Journal of Big Data

RESEARCH Open Access

Defining user spectra to classify Ethereum users based on their behavior

Gianluca Bonifazi, Enrico Corradini, Domenico Ursino* and Luca Virgili

*Correspondence:
d.ursino@univpm.it
Department of Information Engineering, Polytechnic University of Marche, Ancona, Italy

Abstract

Purpose: In this paper, we define the concept of user spectrum and adopt it to classify Ethereum users based on their behavior.

Design/methodology/approach: Given a time period, our approach associates each user with a spectrum showing the trend of some behavioral features obtained from a social network-based representation of Ethereum. Each class of users has its own spectrum, obtained by averaging the spectra of its users. In order to evaluate the similarity between the spectrum of a class and the one of a user, we propose a tailored similarity measure obtained by adapting to this context some general measures provided in the past. Finally, we test our approach on a dataset of Ethereum transactions.

Findings: We define a social network-based model to represent Ethereum. We also define a spectrum for a user and a class of users (i.e., token contract, exchange, bancor and uniswap), consisting of suitable multivariate time series. Furthermore, we propose an approach to classify new users. The core of this approach is a metric capable of measuring the similarity degree between the spectrum of a user and the one of a class of users. This metric is obtained by adapting the Eros distance (i.e., Extended Frobenius Norm) to this scenario.

Originality/value: This paper introduces the concept of spectrum of a user and a class of users, which is new for blockchains. Differently from past models, which represented user behavior by means of univariate time series, the user spectrum here proposed exploits multivariate time series. Moreover, this paper shows that the original Eros distance does not return satisfactory results when applied to user and class spectra, and proposes a modified version of it, tailored to the reference scenario, which reaches a very high accuracy. Finally, it adopts spectra and the modified Eros distance to classify Ethereum users based on their past behavior. Currently, no multi-class automatic classification approach tailored to Ethereum exists yet, albeit some single-class ones have been recently proposed. Therefore, the only way to classify users in Ethereum are online services (e.g., Etherscan), where users are classified after a request from them. However, the fraction of users thus classified is low. To address this issue, we present an automatic approach for a multi-class classification of Ethereum users based on their past behavior.

Keywords: Blockchain, Classification algorithm, Ethereum, User spectrum, Multivariate Time Series, Eros distance, Extended Frobenius Norm, Etherscan

Introduction

In recent years, we have witnessed an impressive development of the blockchain technology [1]. It was initially introduced by Satoshi Nakamoto [2] to support the development of the cryptocurrency Bitcoin [3]. Later, smart contracts were introduced in Ethereum[1] and this technology has spread to a variety of applications in the financial sector. Finally, it is now starting to be adopted in an increasing number of sectors.

Anyone can participate to a blockchain network; therefore, different actors can be identified in this ecosystem [4]. For example, if we consider a blockchain like Ethereum, some actors (called miners) maintain the blockchain network, while others allow users to trade different cryptocurrencies and/or make banking transactions. Some others deal with auctions, others offer games or services, and so on. In some cases, there are online systems that provide a classification of the users of a blockchain network, even if the fraction of classified users is very small. The most known of such systems is Etherscan[2], which provides this service for Ethereum. Through Etherscan, the developer of a smart contract can publish the corresponding code and request verification. Etherscan performs such a task and, if positive, also provides a categorization of the corresponding user[3].

Knowing a user's category can be extremely relevant in the context of blockchain networks [5, 6]. For example, such a knowledge allows us to find a set of competitors of a user performing a certain activity (exchange, bancor, etc.). In addition, through appropriate analyses, it is possible to identify whether, within a category, there are backbones of users connected to each other to avoid competing with one another or to gain dominant positions over others. Again, thanks to even more complex analysis, it is possible to understand the different strategies carried out by users of the same category and which of them is the winning one.

Despite the importance of this knowledge, in the past literature, there exist very few approaches that, given a user of a blockchain network, can automatically derive her category [6–11]. Furthermore, the few categorization approaches currently existing are usually tailored to the Bitcoin blockchain, while general ones have been tested on small specific blockchains. As for Ethereum, several approaches to identify users belonging to a certain category of interest have been proposed in the past. Instead, to the best of our knowledge, no tailored classification approach, like the ones presented for Bitcoin, have been proposed for Ethereum. As a consequence, the only current way to classify users in this blockchain is based on the activity of providers of this service, like Etherscan. However, they can classify at most those users who submit their smart contracts to them for verification. Unfortunately, such users are only a small minority of those present in Ethereum.

In this paper, we aim at filling this gap by proposing an automatic approach for classifying users in Ethereum. The starting point of our approach is that each Ethereum user has an address in order to carry out her activities. It is an alphanumeric code allowing a

[1] https://ethereum.org/.

[2] https://etherscan.io/.

[3] https://etherscan.io/labelcloud.

user to be identified in the blockchain network and to carry out transactions with other users[4]. All the transactions made by a user in a certain time period allow us to reconstruct, at least partially, her behavior in that period.

More specifically, in order to define user behaviors in a certain time interval, our approach first builds a social network representing the users involved in Ethereum and their transactions. Then, starting from this social network, it defines and computes a set of features for each user. They are the number of incoming and outgoing arcs of the node corresponding to the user, the number of incoming and outgoing transactions, the amount of incoming and outgoing money (expressed in Ether, the Ethereum cryptocurrency), the clustering coefficient and the PageRank. The values of these features can change over time. Given a time period T and a user u_j, we call the spectrum of u_j in T the set of time series expressing the values of the features for u_j in T. The spectrum of u_j provides a concise, but accurate, picture of the behavior of u_j during T.

Having a spectrum for each user might lead to think that categorizing users is a simple task. In fact, in principle, one could build a spectrum for each class starting from the spectra of the users belonging to it, identified from the training data. At this point, given a new user, whose spectrum is known, she could be assigned to the class with the spectrum most similar to her own. Although this procedure seems simple at an abstract level, it is much more complex in reality. In fact, we have seen that the spectrum of a user (and, consequently, the spectrum of a class) consists of a set of time series, one for each feature. As a consequence, it is necessary to define a similarity measure between two sets of time series. Furthermore, the various features are not totally independent of each other. In fact, as we will see later, a correlation study on them showed us that some features are totally or partially correlated. Therefore, the spectrum of a user must be managed as a multivariate time series.

As a consequence, we must face a classification problem in which each element to classify and each available class are represented by multivariate time series. To the best of our knowledge, there is no out-of-the-box classification algorithm with these characteristics. Thus, it is necessary to define a new one. The core of such an algorithm consists of a metric capable of measuring the similarity degree between two multivariate time series (which, in our case, are the spectrum of the user to be classified and the spectrum of each class). Several metrics proposed for this purpose exist in the literature. Among them, we mention the Dynamic Time Warping [12], the Weighted Sum SVD [13], and the Eros distance, also known as Extended Frobenius Norm [14]. The latter has been shown to outperform the other more traditional metrics [14]. Hence, it would represent the natural choice in our case. Unfortunately, as we will see in "Evaluation" section, the results obtained by applying the Eros distance to our reference scenario were not satisfactory. However, we managed to define a variant of it. Even if more expensive in terms of computation time (albeit, as we shall see, these costs are largely acceptable), this variant achieves a very high classification accuracy. It represents the core of our classification approach and will be described in detail in this paper, from both a theoretical and an experimental point of view.

[4] In this paper, we do not consider the case in which different Ethereum addresses are handled by the same user. When this happens, we investigate the addresses separately because we assume that the corresponding user could deliberately show different behaviors in the different addresses.

A more formal definition of the proposed approach involves structuring it into the following steps:

- Construction of the support social network starting from the training set.
- Construction of the spectrum of each user from the data about her behavior stored in the dataset and the social network built in the previous step.
- Selection of the classes of interest.
- Construction of the spectrum of each class from the spectra of the corresponding users.
- Definition of a new version of the Eros distance tailored to our scenario.
- For each new user:
  - Computation of the Eros distance between her spectrum and the one of each class.
  - Assignment of the user to the nearest class (or to no class, if her spectrum is very far from the ones of all classes) based on the values of the Eros distance computed in the previous step.

We highlight that our approach, albeit designed to classify Ethereum users, can be easily extended to other blockchains. In fact, as we will see below, the social network-based representation of a blockchain, the definition of the spectrum of a user or a class, and the classification algorithm are characterized by a high abstraction level. Therefore, they can be easily applied to many different blockchains.

In summary, the main contributions of this paper are as follows: (i) we define a social-network based model to represent Ethereum; (ii) we introduce the concept of spectrum of a user or a class of users; (iii) we propose a multivariate representation, instead of a univariate one, of a user's behavior; (iv) we introduce a modified version of the Eros distance to measure the distances between spectra; (v) we propose an automatic multi-class algorithm (instead of the single-class existing ones) for classifying Ethereum users based on their past behavior.

The outline of this paper is as follows: In "Related literature" section, we present the related literature. Then, in "Proposed method" section, we provide the description of the proposed approach. In "Experiments" section, we present some experiments aimed to perform an Exploratory Data Analysis on our dataset and tune our approach. In "Evaluation" section, we present the tasks we performed to evaluate our approach and the results obtained. In "Discussion" section, we highlight the strength of our approach, compared to the current state of the art described in detail in "Related literature" section . Finally, in "Conclusion" section, we draw our conclusions and highlight some possible future developments of our research.

Related literature

After the introduction of Bitcoin in 2008 [2], many cryptocurrencies have been created and have spread [15]. This prompted researchers to investigate both the development of this phenomenon and the issues related to it [16–21]. Indeed, while the growth of cryptocurrencies has opened new opportunities, it also led to new challenges to face and several problems to overcome.

As a matter of fact, malicious users have found in cryptocurrencies new opportunities for profit by deceiving newcomers [22], thanks also to the fact that blockchains guarantee a certain degree of anonymity [23, 24]. Many researchers have proposed approaches to detect frauds, scams and, generally, illegal transactions on several cryptocurrencies, such as Bitcoin and Ethereum [25–28]. Other ones have focused on tracking accounts and people, or groups of people, who performed these illegal acts [29–31]. This last challenging issue has paved the way to the more general problem of classifying and characterizing accounts, addresses and smart contracts in a blockchain [32, 33].

As for this topic, the authors of [7] propose to characterize an entity in the Bitcoin blockchain by analyzing information revealed by the patterns of the transactions made by its neighbors. Here, the term "entity" is used to denote the set of the addresses of a single user. This way of proceeding is motivated by the fact that a user can have associated several addresses in the Bitcoin blockchain. The approach of [7] models the Bitcoin blockchain as a directed weighted bipartite graph. Using the WalletExplorer website, the authors of [7] obtain a final labeled dataset with 30,331,700 addresses, associated with 272 entities. These are divided into five categories, namely Exchange, Service, Gambling, Mining Pool and DarkNet Marketplace. Classification is performed using 315 features belonging to five different categories, namely address, entity, time, centrality and motif. This approach achieves an overall accuracy of 0.85 with the Logistic Regression classifier and 0.92 with the LightGBM one.

The authors of [6, 8] propose a multi-class service identification of Bitcoin addresses based on a summarization of transaction history. Specifically, the authors of [6] consider eight parameters to perform this task. Using WalletExplorer and Blockchain.info, they identify seven types of Bitcoin-enabled services, along with a set of more than 26,000 addresses associated with them. Starting from this training set, they achieve a classification accuracy of 0.70 (resp., 0.72) in the address-based (resp., owner-based) scheme using a random forest classifier. The authors of [8] start from the approach proposed in [6] but add two more parameters to support classification. Using a dataset of 13 million transactions, they evaluate the new set of features with eight different classifiers. Proceeding in this way, they manage to improve the results obtained in [6]. In fact, they achieve a Micro F1 score equal to 0.87 and a Macro F1 score equal to 0.86 with the LightGBM classifier.

The authors of [9] present a new approach to decrease the anonymity of Bitcoin through entity characterization based on a cascade of machine learning models. This approach uses data on entities, addresses and motifs as classification features. The simultaneous usage of several machine learning models, each inserted several times in a cascade, allows the authors to reach a very high global accuracy, equal to 0.9968. This result is obtained after an appropriate training of all the models involved that, de-facto, tailors the overall model on the training data. The testing campaign was performed using the WalletExplorer dataset and the Gradient Boosting model.

The authors of [34] propose an approach focused on the detection of entities belonging to a single class, i.e., Exchange. First, they model the Bitcoin blockchain as a directed hypergraph. Then, they use this hypergraph to build classification models capable of detecting a set of discriminating features. Finally, they employ these features to decide whether an entity belongs to the Exchange class. The accuracy achieved by this approach is equal to 0.80, which is lower than that of other ones. However, this approach has the important advantage of exploiting only purely structural features of the hypergraph.

Finally, in [10, 11, 35], the authors propose two different methods that perform classification and clustering of addresses in a blockchain starting from the behavior of the corresponding users. In particular, the authors of [11] propose a deep learning based classification method called PeerClassifier. Instead, those of [10] propose a clustering method that uses the Dynamic Time Warping similarity measure applied to two sequences represented as two univariate time series. In both cases, the experimental campaign is conducted on a real blockchain operating on stock trading.

In the past literature, there are some Etherscan-based approaches to classify Ethereum users in a multi-class hierarchy. Furthermore, there are few automatic approaches that aim at identifying addresses belonging to a specific category of Ethereum users [5, 36–41]. All of them, are single-class, i.e., they were conceived to identify users belonging to a certain class. For example, the authors of [36] (similarly to [42–44]) propose an approach to store Ethereum data in a graph database in order to carry out analyses on it. They derive the data of interest from Etherscan and create a hierarchy representing addresses and their transactions. The authors of [37] propose a methodology for labeling the addresses of cryptocurrencies. First of all, they classify cryptocurrencies in two groups. The former includes those with an Unspent Transaction Output, such as Bitcoin and Litecoin[5]. The latter comprises account-based cryptocurrencies, such as Ethereum and EOS[6]. After this, based on their experience in the field (they work at Binance.com), they propose an approach to label addresses of the first and second cryptocurrency type and verify it on Bitcoin and Ethereum. The authors of [5, 39, 45] propose an approach to detect phishing accounts in Ethereum. It first collects the data of interest from Etherscan and uses this data to build an Ethereum transaction network. Then, it applies a network embedding method to extract latent features of the accounts performing phishing activities. Finally, it uses these features to train a one-class Support Vector Machine. In [38], the author proposes an approach to cluster Ethereum addresses in order to identify entities controlling multiple addresses. The clustering task is done considering the following features: deposit address, multiple participation in airdrops and token authorization mechanisms. The author shows that his approach can cluster 17.9% of all active externally owned account addresses. He also finds that there are more than 340,000 entities that likely control multiple addresses. The authors of [40] propose an approach to detect Ponzi schemes implemented as smart contracts in Ethereum (also called "smart Ponzi schemes"). First, they manually identify 200 smart Ponzi schemes in Ethereum. Then, starting from the analysis of these schemes, they extract features to recognize smart Ponzi schemes. Finally, they use the extracted features to identify new smart Ponzi schemes. They show that their approach achieves a very good accuracy and estimate that there are at least 500 smart Ponzi schemes running on Ethereum.

[5] https://litecoin.org/.

[6] https://eos.io/.

Proposed method

In this section, we present our approach. As mentioned in the Introduction, it consists of several steps, each introducing innovations with respect to the corresponding tasks proposed in the past. More specifically, the outline of our approach is as follows:

1. Construction of a social network supporting the representation of a training set concerning Ethereum users and their behavior.
2. Construction of the spectrum of users of the training set from their data stored in the dataset and some metrics computed on the social network built at Step 1.
3. Selection of the classes of interest. These are presumably the ones most prevalent in the dataset and, thus, in Ethereum. However, if we want to focus on one or more uncommon classes (e.g., for studying an outlier class), we can do it.
4. Construction of the spectrum of each class selected at Step 3 starting from the spectra of the users of the training set associated with that class.
5. Definition of a new version of the Eros distance tailored to our scenario and computation of the corresponding weights starting from the dataset.
6. For each user to be classified (whether she belongs to the test set or is a new user of whom nothing is known):

    (a) Construction of the corresponding spectrum.
    (b) Computation of the Eros distance between the spectrum built at Step 6(a) and the class spectra built at Step 4.
    (c) Assignment of the user to the nearest class according to the values of the Eros distance computed at Step 6(b). Otherwise, assignment of the user to no class if the Eros distance between her spectrum and that of all available classes is higher than a certain threshold.

In the next subsections, we describe the various steps of our approach in detail.

Modeling a blockchain as a social network

A blockchain can be modeled through a social network in a very direct way. In fact, the social network nodes can represent the blockchain addresses, while its arcs can denote the transactions between the addresses corresponding to the involved nodes. The capability of building such a model for a blockchain leads to the possibility of extracting knowledge about the behavior of blockchain actors by employing the Social Network Analysis based techniques proposed in the past [46–48]. In the following, we show this property taking Ethereum as the reference blockchain because it is the blockchain of interest for this paper. However, we point out again that our approach to build and characterize a social network from a blockchain (and, consequently, the next classification approach representing the core of this paper) can be applied to most blockchains. Indeed, the features used to model Ethereum as a social network (such as the sender, receiver and timestamp of a transaction, and the amount of transferred money) are also present in many other blockchains, like Bitcoin, Litecoin, and so on.

After this necessary preliminary remark, we can now see how a social network $\mathcal{G}$, representing the Ethereum blockchain, can be built. Specifically:

$$\mathcal{G} = \langle N, A \rangle$$

Here, N is the set of nodes of $\mathcal{G}$. A node $n \in N$ corresponds to an Ethereum address that has made at least one transaction. Since there is a biunivocal correspondence between a node of $\mathcal{G}$ and an Ethereum address, in the following we will use these two terms interchangeably. Each node n has associated a label l_n, indicating the class which it belongs to (see below); l_n is set to null if no class has been assigned to n yet.

A represents the set of arcs of $\mathcal{G}$. There is an arc $a = (n_i, n_j, TrS_{ij}) \in A$ if there was at least one transaction from n_i to n_j. TrS_{ij} consists of a set of triplets $(tr_{ij_k}, \tau_{ij_k}, v_{ij_k})$, where tr_{ij_k} represents the k^{th} transaction from n_i to n_j, τ_{ij_k} indicates the corresponding timestamp and v_{ij_k} denotes the amount of Wei[7] transferred from n_i to n_j through tr_{ij_k}.

Modeling Ethereum as a social network allows us to use various Social Network Analysis measures to characterize each Ethereum address. In particular, we chose a set F of features that can support in distinguishing one class from another. They are:

- `In-degree`: it represents the number of arcs incoming to n_i and, therefore, the number of nodes of $\mathcal{G}$ pointing to n_i. It can be determined by computing the cardinality of the set:

  $$IN_i = \{n_j | (n_j, n_i, TrS_{ji}) \in A\}$$

- `Out-degree`: it denotes the number of arcs outgoing from n_i and, therefore, the number of nodes of $\mathcal{G}$ which n_i points to. It can be determined by computing the cardinality of the set:

  $$OUT_i = \{n_j | (n_i, n_j, TrS_{ij}) \in A\}$$

- `In-transaction`: it indicates the number of transactions towards n_i made by the nodes of $\mathcal{G}$. It can be computed as:

  $$\sum_{n_j \in IN_i} |TrS_{ji}|$$

  where $|TrS_{ji}|$ denotes the cardinality of the set TrS_{ji}.
- `Out-transaction`: it represents the number of transactions towards the nodes of $\mathcal{G}$ made by n_i. It can be computed as:

  $$\sum_{n_j \in OUT_i} |TrS_{ij}|$$

- `In-value`: it denotes the total amount of Wei received by n_i. It can be computed as:

  $$\sum_{n_j \in IN_i} \sum_{k=1..|TrS_{ji}|} v_{ji_k}$$

- `Out-value`: it indicates the total amount of Wei sent by n_i. It can be computed as:

[7] Wei is the smallest fraction of Ether; it corresponds to 10^{-18} Ethers.

$$\sum_{n_j \in OUT_i} \sum_{k=1..|TrS_{ij}|} v_{ij_k}$$

- `Clustering-coefficient`: it represents the clustering coefficient of n_i. Recall that, in Social Network Analysis, this parameter is an indicator of the tendency of n_i and its neighbors to form a cluster.
- `PageRank`: it denotes the PageRank of n_i. This parameter is an indicator of the number of links received by n_i, the centrality of the neighbors of n_i and their propensity to link to each other [49].

In our reference scenario, the time factor plays a key role. As a consequence, our model should take time into account. In fact, users continuously make transactions on Ethereum, which leads to continuous changes in the structure of the corresponding social network and the labels of its arcs.

In order to take time into consideration, given a time instant t, we denote with $\mathcal{G}(t)$ the social network associated with Ethereum that considers the transactions made on that blockchain from its appearance until t and, therefore, the transactions whose timestamp is less than or equal to t.

Similarly, given two time instants t_α and t_β, we can build a social network $\mathcal{G}(t_\alpha, t_\beta)$ representing Ethereum, and the transactions made on it, in the time interval $(t_\alpha, t_\beta]$. More formally, $\mathcal{G}(t_\alpha, t_\beta)$ considers only the transactions on Ethereum such that the corresponding timestamp is higher than t_α and less than or equal to t_β.

Defining the spectrum of a user or a class of users

We have introduced the eight features able to characterize an Ethereum address and we have presented the social network $\mathcal{G}(t_\alpha, t_\beta)$, modeling Ethereum in the time interval $(t_\alpha, t_\beta]$. We are now able to define the concept of spectrum of an Ethereum address in $(t_\alpha, t_\beta]$.

Let F be the set of features introduced in the previous section and let $T = (t_\alpha, t_\beta]$ be a time interval. We assume that T consists of a certain number of days. Let d_h be the h^{th} day of T. T can be represented as a succession $T = \{d_{\alpha+1} = d_1, d_2, \cdots, d_h, \cdots, d_q = d_\beta\}$ of q days. Let f_p be a parameter of F. It can have associated a time series $\Phi_p = \{\phi_{p_1}, \phi_{p_2}, \cdots, \phi_{p_h}, \cdots, \phi_{p_q}\}$, where ϕ_{p_h} is the value assumed by f_p at a constant and default time of d_h (for instance, at 12:00 am).

We define the spectrum $\mathcal{S}_i^T$ of a node n_i in the time interval T as the set $\mathcal{S}_i^T = \{\phi_{p_i} | f_p \in F$ and ϕ_{p_i} is the succession of the values assumed by f_p in n_i during $T\}$. In other words, the spectrum of n_i in T is given by a set of successions, one for each feature of F. Each succession is made of the values assumed by the corresponding feature for the Ethereum address associated with n_i for the days belonging to T.

The spectrum $\mathcal{S}_i^T$ can be represented by a matrix that has q rows (one for each day of T) and nine columns. The first column is used to indicate the date, while the other eight ones correspond to the features of F. In particular, the semantics of the columns is as follows:

1. `Day`: its h^{th} element indicates the date corresponding to d_h.
2. `In-degree`: its h^{th} element denotes the number of addresses from which n_i received transactions during the time interval τ_h between 12:00 am of d_{h-1} and 12:00 am of d_h.

3. `Out-degree`: its h^{th} element indicates the number of addresses to which n_i has made transactions during τ_h.
4. `In-transaction`: its h^{th} element denotes the number of transactions received by n_i during τ_h.
5. `Out-transaction`: its h^{th} element indicates the number of transactions made by n_i during τ_h.
6. `In-value`: its h^{th} element denotes the amount of Wei received from n_i during τ_h.
7. `Out-value`: its h^{th} element indicates the amount of Wei sent by n_i during τ_h.
8. `Clustering-coefficient`: its h^{th} element denotes the clustering coefficient of n_i in the social network $\mathcal{G}(d_{h-1}, d_h)$.
9. `PageRank`: its h^{th} element indicates the PageRank of n_i in $\mathcal{G}(d_{h-1}, d_h)$.

Defining the new version of the Eros Distance

The algorithm for the Eros distance computation applies Principal Component Analysis [50] to two multivariate time series, each represented by means of a matrix. First it generates the principal components and their corresponding eigenvalues and eigenvectors. In our case, the eigenvectors are associated with the eight spectrum features. More specifically, each eigenvector corresponds to a feature and the associated eigenvalue represents the importance of that feature for the characterization of the address or the class which the spectrum refers to. Then, the algorithm uses principal components and their associated eigenvectors to compute the similarity of the two matrices associated with the multivariate time series under consideration. It is easy and fast to implement; at the same time, as stated in [14], the Eros distance outperforms other traditional similarity measures for multivariate time series, such as the Dynamic Time Warping [12], the Weighted Sum SVD [13], and so forth.

We selected the Eros distance as the reference metric for computing spectra similarities in our classification algorithm. In fact, this computes the distance between a blockchain address to be classified and each possible class and assigns the address to the closest class. In this context, the Eros distance allows us to measure the similarity degree between two multivariate time series representing the spectrum of the address to classify and the one of a class.

The way our algorithm proceeds and the adoption of the Eros distance allow us to perform the address classification in a way that minimizes the distances between the spectra of the addresses of the same class and maximizes the distances between the spectra of the addresses of different classes.

The algorithm for the Eros distance computation uses some weights, one for each time series considered and, therefore, one for each feature. Each weight denotes the relative importance of the corresponding time series (and, therefore, of the corresponding feature) with respect to all the other ones.

The original version of the Eros distance described in [14] obtains these weights from the eigenvalues associated with the eigenvectors representing the time series being considered. Initially, we applied this version but, as we will see in "Evaluation" section, the results of the classification obtained in this way were not particularly satisfactory.

Nevertheless, we considered that the possibility, offered by the Eros distance, to associate a single value with the distance between two sets of multivariate time series was a key feature for our context. Therefore, we planned to define a new version of the Eros distance in which the weights are computed in a way tailored to our reference scenario. Regarding this, we recall that, in our case, whenever the Eros distance measures the similarity degree of two spectra, it has to consider two sets, each consisting of 8 time series. Each time series has associated a weight and the overall sum of the weights must be equal to 1. Therefore, in principle, we should consider 2 sets of 8 weights that can vary in any way between 0 and 1, with the only constraint that their overall sum must be equal to 1. It is reasonable to assume that the weights are decimal numbers with two digits after the decimal point. Even with this assumption, the problem is still NP-hard, because it would be necessary to exhaustively examine all the possible valid combinations of weights. As a consequence, despite the fact that, at the moment, the classes are only 4 and the features are only 8, we have judged opportune to preserve the scalability of our approach and to determine since now a heuristics to solve it. We have defined such a heuristics, which is reported in Algorithm 1.

Algorithm 1 Heuristics for computing the best weight combination for each class

```
Input
  ■ $Cl$: the set of the classes of interest
  ■ $\mathcal{S}_{Cl}$: the set of the spectra of the classes of $Cl$
  ■ $\mathcal{S}_{train}$: a set of sets of address spectra; $\mathcal{S}^i_{train}$ comprises all the addresses of the training set already assigned to
    the class $Cl_i$
  ■ $step$: a decimal number in the interval $[0, 1]$
Output
  ■ $\mathcal{W}_{best}$: a set of weight sets such that $\mathcal{W}^i_{best}$ comprises all the weights computed for the class $Cl_i$
Require: $min_d$, $max_d$, $d$, $min_q$, $max_q$: a real; $Eros(S_x, S_y, w)$: a function computing the Eros distance between the
  spectra $S_x$ and $S_y$ using the set $w$ of weights; $w_t$: a set of weights such that each weight is a two-digit decimal number
  in the interval $[0, 1]$ and $\sum_{k=1}^{8} w_t^k = 1$; $\mathcal{W}_{temp}$: a set of weight sets

for $Cl_i \in Cl$ do
  $max_d = 0$
  $min_d = +\infty$
  initialize $w_t$ as a random combination of two-digit decimal numbers such that $\sum_{k=1}^{8} w_t^k = 1$
  $\mathcal{W}_{temp} = \{w_t\}$
  Add to $\mathcal{W}_{temp}$ all the possible sets of weights obtained by increasing one component of $w_t$ of one or more steps and
  decreasing another component of $w_t$ of the same number of steps
  for $w_q \in \mathcal{W}_{temp}$ do
    $max_t = 0$
    $min_t = +\infty$
    for $S_j \in \mathcal{S}^i_{train}$ do
      $d = Eros(S_i, S_j, w_q)$
      if $d < min_q$ then
        $min_q = d$
      end if
    end for
    for $S_j \in \mathcal{S}^k_{train}, k \neq i$ do
      $d = Eros(S_i, S_j, w_q)$
      if $d > max_q$ then
        $max_q = d$
      end if
    end for
    if $(max_d < max_q)$ and $(min_d > min_q)$ then
      $\mathcal{W}^i_{best} = w_q$
      $max_d = max_q$
      $min_d = min_q$
    end if
  end for
end for
return $\mathcal{W}_{best}$
```

Our heuristics receives in input:

- The set Cl of the classes of interest; in our case, this set consists of the classes "Token Contract", "Exchange", "Bancor" and "Uniswap".
- The set $\mathcal{S}_{Cl}$ of the spectra of the classes of Cl; as for our dataset, these are the spectra shown in Figures 4, 6, 8 and 10.
- The set $\mathcal{S}_{train}$ of the spectra of the training addresses; the element $\mathcal{S}^i_{train}$ represents the set of spectra of the training addresses assigned to the class Cl_i.
- The parameter *step*, which is a decimal number in the range [0, 1]. As we will see below, it allows the management of a tradeoff between the accuracy and the computation time of our heuristics. In fact, the smaller the step, the more accurate the output of our heuristics, but the longer its computation time.

Our heuristics returns a set $\mathcal{W}_{best}$ of weights sets, one for each class. $\mathcal{W}_{best}$ is computed in such a way as to minimize the Eros distance between the spectra of the addresses of the same class and maximize the Eros distance between the spectra of the addresses of different classes. It also uses a function *Eros* that receives two spectra S_x and S_y and a set w of weights and computes the Eros distance between S_x and S_y using the weights specified in w.

For each class Cl_i belonging to Cl, our heuristics builds the set w_t of weights as a random combination of two-digit decimal numbers such that $\sum_{k=1}^{8} w_t^k = 1$. This last condition is required by the Eros distance and must be verified by any admissible set of weights.

Starting with w_t as seed, our heuristics builds a set $\mathcal{W}_{temp}$ by increasing one of the weights of w_t of a value equal to *step* and decreasing another one of the same value. It repeats this procedure for any pair of weights of w_t. In doing so, it may happen that some of the new combinations obtained are not admissible because one or both of the modified weights do not fall within the range [0, 1]. These combinations are discarded.

Once the construction of this initial version of $\mathcal{W}_{temp}$ is finished, our heuristics proceeds with its enrichment. For this purpose, it repeats the same procedure by increasing a weight of w_t of a value equal to $2 \cdot step$ and decreasing another one of the same value. After this second iteration has been finished, it repeats the same procedure by increasing and decreasing the weights of w_t of a value equal to $3 \cdot step$, $4 \cdot step$, and so on. The enrichment of $\mathcal{W}_{temp}$ terminates when, during one iteration of this procedure, no new admissible pair is obtained.

From this description, we can see how *step* acts as a regulator between accuracy and computation time. In fact, the lower its value, the higher the number of weight sets present in $\mathcal{W}_{temp}$ and, consequently, the higher the accuracy of our heuristics, but the longer its computation time. On the contrary, the higher the value of *step*, the lower the accuracy of our heuristics but the smaller its computation time.

At this point, $\mathcal{W}_{temp}$ has been completely constructed. Now, for each set $w_q \in \mathcal{W}_{temp}$, our heuristics applies the *Eros* function, with the set w_q of weights, for computing the minimum distance min_q between the spectrum S_i of Cl_i and the spectrum S_j of any address assigned to Cl_i. Then, it applies *Eros*, with the same set of weights, for computing the maximum distance max_q between S_i and the spectrum S_j of any address assigned to a class different from Cl_i.

If the minimum current distance min_d concerning Cl_i is greater than min_q and the maximum current distance max_d concerning Cl_i is less than max_q, then max_d is set to max_q, min_d is set to min_q, w_q becomes the new best current set of weights for Cl_i and is assigned to $\mathcal{W}^i_{best}$.

After all the sets of weights of $\mathcal{W}_{temp}$ have been examined, the current value of $\mathcal{W}^i_{best}$ becomes final. At this point, a new class of Cl is selected and the whole procedure described above is repeated. After all the classes of Cl have been examined, our heuristics terminates and returns $\mathcal{W}_{best}$.

We end this description of the heuristics with some considerations regarding its accuracy and computation time. As mentioned above, our heuristics has one parameter, namely *step*, which acts as regulator. Its presence guarantees that our heuristics terminates (in fact, it would be enough to choose a high value of *step*). Clearly, this is not enough to say that our heuristics is adequate for the problem for which it was designed. In fact, it is necessary: (i) to show that the accuracy of results is acceptable; (ii) to verify that the computation time is acceptable and, in any case, much less than the time taken by an exhaustive approach for defining weights; (iii) if possible, to find a default value for *step* that can guarantee in most cases an excellent tradeoff between accuracy and computation time. We will devote Section of the paper to address these issues. For now we anticipate that: (i) we found that setting *step* to 0.05 guarantees an excellent tradeoff between accuracy and computation time; (ii) the accuracy of the results obtained by our heuristics proved to be comparable with the one of the exhaustive approach; (iii) the computation time employed by our heuristics is much (in particular, several orders of magnitude) less than that of the exhaustive approach. In light of these results, we can say that our heuristics is adequate for the problem it aims to address.

Classifying users based on their spectra

In this section, we define a classification algorithm that, given a time interval T and an address a_j whose spectrum in T is known, assuming that the spectra of the four classes of interest in T are known, is able to classify a_j. In particular, the algorithm may assign a_j to one of the four classes or may conclude that a_j does not belong to any of them.

We observe that the classification problem we are considering is complex because it involves comparing spectra and calculating a similarity degree between them. In particular, each spectrum consists of a set of time series. As we saw in "Defining class spectra" section, these are not independent of each other but are correlated. Even if, given two features with a correlation degree equal to 1, we remove one of them and keep the other, we would not have solved the problem because the remaining features would still be partially correlated to each other. As a consequence, we must handle multivariate time series.

Recall that, as stated in the Introduction, the past literature provides some approaches to classify multivariate time series [51–53]. We have also specified that, to the best of our knowledge, there is no out-of-the-box classification approach that can be easily implemented in our case. Therefore, we preferred to define a new technique tailored to the characteristics of the problem we want to face. This technique involves the modeling of the blockchain as a social network and the next derivation of the appropriate features from it.

The core of such an algorithm consists of a metric able to compute a similarity degree between multivariate time series. In order to perform this task, we rely on the Eros distance, also known as Extended Frobenius Norm [14].

Once the weights of $\mathcal{W}_{temp}$ have been computed, the definition of the classification algorithm is straightforward. In fact, given an address a_j to be classified, it is sufficient to compute the Eros distance between the spectrum $\mathcal{S}_j$ of a_j and the spectrum of each available class. a_j will be assigned to the class with the minimum distance. We report the corresponding pseudo-code in Algorithm 2.

Algorithm 2 Algorithm for classifying a new address

Input

- a_j: the Ethereum address to be classified
- $\mathcal{S}_j$: the spectrum of a_j
- Cl: the set of the classes of interest
- $\mathcal{S}_{Cl}$: the set of the spectra of the classes of Cl
- $\mathcal{W}_{best}$: the set of the best weights identified by our heuristics

Output

- the class $\overline{Cl}$ which a_j is assigned to

```
Require: $min_d$, $th_{dmax}$, $d$: a real; $Eros(S_x, S_y, w)$: a function computing the Eros distance between the spectra $S_x$ and $S_y$ using the set $w$ of weights;

$\overline{Cl} = null$;
$min_d = +\infty$
for $Cl_i \in Cl$ do
    $d = Eros(S_i, S_j, \mathcal{W}^i_{best})$
    if $d < min_d$ then
        $min_d = d$
        $\overline{Cl} = Cl_i$
    end if
end for
return $\overline{Cl}$
```

Experiments

In this section, we present several experiments that helped us to define the details of our approach. In particular, in "Dataset" section, we present the dataset we used for training and testing it. In "An example of user spectrum" section, we describe an example of user spectrum. In "Defining the classes of interest" section, we present the process that led us to define the classes of interest. In "Defining class spectra" section, we illustrate the spectra of the selected classes. Finally, in "Weights of the Eros distance" section, we present the application, to the dataset of interest, of the method for computing the weights of the Eros distance.

In order to carry out our experiments, we used a server equipped with 16 Intel Xeon E5520 CPUs and 96 GB RAM with the Ubuntu 18.04.3 operating system. We adopted Python 3.6 as programming language, its library Pandas to perform ETL operations on data, and its library NetworkX to carry out operations on networks.

Table 1 Some preliminary statistics performed on our dataset

Parameter	Value
Number of transactions	41,420,435
Total number of addresses	5,553,645
Total number of `from_address`	4,980,691
Total number of `to_addresses`	4,471,985
Cardinality of the intersection between `from_address` and `to_address`	3,899,031
Number of null `from_address`	1
Number of null `to_address`	2

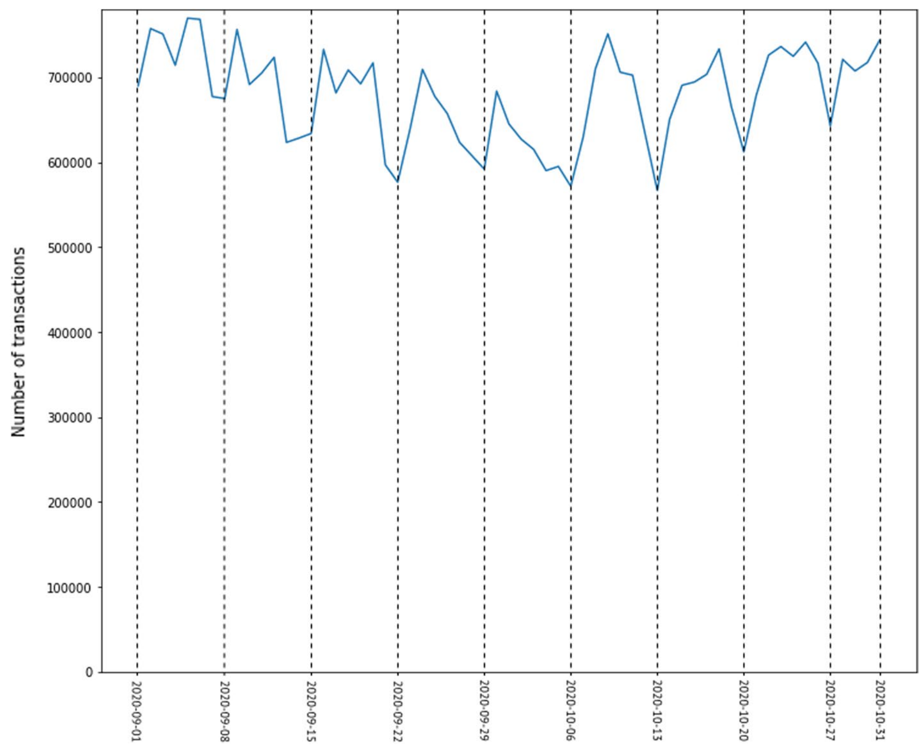

Fig. 1 Number of transactions over time

Dataset

In order to carry out our analyses, we derived a dataset from Ethereum. In particular, we downloaded the corresponding data from Google BigQuery[8]. The data we selected covers a period from September 1^{st}, 2019 to October 31^{st}, 2019. We chose it because we wanted to test our approach in a "normal" period for Ethereum, i.e., a period when there were no particular speculative bubbles. In fact the latter can heavily modify user behaviors and deserve a separate study [4]. We selected all the transactions made on Ethereum in that period. The total number of transactions considered in the dataset is 41,420,435, whereas the total number of addresses is 5,553,645. We computed some statistics on the dataset; they are reported in Table 1.

[8] https://www.kaggle.com/bigquery/ethereum-blockchain.

The distribution of transactions over time is reported in Figure 1. From the analysis of this figure we can see that the number of transactions is always in a range between 600,000 and 800,000. This trend is substantially constant with a slight decrease observed in the second half of September balanced by an increase in the first half of October. In any case, in the time interval of our dataset, we do not observe significant peaks that could suggest the presence of a speculative bubble.

During the dataset construction we had to perform some ETL (Extraction, Transformation and Loading) operations. In particular, first we removed some duplicate transactions that were present in the dataset since they cannot exist in a blockchain. Their presence was likely due to a download error. In addition, we removed all transactions in which at least one field had a null value. In fact, this type of transactions could not be used for our tests. After these basic tasks, we performed some additional, more specific, ones. In particular, we removed transactions in which at least one of the addresses involved had a wrong hexadecimal value, different from the standard expected by Ethereum. We also removed transactions in which a "dead address" was present, i.e., those transactions in which tokens are sent to be burned. Last but not least, we unified all amounts of money exchanged by representing them with a single currency, i.e., Wei.

After them, we were able to associate a dataset row with each transaction. Each row consists of four columns, namely: *(i)* `from_address`, representing the blockchain address starting the transaction; *(ii)* `to_address`, denoting the blockchain address receiving the transaction; *(iii)* `timestamp`, indicating the transaction timestamp; *(iv)* `value`, representing the amount of Wei transferred during the transaction.

We split our dataset into two parts. The former contains all the transactions made in September 2019; it consists of 20,465,806 transactions and was used for training. The latter comprises all the transactions made in October 2019; it consists of 20,954,629 transactions and was employed for testing.

Everything we describe in this section refers to an Exploratory Data Analysis on the dataset, as well as on training activities. Instead, we will describe the testing activities in "Evaluation" section.

An example of user spectrum

An example of user spectrum is shown in Table 2. It refers to the Ethereum address encoded as `0xf0ee6b27b759c9893ce4f094b49ad28fd15a23e4` and to the time interval T ranging from September 1^{st}, 2019 to September 30^{th}, 2019.

Defining the classes of interest

In order to define our classification approach, it was necessary to identify the classes of interest. For this purpose, we exploited information provided by Etherscan. At the time of writing, this service provider has defined 426 possible classes. Clearly, it is impractical to think of building a classification approach with such a large number of classes. Therefore, it seemed appropriate to detect the most common ones by checking the

Table 2 An example of a user spectrum

Day	In-degree	Out-degree	In-transactions	Out-transactions	In-value	Out- value	Clustering-coefficient	PageRank
2019-09-01	14	0	36	0	36	0	0.000020	0.021978
2019-09-02	11	0	24	0	24	0	0.000014	0.010526
2019-09-03	30	0	45	0	45	0	0.000019	0.003171
2019-09-04	21	0	36	0	36	0	0.000015	0.003025
2019-09-05	16	0	28	0	28	0	0.000013	0.002261
2019-09-06	22	0	46	0	46	0	0.000013	0.002272
2019-09-07	25	0	54	0	54	0	0.000014	0.002922
2019-09-08	18	0	46	0	46	0	0.000026	0.002871
2019-09-09	15	0	45	0	45	0	0.000026	0.002669
2019-09-10	22	0	63	0	63	0	0.000028	0.002312
2019-09-11	24	0	78	0	78	0	0.000031	0.002150
2019-09-12	25	0	85	0	85	0	0.000031	0.002070
2019-09-13	18	0	49	0	49	0	0.000031	0.002020
2019-09-14	8	0	22	0	22	0	0.000030	0.001925
2019-09-15	10	0	12	0	12	0	0.000029	0.001733
2019-09-16	24	0	34	0	34	0	0.000031	0.001689
2019-09-17	12	0	18	0	18	0	0.000030	0.001578
2019-09-18	24	0	34	0	34	0	0.000031	0.001543
2019-09-19	13	0	16	0	16	0	0.000031	0.001587
2019-09-20	24	0	35	0	35	0	0.000031	0.001542
2019-09-21	23	0	29	0	29	0	0.000031	0.001501
2019-09-22	12	0	20	0	20	0	0.000032	0.001494
2019-09-23	15	0	29	0	29	0	0.000032	0.001462
2019-09-24	19	0	43	0	43	0	0.000031	0.001436
2019-09-25	28	0	55	0	55	0	0.000032	0.001481
2019-09-26	20	0	31	0	31	0	0.000031	0.001436
2019-09-27	15	0	33	0	33	0	0.000031	0.001440
2019-09-28	17	0	29	0	29	0	0.000032	0.001339
2019-09-29	27	0	57	0	57	0	0.000033	0.001308
2019-09-30	19	0	27	0	27	0	0.000033	0.001308

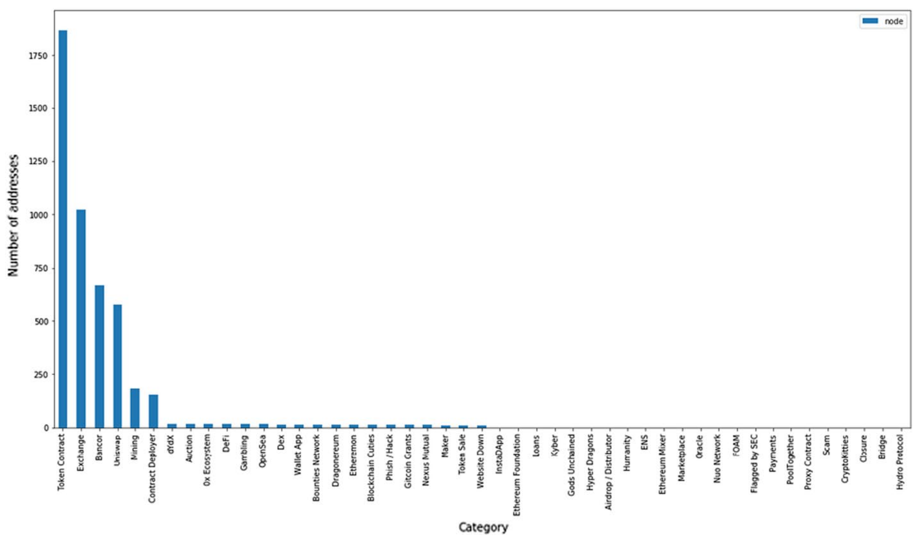

Fig. 2 Distribution of Ethereum training addresses against the main Etherscan classes

Table 3 Number of addresses belonging to each class of interest for our investigation

Class	Number of addresses
Token Contract	1866
Exchange	1021
Bancor	666
Uniswap	577

distribution of the current addresses against the classes provided by Etherscan. To this end, we selected uniformly at random a set of 2,010,729 Ethereum addresses from the training data of our dataset and verified their classes (if any) on Etherscan. This check returned a class for 4,443 of them. Figure 2 shows the distribution of these addresses against the main classes handled by Etherscan.

From the analysis of this figure, it is clear that the distribution follows a power law. The majority of the addresses (41.99%) belongs to the class "Token Contracts". Immediately after, there are the classes "Exchange" (22.97%), "Bancor" (14.98%) and "Uniswap" (12.98%). Overall, these four classes cover 92.92% of Ethereum addresses labeled by Etherscan. For this reason, we decided to focus our classification approach on them in order to reconstruct, for each class, a very precise profile, clearly distinguishing it from the others. The addition of more classes would have risked creating partially overlapping class profiles with a negligible increase in the number of addresses that could be classified. The semantics of the four classes we chose is as follows:

- The *"Token Contract"* class includes addresses using tokens instead of Ether. Tokens are an alternative currency to Ether, used to fasten up and simplify processes.
- The *"Exchange"* class includes addresses acting as money changers; these allow clients to buy and sell cryptocurrencies.
- The *"Bancor"* class includes addresses acting as banks. A bancor allows clients to deposit and convert each available token in the network, without counterparts, automatically at a given price, using a simple web wallet.
- The *"Uniswap"* class includes addresses using the "Uniswap"[9] protocol for the automatic exchange of tokens in Ethereum.

In Table 3, we report the number of addresses for each of these classes.

Defining class spectra

After determining the classes of interest, in this section we want to define the spectrum of each class. As a first step, we need to check if all the features identified in "Defining the spectrum of a user or a class of users" section are independent from each other or if there are correlations between them.

To answer this question, for all the addresses of our training set, we computed the spectrum with reference to the corresponding time interval, i.e., from September 1st, 2019 to September 30th, 2019. Then, we computed the overall correlation matrix

[9] https://uniswap.org.

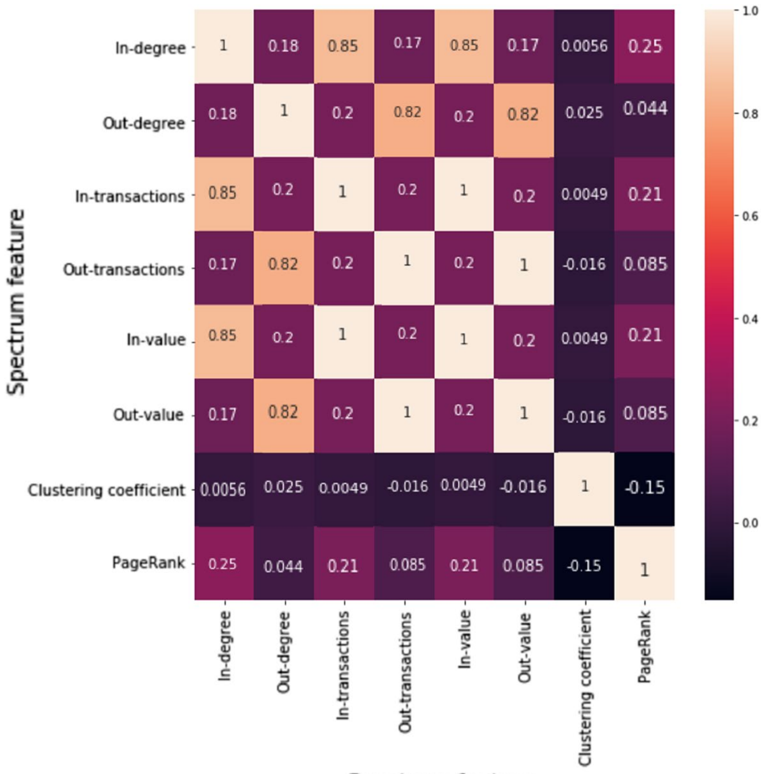

Fig. 3 Correlation matrix for the spectrum features of all the addresses in the training data set

associated with all the addresses of our training set. For this purpose, we set the value of each element of the matrix equal to the average of the values of the corresponding elements for all addresses. The matrix thus obtained is shown in Figure 3.

From the analysis of this figure we can see that there are totally correlated features. In fact, `In-transaction` is totally correlated with `In-value`, while `Out-transaction` is totally correlated with `Out-value`. Furthermore, there are other strong correlations. For instance, `In-degree` is strongly correlated with `In-transaction` and `In-value`, while `Out-degree` is strongly correlated with `Out-transaction` and `Out-value`.

This result is extremely important because it allows us to draw the following two relevant conclusions:

- In principle, we could remove one feature between `In-transaction` and `In-value` and one feature between `Out-transaction` and `Out-value` from the spectrum. We decided not to do so because the result refers to a specific time interval. We believe it is plausible that it applies to the other time intervals as well. However, since a formal proof of this is not possible, we felt it appropriate to preserve all features. As a consequence of this decision, it is to be expected that some spectrum features will have perfectly coincident trends in the following.

Table 4 Minimum, maximum, mean and standard deviation of the values of the spectrum features for the class "Token Contract"

Feature	Minimum Value	Maximum Value	Mean Value	Standard Deviation
`In-degree`	4.65	91.40	20.52	18.60
`Out-degree`	0	0	0	0
`In-transaction`	10.80	354.44	59.24	70.76
`Out-transaction`	0	0	0	0
`In-value`	10.81	314.44	59.24	70.76
`Out-value`	0	0	0	0
`Clustering-coefficient`	$5.80 \cdot 10^{-4}$	$2.90 \cdot 10^{-2}$	$8.40 \cdot 10^{-3}$	$7.30 \cdot 10^{-3}$
`PageRank`	$1.61 \cdot 10^{-5}$	$9.41 \cdot 10^{-5}$	$5.97 \cdot 10^{-5}$	$2.24 \cdot 10^{-5}$

- There are strong correlations between several spectrum features. Consequently, they cannot be considered independent of each other and the spectrum of an address in a time interval must be analyzed as a multivariate time series.

After considering the overall spectrum representing all users in the dataset, in the next subsections we examine the spectrum of the four classes of interest determined above.

Spectrum of the class "Token Contract"

Given all the nodes of the class "Token contract" in the training period, we computed the minimum, maximum, mean and standard deviation of the values of the spectrum features. They are shown in Table 4.

Then, in order to generate the spectrum of this class, we considered, for each feature and for each day of the training period, the average of the corresponding values for all the nodes of that class. The corresponding result is shown in Figure 4.

As can be seen in this figure, there are spectrum features having an identical trend, as we expected based on what we said in "Defining class spectra" section . These are `In-transaction` and `In-Degree`, on the one hand, and `Out-transaction`, `Out-degree` and `Out-value`, on the other hand. In addition, there are strong similarities between the trends of `In-degree` on the one hand, and `In-transaction` and `In-value` on the other hand. To quantify this fact, we computed the correlation matrix for the spectrum features of this class. It is shown in Figure 5. This figure also reveals another interesting correlation, i.e., a strong inverse correlation between `Clustering-coefficient` and `PageRank`.

Spectrum of the class "Exchange"

The minimum, maximum, mean and standard deviation of the values of the spectrum features for the class "Exchange" are reported in Table 5. Figure 6 shows the spectrum of this class.

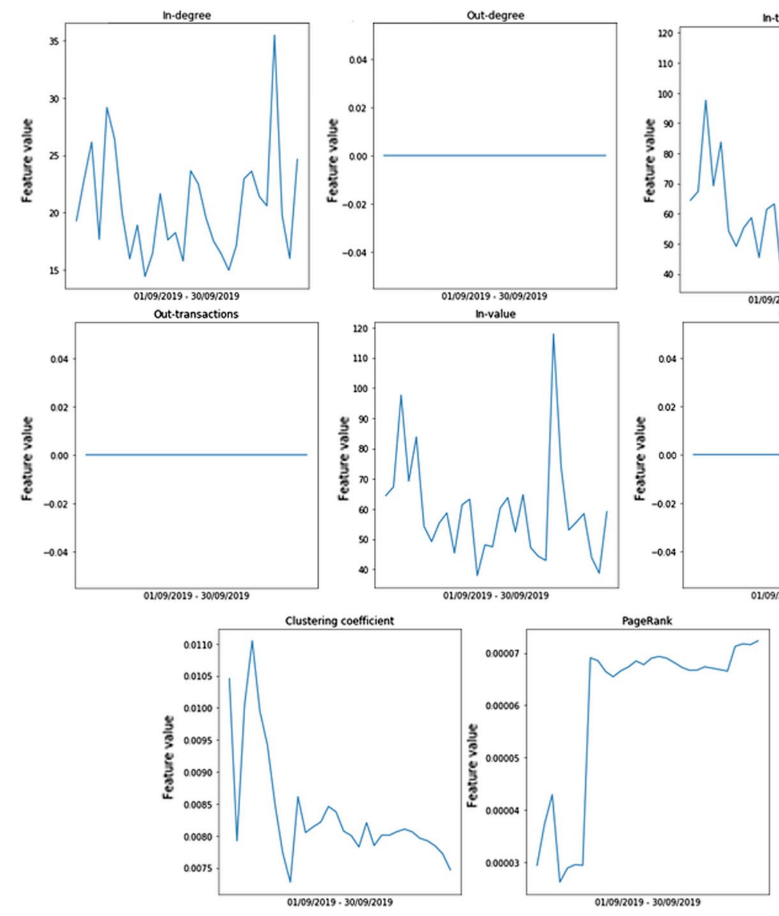

Fig. 4 Spectrum of the class "Token Contract"

One interesting characteristic that can be observed in this spectrum is the absence of features with constant null value. As we will see in the next subsections, when we will examine the spectrum of the other classes, this characteristic is specific of the class "Exchange" and cannot be found in any other classes. Already from a visual analysis of this spectrum, we can observe that the trends of `In-transaction`, `In-degree` and `In-value` are identical. Similarly, the trends of `Out-transaction` and `Out-value` are identical. There is also a strong correlation between these last trends and the one of `Out-degree`.

Again, we computed the correlation matrix for the features of this class. It is reported in Figure 7. It shows a correlation value equal to 1 between `In-degree`, `In-transactions` and `In-value`, as well as between `Out-transactions` and `Out-value`. There is also a very high correlation, equal to 0.92, between `Out-degree` and `Out-transactions` and between `Out-degree` and `Out-value`. All these values fully confirm what we have deduced above from the direct observations of the trends in Figure 6.

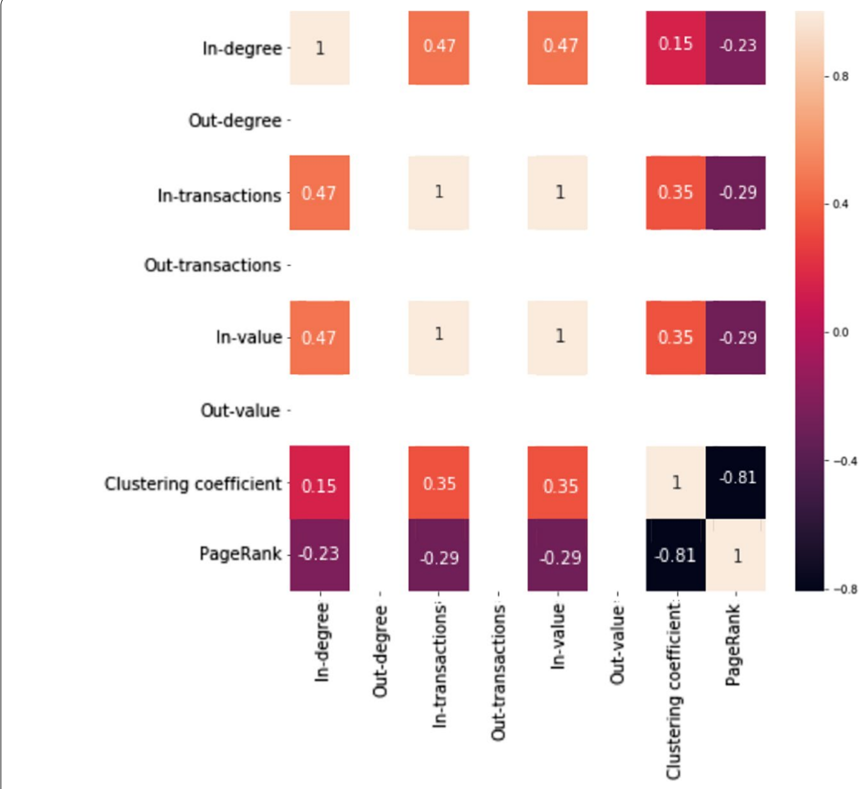

Fig. 5 Correlation matrix for the spectrum features of the class "Token Contract"

Table 5 Minimum, maximum, mean and standard deviation of the values of the spectrum features for the class "Exchange"

Feature	Minimum Value	Maximum Value	Mean Value	Standard Deviation
`In-degree`	73.00	322.05	145.22	96.60
`Out-degree`	21.40	190.13	83.78	55.43
`In-transaction`	84.56	387.67	173.85	81.61
`Out-transaction`	76.37	417.83	185.35	93.10
`In-value`	84.56	387.67	173.85	81.61
`Out-value`	76.37	417.83	185.33	93.10
`Clustering-coefficient`	$5.26 \cdot 10^{-4}$	$1.99 \cdot 10^{-2}$	$4.99 \cdot 10^{-3}$	$5.02 \cdot 10^{-3}$
`PageRank`	$2.76 \cdot 10^{-4}$	$5.68 \cdot 10^{-4}$	$4.43 \cdot 10^{-4}$	$8.00 \cdot 10^{-5}$

Spectrum of the class "Bancor"

In Table 6, we report the minimum, maximum, mean and standard deviation of the values of the spectrum features for the class "Bancor". In Figure 8, we show the spectrum of this class.

From the analysis of this spectrum we can see that the trends of `Out-transaction`, `Out-degree` and `Out-value` are identical. An analogous discourse is valid for the trends of `In-transaction` and `In-value`, which, in turn, show a strong correlation with the trend of `In-degree`.

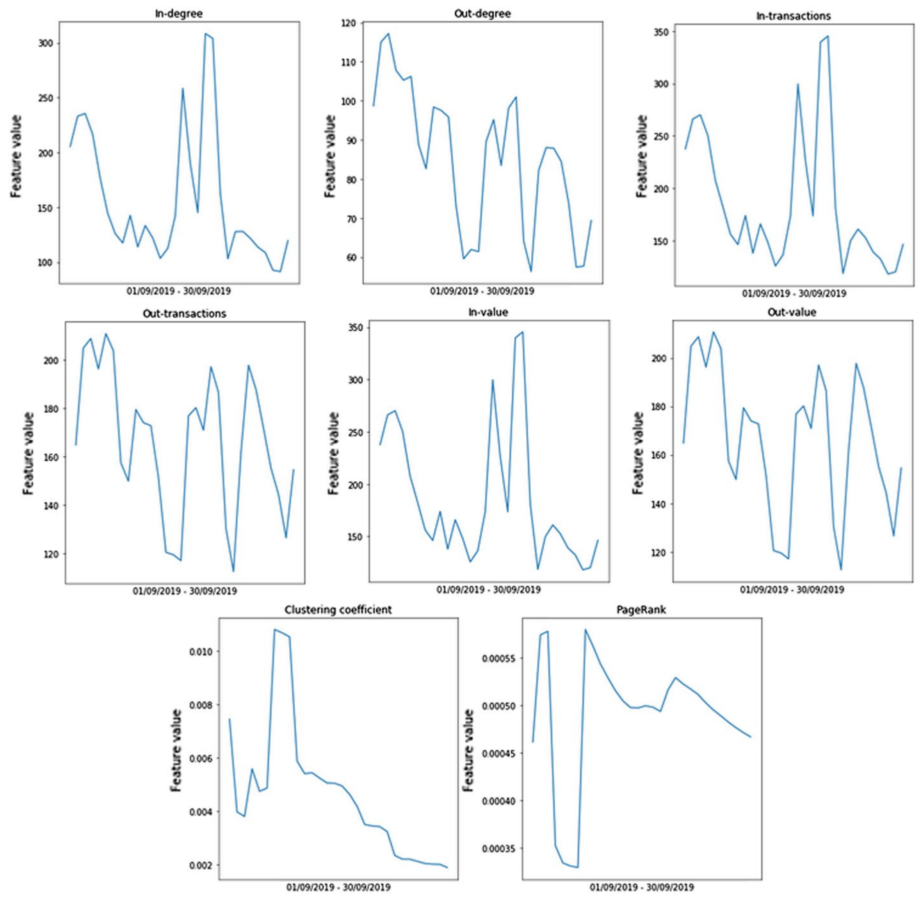

Fig. 6 Spectrum of the class "Exchange"

Also for this class we quantified these correlations by computing the correlation matrix for the features of its spectrum. In Figure 9, we report such a matrix. Its analysis confirms all the previous observations and also highlights a good correlation between `Clustering-coefficient` and `In-degree`. It also reveals a strong correlation between `In-transaction`, `In-value` and `In-degree`, on one hand, and `Out-transaction`, `Out-value` and `Out-degree`, on the other hand. This is typical of this class of addresses that represents bankers.

Spectrum of the class "Uniswap"

The minimum, maximum, mean and standard deviation of the values of the spectrum features for the class "Uniswap" are reported in Table 7. The spectrum of this class is shown in Figure 10.

From the analysis of this spectrum, we can see that the trends of `Out-transaction`, `Out-degree` and `Out-value` are identical. The same conclusion applies to the trends of `In-transaction` and `In-value`. In addition, we can observe a strong correlation between the trend of `In-degree` and the ones of `In-value` and `In-transaction`.

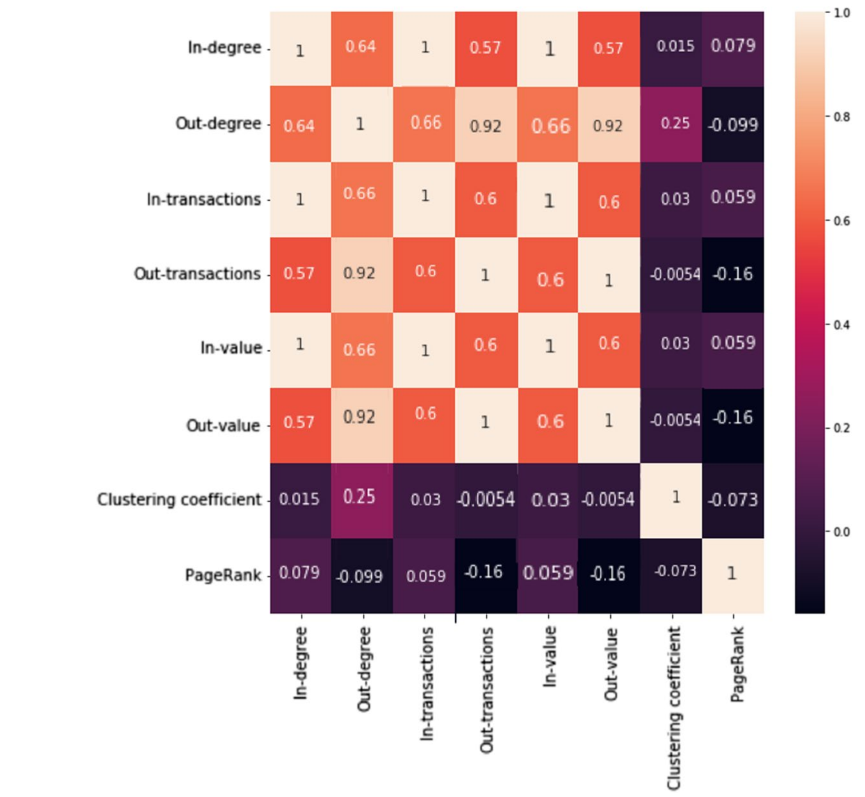

Fig. 7 Correlation matrix for the spectrum features of the class "Exchange"

Table 6 Minimum, maximum, mean and standard deviation of the values of the spectrum features for the class "Bancor"

Feature	Minimum Value	Maximum Value	Mean Value	Standard Deviation
`In-degree`	0.42	9.63	3.10	2.23
`Out-degree`	0	0	0	0
`In-transaction`	1.57	37.40	9.47	8.04
`Out-transaction`	0	0	0	0
`In-value`	1.57	37.47	9.47	8.04
`Out-value`	0	0	0	0
`Clustering-coefficient`	$1.87 \cdot 10^{-4}$	$4.27 \cdot 10^{-3}$	$1.32 \cdot 10^{-3}$	$1.01 \cdot 10^{-3}$
`PageRank`	$8.99 \cdot 10^{-7}$	$3.57 \cdot 10^{-6}$	$1.49 \cdot 10^{-6}$	$6.21 \cdot 10^{-7}$

In Figure 11, we report the correlation matrix for the features of this spectrum. This figure confirms all the previous observations. As for this class, it also shows a strong correlation between `Clustering-coefficient` and `PageRank` and a good correlation between `PageRank` and `In-Degree`.

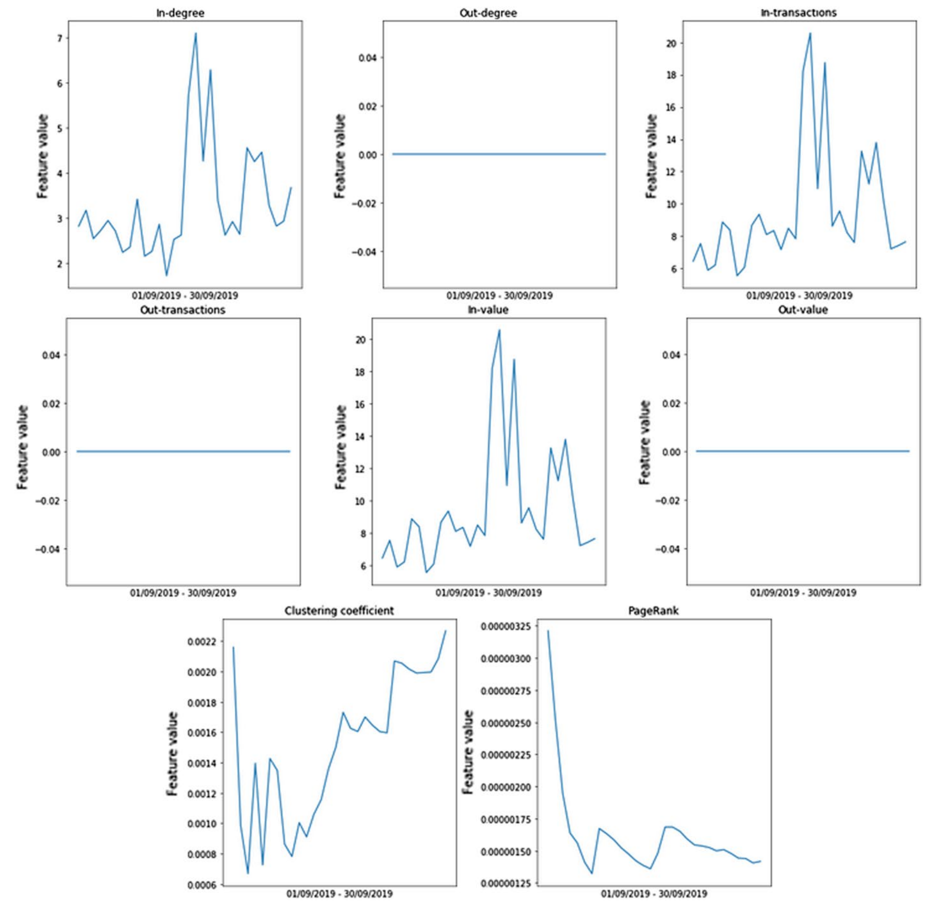

Fig. 8 Spectrum of the class "Bancor"

Weights of the Eros distance

In order to give an idea of the behavior of our heuristics for determining the weights of the Eros distance, in Table 8 we report the set of the weights of $\mathcal{W}_{temp}$ for the training data of our dataset. The examination of this table provides us with the following information:

- As for the class "Token Contract" the most important features are `In-transactions` and `In-value`. An intermediate weight is assigned to `In-degree`, `Clustering-coefficient` and `PageRank`. Finally, `Out-degree`, `Out-transactions` and `Out-value` have no weight.
- As far as the class "Exchange" is concerned, all features have roughly similar weights.
- Regarding the class "Bancor", the most important features are `In-transactions`, `In-value` e `In-degree`. A fairly small weight is assigned to `PageRank` and `Clustering-coefficient`. Finally, the other ones have no weight.
- As far as the class "Uniswap" is concerned, the most important features are `PageRank` and `Clustering-coefficient`. A small to medium weight is assigned to the

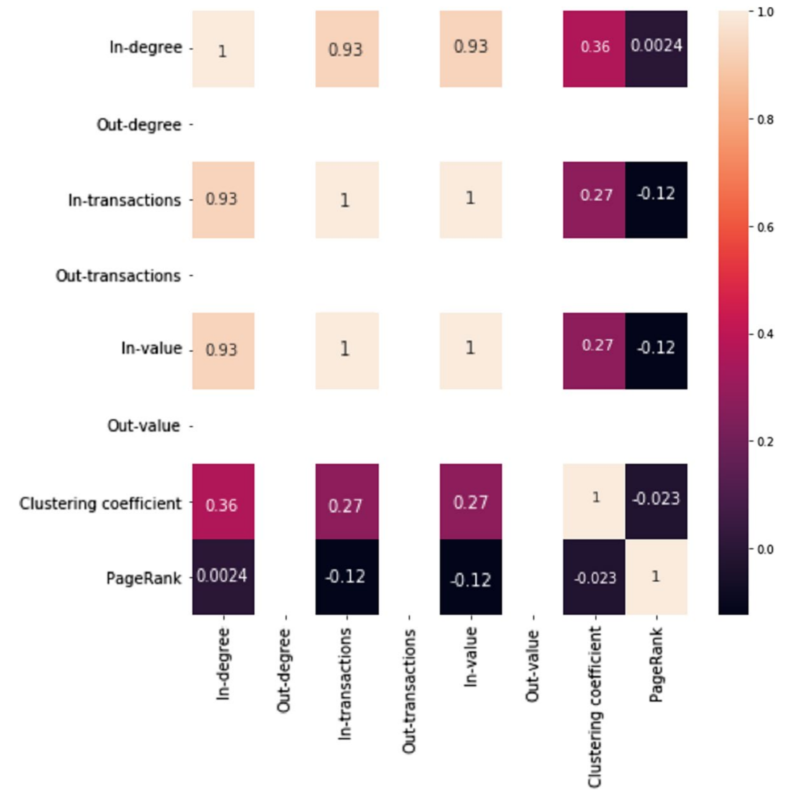

Fig. 9 Correlation matrix for the spectrum features of the class "Bancor"

Table 7 Minimum, maximum, mean and standard deviation of the values of the spectrum features for the class "Uniswap"

Feature	Minimum Value	Maximum Value	Mean Value	Standard Deviation
`In-degree`	0.42	9.63	3.10	2.23
`Out-degree`	0	0	0	0
`In-transaction`	1.57	37.40	9.47	8.04
`Out-transaction`	0	0	0	0
`In-value`	1.57	37.47	9.47	8.04
`Out-value`	0	0	0	0
`Clustering-coefficient`	$1.87 \cdot 10^{-4}$	$4.27 \cdot 10^{-3}$	$1.32 \cdot 10^{-3}$	$1.01 \cdot 10^{-3}$
`PageRank`	$8.99 \cdot 10^{-7}$	$3.57 \cdot 10^{-6}$	$1.49 \cdot 10^{-6}$	$6.21 \cdot 10^{-7}$

features `In-degree`, `In-transactions` and `In-value`. The other ones have no weight.

Comparing the weights shown in Table 8 with the spectra shown in Figures 4, 6, 8 and 10 and with the correlation matrices reported in Figures 5, 7, 9 and 11, the results obtained by our heuristics appear compatible with the knowledge that a human expert could derive from those figures. Clearly their actual validity must be confirmed by experiments; these will be illustrated in the next section.

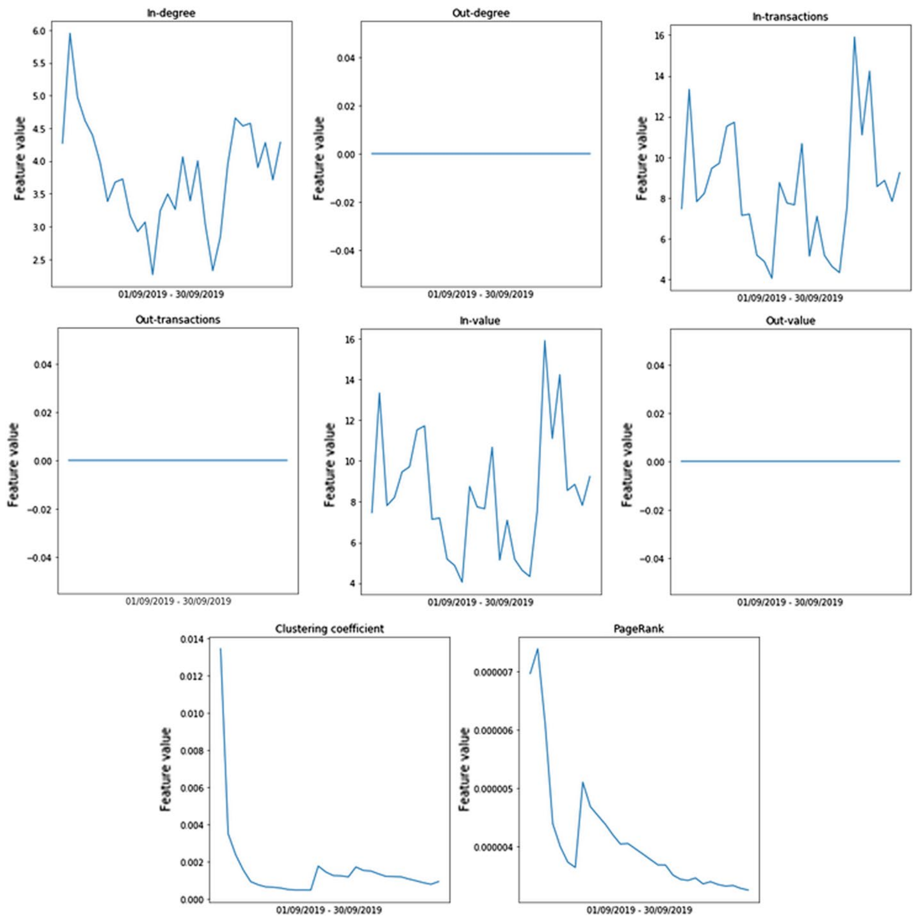

Fig. 10 Spectrum of the class "Uniswap"

Evaluation

In this section, we present the tests we carried out to evaluate the performance of our classification approach. Specifically, in "Evaluating our approach with the original Eros distance" section, we analyze our classification approach with the original Eros distance. In "Evaluating our approach with an exhaustive examination of all weight com-binations for the Eros distance" section, we consider our classification approach with an exhaustive examination of all the combinations of the weights of the Eros distance. In "Evaluating our approach with our version of the Eros distance" section, we analyze our classification approach supported with the new Eros distance with *step* set to 0.05, which proved able to guarantee an excellent tradeoff between accuracy and computation time. Finally, in "Computation time analysis" section, we give an idea of the computation times associated with the various steps of our approach.

As mentioned in "Dataset" section, testing data in our dataset includes 20,954,629 transactions (i.e., all the transactions carried out on Ethereum from October 1st, 2019 to October 31st, 2019). Similarly to what we did for training data (see "Defining class spectra" section), we selected 2,120,834 Ethereum addresses uniformly at random from testing data and derived the corresponding classes from Etherscan. It was able to label 4,568 addresses whose distribution is shown in Figure 12.

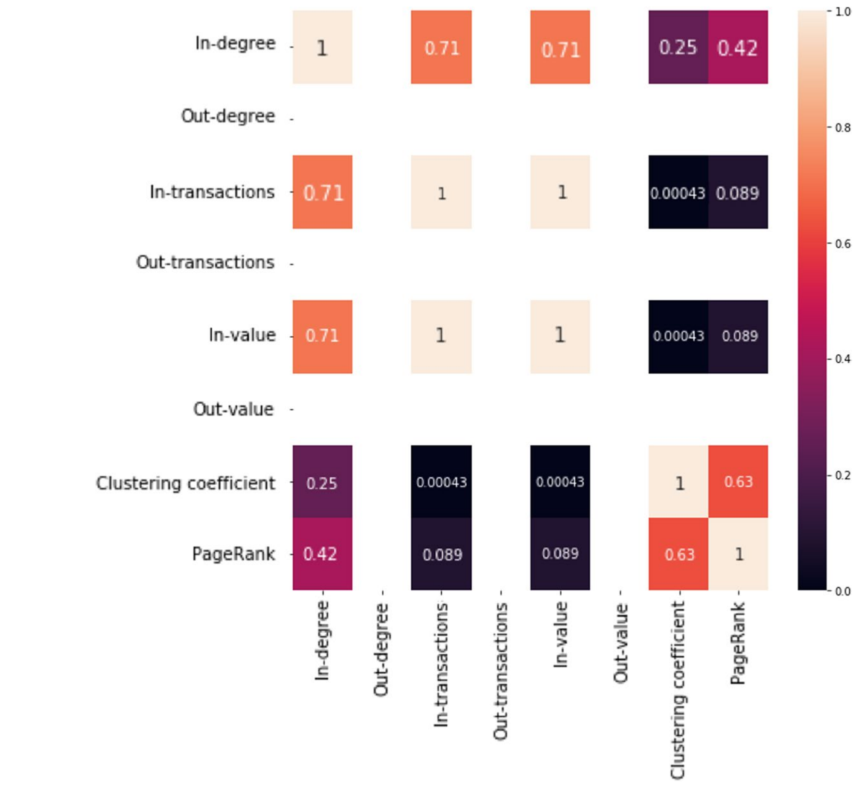

Fig. 11 Correlation matrix for the spectrum features of the class "Uniswap"

As reported in this figure, the first four classes were "Token Contract", "Exchange", "Bancor" and "Uniswap". They covered 93.73% of the Ethereum addresses labeled by Etherscan. Table 9 reports the number of addresses assigned by Etherscan to these classes. These assignments represent the ground truth for the experiments described in the next subsections.

Evaluating our approach with the original Eros distance

In this section, we evaluate our classification approach with the original version of the Eros distance for computing the similarity degree of two spectra. Recall that, in this version, the weights are obtained from the eigenvalues associated with the eigenvectors representing the time series under consideration. To perform our evaluation, we applied our classification algorithm with the original Eros distance providing as input to it the 4,568 testing addresses already labeled by Etherscan.

The computation time of this algorithm, when adopting the hardware framework described in "Dataset" section, is equal to 21 seconds. It is acceptable if we consider that we are managing multivariate time series. However, it is still high compared to a classic classification algorithm, in which each class is represented by the value of a single parameter.

The confusion matrix we obtained is shown in Table 10. From the analysis of this matrix we can see that the results, albeit acceptable, are not particularly satisfactory. In order to have numerical indicators capable of quantifying the goodness of

Table 8 Weights combination for the Eros distance relative to each class of interest

Class	Weights
Token Contract	In-degree: 0.18
	Out-degree: 0
	In-transactions: 0.26
	Out-transactions: 0
	In-value: 0.26
	Out-value: 0
	PageRank: 0.14
	Clustering-coefficient: 0.16
Exchange	In-degree: 0.13
	Out-degree: 0.15
	In-transactions: 0.13
	Out-transactions: 0.15
	In-value: 0.13
	Out-value: 0.15
	PageRank: 0.10
	Clustering-coefficient: 0.06
Bancor	In-degree: 0.27
	Out-degree: 0
	In-transactions: 0.27
	Out-transactions: 0
	In-value: 0.27
	Out-value: 0
	PageRank: 0.10
	Clustering-coefficient: 0.09
Uniswap	In-degree: 0.12
	Out-degree: 0
	In-transactions: 0.12
	Out-transactions: 0
	In-value: 0.12
	Out-value: 0
	PageRank: 0.31
	Clustering-coefficient: 0.33

the results obtained, we computed the Micro- and Macro- Average Precision, Average Recall and Average F1-Score, as well as the overall Accuracy.

Recall that, in a multi-class classification, Micro-Average means computing Precision, Recall and F1-Score considering true positives, true negatives, false positives and false negatives together, without distinguishing between classes. On the contrary, Macro-Average means computing the metrics independently for each class and, then, computing the average of the values thus obtained. Instead, the overall Accuracy is simply defined as the ratio of the number of correctly classified instances to the total number of instances. All the seven parameters of our interest have a value ranging in the real interval [0, 1]; the higher the value, the higher the goodness of the approach being evaluated [54].

As for our experiment, the values of Micro- and Macro- Average parameters and the one of Accuracy are reported in Table 11.

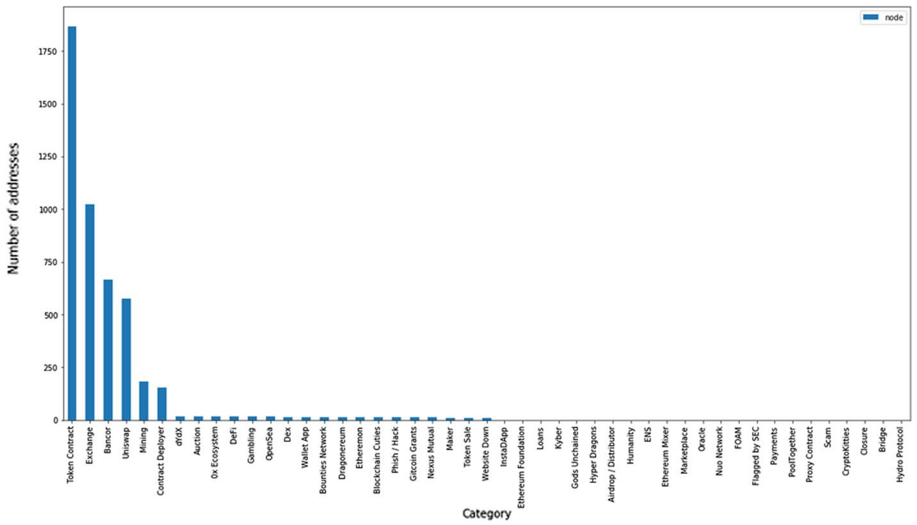

Fig. 12 Distribution of Ethereum testing addresses against the main categories of Etherscan

Table 9 Number of addresses belonging to each class of interest

Class	Number of addresses
Token Contract	1954
Exchange	1052
Bancor	684
Uniswap	592

Table 10 Confusion matrix of our classification algorithm with the classical version of the Eros distance

	Token Contract	Exchange	Bancor	Uniswap
Token Contract	1632	88	224	10
Exchange	62	964	54	72
Bancor	124	20	523	17
Uniswap	18	70	20	484

Table 11 Values of some quality metrics obtained by applying our classification algorithm with the original Eros distance on testing data

Metric	Value
Accuracy	0.75
Micro Average Precision	0.74
Macro Average Precision	0.62
Micro Average Recall	0.74
Macro Average Recall	0.63
Micro Average F1-Score	0.76
Macro Average F1-Score	0.76

Table 12 The best weight combination for the Eros distance obtained after an exhaustive examination of all the possible combinations on testing data

Class	Weights
Token Contract	In-degree: 0.15
	Out-degree: 0
	In-transactions: 0.30
	Out-transactions: 0
	In-value: 0.30
	Out-value: 0
	PageRank: 0.12
	Clustering-coefficient: 0.13
Exchange	In-degree: 0.12
	Out-degree: 0.16
	In-transactions: 0.12
	Out-transactions: 0.16
	In-value: 0.12
	Out-value: 0.16
	PageRank: 0.11
	Clustering-coefficient: 0.09
Bancor	In-degree: 0.30
	Out-degree: 0
	In-transactions: 0.30
	Out-transactions: 0
	In-value: 0.30
	Out-value: 0
	PageRank: 0.06
	Clustering-coefficient: 0.04
Uniswap	In-degree: 0.10
	Out-degree: 0
	In-transactions: 0.10
	Out-transactions: 0
	In-value: 0.10
	Out-value: 0
	PageRank: 0.34
	Clustering-coefficient: 0.36

Table 13 Confusion matrix of our classification algorithm with an exhaustive examination of all the possible weight combinations for the Eros distance

	Token contract	Exchange	Bancor	Uniswap
Token Contract	1896	18	32	8
Exchange	21	984	24	23
Bancor	36	15	621	12
Uniswap	12	32	16	532

Table 14 Values of some quality metrics obtained by applying our classification algorithm with an exhaustive examination of all the possible weight combinations for the Eros distance

Metric	Value
Accuracy	0.97
Micro Average Precision	0.94
Macro Average Precision	0.93
Micro Average Recall	0.94
Macro Average Recall	0.93
Micro Average F1-Score	0.94
Macro Average F1-Score	0.93

This table confirms, from a quantitative viewpoint, what we have qualitatively observed above, namely that the original eigenvalues-based method for computing the Eros distance is not suitable for our context.

Evaluating our approach with an exhaustive examination of all weight combinations for the Eros distance

In this section, we want to test whether satisfactory accuracy results are obtained with a modified version of the Eros distance. In particular, we considered all the possible combinations of weights relative to the four classes of interest and chose the best one. It is reported in Table 12.

Then, we applied our classification algorithm with the modified Eros distance and this combination of weights. In Table 13, we report the obtained confusion matrix, while in Table 14 we show the values of Accuracy and Micro- and Macro- Average Precision, Average Recall and Average F1-Score.

From the analysis of these tables, we can see that the results obtained in this case are really excellent. However, the main problem with this approach is its computation time. In fact, in order to classify 4568 testing addresses, our algorithm required 195,641 seconds. This is a much longer time than the one required by the original version of the Eros distance. While this is still acceptable for about 4500 testing addresses, it becomes impractical as the number of the addresseses to classify starts to increase.

Evaluating our approach with our version of the Eros distance

In this section, we want to test the performance of our classification algorithm with our version of the Eros distance. Specifically, in this case, the weights to be adopted for the computation of the Eros distance are determined by means of our heuristics described in Algorithm 1. In applying it, we set the value of the parameter *step* to 0.05, which has proven to return an excellent tradeoff between accuracy and computation time.

Proceeding in this way, we obtained the weight combination shown in Table 15. Comparing it with the optimal one, provided in Table 12, we can see that the differences are very small.

Then, we applied our classification algorithm, equipped with the modified Eros distance and this weight combination. In Table 16, we report the confusion matrix, while

Table 15 The best weight combination for the Eros distance obtained by applying our heuristics on testing data

Class	Weights
Token Contract	In-degree: 0.17
	Out-degree: 0
	In-transactions: 0.28
	Out-transactions: 0
	In-value: 0.28
	Out-value: 0
	PageRank: 0.14
	Clustering-coefficient: 0.13
Exchange	In-degree: 0.13
	Out-degree: 0.13
	In-transactions: 0.13
	Out-transactions: 0.13
	In-value: 0.13
	Out-value: 0.13
	PageRank: 0.12
	Clustering-coefficient: 0.10
Bancor	In-degree: 0.29
	Out-degree: 0
	In-transactions: 0.29
	Out-transactions: 0
	In-value: 0.20
	Out-value: 0
	PageRank: 0.08
	Clustering-coefficient: 0.05
Uniswap	In-degree: 0.12
	Out-degree: 0
	In-transactions: 0.12
	Out-transactions: 0
	In-value: 0.12
	Out-value: 0
	PageRank: 0.31
	Clustering-coefficient: 0.33

Table 16 Confusion matrix of our classification algorithm with our version of the Eros distance

	Token contract	Exchange	Bancor	Uniswap
Token contract	1838	44	54	18
Exchange	33	956	31	33
Bancor	42	18	608	16
Uniswap	14	46	18	514

in Table 17 we report the values of Accuracy and Micro- and Macro- Average Precision, Average Recall and Average F1-Score.

These tables show that the goodness of our algorithm slightly degrades, compared to the one obtained by an exhaustive approach. However, it continues to be very high.

Table 17 Values of some quality metrics obtained by applying our classification algorithm with our version of the Eros distance

Metric	Value
Accuracy	0.91
Micro Average Precision	0.91
Macro Average Precision	0.90
Micro Average Recall	0.91
Macro Average Recall	0.89
Micro Average F1-Score	0.91
Macro Average F1-Score	0.89

In order to classify the 4568 testing addresses, our algorithm required 1410 seconds. This is a longer time than the one required by the original version of the Eros distance. However, it is much shorter than the one required by the exhaustive approach. This is already an important result, but the most relevant fact is that this computation time does not grow exponentially with the number of classes and/or the number of features, thus ensuring the scalability of our approach.

Another very interesting characteristic is that the user can tune the tradeoff between accuracy and computation time by simply setting the value of *step*, depending on the number of classes and features she needs to consider, the accuracy degree she desires and the time she has available. In our opinion, this tuning feature represents an additional characteristic of our approach, generally not present in the related ones proposed in the literature and that can be extremely useful in real contexts.

Computation time analysis

In this section, we conclude the evaluation of our approach by discussing the computation time of its steps. In particular, we consider the application of our approach on the dataset we used in this paper (Section). With our computational resources (see Section for all details on them), the time required for the tasks of our experiments are as follows:

- The time required to build the training (resp., testing) network was 2,522 (resp., 2,734) seconds.
- The time necessary to compute the spectra of the training (resp., testing) users was 9,234 (resp., 9,624) seconds. This is the largest computation time. It was necessary because, for the computation of the spectrum of a user, it is necessary to compute the clustering coefficient of the corresponding network node, which requires most of the time indicated above.
- The time required to compute the spectra of the training and testing classes from the ones of the corresponding users is negligible.
- The time required for classifying the training (resp., testing) users adopting our version of the Eros distance was 1,242 (resp., 1,410) seconds.

Regarding these times, we observe that they are acceptable. This conclusion is also reinforced by the consideration that the class of a user is invariant, or at least varies very slowly over time. Therefore, the classification of a user must be carried out only once or, at least, very rarely.

Discussion

Our approach to classify Ethereum users based on their behavior has a peculiarity that differentiates it from all the other classification approaches operating on Ethereum. In fact, it is *automatic* and, at the same time, *multi-class*. Let us now take a closer look at the importance of this peculiarity. The current approaches to classify Ethereum users are based on the analysis of users' smart contracts that they voluntarily submit to a provider of this service, such as Etherscan. However, the fraction of users thus classified is extremely low (more specifically, at the time of writing, it is equal to 0.236%). To overcome this difficulty, several automatic approaches to classify Ethereum users have been proposed. However, they are all single-class. In fact, they aim to find all users belonging to a certain class [5, 36–40]. They certainly represent a first response to the need for approaches capable of classifying a huge number of users. However, such an answer is still limited because, as we have seen in "Defining class spectra" section, more than 400 classes exist on Ethereum. And, although the most important ones are few, these approaches have been targeted for a very specific class. Therefore, they cannot be easily extended to find users of another class so as to simulate multi-class behavior by calling them multiple times, once for each class of interest. Instead, our approach is automatic, multi-class and incremental; therefore, it allows the classification of all the addresses belonging to classes whose spectrum is known. From this point of view, it solves an open problem and becomes an indispensable tool for all those applications needing user classification to operate [5, 6, 55].

All the automatic multi-class approaches for classifying blockchain users that we presented in Section have many differences from the one proposed in this paper. First, they were all designed for the Bitcoin blockchain, except the ones described in [10, 11]. In principle, these could be employed on any blockchain, but were tested on a very specific one, operating on stock trading. Instead, our approach is designed to operate on Ethereum, even if its guidelines are general and can be fit to other blockchains in the future.

An important difference between our approach and the related ones proposed in the past literature lies in the fact that it introduces the concept of spectrum of a user and a class of users. In this concept, a crucial role is played by the "time" variable. Instead, this variable is not taken into account by most of the approaches seen in "Related literature" section, more specifically by the ones described in [6–9, 11, 34]. The only approach that takes time into account is the one proposed in [10]. However, it operates on univariate time series, assuming that there is no form of correlation between features. This assumption is very strong in reality and, if not verified, would lead to a decrease in the accuracy of the results proportional to the correlation degree of features. In our approach, the concept of spectrum allows us to consider not only the temporal evolution of features but also their correlation. In fact, we measured the correlation degree of each pair of features adopted and found that some of them are totally or partially correlated (see Section). As a consequence, we decided to operate on multivariate time series, instead of univariate ones. Clearly, this makes our approach a bit more complex but allows it to achieve very accurate results, as we have shown in Section .

Another very important feature of our approach concerns the measure of similarity between spectra, and thus between multivariate time series. To perform this task, we start from the Eros distance [14]. This measure is very simple and easy to implement and, at the same time, outperforms other similarity measures for multivariate time series previously proposed in the literature [14]. Regarding this, our approach makes an additional contribution. Indeed, it first shows, through some experiments, that the original Eros distance does not return satisfactory results in our context. Then, it proposes a modified version of this distance which, at the price of an acceptable increase of the computational time, manages to reach very high accuracy values, as shown in Section .

Conclusion

In this paper, we proposed an automatic social network based approach to classify Ethereum users. First, we saw that the classification of a user in Ethereum currently occurs only when she requests the validation of her smart contract to a provider in charge of this service, such as Etherscan. As a result, only a small fraction of Ethereum users is presently classified. Our approach is automatic and, therefore, can classify any Ethereum user. The classification of a user is based on her past behavior modeled through the time evolution of eight parameters forming a multivariate time series, which represents her spectrum. In order to compute the similarity between the spectrum of a user and that of a class, we had to fit the Eros distance to our context. We also tested our approach on a dataset derived from Ethereum and obtained very satisfactory results in terms of both accuracy and computation time.

In the future, we plan to develop the research topics described in this paper along several directions. First, we would like to extend our approach in order to classify Ethereum entities. We recall that, in the past literature, the term "entity" has been used to denote the set of addresses of a single user. Investigating the exploitation of multiple addresses by a single user is a challenging issue. Indeed, it is first necessary to understand why a user is doing it. Then, it is needed to evaluate if and when it makes sense considering the addresses all together or separately.

Afterwards, we aim at extending the way of proceeding underlying our approach in order to define a similar approach for Bitcoin and compare it with the ones already proposed for this blockchain.

A third extension might be in depth rather than in breadth. In fact, so far we have modeled user behavior by means of a spectrum comprising eight "structural" features related to transactions made by users. None of these features takes transaction reasons into account. This information, although difficult to extract and process, could be a valuable source for understanding user behavior and being able to classify users more accurately. In the future, we plan to investigate this issue to understand whether the benefits brought by the analysis of transaction reasons outweigh the corresponding costs.

Finally, we believe it is possible to apply graph mining techniques on the social network modeling Ethereum. This could lead to the identification of possible recurring structures and motifs. The discovery of such structures could allow us to define an approach for the detection of ransom demands, fraud, blackmail spread over the network or, even, activities carried out in cooperation by a group of criminals.

Acknowledgements
Not applicable.

Authors' contributions
In this paper GB, EC, DU and LV interacted with each other in all the tasks connected with the presented research. Their contribution is equal and this is testified by the alphabetical order used in the Author List.

Funding
Not applicable.

Availability of data and materials
The datasets used and analyzed during the current study are available from the corresponding author on request.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 19 November 2021 Accepted: 23 March 2022
Published online: 04 April 2022

References

1. Zheng Z, Xie S, Dai HN, Chen X, Wang H. Blockchain challenges and opportunities: a survey. Int J Web Grid Serv. 2018;14(4):352–75.
2. Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. The cCryptography mailing list. 2008.
3. Mukhopadhyay U, Skjellum A, Hambolu O, Oakley J, Yu L, Brooks R. A brief survey of cryptocurrency systems. In: Proceedings of the international conference on privacy, security and trust (PST'16). Auckland, New Zealand. IEEE; 2016. p. 745-52.
4. Bonifazi G, Corradini E, Ursino D, Virgili L. A Social Network Analysis based approach to investigate user behavior during a cryptocurrency speculative bubble. J Inf Sci. 2021.
5. Yuan Q, Huang B, Zhang J, Wu J, Zhang H, Zhang X. Detecting Phishing Scams on Ethereum Based on Transaction Records. In: Proceedings of the international symposium on circuits and systems (ISCAS'20). Seville, Spain. IEEE; 2020. p. 1-5.
6. Toyoda K, Ohtsuki T, Mathiopoulos PT. Multi-class bitcoin-enabled service identification based on transaction history summarization. In: Proceedings of the IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData). Halifax, NS, Canada. IEEE; 2018. p. 1153-60.
7. Jourdan M, Blandin S, Wynter L, Deshpande P. Characterizing entities in the bitcoin blockchain. In: Proceedings of the international conference on data mining workshops (ICDMW'18). Singapore. IEEE; 2018. p. 55-62.
8. Lin YJ, Wu PW, Hsu CH, Tu IP, Liao SW. An evaluation of bitcoin address classification based on transaction history summarization. In: Proceedings of the IEEE international conference on blockchain and cryptocurrency (ICBC'19). Seoul, South Korea. IEEE; 2019. p. 302-10.
9. Zola F, Eguimendia M, Bruse JL, Urrutia RO. Cascading machine learning to attack bitcoin anonymity. In: Proceedings of the international conference on blockchain (ICBC'19). Atlanta, GA, USA. IEEE; 2019. p. 10-7.
10. Huang B, Liu Z, Chen J, Liu A, Liu Q, He Q. Behavior pattern clustering in blockchain networks. Multimed Tools Appl. 2017;76(19):20099–110.
11. Tang H, Jiao Y, Huang B, Lin C, Goyal S, Wang B. Learning to classify blockchain peers according to their behavior sequences. IEEE Access. 2018;6:71208–15.
12. Berndt DJ, Clifford J. Using dynamic time warping to find patterns in time series. In: Proceedings of the international conference on knowledge discovery in databases (KDD'94), vol. 10. Seattle, WA, USA. AAAI Press; 1994. p. 359-70.
13. Shahabi C, Yan D. Real-time Pattern Isolation and Recognition Over Immersive Sensor Data Streams. In: Proceedings of the international conference on multimedia modeling (MMM'03). Taipei, Taiwan; 2003. p. 93-113.
14. Yang K, Shahabi C. A PCA-based similarity measure for multivariate time series. In: Proceedings of the international workshop on multimedia databases (MMDB'04). Washington, DC, USA. ACM; 2004. p. 65-74.
15. Corbet S, Lucey B, Urquhart A, Yarovaya L. Cryptocurrencies as a financial asset: a systematic analysis. Int Rev Financ Anal. 2019;62:182–99.

16. Li X, Jiang P, Chen T, Luo X, Wen Q. A survey on the security of blockchain systems. Future Gener Comput Syst. 2020;107:841–53.
17. ElBahrawy A, Alessandretti L, Kandler A, Pastor-Satorras R, Baronchelli A. Evolutionary dynamics of the cryptocurrency market. R Soc Open Sci. 2017;4(11):170623.
18. Antonakakis N, Chatziantoniou I, Gabauer D. Cryptocurrency market contagion: market uncertainty, market complexity, and dynamic portfolios. J Int Financ Mark Inst Money. 2019;61:37–51.
19. Sun H, Ruan N, Liu H. Ethereum Analysis via Node Clustering. In: Proceedings of the international conference on network and system security (NSS'19). Sapporo, Japan: Springer; 2019. p. 114-29.
20. Thelwall M. Can social news websites pay for content and curation? The SteemIt cryptocurrency model. J Inf Sci. 2018;44(6):736–51.
21. Wu J, Liu J, Zhao Y, Zheng Z. Analysis of cryptocurrency transactions from a network perspective: an overview. J Netw Comput Appl. 2021;190:103139.
22. Vasek M, Moore T. Analyzing the Bitcoin Ponzi scheme ecosystem. In: Proceedings of the international conference on financial cryptography and data sSecurity (FC'18). Nieuwport, Curaçao; International Financial Cryptography Association; 2018. p. 101-12.
23. Reid F, Harrigan M. An analysis of anonymity in the bitcoin system. In: Security and privacy in social networks. Springer; 2013. p. 197-223.
24. Shen J, Zhou J, Xie Y, Yu S, Xuan Q. Identity inference on blockchain using graph neural network. In: Proceedings of the international conference on blockchain and trustworthy systems (BlockSys21). Virtual location. Springer; 2021. p. 3-17.
25. Camino R, Torres CF, Baden M, State R. A data science approach for honeypot detection in Ethereum. arXiv preprint arXiv:191001449. 2019. ArXiv.
26. Chen W, Zheng Z, Cui J, Ngai E, Zheng P, Zhou Y. Detecting Ponzi schemes on Ethereum: Towards healthier blockchain technology. In: Proceedings of the international world wide web conference (WWW'18). Lyon, France. ACM; 2018. p. 1409-18.
27. Bartoletti M, Pes B, Serusi S. Data mining for detecting Bitcoin Ponzi schemes. In: Proceedings of the international crypto valley conference on blockchain technology (CVCBT '18). Zug, Switzerland. IEEE; 2018. p. 75-84.
28. Lee C, Maharjan S, Ko K, Hong JWK. Toward Detecting Illegal Transactions on Bitcoin Using Machine-Learning Methods. In: Proceedings of the international conference on blockchain and trustworthy systems (BlockSys'19). Guangzhou, China. Springer; 2019. p. 520-33.
29. Li Y, Cai Y, Tian H, Xue G, Zheng Z. Identifying illicit addresses in Bitcoin network. In: Proceedings of the international conference on blockchain and trustworthy systems (BlockSys '19). Guangzhou, China. Springer; 2020. p. 99-111.
30. Kumar N, Singh A, Handa A, Shukla SK. Detecting malicious accounts on the Ethereum blockchain with supervised learning. In: Proceedings of the international symposium on cyber security cryptography and machine learning (CSCML'20). Be'er Sheva, Israel. Springer; 2020. p. 94-109.
31. Bartoletti M, Carta S, Cimoli T, Saia R. Dissecting Ponzi schemes on Ethereum: identification, analysis, and impact. Future Gener Comput Syst. 2020;102:259–77.
32. Lee C, Maharjan S, Ko K, Woo J, Hong JWK. Machine learning based bitcoin address classification. In: Proceedings of the international conference on blockchain and trustworthy systems (BlockSys'20). Dali, China. Springer; 2020. p. 517-31.
33. L Kiffer and D Levin and A Mislove. Analyzing Ethereum's contract topology. In: Proceedings of the internet measurement conference (IMC'18). Boston, MA, USA. ACM; 2018. p. 494-9.
34. Ranshous S, Joslyn CA, Kreyling S, Nowak K, Samatova NF, West CL, et al. Exchange pattern mining in the bitcoin transaction directed hypergraph. In: Proceedings of the international conference on financial cryptography and data security (FC'17). Malta. Springer; 2017. p. 248-63.
35. Wu SW, Wu Z, Chen S, Li G, Zhang S. Community detection in blockchain social networks. J Commun Inf Netw. 2021;6(1):59–71.
36. Chan W, Olmsted A. Ethereum transaction graph analysis. In: Proc. of the International conference for internet technology and secured transactions (ICITST'17). Cambridge, MA, USA. IEEE; 2017. p. 498-500.
37. Wang M, Ichijo H, Xiao B. Cryptocurrency address clustering and labeling. arXiv preprint arXiv:200313399. 2020.
38. Victor F. Address clustering heuristics for Ethereum. In: Proceedings of the international conference on financial cryptography and data security (FC'20). Kota Kinabalu, Malaysia. Springer; 2020. p. 617-33.
39. Wu J, Yuan Q, Lin D, You W, Chen W, Chen C, et al. Who are the phishers? Phishing scam detection on Ethereum via network embedding. IEEE Trans Syst Man Cybernet Syst. 2020:1-11.
40. Chen W, Zheng Z, Ngai ECH, Zheng P, Zhou Y. Exploiting blockchain data to detect smart Ponzi schemes on Ethereum. IEEE Access. 2019;7:37575–86.
41. Wang J, Chen P, Yu S, Xuan Q. Tsgn: Transaction subgraph networks for identifying ethereum phishing accounts. In: Proceedings of the international conference on blockchain and trustworthy systems (BlockSys'21). Virtual location. Springer; 2021. p. 187-200.
42. Lin D, Chen J, Wu J, Zheng Z. Evolution of ethereum transaction relationships: Toward understanding global driving factors from microscopic patterns. IEEE Trans Comput Soc Syst. 2021:1-12.
43. Xie Y, Zhou J, Wang J, Zhang J, Sheng Y, Wu J, et al. Understanding ethereum transactions via network approach. In: Graph data mining. Springer; 2021. p. 155-76.
44. Xie Y, Jin J, Zhang J, Yu S, Xuan Q. Temporal-Amount Snapshot MultiGraph for Ethereum Transaction Tracking. In: Proceedings of the international conference on blockchain and trustworthy systems (BlockSys21). Virtual location. Springer; 2021. p. 133-46.
45. Zhang D, Chen J, Lu X. Blockchain Phishing scam detection via multi-channel graph classification. In: Proceedings of the international conference on blockchain and trustworthy systems (BlockSys'21). Virtual Location. Springer; 2021. p. 241–56.
46. Koohi-Var T, Zahedi M. Cross-domain graph based similarity measurement of workflows. J Big Data. 2018;5(1):1–16.

47. Ebrahimi F, Asemi A, Nezarat A, Ko A. Developing a mathematical model of the co-author recommender system using graph mining techniques and big data applications. J Big Data. 2021;8(1):1–15.
48. Maduako I, Wachowicz M, Hanson T. STVG: an evolutionary graph framework for analyzing fast-evolving networks. J Big Data. 2019;6(1):1–24.
49. Maslov S, Redner S. Promise and pitfalls of extending Google's PageRank algorithm to citation networks. J Neurosci. 2008;28(44):11103–5.
50. Wold S, Esbensen K, Geladi P. Principal component analysis. Chemom Intell Lab syst. 1987;2(1–3):37–52.
51. Karim F, Majumdar S, Darabi H, Harford S. Multivariate LSTM-FCNs for time series classification. Neural Netw. 2019;116:237–45.
52. Baydogan MG, Runger G. Learning a symbolic representation for multivariate time series classification. Data Min Knowl Discov. 2015;29(2):400–22.
53. Schäfer P, Leser U. Multivariate time series classification with WEASEL+ MUSE. arXiv preprint arXiv:171111343. 2017.
54. Hossin M, Sulaiman MN. A review on evaluation metrics for data classification evaluations. Int J Data Min Knowl Manage Process. 2015;5(2):1.
55. Chowdhury S, Khanzadeh M, Akula R, Zhang F, Zhang S, Medal H, et al. Botnet detection using graph-based feature clustering. J Big Data. 2017;4(1):1–23.

Publisher's Note