EPJ Data Science
a SpringerOpen Journal

REGULAR ARTICLE

Open Access

Efficient modeling of higher-order dependencies in networks: from algorithm to application for anomaly detection

Mandana Saebi[1†], Jian Xu[1†], Lance M. Kaplan[2], Bruno Ribeiro[3] and Nitesh V. Chawla[1*]

*Correspondence: nchawla@nd.edu
[1] University of Notre Dame, Notre Dame, USA
Full list of author information is available at the end of the article
[†] Equal contributors

Abstract

Complex systems, represented as dynamic networks, comprise of components that influence each other via direct and/or indirect interactions. Recent research has shown the importance of using Higher-Order Networks (HONs) for modeling and analyzing such complex systems, as the typical Markovian assumption in developing the First Order Network (FON) can be limiting. This higher-order network representation not only creates a more accurate representation of the underlying complex system, but also leads to more accurate network analysis. In this paper, we first present a scalable and accurate model, `BuildHON+`, for higher-order network representation of data derived from a complex system with various orders of dependencies. Then, we show that this higher-order network representation modeled by `BuildHON+` is significantly more accurate in identifying anomalies than FON, demonstrating a need for the higher-order network representation and modeling of complex systems for deriving meaningful conclusions.

Keywords: Higher-order network; Dynamic network; Anomaly detection; Sequential data

1 Introduction

Networks are a popular way of representing rich and sparse interactions among the components of a complex system. It is, thus, critical for the network to truly represent the inherent phenomena in the complex system to avoid incorrect conclusions. Conventionally, edges in networks represent the pairwise interactions of the nodes, assuming the naive Markovian property for node interactions, resulting in the first-order network representation (FON). However, the key question is—*is this accurately representing the underlying phenomena in the complex systems?* And if the network is not accurately representing the inherent dependencies in the complex system, can we trust the analysis and results stemming from this network? The Markovian assumption for network modeling of complex system can be limiting for network analysis tasks, including community detection [3, 4], node ranking [5], and dynamic processes [6] in time-varying complex systems.

Recent research has brought to fore challenges with the FON view, especially its limitations on capturing the sequential patterns or higher- and variable-order of dependencies

Springer

in a complex system and its impact on resulting network analysis. This has led to the development of network representation models that capture such higher-order dependencies, going beyond the traditional pairwise Markovian network representation [1, 2].

Our prior work [2] tackles the limitations stemming from the Markovian assumption for node interactions (as in FON), and proposes `BuildHON` for extracting higher-order dependencies from sequential data to build the Higher-Order Network (HON) representation. `BuildHON`, although accurate, faced the challenge of computational complexity as well as parameter dependency. In this work, we address these limitations by proposing a scalable and parameter-free algorithm, `BuildHON+`, for accurate extraction of higher-order dependencies from sequential data. Given `BuildHON+`, we are also interested in downstream network analysis tasks, adn we focus on the following question in this paper that has not been addressed in prior HON work: *Does incorporating higher-order dependencies improve the performance of existing network-based methods for detecting anomalous signals in the sequential data?*

To answer the above question, we define anomalies (or change points) as deviations from the norm or expected behavior of a complex system. We note that the anomalies could also be important change points in the behavior of the complex system. The key here is to be able to accurately flag such deviations or events in a complex system. While there exists a wide range of anomaly detection methods on dynamic networks [7, 8], all of them use the first-order network (FON) to represent the underlying raw data (such as clickstreams, taxi movements, or event sequences), which can lose important higher-order information [2, 3]. As FON is an oversimplification of higher-order dynamics, we hypothesize that anomaly detection algorithms that rely on FONs will miss important changes in the network, thus leaving anomalies undetected. We systematically demonstrate why existing network-based anomaly detection methods can leave certain signals undetected, and propose a higher-order network anomaly detection framework. Consider the following example.

Example Fig. 1 illustrates the challenge of detecting certain types of anomalies, using a minimal example of web clickstreams data (sequences of web page views produced by users) collected by a local media company. Given the web clickstreams as the input to network-based anomaly detection methods, conventionally, a web traffic network is built for each time window (two one-hour windows illustrated here), with the nodes representing web pages and the edges representing total traffic between web pages. A change in the network topology indicates an anomaly in web traffic patterns. According to the original clickstreams, in the first hour, all users coming from the soccer web page to the weather page proceed to the ticket page, and all users coming from the skating page to the weather page go to TV schedules. But the flow of users is completely flipped in the next hour, possibly the weather forecast has updated with much colder weather which is in favor of winter activities. However, despite the significant changes in user web viewing patterns, the pairwise traffic between web pages in this example remains the same, thus the FON topology shows no changes. Therefore, no matter what network-based anomaly detection method is used, if the method relies on FON, the company will not be able to detect such type of anomalies, thus failing to respond (e.g., caching pages for visits, or targeted promotion of pages) to the changes in user behaviors.

Contributions. We make three main contributions in the paper.

Figure 1 Higher-order anomalies cannot be detected by network-based anomaly detection methods if FON is used
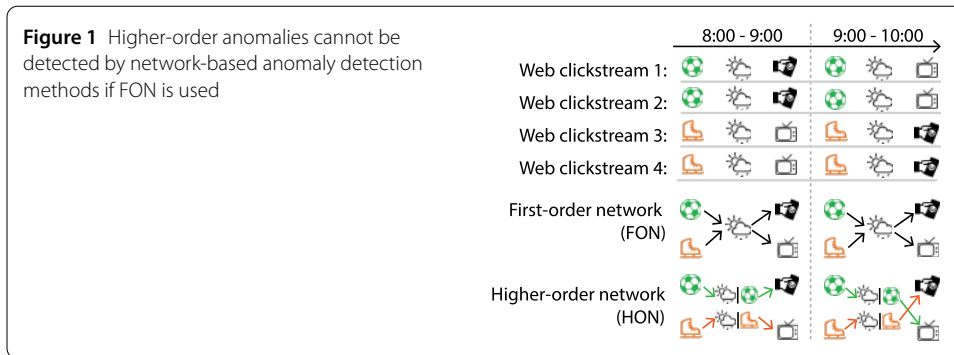
- We develop a scalable and parameter-free algorithm for higher-order network representation, `BuildHON+`, building on our prior work [2]. We demonstrate the efficiency of `BuildHON+` through comprehensive complexity and performance analysis on the global ship movement data, which is known to exhibit dependencies beyond the fifth order.
- We showcase the performance of `BuildHON+` in the task of network-based anomaly detection on a real-world taxi trajectory data. We explain why the parameter dependency in our prior work can be limiting for efficient network construction and as a result, anomaly detection.
- Using a large-scale synthetic taxi movement data with 11 billion taxi movements, we show how multiple existing anomaly detection methods that depend on FON collectively fail to capture anomalous navigation behaviors beyond first-order, and how `BuildHON+` can solve the problem.

2 Related work

Higher-order networks. Recent research has highlighted the limitations of the conventional network model for representing the sequential and indirect dependencies between the components of complex systems. Multi-layer higher-order models [9, 10], motif and clique-based higher-order models [4, 11, 12], and non-Markovian higher-order models [2, 3, 6] try to embed complex patterns that are stemming from the raw data into the network representation. Specifically, non-Markovian network models has gained a lot of attraction in many applications including social networks [13, 14], human transportation networks [2, 3, 6, 15], trade networks [16, 17], and citation networks [3]. Several research studies show how incorporating higher-order dependencies affects various network analysis tasks, including community detection [3, 4], node ranking [5], and dynamic processes [6] in the network. However, from current research studies, it is unclear what is the effect of using a higher-order network model on detecting anomalies in dynamic networks.

Anomaly detection in dynamic networks. Unlike the task of detecting anomalous nodes and edges in a single static network (such as [18]), anomaly detection in dynamic networks [7, 19] uses multiple snapshots of networks to represent the interactions of interest (such as interacting molecules [20], elements in frames of videos [21], flow of invasive species [22], etc.), then identifies the time when the network topology shows significant changes, using network distance metrics [23–25], probability methods [26], subgraph methods like [27] and more. There are many advantages of using network-based methods for the task of anomaly detection in sequential data. Aside from the availability of several different

networks, a graph structure represents the relational nature of the data, which is essential for addressing the anomaly detection problem [7]. Furthermore, the inter-dependencies of the raw data can be captured more efficiently with graph representation. This feature can be further enhanced in the higher-order representation of the network, as done in this work. The importance of higher-order patterns in different network analysis tasks has gained a lot of attention recently [1, 28]. However, one of the major challenges is that the graph search space is very large, requiring the anomaly detection methods to be scalable and efficient for large data sets [7].

Moreover, using snapshots of networks may cause the fine-grained time-stamps to be lost. Therefore, the optimal time-stamp is often data-dependent and should be identified empirically through sufficient experiments.

Nevertheless, existing methods on anomaly detection rely on conventional FON; as we will show, certain types of anomalies cannot be detected with any network-based anomaly detection methods if FON is used. Rather than proposing another approach to identify the anomalous network from a series of networks, our innovation lies in the network construction step, which ensures anomalous signals are preserved in the network in the first place.
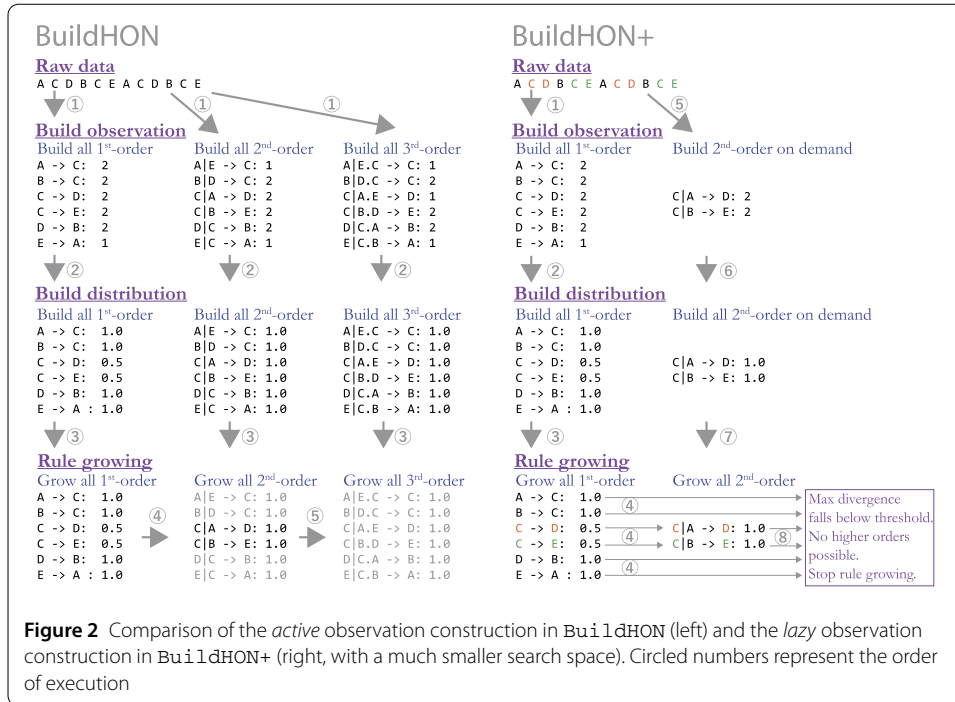
3 Methods

We first present a scalable and parameter-free approach for constructing HON, namely `BuildHON+`. We then show how this new approach enables more accurate anomaly detection (compared to using FON) by incorporating several different network distance measures. Our previous algorithm, `BuildHON` required two parameters that had to be specified experimentally, depending on the data set. Furthermore, it uses an exhaustive search for extracting the dependency rules and constructing the network, which becomes impractical for various network analysis tasks, including anomaly detection. It needs two parameters in addition to the detection threshold: a *MaxOrder* parameter which governs how many orders of dependencies the algorithm will consider in HON, and a *MinSupport* parameter that discards infrequent observations. These limitations mitigate its applicability to Big Data.

3.1 `BuildHON+`: building HON from big data

Here we introduce `BuildHON+`, a parameter-free algorithm that constructs HON from big data sets. `BuildHON+` is a practical approach that preserves higher-order signals in the network representation step ($S_i \rightarrow G_i$) which is essential for anomaly detection. The difference between `BuildHON` and `BuildHON+` is similar to the difference between pruning and early stopping in decision trees. `BuildHON` first builds a HON of all orders from first-order to *MaxOrder* and then selects branches showing significant higher-order dependencies. `BuildHON+` reduces the search space beforehand by checking in each step if increasing the order may produce significant dependencies. Furthermore, `BuildHON` can only discover dependencies up to *MaxOrder*. `BuildHON+` however, finds the appropriate dependency order hidden in the raw data and is not limited by *MaxOrder*. Therefore, the output network resulting from `BuildHON+` is a more reliable and accurate representation of the raw data, which is essential for the task of anomaly detection.

The core of `BuildHON` is the dependency rule extraction step, which answers whether higher-order dependencies exist in the raw sequential data, and how high the orders are.

Figure 2 Comparison of the *active* observation construction in `BuildHON` (left) and the *lazy* observation construction in `BuildHON+` (right, with a much smaller search space). Circled numbers represent the order of execution

The dependency rules extracted are then converted to higher-order nodes and edges as the building blocks of HON. Rather than deriving a fixed order of dependency for the whole network, the method allows for variable orders of dependencies for more compact representation. Figure 2 illustrates the dependency rule extraction step. `BuildHON` first counts the observed n-grams in the raw data (step ①), then compute probability distributions for the next steps given the current and previous steps (step ②). Finally test if knowing one more previous step significantly changes the distribution for the next step—if so, higher-order dependency exists for the path (step ④); this procedure ("rule growing") is iterated recursively until a pre-defined *MaxOrder* (shown here *MaxOrder* = 3). In this example, the probability distribution of the next steps from C changes significantly if the previous step (coming to C from A or B) is known (step ④), but knowing more previous steps (coming to C from $E \to A$ or $D \to B$) does not make a difference (step ⑤); therefore, paths $C|A \to D$ and $C|A \to E$ demonstrate second-order dependencies.

Formally, the "rule growing" process works as follows: for each path (n-gram) $\mathcal{S} = [S_{t-k}, S_{t-(k-1)}, \ldots, S_t]$ of order k, starting from the first-order $k = 1$, assume k is the true order of dependency, which $\mathcal{S}$ has the distribution D for the next step. Then extend $\mathcal{S}$ to $\mathcal{S}_{\text{ext}} = [S_{t-(k+1)}, S_{t-k}, S_{t-(k-1)}, \ldots, S_t]$ by adding one more previous step; $\mathcal{S}_{\text{ext}}$ has order $k_{\text{ext}} = k + 1$ and distribution D_{ext}. Next, test if D_{ext} is significantly different than that of D using Kullback–Leibler divergence [29] as $\mathcal{D}_{\text{KL}}(D_{\text{ext}} || D)$, and compare with a dynamic threshold δ—if the divergence is larger than δ, order $k + 1$ is assumed instead of k for the path $\mathcal{S}_{\text{ext}}$. The dynamic threshold δ is defined as $\delta = \frac{k_{\text{ext}}}{\log_2(1+Support_{\mathcal{S}_{\text{ext}}})}$, so that lower orders are preferred than higher-orders, unless higher-order paths have sufficient support (number of observations). The whole process is iterated recursively until *MaxOrder*.

3.1.1 Eliminating all parameters

The reason for having the *MaxOrder* and *MinSupport* parameters in `BuildHON` is to set a hard stop for the rule growing process, otherwise, it will iterate indefinitely and keep extending $\mathcal{S}$. However, we show that we can pre-determine if extending $\mathcal{S}$ will not produce significantly different distributions, which forms an important basis for `BuildHON+`.

Lemma 1 *The significance threshold* $\delta = \frac{k_{\text{ext}}}{\log_2(1+Support_{\mathcal{S}_{\text{ext}}})}$ *increases monotonically in rule growing when expanding* $\mathcal{S}$ *to* $\mathcal{S}_{\text{ext}}$.

Proof On the numerator, the order k_{ext} of the extended sequence $\mathcal{S}_{\text{ext}}$ increases monotonically with the inclusion of more previous steps. Meanwhile, every observations of $[S_{t-(k+1)}, S_{t-k}, \ldots, S_{t-1}, S_t]$ in the raw data can find a corresponding observation of $[S_{t-k}, \ldots, S_{t-1}, S_t]$, but not the other way around. Therefore, the support of $\mathcal{S}_{\text{ext}}$, $Support_{\mathcal{S}_{\text{ext}}} \leq Support_{\mathcal{S}}$ of the lower order $k = k_{\text{ext}} - 1$. As a result, the denominator decreases monotonically with the rule growing process. $\square$

Given the next step distribution $D = [P_1, P_2, \ldots, P_N]$ of sequence $\mathcal{S}$, we can derive an upper-bound of possible divergence:

$$
\begin{aligned}
&\max\big(\mathcal{D}_{\text{KL}}\big(D_{\text{ext}} \parallel D\big)\big) \\
&= \max\bigg(\sum_{i \in D} P_{\text{ext}}(i) \times \log_2 \frac{P_{\text{ext}}(i)}{P(i)} \bigg) \\
&\leq 1 \times \log_2 \frac{1}{\min(P(i))} + 0 + 0 + \cdots \\
&= -\log_2\big(\min\big(P(i)\big)\big)
\end{aligned}
\tag{1}
$$

The equal sign (maximum possible divergence) is taken iff the least likely option for the next step $P(i)$ in $\mathcal{S}$ becomes the most likely option $P_{\text{ext}}(i) = 1$ in $\mathcal{S}_{\text{ext}}$, and all other options have $P = 0$. Therefore, we can test if $-\log_2(\min(P_{\text{Distr}}(i))) < \delta$ holds during the rule growing process; if it holds, then further increasing the order (adding more previous steps) will not produce significantly different distributions, so we can stop the rule growing process and take the last known k (which passed the actual divergence test, not the order which passes the maximum divergence test) as the true order of dependency. Note that, the dynamic threshold is chosen heuristically in its current form. This threshold meets our design requirements: (1) enforce higher support for higher-orders, and (2) fast to compute, as it is a frequently used module in the innermost loop.

Furthermore, `BuildHON+` no longer requires a *MinSupport* parameter. Recall that using *MinSupport* > 1 in `BuildHON` helps reduce the search space as a crude form of early stopping, with the risk of losing valid higher-order patterns. In `BuildHON+`, the dynamic threshold takes care of early stopping without requiring any extra parameter (*MinSupport*) to limit the search space. This parameter is left in the algorithm only for backward compatibility and is set to 1 by default, but does not serve any initial seeding purpose. In other words, MinSupport is not used in `BuildHON+`.

An advantage of this proposed parameter-free approach is that rather than terminating the rule growing process prematurely by the *MaxOrder* threshold, the algorithm can now extract *arbitrarily orders of dependency*.

3.1.2 Scalability for higher-orders

`BuildHON` builds all observations and distributions up to *MaxOrder* ahead of the rule growing process (Fig. 2 left). This procedure becomes prohibitively expensive for big data with high orders of dependencies: to extract sparse tenth order dependencies, `BuildHON` needs to enumerate n-grams from first-order to tenth order and compare probability distributions, which already exceeds a personal computer's capacity using a typical real-world data set (see Sect. 4).

`BuildHON+`, on the other hand, uses a *lazy construction* of observations and distributions that has a much smaller search space, and can easily scale to arbitrarily high order of dependency. Specifically, `BuildHON+` does not require the counting of the occurrences of n-grams or calculating the distribution of the next steps, until the rule growing step explicitly asks for such information.

Example `BuildHON+` first builds all first-order observations and distributions (Fig. 2 right step ①–③). Given that $A \to C$, $B \to C$, $D \to B$, $E \to A$ all have single deterministic options for the next step with $P = 1$, according to $-\log_2(\min(P_{\text{Distr}}(i))) = 0 < \delta$, `Build-HON+` knows no higher-order dependencies can possibly exist by extending these bigrams (step ④). Only the two paths $C \to D$ and $C \to E$ will be extended; since the corresponding second-order observations and distributions are not known yet, `BuildHON+` *selectively* derives the necessary information from the raw data (Fig. 2 right step ⑤–⑦), and finds that the second-order distributions show significant changes. At this point, both $C|A \to D$ and $C|B \to E$ have single deterministic options for the next step, so again, `BuildHON+` determines no dependencies beyond second-order can exist (step ⑧), so the rule growing procedure stops, without the need for further generation and comparison of distributions.

The challenge is how to count the n-gram of interest on demand—seemingly every on-demand construction requires a traversal of the raw sequential data with the complexity of $\Theta(L)$. However, given the following knowledge:

Lemma 2 *All observations of the sequence $[S_{t-k-1}, S_{t-k}, \ldots, S_{t-1}, S_t]$ can be found exactly at the current and one preceding locations of all observations of sequence $[S_{t-k}, \ldots, S_{t-1}, S_t]$ in the raw data.*

Proof Instead of traversing the raw data, we use an *indexing cache* to store the locations of known observations, then use that to narrow down higher-order n-gram look-ups. As illustrated in Fig. 2, if we cache the locations of $C \to D$ and $C \to E$ in the raw sequential data, then $C|A \to D$ and $C|B \to E$ can be found at the same locations.

During the rule growing process, if $\mathcal{S}_{\text{ext}}$ has not been observed, *recursively* check if the lower-order observation is in the indexing cache, and use those cached indexes to perform a fast lookup in the raw data. New observations from $\mathcal{S}_{\text{ext}}$ are then added to the indexing cache. This procedure guarantees the identification of observations of the previously unseen $\mathcal{S}_{\text{ext}}$, and the lookup time for each observation is $\Theta(1)$ when the indexing cache is implemented with hash tables.   □

Complexity analysis. We formally analyze and compare the computational complexity of `BuildHON` and `BuildHON+`.

`BuildHON`. Suppose the size of raw sequential data is L, and there are D_i distinct n-grams of order of i. All first-order observations (bigrams) take $\Theta(2D_2)$ space, second order observations (trigrams) take $\Theta(3D_3)$ space, and so on; building observations and distributions up to k^{th} order takes $\Theta(2D_2 + 3D_3 + \cdots + kD_k)$ storage, with k being the maximum order allowed, because `BuildHON` always keeps raising order until k is reached, while keeping all the breadth-first search results for lower orders. with $D_3 \geq D_2$, $D_4 \geq D_3$, resulting in a complexity of $\Theta(k^2 D_2)$.

If N is the number of unique entities in the raw data, then the time complexity of the algorithm is $\Theta(Nk^2 D_2)$. All observations will be traversed at least once, and evaluating if adding a previous step significantly changes the probability distribution of the next step takes up to $\Theta(N)$ time (assuming Kullback–Leibler divergence [29] is used).
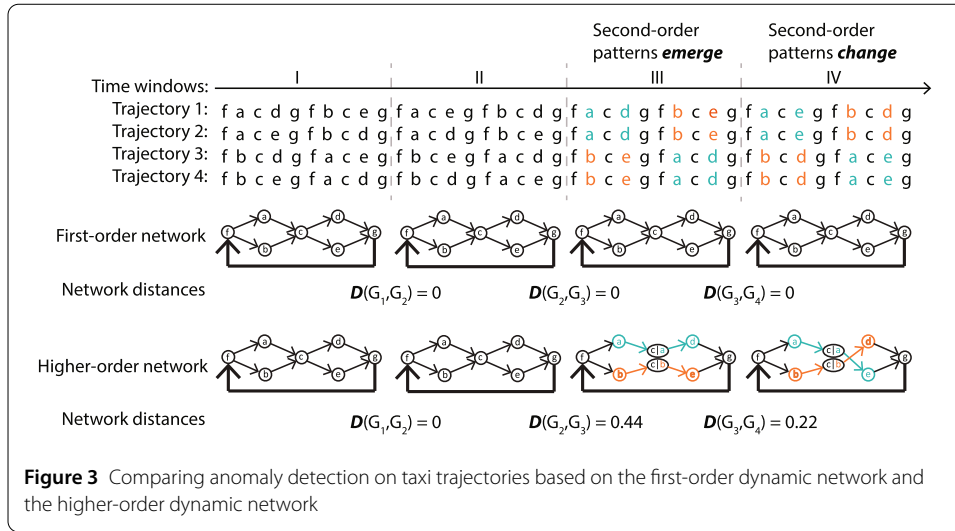
`BuildHON+`. Assume there are R_i distinct n-grams that are *exactly* of order i. By definition, we have $R_i \leq D_i$. Therefore, `BuildHON+`'s space complexity is $\Theta(2R_2 + 3R_3 + \cdots + tR_t)$ (including observations, distributions, and the indexing cache) where R_k is the *exact* number of higher-order dependency rules for order k. Note that, $R_k \leq L$, but it is not necessarily $R_{(i+1)} \geq R_{(i)}$. Also $t \leq k$.

In practice, what makes `BuildHON+` different from `BuildHON` is its sensitivity to the underlying data. If the dataset contains very few non-significant n-grams up to maximum specified order, the space complexity of `BuildHON+` would not be very different from `BuildHON`. However, for very noisy data ($D_i \gg R_i$) or data with an actual order much smaller than the specified maximum order ($t \ll k$), the space complexity of `BuildHON+` would be significantly smaller than `BuildHON`. The same applies to time complexity: while `BuildHON` has $\Theta(Nk^2 D_2)$, `BuildHON+` has $\Theta(N(2R_1 + 3R_2 + \cdots))$. A side-by-side comparison between `BuildHON` and `BuildHON+` in running time and memory consumption on a real-world data set is provided in Sect. 4.

3.2 Higher-order anomaly detection

Definition. The procedure of a network-based anomaly detection method takes the sequential data, $\mathcal{S} = [S_1, S_2, \ldots, S_T]$ which is divided into T time windows $t \in [1, T]$ as the input. In each time window, the sequential data is represented as a network, i.e., $S_i \rightarrow G_i$, yielding a dynamic network $\mathcal{G} = [G_1, G_2, \ldots, G_T]$ composed of the sequence of networks. The dynamic network $\mathcal{G}$ is then used to find the change point(s) $t \in [1, T]$ when G_t is significantly different from G_{t-1}. The difference between networks in neighboring time intervals, i.e., $d_t = \mathcal{D}(G_{t-1}, G_t)$, can be quantified by network distance metrics $\mathcal{D}$ (e.g., [23–25]). Then the problem of anomaly detection in networks reduces to anomaly detection in the time series of $[d_2, d_3, \ldots, d_T]$. Next, to determine if the network difference d_t is significantly high, straightforwardly, if d_t is larger than a fixed threshold Δ, G_t is anomalously different than G_{t-1}. Another more robust way is to establish the norm of network differences by computing the running average and standard deviation of network differences in the last k time intervals, the null hypothesis being d_t not significantly large; if d_t deviates from the running average by two standard deviations, the null hypothesis is rejected and time t is considered a change point.

Existing network-based anomaly detection methods mostly differ at the network distance calculation step. However, for the $S_i \rightarrow G_i$ step, i.e., where raw sequential data is represented as networks, existing methods all use FON as G to represent the underlying sequential data S, by counting the occurrences of pairs (bigrams) as edge weights in the

Figure 3 Comparing anomaly detection on taxi trajectories based on the first-order dynamic network and the higher-order dynamic network

network. Here, we propose to use the higher-order network (HON) that selectively embeds n-grams for the $S_i \to G_i$ step. HON, using `BuildHON+`, keeps all structures of FON, and when higher-order dependencies exist in the raw sequential data, it splits a node into multiple nodes representing previous steps. We show that certain types of anomalies will remain undetected for all existing network-based anomaly detection methods using FON, but can be revealed by using HON.

Example Fig. 3 illustrates a side-by-side comparison of FON and HON in the network representation step. Suppose there are four taxi trajectories in the raw data. In time window I, taxis in location c randomly navigate to d or e, regardless if the taxis came to location c from a or b. In this time window, HON is identical to FON and there are no higher-order dependencies. In time window II, the traffic patterns are randomly shuffled, and the pairwise traffic between pages a, b, c, d, e remains the same as time window I. Neither FON nor HON shows changes.

In time window III, second-order patterns *emerge*: all taxis that had navigated from a to c go to d, and all taxis from b to c go to e. Since the aggregated traffic from c to d and e remains the same, the FON remains *exactly the same*, missing this newly emerged pattern. In contrast, HON uses additional higher-order nodes and edges to capture higher-order dependencies: node c is now splitted into a new node $c|a$ (representing c given the last step being a) and node $c|b$ (representing c given the last step being b). The path $a \to c \to d$ now becomes $a \to c|a \to d$; the edge $c \to e$ rewired similarly. Therefore, the emergence of the second-order pattern in the raw data is reflected by the non-trivial changes in the topology of HON. If we use the weight distance [23] defined as

$$\mathcal{D}(G, H) = |E_G \cup E_H|^{-1} \sum_{u,v \in V} \frac{|w^G(u, v) - w^H(u, v)|}{\max\{w^G(u, v), w^H(u, v)\}} \tag{2}$$

with w being the edge weights and $|E|$ being the total number of edges, due to the complete changes in four out of the nine edges on HON, the network distance $\mathcal{D}(G_2, G_3) = 0.44 > 0$, successfully captures this *higher-order anomaly* (a significant change in higher-order navigation patterns).

In time window IV, the second-order navigation pattern *changes*: all taxis that navigated from location a to c now visit e instead of d, and all from b to c now visit d instead of e. Since the pairwise traffic from c to d and e remains the same, FON remains the same. However, HON captures the changes with two edge rewirings: now $c|a \rightarrow e$ and $c|b \rightarrow d$, resulting in $\mathcal{D}(G_3, G_4) = 0.22 > 0$.

In brief, FON is an oversimplification of sequential data produced by complex systems, and conventional network-based anomaly detection methods that use FON may fail to capture the emergence and changes of higher-order navigation patterns. If HON is used instead, without changes to distance metrics, existing methods can capture these previously ignored anomalies.

3.2.1 Distance metrics

After successful construction of HON (using `BuildHON+`) we apply five network distance measures to detect anomalies.

1. *Weight distance.* This metric was introduced earlier (Equation (2)).
2. *Maximum common subgraph (MCS).* The MCS distance is defined similarly to the weight distance in Equation (2) but operates on MCS [23]:

$$\mathcal{D}(G,H) = |E_G \cap E_H|^{-1} \sum_{u,v \in V} \frac{|w^G(u,v) - w^H(u,v)|}{\max\{w^G(u,v), w^H(u,v)\}} \qquad (3)$$

3. *Modality.* This distance function can be defined as follows [24]:

$$\mathcal{D}(G,H) = \big\| \pi(G) - \pi(H) \big\| \qquad (4)$$

    where $\pi(G)$ and $\pi(H)$ are the Perron vectors of graphs G and H, respectively.
4. *Entropy graph distance.* This can be defined as follows [25]:

$$\mathcal{D}(G,H) = E(G) - E(H) \qquad (5)$$

    where $E(*)$ is the entropy measure of the edges:

$$E(*) = \sum_{e \in E_*} \widetilde{W}_*^e - \sum_{e \in E_*} \ln \widetilde{W}_*^e \qquad (6)$$
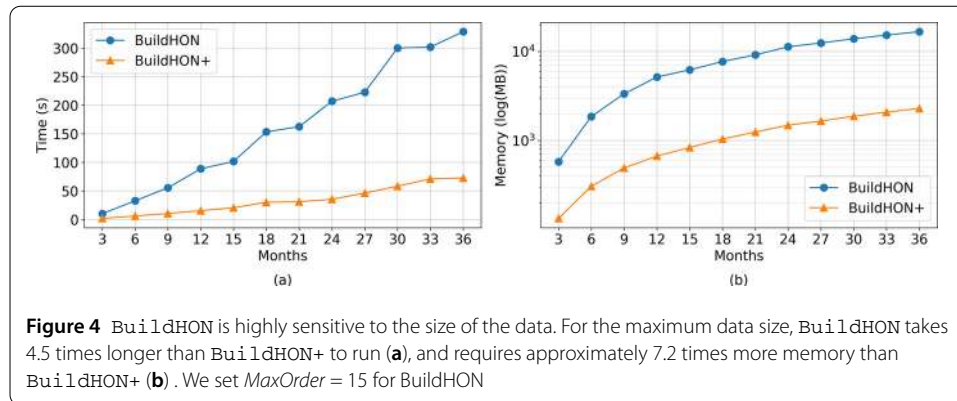
    and:

$$\widetilde{W}_*^e = \frac{W_*^e}{\sum_{e \in E_*} W_*^e} \qquad (7)$$

    is the normalized weight for edge e.
5. Finally, we also use the *spectral distance*, which is defined as [25]:

$$\mathcal{D}(G,H) = \sqrt{\frac{\sum_{i=1}^{k}(\lambda_i - \mu_i)^2}{\min(\sum_{i=1}^{k}\lambda_i^2, \sum_{i=1}^{k}\mu_i^2)}} \qquad (8)$$

    where λ_i and μ_i represent the eigenvalues of the Laplacian matrix for graph G and G, respectively.

Figure 4 `BuildHON` is highly sensitive to the size of the data. For the maximum data size, `BuildHON` takes 4.5 times longer than `BuildHON+` to run (**a**), and requires approximately 7.2 times more memory than `BuildHON+` (**b**) . We set *MaxOrder* = 15 for BuildHON

Note that, in order to calculate network distance in HON, all higher-order nodes are treated as first-order ones. That is, a change from $D|B,C \rightarrow E$ to $D|B,C,A \rightarrow E$ results in total removal of $D|B,C$ and new addition of node $D|B,C,A$. The reason is that in many cases, anomalous patterns result in a change of higher-order patterns. It is desirable that the anomaly detection method detects the "emergence", "change" and "dissipation" of higher-order patterns. We leave the task of classifying different higher-order anomalies for future work.

4 Results

In this section, we first compare `BuildHON+` with `BuildHON` in terms of running time and memory consumption on real-world data of various sizes and multiple orders of dependency. Next, we present the anomaly detection results.
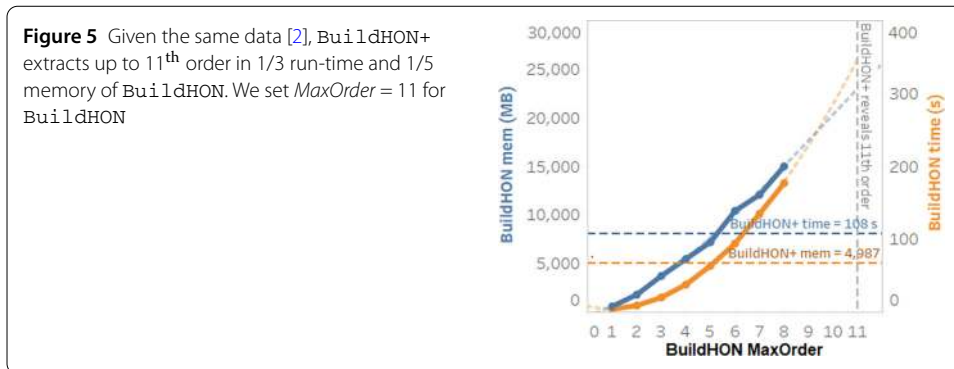
For the anomaly detection experiments, we first construct a large-scale synthetic taxi movement data with 11 billion movements and variable orders of dependencies, and show that five existing anomaly detection methods based on FON collectively fail to capture anomalous navigation behaviors beyond first-order, while using our framework, all methods show significant improvements.

We also demonstrate HON on real-world taxi trajectory data, showing its ability in capturing the higher-order anomaly signals and revealing the exact location of anomalies.

4.1 Scalability analysis: performance improvement of `BuildHON+` over `BuildHON`

To highlight the scalability advantage of `BuildHON+`, instead of the taxi data or the synthetic data (which demonstrates up to third order of dependency), we use the same shipping trajectories data as used in the HON paper [2]. This data was shown to demonstrate dependencies of more than the fifth-order due to ships' cyclic movement patterns. It consists of up to three years of shipping data (between May 1st, 1997 and April 30th, 2003), aggregated over 3-months intervals. The smallest and largest data contains 372,500 and 4,721,936 voyages, respectively.

For a fair comparison, we use the Python implementation for both `BuildHON+` and `BuildHON`. Both implementations run single-threaded on the same Linux machine (Intel Quad 16-core @ 2.10 GHz, 128 GB RAM). `BuildHON+` is parameter-free (no limit to the maximum order, optional *MinSupport* = 1) and does not require further configuration. We set *MinSupport* = 1 and *MaxOrder* = 15 for `BuildHON`. We start with the first 3-months of the data and aggregate the trajectories over the next 6 months, 9 months, and

Figure 5 Given the same data [2], `BuildHON+` extracts up to 11th order in 1/3 run-time and 1/5 memory of `BuildHON`. We set *MaxOrder* = 11 for `BuildHON`

so on. Figure 4 illustrate the time and memory usage of both algorithms as the size of the data increases. We observe that `BuildHON` is highly sensitive to the size of the data. For the maximum data size, `BuildHON` requires approximately 7.2 times more memory than `BuildHON+` and takes 4.5 times longer to run.

We further analyze the run time and memory usage of both algorithms on the same shipping dataset to analyze the effect of setting different values for *MaxOrder*. For this experiment, we use one year of data which consists of 3,415,577 voyages between May 1st, 2012 and April 30th, 2013.

We set *MinSupport* = 1 for `BuildHON`, and gradually increase *MaxOrder* from the first-order. Same as above, `BuildHON+` does not require further configuration. `BuildHON+` was able to find up to 11th order within 2 minutes, with a peak memory usage less than 5 GB, as the reference lines displayed in Fig. 5. In comparison, `BuildHON` already exceeds the running time and memory consumption of `BuildHON+` at 6th order, reaches the physical memory limit at 8th order, and would need about 22 GB memory and 6 minutes (3× time and 5× memory than `BuildHON+`) to achieve the same results as `Build-HON+` can. Both implementations run single-threaded on the same laptop (Intel i7-6600U @ 2.60 GHz, 16 GB RAM, SSD).

4.2 Anomaly detection: large-scale synthetic taxi movements

We first use the synthetic data with known higher-order anomalies to test the effectiveness of the HON-based anomaly detection framework. With synthetic data, we know exactly when, where, and what types of anomalies exist. To begin with, we assume 100,000 taxis are navigating through a 10×10 grid with cells numbered from 00 to 99. At each timestamp, every taxi moves 100 steps, resulting in 10,000,000 movements.

Our goal is to synthesize input sequences with variable orders of taxi navigating patterns. We start from the basic case where all taxis navigate randomly, then gradually add or change first-order and higher-order navigation rules, and see if the proposed method can successfully identify these anomalies.

For each of the following 11 cases, we maintain the taxi navigation behavior for 100 time windows. In total, we generate 11,000,000,000 taxi movements for the subsequent anomaly detection task. The full process to synthesize the input trajectories is illustrated in Fig. 6.

Initial random movement case. At $t = [0, 99]$, each taxi has a 50% chance of navigating to the cell on the right and 50% chance of navigating down in each move.
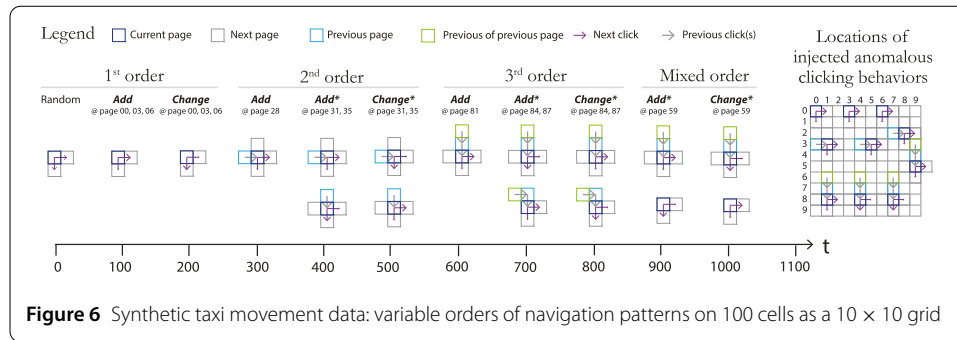
Figure 6 Synthetic taxi movement data: variable orders of navigation patterns on 100 cells as a 10 × 10 grid

Emergence of the first-order dependency. At $t = [100, 199]$, we impose the following first-order rule of movement: all taxis coming to cell 00, 03 and 06 will have a 90% chance of moving to the right and 10% chance of moving down in the next step. This new rule incurs a significant change of first-order traffic at $t = 100$ between pairs of cells 00–01, 00–10, 03–04, 03–13, 06–07 and 06–16. The locations of these dependency rules are highlighted on the right of Fig. 6.

Change of the first-order dependency. At $t = [200, 299]$, we change the existing first-order movement rules: all taxis coming to cell 00, 03 and 06 will now have a 90% chance of moving down in the next step, and a 10% chance of moving right. This change at $t = 200$ should also be reflected in both FON and HON.
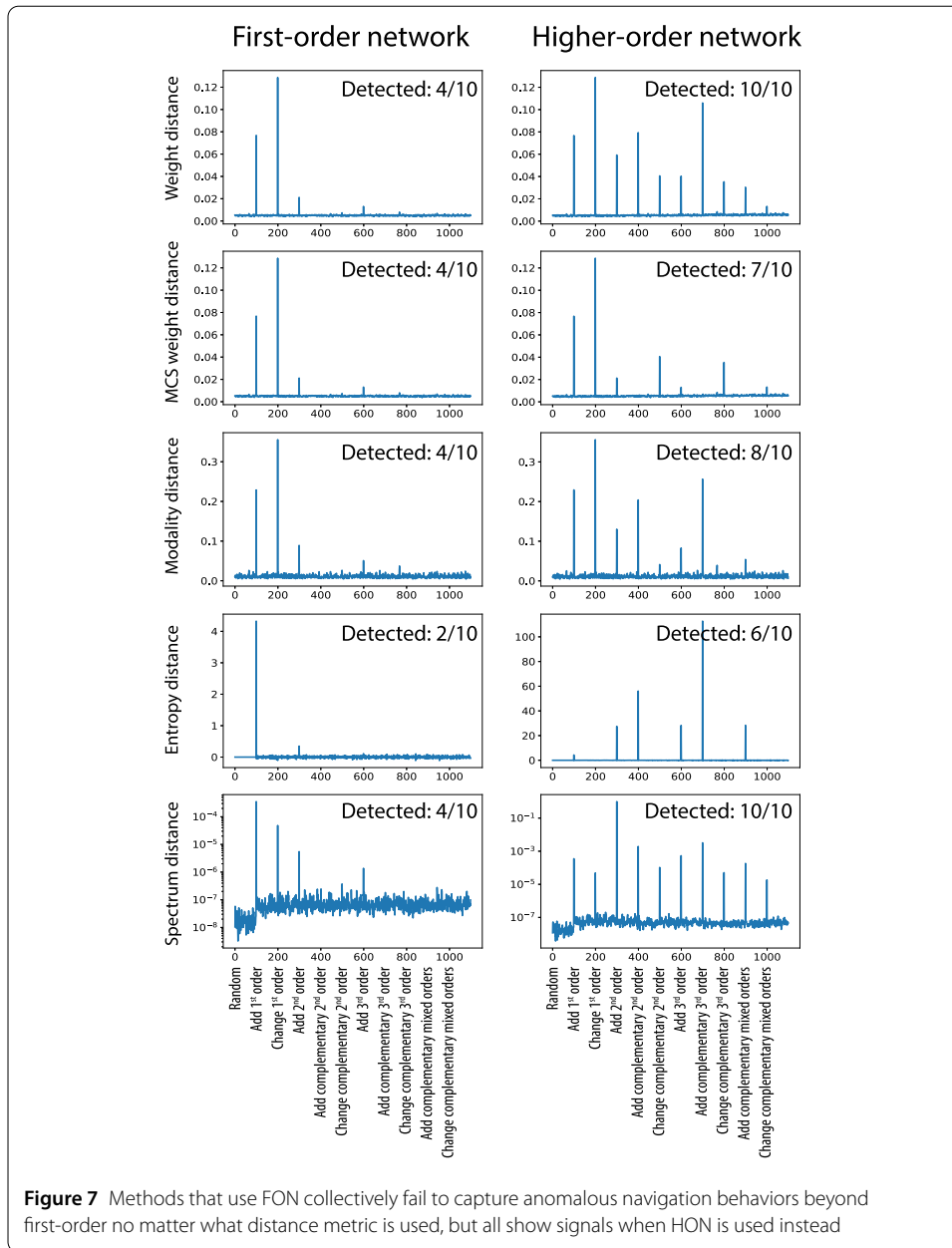
Emergence of second-order dependency. At $t = [300, 399]$, we keep the previous first-order rules and impose a new second-order rule: all taxis coming from cell 27 to 28 will have a 90% chance of moving to the right in the next step, and a 10% chance of moving down. This change at $t = 300$ not only introduces new higher-order dependencies, but also slightly influences first-order traffic (traffic of $27 \rightarrow 28 \rightarrow 29/38$ changes from 1:1 to 7:3).

Emergence of complementary second-order dependencies. At $t = [400, 499]$, we impose a pair of new second-order rules: (1) all taxis coming from cell 30 to 31 (and 34 to 35) will have a 90% chance of moving to the right in the next step, and a 10% chance of moving down; (2) all taxis coming from page 21 to 31 (and 25 to 35) will have a 90% chance of moving down, and a 10% chance of moving right. The combined effect of these two new complementary second-order dependencies at $t = 400$ is that the first-order taxi traffic from cell 31 and 35 remains unchanged.

Change of complementary second-order dependencies. At $t = [500, 599]$, we flip the rules for the complementary second-order dependencies: (1) all taxis coming from cell 30 to 31 (and 34 to 35) will have a 90% chance of moving down, and a 10% chance of moving right; (2) all taxis coming from page 21 to 31 (and 25 to 35) will have a 90% chance of moving right, and a 10% chance of moving down. At $t = 500$ the first-order taxi traffic from cell 31 and 35 still remains unchanged.

Emergence of third-order dependency. At $t = [600, 699]$, we impose a new third-order rule: all taxis coming from cell 61 through 71 to 81 will have a 90% chance of moving to the right in the next step, and a 10% chance of moving down. This introduction of third-order dependencies at $t = 600$ also slightly influences the first-order traffic (from 1:1 to 3:2).
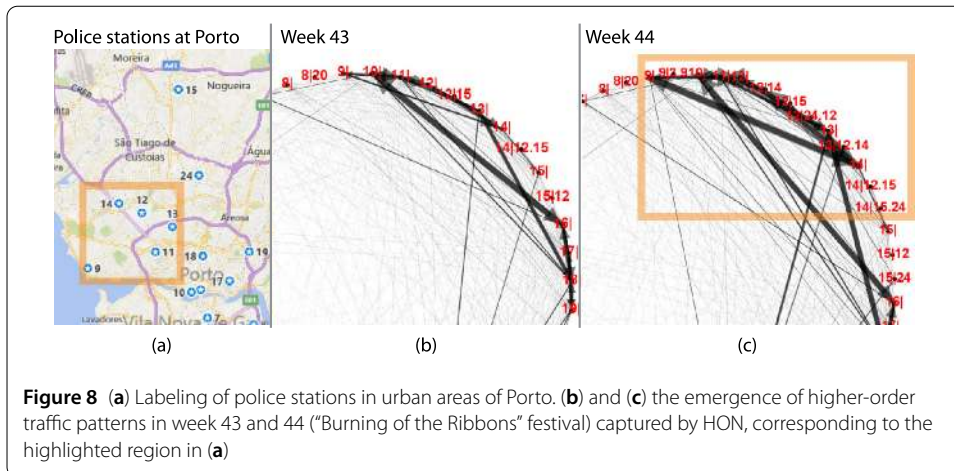
Emergence of complementary third-order dependencies. At $t = [700, 799]$, we impose a pair of new third-order rules: (1) all taxis coming from cell 64 through 74 to 84 (and 67 through 77 to 87) will have a 90% chance of moving to the right in the next step, and a 10%

Figure 7 Methods that use FON collectively fail to capture anomalous navigation behaviors beyond first-order no matter what distance metric is used, but all show signals when HON is used instead

chance of moving down; (2) all taxis coming from 73 through 74 to 84 (and 76 through 77 to 87) will have a 90% chance of moving down, and a 10% chance of moving right. Here at $t = 700$ first-order traffic does not change when these two complementary third-order dependencies are introduced together.

Change of complementary third-order dependencies. At $t = [800, 899]$, we flip the rules for the complementary third-order dependencies. First-order traffic at $t = 800$ again remains unchanged.

Emergence of complementary mixed-order dependency. At $t = [900, 999]$, we impose a new third-order rule and a first-order rule: (1) all taxis coming from cell 39 through 49 to 59 will have a 90% chance of moving to the right in the next step, and a 10% chance of moving down; (2) all taxis at cell 59 will have 11/30 chance of moving right and 19/30 chance of

Figure 8 (**a**) Labeling of police stations in urban areas of Porto. (**b**) and (**c**) the emergence of higher-order traffic patterns in week 43 and 44 ("Burning of the Ribbons" festival) captured by HON, corresponding to the highlighted region in (**a**)

moving down. At $t = 900$ first-order traffic does not change, because the influence of the new third-order rule on pairwise traffic is canceled by the new first-order rule.
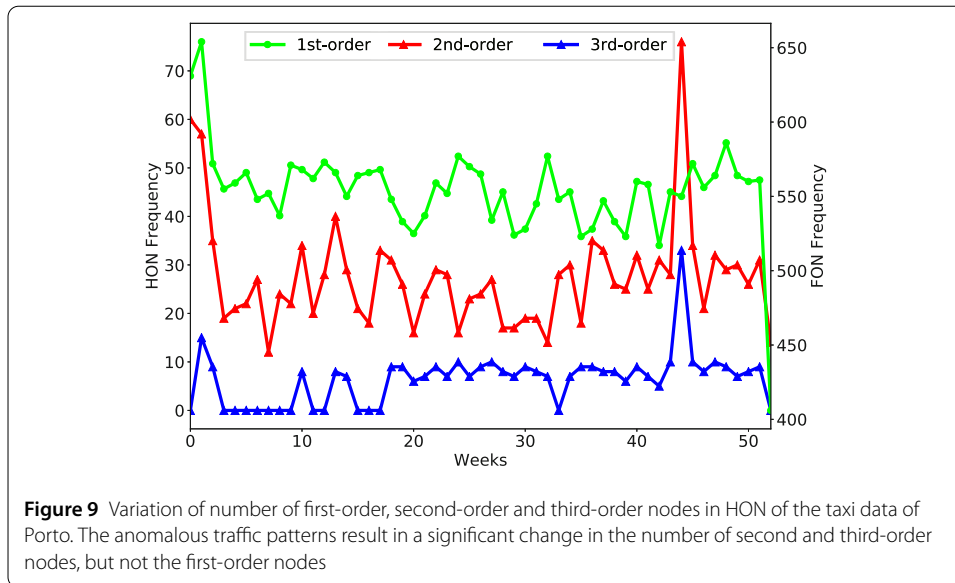
Change of complementary mixed-order dependency. At $t = [1000, 1099]$, we flip the rules for the mixed-order dependencies. First-order traffic at $t = 1000$ remains unchanged.

4.2.1 Results

For all five distance metrics, we present a side-by-side comparison between anomaly detection results using FON and HON in Fig. 7. The Y-axis shows the graph distances between neighboring time windows; given that we have injected 10 anomalous movement patterns at $t = [100, 200, \ldots, 1000]$, we should expect to see 10 "spikes" in graph distances.

Methods using FON can detect at most 4 out of the 10 anomalies: the addition and changes in first-order movement patterns ($t = 100$, $t = 200$), the addition of second-order ($t = 300$), and the addition of third-order ($t = 600$) movement patterns. Because the latter two cases also slightly change the first-order traffic, FON does reflects the changes, but the spikes incurred are not as significant as when changes are made directly to first-order rules. For the other six cases, all five distance metrics appear as if there are no anomalies, as long as they rely on FON topology.

In contrast, methods using HON (1) capture all first-order anomalies ($t = 100$, $t = 200$); (2) show stronger signals for the addition of second-order and third-order rules ($t = 300$, $t = 600$) because not only the first-order traffic is changed but `BuildHON+` also creates additional higher-order nodes and edges for higher-order dependencies; (3) capture the six additional cases where higher-order movement patterns are changed but first-order traffic remains the same. Here the topological changes of HON are best reflected with weight distance and spectrum distance (detecting 10/10 anomalies); MCS weight method misses the addition of higher-order nodes and edges ($t = 400, 700, 900$) because those topological changes are excluded from common subgraphs; entropy method misses changes in edge weights ($t = 200, 500, 800, 1000$), also because by definition a swap in edge weights do not change a graph's entropy. Nevertheless, all these distance metrics are able to identify more types of anomalous signals simply by using HON instead of FON, with no changes to these distance metrics. In other words, `BuildHON+` can be plugged into existing network-based anomaly detection methods directly, and extend their ability in detecting higher-order anomalies.
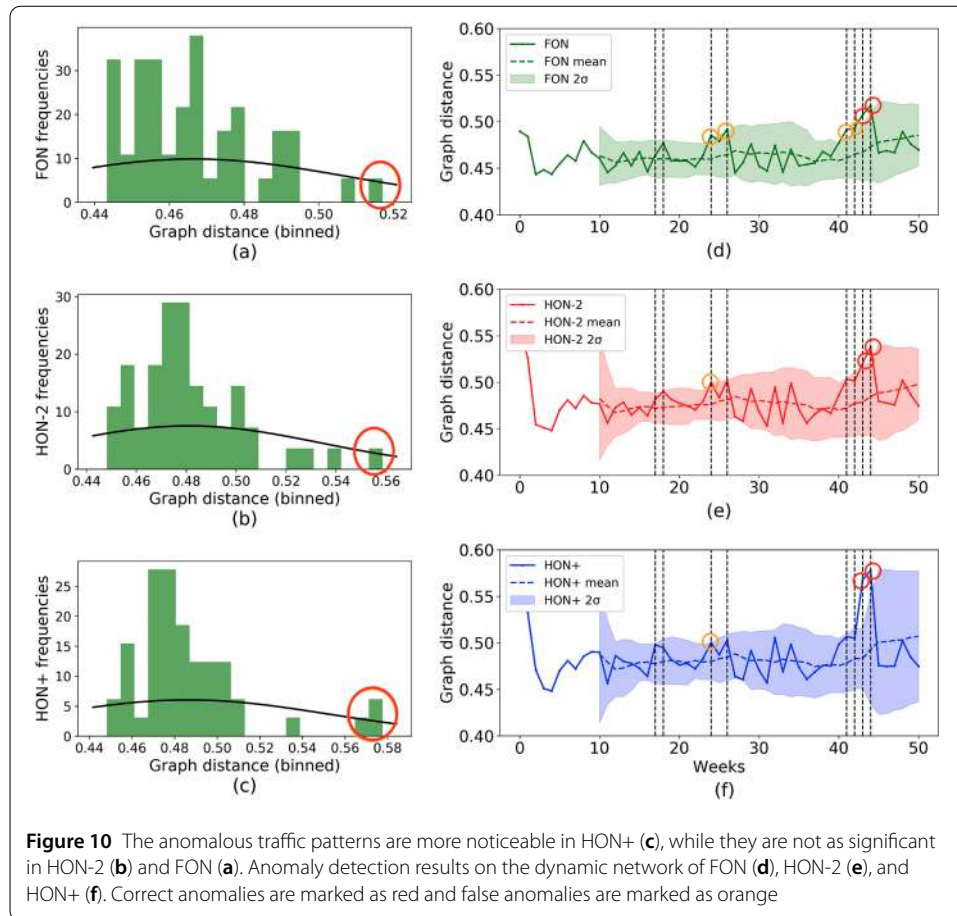
Figure 9 Variation of number of first-order, second-order and third-order nodes in HON of the taxi data of Porto. The anomalous traffic patterns result in a significant change in the number of second and third-order nodes, but not the first-order nodes

4.3 Anomaly detection: real world taxi data

We use the ECML/PKDD 2015 challenge data,[a] which contains one year (Jul. 1, 2013 to Jun. 30, 2014) of all the 442 taxi GPS trajectories in Porto, Portugal. The coordinates of each taxi were collected every 15 seconds. To discretize the geolocation data into points of interest that are representative of population density, we map all coordinates to the nearest 41 police stations (Fig. 8). As a pre-processing step and to void introduction of bias / noise, we removed the taxis that have been idle for more than 5 days because that can arise due to data collection errors (on average 5.29% of the trajectories were removed). The highlighted box in Fig. 8 indicates the detection of anomalies. Figure 9 shows the week to week difference in higher-order dependencies, yielding in 52 time windows and 442 trajectories of points of interest. We consider both FON and `BuildHON+` with a fixed maximum order of 2 and `BuildHON+` with a variable higher-order (discovered to be 3 by the algorithm). We show that `BuildHON+` when allowed to discover the maximum order, results in the highest indication of potential anomalies.

Note that the choice of time-window is quite data-dependent. We initially attempted daily time-windows but noticed that the weekly fluctuation patterns (weekday commute traffic vs weekend recreational traffic) dominate any other signals. Besides, daily time windows have sparser observations, resulting in a very sparse network for each time step. On the other hand, because anomalous traffic patterns usually last for no more than a few days, using monthly aggregation will dilute the signal and result in too coarse a granularity.

4.3.1 Graph distance analysis

We compare the 52 networks for FON, *MaxOrder* of 2 as a constraint for `BuildHON+` (indicated as HON-2), and no *MaxOrder* constraint on `BuildHON+` (indicated as HON+) in Fig. 10. Our goal is to see the improvements afforded by allowing `BuildHON+` to automatically discover the requisite higher-order for a given data, versus specifying the maximum order of 2 using `BuildHON` and the FON representation. We compute the graph distances (using weight distance) for neighboring time windows. The histograms of graph distances for each network is shown in Fig. 10 (a), (b), and (c). We also compute the running average and standard deviation using the graph distances in the previous 10 weeks, with

Figure 10 The anomalous traffic patterns are more noticeable in HON+ (**c**), while they are not as significant in HON-2 (**b**) and FON (**a**). Anomaly detection results on the dynamic network of FON (**d**), HON-2 (**e**), and HON+ (**f**). Correct anomalies are marked as red and false anomalies are marked as orange

the null hypothesis as "the network is not significantly different if the graph distance does not deviate more than 2σ from the mean". Variation of graph distances for FON, HON-2 and HON+ is shown in Fig. 10 (d), (e), and (f) respectively. While the trend of HON+ resembles that of HON-2 and FON, the graph distances in weeks 43 and 44 are particularly more significant in HON+ than HON-2 and FON (HON-2 offers more significance over FON as well). Such differences are also indicated in the histograms of graph distances in Fig. 10 (a), (b), and (c), where the red circles highlight the correct anomalous signals, which is observable in HON+, while it is not as significant in FON and even HON-2.

We focus on the case of week 43 and 44 to understand why HON+ produces a stronger signal than HON-2 and FON in this time window. We notice that Porto's second most important festival, "Burning of the Ribbons", lasts from May 2 to May 9 in 2014 and falls within the end of week 43 and the entire week 44 of our study. The festival involves parades, road closures, and is popular among tourists, which could be the underlying reason for the changes in taxis' movement patterns. After plotting the traffic HON+ of week 43 and week 44 in Fig. 8 (b) and (c), we notice that multiple higher-order nodes and edges emerge in these weeks, indicating the emergence of higher-order traffic patterns. The newly emerged higher-order patterns correspond to police stations labeled from 9 to 14, which is where the event's main venue (Queimódromo in the City Park) and participating universities are located.[b] We further compared the fluctuations in the number of higher-order nodes (obtained from HON+) in Fig. 9. We notice that the number of first-order

nodes does not change significantly, while the number of second and third-order nodes shows a sharp change in week 44 of the data. FON (although showing deviation from average at week 44) does not capture the change in additional higher-order nodes, and HON-2 does not capture the change in third-order nodes. HON+ is more effective in deciphering the anomalous signal. This analysis shows the importance of including variable and higher-order dependencies for anomaly detection, and the applicability of `BuildHON+` in discovering the appropriate orders given the data. Depending on the data, the *Max-Order* value required for accurate detection of anomalies can be different. `BuildHON+` removes this dependency, ensuring accurate detection of changes in the network.
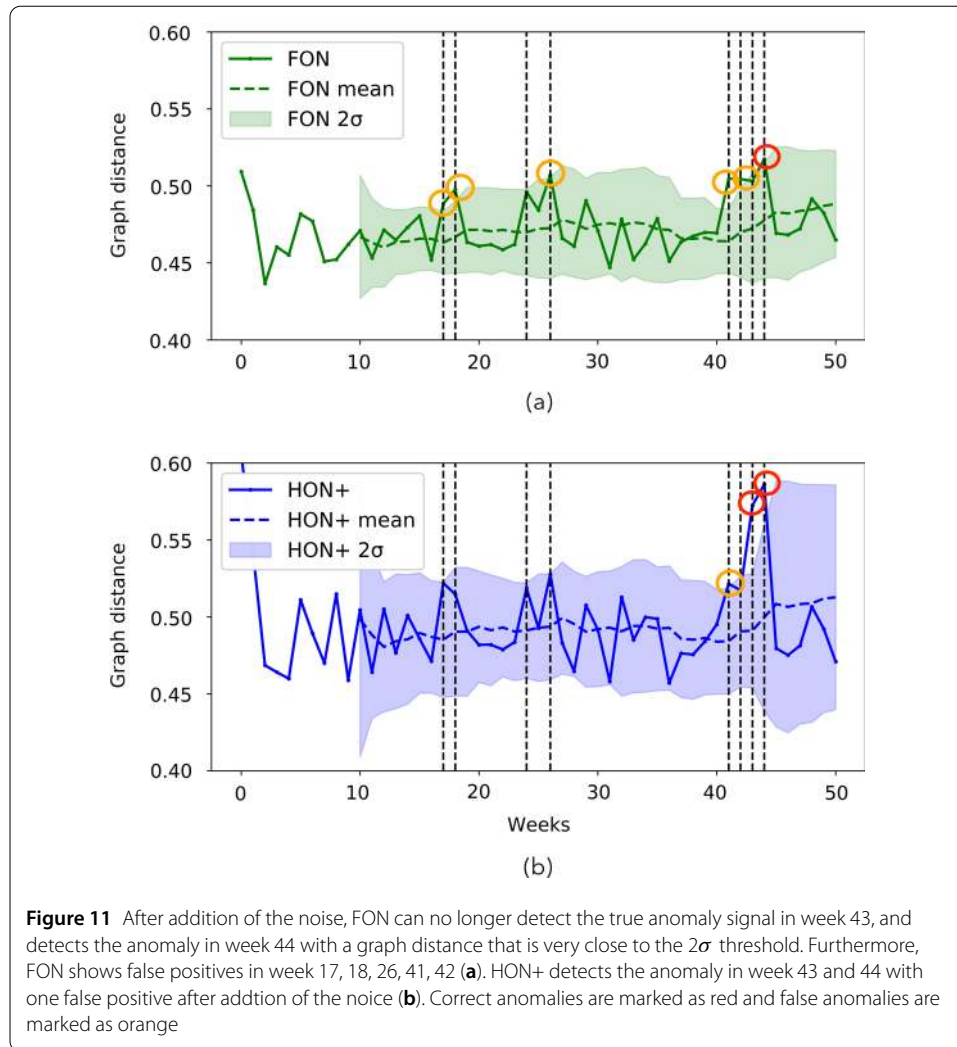
4.4 Robustness to noise

We notice that FON graph distance in week 43 and 44 falls slightly outside the 2σ threshold as well. However, HON+ deviation from the 2σ threshold is 3.25 times bigger that of FON in week 44 and 5.2 times bigger in week 43. This becomes important in the presence of noise where anomalies by FON may not be detected. Furthermore, based on FON graph distances, weeks 24, 26, 41, and 42 are also anomalous events (Table 1 and Fig. 10 (d)). However, no significant event happened during these weeks. Thus, without any noise, FON can detect anomaly but with a higher false positive rate (4), while HON+ can also correctly detect the anomalies but with only 1 false positive for week 24, with a very small value above the 2σ threshold, as indicated in Fig. 10 (f) and Table 1.

To illustrate the above point, we designed an experiment to show the robustness of HON+ and FON against noise. We randomly assigned 10% of all taxis to the next closest police station and constructed the corresponding HON+ and FON. The graph distances for FON and HON+ after adding the noise is shown in Fig. 11 (a) and Fig. 11 (b), respectively. The detected anomalies are presented in Table 1 where the values represent the difference between the graph distance and the 2σ threshold. Before adding the noise, FON detects the anomalies at week 44 and week 43 with a small margin from the 2σ threshold. Furthermore, it has a higher false positive rate (Fig. 10 (d)). After adding the noise, FON shows false positives in weeks 17, 18, 26, 41, 42 and one correct anomaly at week 44 which is very close to the 2σ threshold (Fig. 11 (a) and Table 1). Furthermore, FON does not

Table 1 Represents all the data points in which the graph distance (using FON and HON+) falls outside the 2σ threshold. The column "value" indicates the difference between the graph distance (obtained from FON or HON+) and the closest 2σ threshold (with respect to the moving average). TP refers to true positive values (correct anomalies), which are marked as bold. FP refers to false positive values. HON+ correctly detects anomalies, results in much lower false positives, and is more robust in presence of noise

	FON before noise		FON after noise		HON+ before noise		HON+ after noise	
	Week	Value	Week	Value	Week	Value	Week	Value
	24	0.008	17	0.006	24	0.0001	41	0.003
	26	0.003	18	0.008	**43**	**0.052**	**43**	**0.044**
	41	0.009	26	0.002	**44**	**0.026**	**44**	**0.025**
	42	0.002	41	0.020	–	–	–	–
	43	**0.010**	42	0.004	–	–	–	–
	44	**0.008**	**44**	**0.003**	–	–	–	–
TP;FP	2;4		1;5		2;1		2;1	
Precision	0.333		0.167		0.667		0.667	
Recall	1		0.5		1		1	

Figure 11 After addition of the noise, FON can no longer detect the true anomaly signal in week 43, and detects the anomaly in week 44 with a graph distance that is very close to the 2σ threshold. Furthermore, FON shows false positives in week 17, 18, 26, 41, 42 (**a**). HON+ detects the anomaly in week 43 and 44 with one false positive after addtion of the noice (**b**). Correct anomalies are marked as red and false anomalies are marked as orange

detect anything in week 43. HON+, on the other hand, detected the anomalous event (in both weeks 43 and 44) before the noise (Fig. 10 (f)) and after the noise (Fig. 11 (b)) with only one false positive (Table 1). It is important to note that false positives can be very costly and often require manual correction by human labor.

5 Conclusion

This paper presented a scalable and parameter-free algorithm for extracting higher-order dependencies from the sequential data, and demonstrates the success of higher-order network modeling for anomaly detection in dynamic networks. We show that `BuildHON+` is scalable and parameter-free and automates the process of discovering the appropriate variable and higher-order dependencies for each of the nodes in a network. The complexity analysis of `BuildHON+`, as well as running time and memory consumption benchmarking results, demonstrates the scalability of `BuildHON+` to large-scale networks.

We further demonstrate that FONs are weak detectors of higher-order anomalies, especially in the noisy data. This emerges because FONs do not adequately capture the sequential orders or indirect pathways in a complex system, thereby providing a limiting view of the behavior of a complex system in their network representation. `BuildHON+` can accu-

rately capture such anomalies and also work seamlessly with existing anomaly detection methods to enable more accurate detection of anomalies in comparison to using FON.

The higher-order network representation results in a more accurate representation of the underlying trends and patterns in the behavior of a complex system and is the correct way of constructing the network to not miss any important dependencies or signals. This is especially relevant when the data is noisy and has sequential dependencies within indirect pathways. This has numerous applications, ranging from information flow to human interaction activity on a website to transportation to invasive species management to drug and human tracking.

We note that changes in the HON structure can be more complex than changes in the FON structure, such as the emergence and dissipation of higher-order patterns. In order to use the graph distances that are defined for FONs, our current approach treats any changes in the node orders as total removal/addition of that node. This approach may result in more fluctuations in the graph distance and can cause HONs to become less overlapping over time. Regardless, measures defined for FONs can still be used for anomaly detection in HONs, since the 2σ criteria captures the HON fluctuations. One possible improvement can be designing a distance measure for capturing the unique fluctuations in the HON structure.

Another direction for future work is to classify different types of anomalies given different types of node changes in HON, like the emergence and dissipation of higher-order patterns. In addition to the graph distance metrics, one may also consider structure-based metrics [7] that factor in changes of clustering or ranking results and local properties such as motifs on the network. This could be considered as a supervised learning problem, where different categories of anomalies are labeled as classes in the training data and the task is to predict whether those categories of anomalies appear in the testing data. All of these extensions are directly compatible with `BuildHON+`, as the resulting HON representation does not impose a change in the network analysis method.

Supplementary materials are available as Additional file 1.

Supplementary information

Supplementary information accompanies this paper at https://doi.org/10.1140/epjds/s13688-020-00233-y.

> **Additional file 1.** Supplementary information. (PDF 335 kB)

Abbreviations
HON, Higher-order network; FON, First-order network.

Availability of data and materials
The Portugal taxi data is available through: http://www.geolink.pt/ecmlpkdd2015-challenge/data%20set.html. The Code for generating higher-order network and synthetic data is available at: https://github.com/msaebi1993/HON-ANOMALY.

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

Authors' contributions
All the authors participated in the conception of the research study. JX and MS are equal contributors. JX and MS implemented the algorithms and performed the experiments. All the authors participated in analysis and writing of the paper. All authors read and approved the final manuscript.

Author details
[1]University of Notre Dame, Notre Dame, USA. [2]U.S. Army Research Lab, Adelphi, USA. [3]Purdue University, West Lafayette, USA.

Endnotes
[a] http://www.geolink.pt/eomlpkdd2015-challenge/data%20set.html
[b] http://www.maiahoje.pt/noticias/ler-noticia.php?noticia=577

Publisher's Note
Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References
1. Lambiotte R, Rosvall M, Scholtes I (2019) From networks to optimal higher-order models of complex systems. Nat Phys 15:313–320
2. Xu J, Wickramarathne TL, Chawla NV (2016) Representing higher-order dependencies in networks. Sci Adv 2(5):e1600028
3. Rosvall M, Esquivel AV, Lancichinetti A, West JD, Lambiotte R (2014) Memory in network flows and its effects on spreading dynamics and community detection. Nat Commun 5:4630
4. Benson AR, Gleich DF, Leskovec J (2016) Higher-order organization of complex networks. Science 353(6295):163–166
5. Scholtes I, Wider N, Garas A (2016) Higher-order aggregate networks in the analysis of temporal networks: path structures and centralities. Eur Phys J B 89(3):61
6. Scholtes I, Wider N, Pfitzner R, Garas A, Tessone CJ, Schweitzer F (2014) Causality-driven slow-down and speed-up of diffusion in non-Markovian temporal networks. Nat Commun 5:5024
7. Akoglu L, Tong H, Koutra D (2015) Graph based anomaly detection and description: a survey. Data Min Knowl Discov 29(3):626–688
8. Ranshous S, Shen S, Koutra D, Harenberg S, Faloutsos C, Samatova NF (2015) Anomaly detection in dynamic networks: a survey. Wiley Interdiscip Rev: Comput Stat 7(3):223–247
9. Kivelä M, Arenas A, Barthelemy M, Gleeson JP, Moreno Y, Porter MA (2014) Multilayer networks. J Complex Netw 2(3):203–271
10. De Domenico M, Granell C, Porter MA, Arenas A (2016) The physics of spreading processes in multilayer networks. Nat Phys 12(10):901
11. Arenas A, Fernandez A, Fortunato S, Gomez S (2008) Motif-based communities in complex networks. J Phys A, Math Theor 41(22):224001
12. Petri G, Scolamiero M, Donato I, Vaccarino F (2013) Topological strata of weighted complex networks. PLoS ONE 8(6):e66506
13. Karsai M, Kaski K, Kertész J (2012) Correlated dynamics in egocentric communication networks. PLoS ONE 7(7):e40612
14. Wei W, Carley KM (2015) Measuring temporal patterns in dynamic social networks. ACM Trans Knowl Discov Data 10(1):9
15. Matamalas JT, De Domenico M, Arenas A (2016) Assessing reliable human mobility patterns from higher order memory in mobile communications. J R Soc Interface 13(121):20160203
16. Koher A, Lentz HH, Hövel P, Sokolov IM (2016) Infections on temporal networks—a matrix-based approach. PLoS ONE 11(4):e0151209
17. Lebacher M, Thurner PW, Kauermann G (2019) Exploring dependence structures in the international arms trade network: a network autocorrelation approach. Stat Model 20(2):195–218. https://doi.org/10.1177/1471082X18817673
18. Akoglu L, McGlohon M, Faloutsos C (2010) Oddball: spotting anomalies in weighted graphs. In: Advances in knowledge discovery and data mining, pp 410–421
19. Chandola V, Banerjee A, Kumar V (2012) Anomaly detection for discrete sequences: a survey. IEEE Trans Knowl Data Eng 24(5):823–839
20. Ramanathan A, Agarwal PK, Kurnikova M, Langmead CJ (2010) An online approach for mining collective behaviors from molecular dynamics simulations. J Comput Biol 17(3):309–324
21. Saligrama V, Chen Z (2012) Video anomaly detection based on local statistical aggregates. In: IEEE conference on computer vision and pattern recognition (CVPR). IEEE, pp 2112–2119
22. Xu J, Wickramarathne TL, Chawla NV, Grey EK, Steinhaeuser K, Keller RP, Drake JM, Lodge DM (2014) Improving management of aquatic invasions by integrating shipping network, ecological, and environmental data. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 1699–1708
23. Shoubridge P, Kraetzl M, Wallis WAL, Bunke H (2002) Detection of abnormal change in a time series of graphs. J Interconnect Netw 3(01n02):85–101
24. Kraetzl M, Wallis WD (2006) Modality distance between graphs. Util Math 69:97–102

25. Pincombe B (2005) Anomaly detection in time series of graphs using arma processes. ASOR Bull 24(4):2
26. Peel L, Clauset A (2015) Detecting change points in the large-scale structure of evolving networks. In: AAAI, pp 2914–2920
27. Mongiovi M, Bogdanov P, Ranca R, Papalexakis EE, Faloutsos C, Singh AK (2013) NetSpot: spotting significant anomalous regions on dynamic networks. In: Proceedings of the 2013 Siam international conference on data mining. SIAM, Philadelphia, pp 28–36
28. Scholtes I (2017) When is a network a network?: multi-order graphical model selection in pathways and temporal networks. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 1037–1046
29. Kullback S, Leibler RA (1951) On information and sufficiency. Ann Math Stat 22(1):79–86