

Deep Fusion of Lead-lag Graphs: Application to Cryptocurrencies

Hugo Schnoering

Napoleon Group

hugo.schnoering@napoleon-group.com

Hugo Inzirillo

CREST Institut Polytechnique de Paris

Napoleon Group

hugo.inzirillo@napoleon-group.com

Abstract—The study of time series has motivated many researchers, particularly on the area of multivariate-analysis. The study of co-movements and dependency between random variables leads us to develop metrics to describe existing connection between assets. The most commonly used are correlation and causality. Despite the growing literature, some connections remained still undetected. The objective of this paper is to propose a new representation learning algorithm capable to integrate synchronous and asynchronous relationships.

I. INTRODUCTION

Recently, cryptocurrencies have become a center of interest for key players in financial markets and financial institutions. Despite this strong interest in cryptocurrencies, the youth of these new digital assets makes them difficult to model. Connections between pairs are not well established. Several papers point out common behaviour patterns between quotes of several cryptocurrencies [1, 2]. Many efforts have already been devoted to understanding correlations in financial markets and joint dynamics. A large number of research papers are concentrated on Pearson’s correlation coefficient to detect synchronous relations on equity returns [3, 4] and cryptocurrency returns [5]. There are however two limits to this approach:

- it is only sensitive to linear dependencies,
- temporality is broken and it is therefore not sensitive to assets driving each other asynchronously, i.e. to lead-lag relationships.

Statistical analyses using only correlation might not detect any connection between assets. Complexity of financial markets suggests that relationships between assets are not bound to be linear. Several papers already support this hypothesis [6, 7]. In order to detect non linear relations, more general measures may be used: e.g. the Spearman’s rank correlation or the Kendall’s rank correlation. Concerning the lead-lag relationships between assets, they cannot exist in the framework introduced by the standard financial theory. In particular, the Efficient Market Hypothesis (EMH) of Fama [8] states that prices contain all available information. Consequently, it is not possible to predict future returns from the past. It has been therefore demonstrated in several works that this hypothesis is not verified in practice [9, 10]. Interest of some researchers has thus shifted from the analysis of synchronous relationships to the analysis of asynchronous relationships.

Curme et al. [11] study lagged linear relationships between i and j by computing the correlation coefficient between the

returns of i and the forward-shifted returns of j . Fiedor [12] undertakes the same methodology of Curme et al. and replaces the correlation coefficient with the *mutual information* in order to detect non-linear relations between the returns of i and the lagged returns of j . Given a lag T and a set of assets, Fiedor even constructs a *lead-lag graph* in which assets are nodes, and edges represent validated relationships between assets. Such a graph accounts for relations that may exist between assets when information takes T to be propagated. Fiedor also introduces a statistical test to determine whether a found relationship is statistically significant or not. This is done by computing a p-value for each lagged relationship, and then setting a threshold on this p-values.

Empirical results in [12] suggest that the sampling frequency of prices denoted by d plays an important role: relationships between assets are not consistent with the sampling frequency. Our first contribution is to investigate whether statistically significant lagged relationships exist in cryptocurrency markets for sampling periods equal to 1 minute and 5 minutes.

An asset x can be described in different manners: its advisory board, its supply network, its last returns, etc. Finding an embedding / representation of x living in a Euclidean space can be useful because it allows to do basic arithmetic with x . Lots of distance can thus be used to compare two assets, opening up the possibility for clustering. Given a lead-lag graph \mathcal{G} characterized by (T, d) , representations of its nodes / assets can be derived from the graph \mathcal{G} with various approaches such as *Random Walk* [13] or *Node2vec* [14]. Representation of an asset should however not be limited to a lag T and a frequency d , but should take into account the different relationships that may exist between the assets. Given a set of lead-lag graphs obtained with the methodology of Fiedor [12], the main contribution of this article is to learn a fused representation of the assets from several lead-lag graphs.

Several works are dedicated to "fusing graphs" [15, 16, 17]. In this paper, we use the model *deepNF* introduced by Gligorićević et al. [18]. Given a set of graphs sharing the same set of nodes, *deepNF* learns in an unsupervised manner node where the representation accounts for information taken from the different input graphs.

Finally, relations between assets may vary over time. This is especially true in our case, because cryptocurrency markets are in the early adoption phase, and new projects emerge

every day. In our approach, asset representations are regularly updated by taking into account only the near past. A cosine similarity can then be used to monitor the evolution of the similarity between a pair of assets over time. From this similarity measure, we can also derive a similarity matrix which could be used for several purposes.

II. METHODS

Notations:

- n the number of assets
- $p + 1$ the number of dates
- $\{t_\tau\}_{\tau=0}^p \triangleq \{t_0, t_1, t_2, \dots, t_p\}$ the dates arranged in the chronological order
- d the common time span between two consecutive dates
- $P_j(t_i)$ the price of asset j at time t_i
- $r_j(t_i)$ the log-return of asset j between t_{i-1} and t_i
- $\mathbb{E}(X)$ is the expected value of the random variable X
- $\langle u|v \rangle$ is the dot product between vectors u and v
- $\|u\|$ is the Frobenius norm of vector u
- Let \mathbf{M} a matrix. \mathbf{M}_p is the p -th row of \mathbf{M} , $\mathbf{M}_{[p,q]}$ is the submatrix of \mathbf{M} composed of rows $p, p+1, \dots, q-1, q$, and $\mathbf{M}_{-,p}$ is the p -th column of \mathbf{M} .

A. Lead-lag Graph

Here, we present the methodology of [12] to identify non linear dependencies between financial instruments, which itself extends the methodology of [11]. The first step is the computation of the matrix \mathbf{R} of log-returns for the different assets:

$$\mathbf{R}_{i,j} = r_j(t_i) = \log(P_j(t_i)) - \log(P_j(t_{i-1})) \quad (1)$$

Let $T \in \mathbb{N}$ be the lag that will be used for forward shifting. \mathbf{R} is filtered into two matrices, $\mathbf{A}^{(T)}$ and $\mathbf{B}^{(T)}$, in which the last T returns are excluded from $\mathbf{A}^{(T)}$ and the T first returns are excluded from $\mathbf{B}^{(T)}$. From these matrices the matrix \mathbf{C} is constructed using *Mutual Information* of columns of $\mathbf{A}^{(T)}$ and $\mathbf{B}^{(T)}$:

$$C_{m,n} := I_S(\mathbf{A}_{-,m}^{(T)}, \mathbf{B}_{-,n}^{(T)}) \quad (2)$$

where $I_S(\cdot, \cdot)$ is the *Mutual Information* operator between two random variables. *Mutual Information* for two discrete random variables X and Y is defined by equation 3.

$$\begin{aligned} I_S(X, Y) &= D_{KL}\left(p(x, y)||p(x)p(y)\right) \\ &= \mathbb{E}_{p(x,y)} \log\left(\frac{p(x,y)}{p(x)p(y)}\right) \\ &= \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right), \end{aligned} \quad (3)$$

where $p(x, y)$ is the joint probability distribution of (X, Y) , $p(x)$ and $p(y)$ are the marginal probability distributions, and \mathcal{X} and \mathcal{Y} are the sets of values that X and Y can take respectively. For continuous variables the definition remains valid by using integrals and probability density functions. It

is straightforward to demonstrate that the mutual information between two independent variables is equal to 0. In addition, the mutual information is non negative. For this reason, if there is no statistical relation between the lagged returns A_m and the returns B_n , we expect $C_{m,n}$ to be approximately equal to 0.

Even if log-returns are continuous variables, they are discretized for estimating the mutual information. Following [12], the mutual information is estimated with the plug-in estimator, which is the mutual information between the discretized empirical distributions. Once the coefficient $C_{i,j} = I_S(A_i, B_j)$ has been estimated by using the plug-in estimator $\hat{C}_{i,j}$, it is important to determine whether this coefficient is significant or not. As in [12], we use an approximation of the mutual information between two random variables by a gamma distribution in order to build a statistical test of significance. It has been shown in [19] that the mutual information between independent discrete random variables X and Y when estimated from relative frequencies follows a very good approximation of Gamma distribution \mathcal{G} with parameters

$$\alpha = (|\mathcal{X}| - 1)(|\mathcal{Y}| - 1)/2 \quad \text{and} \quad \beta = 1/(N \log(2))$$

where N is the sample size and $|\mathcal{X}|$ and $|\mathcal{Y}|$ denote the numbers of realizations of X and Y respectively. Given a threshold p , the statistical test thus checks the condition 4.

$$\hat{C}_{i,j} \leq \Gamma_{1-p}\left(\frac{1}{2}(|\mathcal{X}| - 1)(|\mathcal{Y}| - 1), \frac{1}{N \log(2)}\right) \quad (4)$$

Where $\Gamma_{1-p}(\alpha, \beta)$ is the $(1 - p)$ quantile of a Gamma distribution $\mathcal{G}(\alpha, \beta)$. If (4) is verified, the test accepts the null hypothesis, i.e. A_m and B_n are independent, and rejects it otherwise. Since we are performing multiple statistical hypothesis testings, the likelihood of incorrectly rejecting the null hypothesis increases. As in [11] and [12], we compensate for that increase by applying the Bonferroni correction : we test each individual hypothesis at a significance level of p/m where $m = n^2$ is the total number of tests.

We denote by $\tilde{\mathbf{C}} \in (\mathbb{R}^+)^{n \times n}$ the matrix for which entry (i, j) is equal to $\hat{C}_{i,j}$ if it is significant, otherwise 0. The matrix $\tilde{\mathbf{C}}$ can be seen as a weighted adjacency matrix of a directed graph. We decide to symmetrize the matrix $\tilde{\mathbf{C}}$ so that we can use most of existing node embedding algorithms:

$$\tilde{\mathbf{C}} \leftarrow \frac{\tilde{\mathbf{C}} + \tilde{\mathbf{C}}^t}{2} \quad (5)$$

B. DeepNF

In this section, we introduce the framework of *DeepNF* introduced by Gligorijević et al. [18]. *DeepNF* learns low-dimensional embedding of nodes, shared across several graphs. We consider a set of N graphs which are represented by their binary adjacency matrices $\{\mathbf{A}^{(i)}\}_{i=1}^N \triangleq \{\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)}\}$, i.e. $A_{i,j}^{(l)} = 1$ if nodes i and j are connected in graph

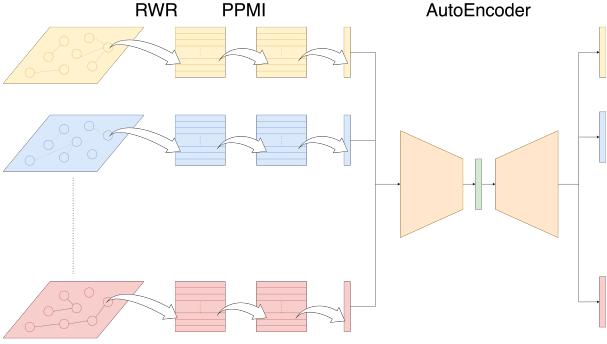


Fig. 1. Deep Network Fusion

l and 0 otherwise. Their approach is composed of three steps: first converting each graph into a high-quality vector representation with the *Random Walk with Restarts* (RWR) method, then constructing from each RWR matrix a *Positive Pointwise Mutual Information* (PPMI) matrix capturing structural information, and finally fusing PPMI matrices by using an *AutoEncoder* like model. The entire methodology is schematized in figure 1.

(1) The *Random Walk* (RW) method is an effective way to transform an unweighted graph into node representations that capture the topological structure of each node. Let $\mathbf{A} \in [0, 1]^{n \times n}$ be a binary adjacency matrix on a set of n nodes \mathcal{N} . We make the following assumption: the node i is self connected, i.e. $A_{i,i} = 1$, if and only if it is isolated, i.e. $\forall j \neq i$, $A_{i,j} = 0$. Given a starting node $i_0 \in \mathcal{N}$ and a length $L \in \mathbb{N}^*$, a random walk is a path $i_0 \rightarrow i_1 \rightarrow \dots \rightarrow i_{L-1}$, where for all $j \in \{1, \dots, L-1\}$, $i_j \in \mathcal{N}$ and i_j is uniformly sampled among the neighbours of i_{j-1} . A *Random Walk with Restart* (RWR) allows at each time step to return to i_0 with a probability $1 - \alpha$, where $\alpha \in [0, 1]$. RWR can be formulated by the recurrence relation of equation 6.

$$\mathbf{p}_i^{(t)} = \alpha \times \mathbf{p}_i^{(t-1)} \hat{\mathbf{A}} + (1 - \alpha) \times \mathbf{p}_i^{(0)}, \quad (6)$$

where for all t , $\mathbf{p}_i^{(t)}$ is a row vector of size n , whose k -th entry indicates the probability of reaching the node k starting from node i after t steps and $\hat{\mathbf{A}}$ is the one-step probability transition matrix obtained from \mathbf{A} by applying the standard row-wise normalization. In order to give a representation $\mathbf{v}_i \in \mathbb{R}^n$ for the node i , Gligorijević et al. [18] adopt the method proposed by Cao et al. [20] given by the equation 7.

$$\mathbf{v}_i = \sum_{t=1}^K \mathbf{p}_i^{(t)}, \quad (7)$$

where K is the total number of RWR steps. As a consequence, the representation v_i of node i accounts for the neighbourhood of node i up to nodes that are distant of K from i . Since $p_i^{(t)}$ is a probability vector, it is worth noticing that $\sum_{j=1}^n v_{i,j} = K$. Applying the RWR step for each graph results in N representation matrices $\{\mathbf{V}^{(1)}, \dots, \mathbf{V}^{(N)}\}$ of size $\mathbb{R}^{n \times n}$:

$$\forall l \in \{1, \dots, N\}, \mathbf{V}^{(l)} = \begin{pmatrix} \mathbf{v}_1^{(l)} \\ \mathbf{v}_2^{(l)} \\ \vdots \\ \mathbf{v}_i^{(l)} \\ \vdots \\ \mathbf{v}_n^{(l)} \end{pmatrix} \quad (8)$$

As a consequence:

$$\forall l \in \{1, \dots, N\}, \forall i, j \in \{0, \dots, n\}, V_{i,j}^{(l)} = \sum_{t=1}^K \mathbf{p}_{i,j}^{(t)} \quad (9)$$

(2) Given a RWR matrix $\mathbf{V}^{(l)}$, we denote by $\tilde{\mathbf{V}}^{(l)}$ the matrix $\mathbf{V}^{(l)}$ normalized:

$$\tilde{\mathbf{V}}^{(l)} = \frac{\mathbf{V}^{(l)}}{\sum_{q=1}^n \sum_{r=1}^n V_{qr}^{(l)}} = \frac{\mathbf{V}^{(l)}}{n \times K}, \quad (10)$$

$\tilde{\mathbf{V}}^{(l)}$ can be seen as the matrix of probabilities of two variables (X, Y) taking their values in $\{1, \dots, n\}$, i.e. $p(X = i, Y = j) = \tilde{\mathbf{V}}_{ij}^{(l)}$. It is straightforward that $p(X = i) = \sum_{j=1}^n \tilde{\mathbf{V}}_{ij}^{(l)} = n^{-1}$ and $p(Y = j) = \sum_{i=1}^n \tilde{\mathbf{V}}_{ij}^{(l)}$. We construct the corresponding PPMI matrix $\mathbf{P}^{(l)}$ as follows:

$$\forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, n\}$$

$$\begin{aligned} \mathbf{P}_{i,j}^{(l)} &= \max \left(0, \log \left(\frac{p(X = i, Y = j)}{p(X = i)p(Y = j)} \right) \right) \\ &= \max \left(0, \log \left(\frac{\tilde{\mathbf{V}}_{ij}^{(l)}}{\left(\sum_{j=1}^n \tilde{\mathbf{V}}_{ij}^{(l)} \right) \left(\sum_{i=1}^n \tilde{\mathbf{V}}_{ij}^{(l)} \right)} \right) \right) \\ &= \max \left(0, \log \left(\frac{n \times \mathbf{V}_{ij}^{(l)}}{\sum_{q=1}^n \mathbf{V}_{qq}^{(l)}} \right) \right). \end{aligned} \quad (11)$$

At the end of this step, each node i is represented by N row vectors $\{\mathbf{P}_i^{(j)}\}_{j=1}^N \triangleq \{\mathbf{P}_i^{(1)}, \dots, \mathbf{P}_i^{(N)}\}$, where for all $l \in \{1, \dots, N\}$, $\mathbf{P}_i^{(l)} \in \mathbb{R}^n$.

(3). Let \mathcal{X} and \mathcal{Z} be two mathematical spaces. An *AutoEncoder* is the combination of two parametrized functions $\Phi : \mathcal{X} \rightarrow \mathcal{Z}$ and $\Psi : \mathcal{Z} \rightarrow \mathcal{X}$, respectively the *encoder* and the *decoder*. The associated parameters are denoted by θ_Φ and θ_Ψ . Let $x \in \mathcal{X}$ be a sample, we define the embedding $z \in \mathcal{Z}$ and the reconstruction $\hat{x} \in \mathcal{X}$ of x respectively by $z := \phi(x, \theta_\Phi)$ and $\hat{x} := \Psi(z, \theta_\Psi)$. If the dimension of \mathcal{Z} is smaller than the dimension of \mathcal{X} , z is a compressed representation of x . In order to find a meaningful representation z , the objective is to minimize a reconstruction error between x and \hat{x} . In the following, we build our encoder and decoder with neural networks called *multilayer perceptron* [21] (MLP).

Multilayer perceptron refers to a feedforward neural network composed of multiple linear layers with non linear activation. A *multilayer perceptron* can thus be described by its activation function and the ordered output dimensions of its linear layers, namely the latent dimensions. An *MLP encoder* (resp. *decoder*) is a multilayer perceptron, whose latent dimensions are decreasing (resp. increasing) w.r.t. the depth. Next, we define the model used by Gligorijević et al. [18] to fuse the different representations (found at the end of step 2) of each node. Given a graph $l \in \{1, \dots, N\}$, we compress the representation $\mathbf{P}_i^{(l)}$ of node i into $\mathbf{z}_i^{(l)}$ using a MLP encoder $e\text{MLP}^{(l)}$ (equation 12).

$$\forall j \in \{1, \dots, N\}, \forall i \in \{1, \dots, n\}, \mathbf{z}_i^{(l)} = e\text{MLP}^{(l)} \left(\mathbf{P}_i^{(l)} \right) \quad (12)$$

Next, we concatenate for each node i its representations $(\mathbf{z}_i^{(l)})_{1 \leq l \leq N}$ obtained previously (equation 13) and feed it to a MLP encoder $e\text{MLP}$ in order to get the fused representation \mathbf{z}_i of node i (equation 14).

$$\forall i \in \{1, \dots, n\}, \mathbf{x}_i = \text{Concat} \left(\mathbf{z}_i^{(1)}, \dots, \mathbf{z}_i^{(N)} \right) \quad (13)$$

$$\forall i \in \{1, \dots, n\}, \mathbf{z}_i = e\text{MLP}(\mathbf{x}_i) \quad (14)$$

Then, \mathbf{z}_i is fed to a MLP decoder $d\text{MLP}$ and give the decompressed representation $\hat{\mathbf{x}}_i$ of \mathbf{z}_i . $\hat{\mathbf{x}}_i$ is then split into N non overlapping vectors $\{\hat{\mathbf{z}}_i^{(1)}, \dots, \hat{\mathbf{z}}_i^{(N)}\}$. Finally, each vector $\hat{\mathbf{z}}_i^{(l)}$ is fed to a MLP decoder $d\text{MLP}^{(l)}$ whose last latent dimension is equal to the dimension of $\mathbf{P}_i^{(l)}$. As a consequence, if we denote by $\hat{\mathbf{P}}_i^{(l)}$ the output of $d\text{MLP}^{(l)}$, $\hat{\mathbf{P}}_i^{(l)}$ has the same size than $\mathbf{P}_i^{(l)}$. This network aims at finding a meaningful representation \mathbf{z}_i of node i , and this representation is validated and refined by attempting to regenerate the inputs $\{\mathbf{P}_i^{(1)}\}_{i=1}^N \triangleq \{\mathbf{P}_i^{(1)}, \dots, \mathbf{P}_i^{(N)}\}$ from \mathbf{z}_i . For this reason, we try to find the optimal parameters that minimize the reconstruction loss \mathcal{L} between each original and reconstructed PPMI matrix (equation 15).

$$\mathcal{L} \triangleq \frac{1}{N} \sum_{l=1}^N L(\hat{\mathbf{P}}_i^{(l)}, \mathbf{P}_i^{(l)}) \quad (15)$$

where L is the mean squared error. This loss function can be optimized by a standard back-propagation algorithm. After the training of the model is done, we extract for each node i its low-dimensional feature vector \mathbf{z}_i , in the following called embedding of asset i .

C. Deep Fusion of Lead-lag Graphs

Let $\{(d_i, T_i)\}_{i=1}^N \triangleq \{(d_1, T_1), (d_2, T_2), \dots, (d_N, T_N)\}$ be a set of tuples of $\mathbb{R} \times \mathbb{N}$ composed of a sampling period d and a lag T . For all $l \in \{1, \dots, N\}$, we denote by $\tilde{\mathbf{C}}^{(l)}$ the $T^{(l)}$ lead-lag network obtained from the series of prices $\{\mathbf{P}_i^{(l)}\}_{i=1}^N \triangleq \{\mathbf{P}^{(1)}, \dots, \mathbf{P}^{(N)}\}$ sampled with the frequency $1/d^{(l)}$. It is worth noticing that two distinct value of d lead to two different matrices of returns (equation 1) because the

series of prices are not sampled with the same frequency. In the following, we will denote by $\mathbf{R}^{(d)}$ the matrix of returns derived from the sampling period d when the confusion is possible. The framework of deepNF enables to find a representation of each asset from $\{\tilde{\mathbf{C}}^{(i)}\}_{i=1}^N \triangleq \{\tilde{\mathbf{C}}^{(1)}, \dots, \tilde{\mathbf{C}}^{(N)}\}$. These representations have the particularity to contain information from multiple multi-scale non-linear lagged relationships between assets. It is worth noticing that this approach is not only limited to lead-lag graphs, any kind of graph whose nodes are assets can be incorporated into the representation learning process, for example the network of cryptocurrency sectors or the network of cryptocurrency investment firms.

D. Dynamic Lead-lag Graphs

Since we want to detect new relations between assets, we have to reevaluate the relations between assets, and then the lead-lag graphs, and to update the representation of each asset on a regular basis. Concretely we choose m distinct dates among the $p+1$ possible dates, it is equivalent to choose a non decreasing injective function $\chi : \{0, \dots, m-1\} \rightarrow \{0, \dots, p\}$. Given an index $k \in \{0, \dots, m-1\}$ and window size $w \in \mathbb{N}^*$, the date of the sample end is $t_{\chi(k)}$, in addition, we keep only data in a lookback window $[t_{\chi(k)-w+1}, t_{\chi(k)}]$ in order to catch recent dynamics. It concretely means to keep only the rows in \mathbf{R} corresponding to the dates $t_{\chi(k)-w+1}, t_{\chi(k)-w+2}, \dots, t_{\chi(k)}$, i.e. $\mathbf{R}_{[\chi(k)-w+1, \chi(k)]}$. From $\mathbf{R}_{[\chi(k)-w+1, \chi(k)]}$, we compute the N lead-lag graphs $\{\tilde{\mathbf{C}}^{(i)}(t_{\chi(k)})\}_{i=1}^N$ and their binary adjacency matrices $\{\mathbf{A}^{(i)}(t_{\chi(k)})\}_{i=1}^N$. DeepNF is then trained to fuse the set of adjacency matrices obtained over all dates $\{\{\mathbf{A}^{(i)}(t_{\chi(k)})\}_{i=1}^N\}_{k=0}^{m-1}$. Finally, embeddings $\{\{z_j(t_{\chi(k)})\}_{j=1}^n\}_{k=0}^{m-1}$ are extracted from the encoder. Given an asset $j \in \{1, \dots, n\}$, m embeddings are available, one for each date $\{t_{\chi(k)}\}_{k=0}^{m-1}$. As a consequence, our framework allows the considered assets to have a time-varying representation.

III. MATERIALS AND RESULTS

A. Lead-lag Graphs

In order to find mutual information-based lagged relationships between cryptoassets, we have downloaded the quotes on a minute-by-minute basis of all assets present in the weekly ranking on market capitalization established by CoinMarketCap (<https://coinmarketcap.com/fr/historical/>) on the 13th October 2019. We have filtered out assets that were not listed on Binance (www.binance.com) at this date and quotes¹ of the resulting $n = 69$ assets² have been downloaded from the Binance API (<https://binance-docs.github.io>). The data cover 705 days between 12th September 2019 and 7th November 2021. Each sample corresponds to n series of quotes from a

¹The quote considered is *token/USDT*.

²The remaining tokens are: ADA, ALGO, ANKR, ATOM, BAND, BAT, BEAM, BNB, BTC, BTT, BUSD, CELR, CHZ, COCOS, COS, CVC, DASH, DENT, DOGE, ENJ, EOS, ETC, ETH, FET, FTM, FUN, GTO, HBAR, HOT, ICX, IOST, KAVA, KEY, LINK, LTC, MATIC, MFT, MITH, MTL, NANO, NEO, NKN, NULS, OMG, ONE, ONG, ONT, PERL, QTUM, REN, RVN, STX, TFUEL, THETA, TOMO, TRX, TUSD, USDC, VET, WAN, WAVES, WIN, XLM, XMR, XRP, XTZ, ZEC, ZIL, ZRX

Data: $\{P_j\}_{j=1}^n, \{(d_i, T_i)\}_{i=1}^N, w, \chi$
Result: Dynamic embeddings $\{\{z_j(t_{\chi(k)})\}_{j=1}^n\}_{k=0}^{m-1}$

Initialization;

```

for  $l = 1, \dots, N$  do
    Compute the matrix of returns  $\mathbf{R}^{(d_l)}$  with the
    frequency  $d_l$ 
end

Training;
for  $k = 1, \dots, m$  do
     $t \leftarrow \chi(k);$ 
    for  $l = 1, \dots, N$  do
         $(d, T) \leftarrow (d_l, T_l);$ 
         $R \leftarrow \mathbf{R}_{t-w+1:t,:}^d;$ 
        Construct the  $T$  lead-lag graph  $\tilde{C}^{(l)}(t)$  from  $R$ ;
        Derive the binary adjacency matrix  $A^{(l)}(t)$ 
        from  $\tilde{C}^{(l)}(t)$ 
    end
end

Train DeepNF to fuse  $\{\{A^{(l)}(\chi(k))\}_{l=1}^N\}_{k=0}^{m-1};$ 
Ending;
Extract  $\{\{z_j(t_{\chi(k)})\}_{j=1}^n\}_{k=0}^{m-1}$  from the encoder of
DeepNF

```

Algorithm 1: Dynamic Deep Fusion

lookback window of size $w = 1440$ minutes, i.e. 24 hours of data, ending at the end of a day. As a result there is no overlap between two consecutive samples. We compute from the series of quotes the series of log-returns using two sampling periods : 1 minute and 5 minutes. For the purpose of estimating mutual information, these log-returns are sample- and asset-wise discretized into 4 distinct states. As in [12], the states represent equal parts, therefore each state is assigned the same number of data points. We have set the uncorrected p-value p (equation 4) equal to 0.01, using the Bonferroni correction this p-value is set to $0.01/69^2$. On figures 2 and 3 we plot for each sample, i.e. for each date, the number of validated mutual information-based links for the first 15 lags for log-returns sampled with periods 1 minute and 5 minutes respectively. In addition, we report on tables I and II some lag-wise statistics for the sample periods 1 minutes and 5 minutes respectively. The lead-lag graphs themselves for different values of T , three distinct dates and both sampling periods are shown on figures 10::27. We recall that nodes are assets, and that an edge from an asset to another is drawn if a significant lagged relationship has been found between those assets.

B. Deep Fusion of Lead-lag Graphs

In the following, for both sampling periods, we will only consider the lead-lag networks obtained for the three first lags. We compute the first representation of each node by applying the first two steps of the deepNF approach. The model used in the third step is an *AutoEncoder* in which we choose the ReLU function as activation function and the output dimensions are $[N \times 25, N \times 10, 30, 15, 30, N \times 10, N \times 25, N \times 100]$, where

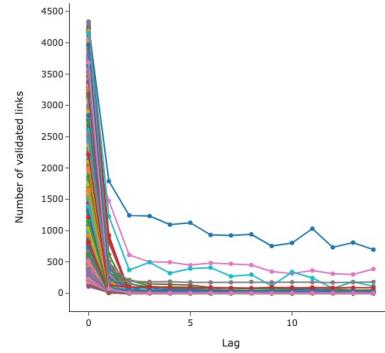


Fig. 2. Sampling period : $d = 1$ minute. Number of validated links for different lags T . One curve per date.

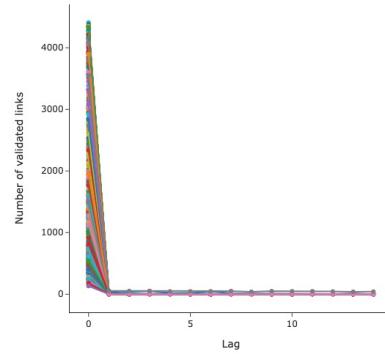


Fig. 3. Sampling period : $d = 5$ minutes. Number of validated links for different lags T . One curve per date.

$N = 6$. The model is schematized on figure 4. The data is split into a train (70%) and a validation (30%) set. We use the mean square error (MSE) as reconstruction error, and train the model by the algorithm Adam with a learning rate set equal to 0.001. The training lasts at most 500 epochs, and is stopped if overfitting is detected in the evolution of validation loss.

C. Asset Representation

At the end of the training, the embeddings are extracted from the encoder. Given an asset, one embedding per sample / day is available. For a given asset i , we will denote by $z_i(t)$ the embedding of i at date t . From all those embeddings, we train a model of *Principal Component Analysis* (PCA) with two components. We plot the projected embeddings of a subset of 16 assets on figure 5.

We also plot the projected embeddings of the whole set of assets for the dates 8th December 2019, 8th August 2020 and 8th September 2021 on figures 6, 7 and 8, respectively.

Finally, a possible application of those representations is to monitor the temporal evolution of the similarity S (equation 16) between two assets. We plot on figure 9 the evolution of

Lag	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Min	103	3	0	0	0	0	0	0	0	0	0	0	0	0	0
Quantile 25 %	601	15	1	0	0	0	0	0	0	0	0	0	0	0	0
Median	1332	20	2	1	1	1	1	1	1	1	0	0	0	0	0
Quantile 75 %	2881	35	4	3	2	2	2	2	2	1	2	1	1	1	1
Max	4341	1790	1243	1232	1095	1125	931	923	940	753	803	1031	732	809	696

TABLE I

SAMPLING PERIOD : $d = 1$ MINUTE. STATISTICS ON THE NUMBER OF VALIDATED LINKS FOR DIFFERENT LAGS T .

Lag	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Min	133	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Quantile 25 %	649	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Median	1339	2	0	0	0	0	0	0	0	0	0	0	0	0	0
Quantile 75 %	3037	3	1	0	0	0	0	0	0	0	0	0	0	0	0
Max	4441	56	53	53	45	47	45	52	50	40	48	49	44	40	42

TABLE II

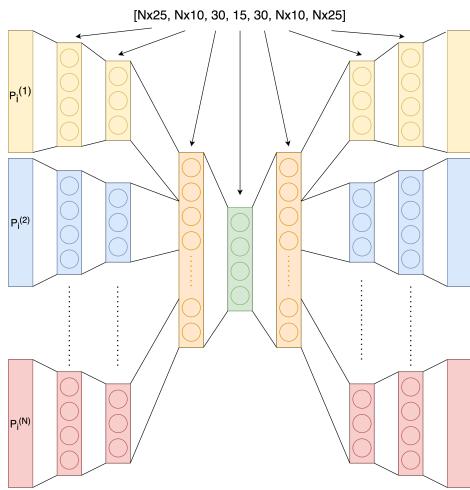
SAMPLING PERIOD : $d = 5$ MINUTE. STATISTICS ON THE NUMBER OF VALIDATED LINKS FOR DIFFERENT LAGS T .

Fig. 4. AutoEncoder

- FTM
- DOGE
- ETC
- ZEC
- BNB
- BTC
- ETH
- ZIL
- ATOM
- THETA
- XTZ
- LINK
- ALGO
- ADA
- LTC
- WAVES

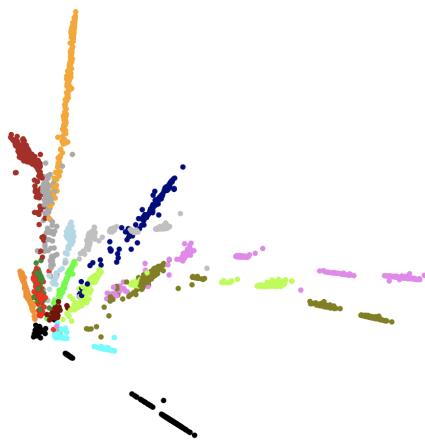


Fig. 5. 2D PCA projection of embeddings for the different dates of a subset of the studied assets.

the similarity between the asset BTC and assets ETH, LTC, BNB and DOGE.

$$\forall i, j \in \{1, \dots, n\}, S_{\cos}(\mathbf{z}_i(t), \mathbf{z}_j(t)) \triangleq \frac{\langle \mathbf{z}_i(t) | \mathbf{z}_j(t) \rangle}{\|\mathbf{z}_i(t)\| \|\mathbf{z}_j(t)\|} \quad (16)$$

IV. DISCUSSION

Figures 2, 3 and tables I and II demonstrate that a significant number of synchronous and asynchronous relations exist between assets. The number of asynchronous relations rapidly decreases to 0 when the lag becomes large. Three dates are exception : the 19th May of 2021, the 12th March of 2020 and the 7th September of 2021, which all correspond to large bearish moves in the cryptocurrency markets. According to tables I and II, there are a larger number of lagged relationships on a minute-to-minute basis than on a basis of 5 minutes. We deduce from the figures 10:27 that the graphs are significantly changing from one sample / day to another, and are unequally dense. In particular the graphs corresponding to the day of the large bearish movement of the 8th September of 2021 is extremely dense.

On figure 5, each asset seems to occupy its own area localized near a line in the plane. We notice that the central area (0,0) concentrates a large number of embeddings. Figures 6, 7 and 8 seem to indicate that embeddings are concentrated during days of large moves (8th September 2021) and are more smeared during normal days (8th December 2019 and 8th August 2020). Stablecoins USDC, USDT and BUSD stand out from the others on figure 8, it is however not surprising because they are not extensively impacted by large market moves. In a further study it would be interesting to study the relationship between the market return and the discrepancy of the embedding space for a given day.

A similarity measure between two assets may be extremely useful, for example to minimize the risk of a portfolio. Similar assets can also be good candidates for pair trading strategies. On figure 16, we can notice that the similarity between LTC and BTC is high and stable, it may be because LTC is a fork of BTC. Similarity between BTC and DOGE became only important since August 2020. In contrary, similarity between BTC and ETH or BNB remains globally weak.

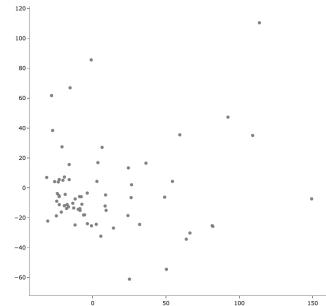


Fig. 6. 2D PCA projection of the embeddings on the 8th December of 2019.

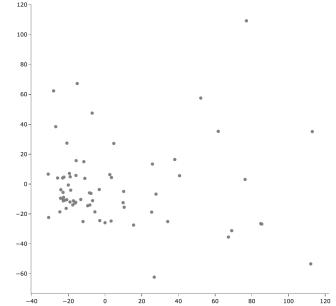


Fig. 7. 2D PCA projection of the embeddings on the 8th August of 2020.

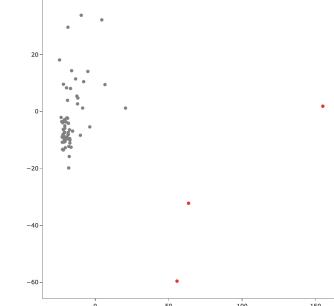


Fig. 8. 2D PCA projection of the embeddings on the 8th September of 2021.

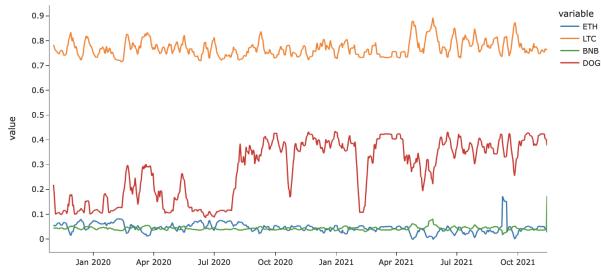


Fig. 9. Evolution of the similarity between the token BTC and the tokens ETH, LTC, BNB et DOGE.

V. CONCLUSION

In this paper, we have proposed a new approach to compute dynamic asset representations, which take into account the synchronous and asynchronous relations that may exist within a basket of assets. At the same time we have demonstrated the existence of those relations for sampling periods 1 minute and 5 minutes. In addition, it has been found that those relations were time-varying. At this point we consider two extensions of this work. First, it would be interesting that our approach supports a possibly time varying asset universe. In fact, cryptocurrencies is a relatively new sector, that is rapidly changing : new projects, and then assets, emerge every day. Secondly, from the computed dynamic embeddings, we can derive a dynamic similarity matrix by using the similarity defined by equation 16. It would be interesting to determine if those matrices can be used to construct a risk-diversified portfolio.

REFERENCES

- [1] K. Gkillas, S. Bekiros, and C. Siriopoulos, “Extreme correlation in cryptocurrency markets,” Available at SSRN 3180934, 2018.
- [2] D. Stosic, D. Stosic, T. B. Ludermir, and T. Stosic, “Collective behavior of cryptocurrency price changes,” *Physica A: Statistical Mechanics and its Applications*, vol. 507, pp. 499–509, 2018.
- [3] G. Buccheri, S. Marmi, and R. N. Mantegna, “Evolution of correlation structure of industrial indices of us equity markets,” *Physical Review E*, vol. 88, no. 1, p. 012806, 2013.
- [4] M. Tumminello, F. Lillo, and R. N. Mantegna, “Correlation, hierarchies, and networks in financial markets,” *Journal of economic behavior & organization*, vol. 75, no. 1, pp. 40–58, 2010.
- [5] S. Lahajnar and A. Rožanec, “The correlation strength of the most important cryptocurrencies in the bull and bear market,” *International Management and Financial Innovation*, vol. 17, no. 3, pp. 67–81, 2020.
- [6] W. A. Brock, W. A. Brock, D. A. Hsieh, B. D. LeBaron, and W. E. Brock, *Nonlinear dynamics, chaos, and instability: statistical theory and economic evidence*. MIT press, 1991.
- [7] D. Sornette and J. V. Andersen, “A nonlinear super-exponential rational model of speculative financial bubbles,” *International Journal of Modern Physics C*, vol. 13, no. 02, pp. 171–187, 2002.
- [8] E. F. Fama, “Efficient market hypothesis,” *Diss. PhD Thesis, Ph. D. dissertation*, 1960.
- [9] T.-H. Lu and Y.-M. Shiu, “Tests for two-day candlestick patterns in the emerging equity market of taiwan,” *Emerging markets finance and trade*, vol. 48, no. sup1, pp. 41–57, 2012.
- [10] J. Nagayasu, *The efficiency of the Japanese equity market*. Emerald Group Publishing Limited, 2003.
- [11] C. Curme, M. Tumminello, R. N. Mantegna, H. E. Stanley, and D. Y. Kenett, “Emergence of statistically validated financial intraday lead-lag relationships,” *Quantitative Finance*, vol. 15, no. 8, pp. 1375–1386, 2015.
- [12] P. Fiedor, “Information-theoretic approach to lead-lag effect on financial markets,” *The European Physical Journal B*, vol. 87, no. 8, pp. 1–9, 2014.
- [13] F. Spitzer, *Principles of random walk*. Springer Science & Business Media, 2013, vol. 34.
- [14] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.
- [15] M. R. Khan and J. E. Blumenstock, “Multi-gcn: Graph convolutional networks for multi-view networks, with ap-

- plications to global poverty,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 606–613.
- [16] W. Zhang, J. Mao, Y. Cao, and C. Xu, “Multiplex graph neural networks for multi-behavior recommendation,” in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2313–2316.
 - [17] Y. Shi, F. Han, X. He, X. He, C. Yang, J. Luo, and J. Han, “mvn2vec: Preservation and collaboration in multi-view network embedding,” *arXiv preprint arXiv:1801.06597*, 2018.
 - [18] V. Gligorijević, M. Barot, and R. Bonneau, “deepnf: deep network fusion for protein function prediction,” *Bioinformatics*, vol. 34, no. 22, pp. 3873–3881, 2018.
 - [19] B. Goebel, Z. Dawy, J. Hagenauer, and J. C. Mueller, “An approximation to the distribution of finite sample size mutual information estimates,” in *IEEE International Conference on Communications, 2005. ICC 2005. 2005*, vol. 2. IEEE, 2005, pp. 1102–1106.
 - [20] S. Cao, W. Lu, and Q. Xu, “Deep neural networks for learning graph representations,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.
 - [21] H. Ramchoun, M. A. J. Idrissi, Y. Ghanou, and M. Ettaoui, “Multilayer perceptron: Architecture optimization and training.” *Int. J. Interact. Multim. Artif. Intell.*, vol. 4, no. 1, pp. 26–30, 2016.
 - [22] R. F. Engle and C. W. Granger, “Co-integration and error correction: Representation, estimation, and testing.” *Econometrica*, vol. 55, no. 2, pp. 251–276, 1987.
 - [23] F. Filho, J. Silva, M. Bertella, and E. Brigatti, “An extensive study of stylized facts displayed by bitcoin returns,” 04 2020.
 - [24] Y. Gong and R. Huser, “Asymmetric tail dependence modeling, with application to cryptocurrency market data.”
 - [25] P. Bryant, “Geometry, statistics, probability: Variations on a common theme,” *The American Statistician*, vol. 38, no. 1, pp. 38–48.
 - [26] I. Sifat, A. Mohamad, and M. M. Shariff, “Lead-lag relationship between bitcoin and ethereum: Evidence from hourly and daily data,” *Research in International Business and Finance*, vol. 50, pp. 306–321, 2019.

APPENDIX

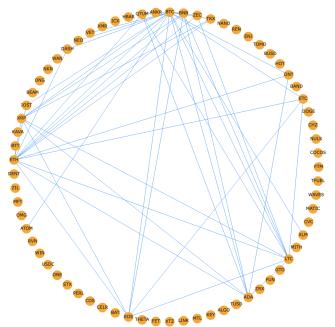


Fig. 10. 08-12-2019. Sampling period : $d = 1$ minute. Lead-lag graph for $T = 0d$.

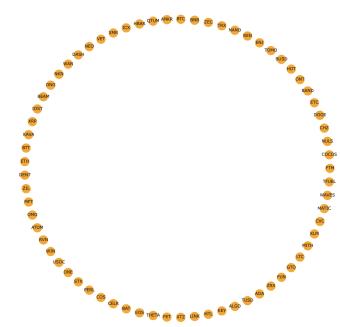


Fig. 11. 08-12-2019. Sampling period : $d = 1$ minute. Lead-lag graph for $T = 1d$.

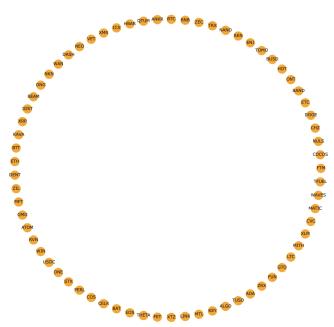


Fig. 12. 08-12-2019. Sampling period : $d = 1$ minute. Lead-lag graph for $T = 2d$.

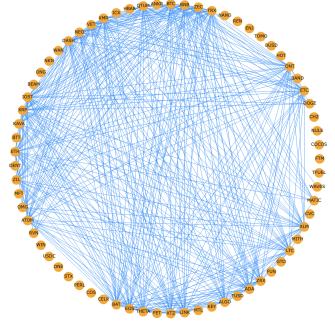


Fig. 13. 2020-08-08. Sampling period : $d = 1$ minute. Lead-lag graph for $T = 0d$.

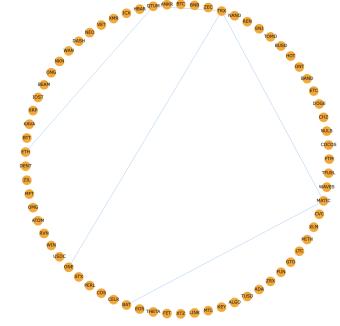


Fig. 14. 2020-08-08. Sampling period : $d = 1$ minute. Lead-lag graph for $T = 1d$.

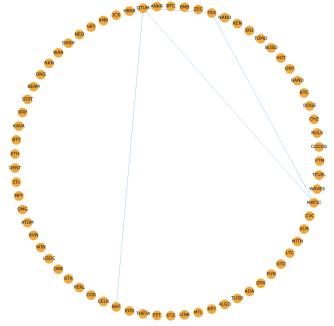


Fig. 15. 2020-08-08. Sampling period : $d = 1$ minute. Lead-lag graph for $T = 2d$.

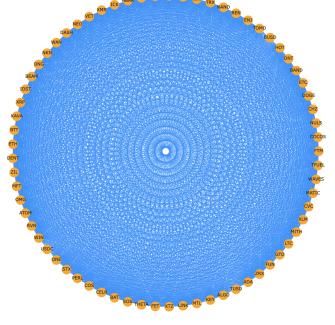


Fig. 16. 2021-09-08. Sampling period : $d = 1$ minute. Lead-lag graph for $T = 0d$.

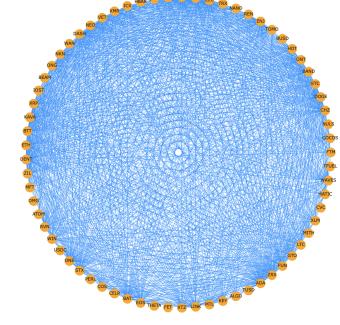


Fig. 17. 2021-09-08. Sampling period : $d = 1$ minute. Lead-lag graph for $T = 1d$.

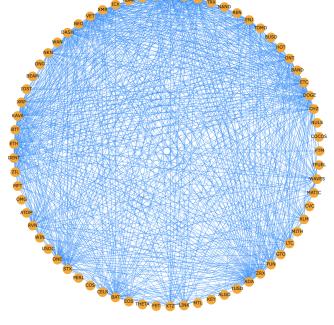


Fig. 18. 2021-09-08. Sampling period : $d = 1$ minute. Lead-lag graph for $T = 2d$.

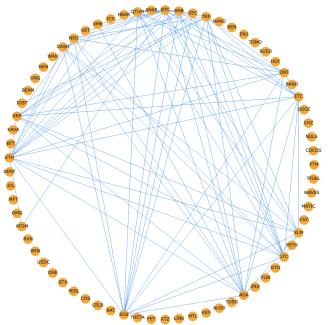


Fig. 19. 08-12-2019. Sampling period : $d = 5$ minute. Lead-lag graph for $T = 0d$.

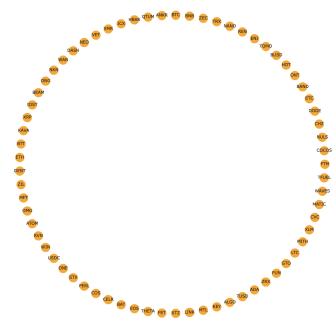


Fig. 20. 08-12-2019. Sampling period : $d = 5$ minute. Lead-lag graph for $T = 1d$.

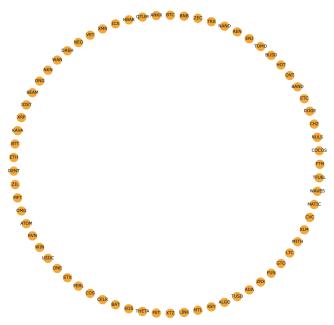


Fig. 21. 08-12-2019. Sampling period : $d = 5$ minute. Lead-lag graph for $T = 2d$.

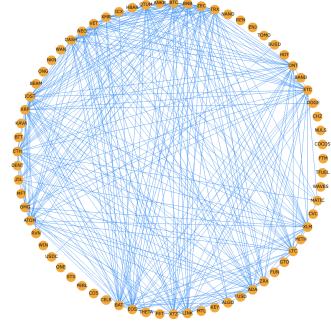


Fig. 22. 2020-08-08. Sampling period : $d = 5$ minute. Lead-lag graph for $T = 0d$.

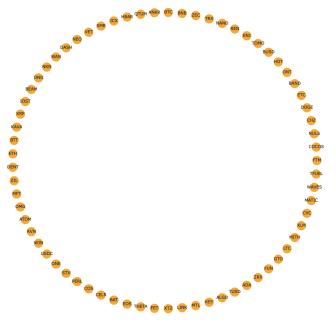


Fig. 23. 2020-08-08. Sampling period : $d = 5$ minute. Lead-lag graph for $T = 1d$.

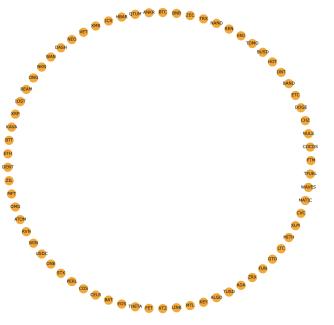


Fig. 24. 2020-08-08. Sampling period : $d = 5$ minute. Lead-lag graph for $T = 2d$.

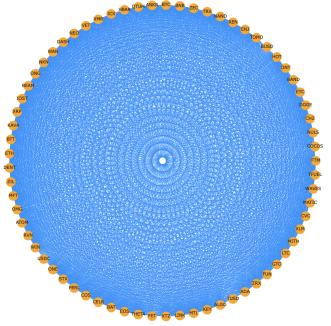


Fig. 25. 2021-09-08. Sampling period : $d = 5$ minute. Lead-lag graph for $T = 0d$.

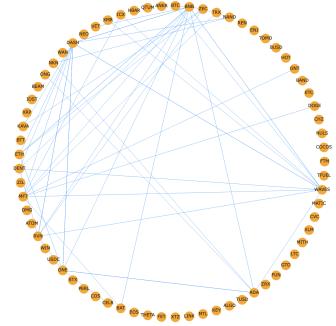


Fig. 26. 2021-09-08. Sampling period : $d = 5$ minute. Lead-lag graph for $T = 1d$.

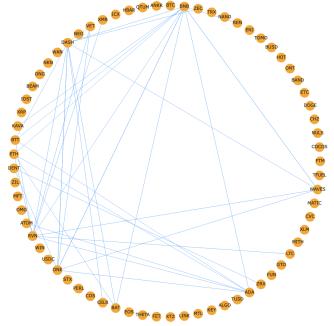


Fig. 27. 2021-09-08. Sampling period : $d = 5$ minute. Lead-lag graph for $T = 2d$.