Development Project (2020YFB1005503), the NSFC Projects (61502209 and U1836116), the Leading-edge Technology Program of Jiangsu Natural Science Foundation (BK20202001), the NSFC of Jiangsu Province Project (BK20201415) and the UK-Jiangsu 20-20 World Class University Initiative programme, and the Natural Science Foundation of the Jiangsu Higher Education Institutions (Grant number: 22KJB520016).

References

[1] Q. Liu, Y. Xu, B. Cao, L. Zhang, M. Peng, Unintentional forking analysis in wireless blockchain networks, Digital Commun. Networks 7 (2021) 335–341.
[2] B. Cao, Z. Zhang, D. Feng, S. Zhang, L. Zhang, M. Peng, Y. Li, Performance analysis and comparison of PoW, PoS and DAG based blockchains, Digital Commun. Networks 6 (2020) 480–485.
[3] S. Nakamoto, Bitcoin: a Peer-To-Peer Electronic Cash System, 2009. https://bitcoin.org/bitcoin.pdf. (Accessed 02 March 2022).
[4] Mt Wikipedia, Gox, 2020. https://en.wikipedia.org/wiki/Mt._Gox. (Accessed 14 March 2022).
[5] K. Nilsson, The Missing MtGox Bitcoins, WizSec, 2015. April, 19.
[6] Wikipedia, Bitfinex, 2020. https://en.wikipedia.org/wiki/Bitfinex. (Accessed 14 March 2022).
[7] F. Reid, M. Harrigan, An analysis of anonymity in the bitcoin system, in: Y. Altshuler, Y. Elovici, A. Cremers, N. Aharony, A. Pentland (Eds.), Security and Privacy in Social Networks, Springer, New York, 2013, pp. 197–223.
[8] D. Ron, A. Shamir, Quantitative analysis of the full bitcoin transaction graph, in: Proceedings of the 2013 International Conference on Financial Cryptography and Data Security, Springer, 2013, pp. 6–24.
[9] M. Fleder, M.S. Kester, S. Pillai, Bitcoin Transaction Graph Analysis, 2015, pp. 1–8, arXiv preprint arXiv:1502.01657.
[10] E. Androulaki, G.O. Karame, M. Roeschlin, T. Scherer, S. Capkun, Evaluating user privacy in bitcoin, in: Proceedings of the 2013 International Conference on Financial Cryptography and Data Security, Springer, 2013, pp. 34–51.
[11] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G.M. Voelker, S. Savage, A fistful of bitcoins: characterizing payments among men with no names, in: Proceedings of the 2013 Conference on Internet Measurement Conference, 2013, pp. 127–140.
[12] M. Harrigan, C. Fretter, The unreasonable effectiveness of address clustering, in: Proceedings of the 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), IEEE, 2016, pp. 368–373.
[13] M. Hong-liang, W. Zhen, H. Min, T. Ji-qiang, S. Meng, Heuristic approaches based clustering of bitcoin addresses, J. Beijing Univ. Posts Telecommun. 41 (2) (2018) 27–31.
[14] D. Ermilov, M. Panov, Y. Yanovich, Automatic bitcoin address clustering, in: Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2017, pp. 461–466.
[15] T. Neudecker, H. Hartenstein, Could network information facilitate address clustering in Bitcoin?, in: Proceedings of the 2017 International Conference on Financial Cryptography and Data Security Springer, 2017, pp. 155–169.
[16] M. Spagnuolo, F. Maggi, S. Zanero, Bitiodine: extracting intelligence from the bitcoin network, in: Proceedings of the 2014 International Conference on Financial Cryptography and Data Security, Springer, 2014, pp. 457–468.
[17] Z. Zhang, T. Zhou, Z. Xie, Bitscope: scaling bitcoin address de-anonymization using multi-resolution clustering, in: Proceedings of the 51st Hawaii International Conference on System Sciences, 2018, pp. 1–11.
[18] C. Kinkeldey, J.-D. Fekete, P. Isenberg, Bitconduite: visualizing and analyzing activity on the bitcoin network, in: Proceedings of the EuroVis 2017-Eurographics Conference on Visualization, Posters Track, 2017, pp. 1–3.
[19] M. Möser, R. Böhme, D. Breuker, An inquiry into money laundering tools in the Bitcoin ecosystem, in: Proceedings of the 2013 APWG eCrime Researchers Summit, Ieee, 2013, pp. 1–14.
[20] C. Zhao, Y. Guan, A graph-based investigation of bitcoin transactions, in: Proceedings of the 2015 IFIP International Conference on Digital Forensics, Springer, 2015, pp. 79–95.
[21] D.D.F. Maesa, A. Marino, L. Ricci, Uncovering the bitcoin blockchain: an analysis of the full users graph, in: Proceedings of the 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), IEEE, 2016, pp. 537–546.
[22] D. Di Francesco Maesa, A. Marino, L. Ricci, Data-driven analysis of bitcoin properties: exploiting the users graph, Int. J. Data Sci. Anal. 6 (2018) 63–80.
[23] M. Jourdan, S. Blandin, L. Wynter, P. Deshpande, Characterizing entities in the bitcoin blockchain, in: Proceedings of the 2018 IEEE International Conference on Data Mining Workshops (ICDMW), IEEE, 2018, pp. 55–62.
[24] J.V. Monaco, Identifying bitcoin users by transaction behavior, in: Proceedings of the 2015 Biometric and Surveillance Technology for Human and Activity Identification XII, SPIE, 2015, pp. 25–39.
[25] A. Pinna, R. Tonelli, M. Orrú, M. Marchesi, A petri nets model for blockchain analysis, Comput. J. 61 (2018) 1374–1388.
[26] A. Pinna, A petri net-based model for investigating disposable addresses in bitcoin system, in: Proceedings of the 2016 KDWeb, 2016, pp. 1–4.
[27] Y. Wu, F. Tao, L. Liu, J. Gu, J. Panneerselvam, R. Zhu, M.N. Shahzad, A bitcoin transaction network analytic method for future blockchain forensic investigation, IEEE Trans. Network Sci. Eng. 8 (2020) 1230–1241.
[28] Y. Wu, A. Luo, D. Xu, Identifying suspicious addresses in Bitcoin thefts, Digit. Invest. 31 (2019) 1–12.
[29] Y. Wu, A. Luo, D. Xu, Forensic analysis of bitcoin transactions, in: Proceedings of the 2019 IEEE International Conference on Intelligence and Security Informatics (ISI), IEEE, 2019, pp. 167–169.
[30] G. Maxwell, CoinJoin: Bitcoin Privacy for the Real World, 2013. https://bitcointalk.org/index.php?topic=279249.0. (Accessed 28 March 2022).