Inference of dynamic hypergraph representations in temporal interaction data

Alec Kirkley[*]
Institute of Data Science, University of Hong Kong, Hong Kong;
Department of Urban Planning and Design, University of Hong Kong, Hong Kong;
and Urban Systems Institute, University of Hong Kong, Hong Kong

(Received 8 September 2023; accepted 8 April 2024; published 6 May 2024)

A range of systems across the social and natural sciences generate data sets consisting of interactions between two distinct categories of items at various instances in time. Online shopping, for example, generates purchasing events of the form (user, product, time of purchase), and mutualistic interactions in plant-pollinator systems generate pollination events of the form (insect, plant, time of pollination). These data sets can be meaningfully modeled as temporal hypergraph snapshots in which multiple items within one category (i.e., online shoppers) share a hyperedge if they interacted with a common item in the other category (i.e., purchased the same product) within a given time window, allowing for the application of hypergraph analysis techniques. However, it is often unclear how to choose the number and duration of these temporal snapshots, which have a strong influence on the final hypergraph representations. Here we propose a principled nonparametric solution to this problem by extracting temporal hypergraph snapshots that optimally capture structural regularities in temporal event data according to the minimum description length principle. We demonstrate our methods on real and synthetic data sets, finding that they can recover planted artificial hypergraph structure in the presence of considerable noise and reveal meaningful activity fluctuations in human mobility data.

DOI: 10.1103/PhysRevE.109.054306

I. INTRODUCTION

The recent fast-paced development of hypergraph modeling tools has opened up many new avenues for understanding the higher-order structure and dynamics of complex systems [1,2]. Data sets arising in applications as diverse as crime prediction [3], social media analytics [4], and epidemiology [5] consist of temporal interaction events between distinct categories of items—for example, users and comment threads in social media data or infected persons and locations in epidemiological data. Such data sets can be represented as temporal hypergraph snapshots, allowing for the application of centrality measures, community detection methods, link prediction algorithms, and dynamical models specifically tailored for capturing the structure and dynamics of higher-order interactions. These higher-order interactions produce emergent behaviors not present in traditional network representations [1,2]. In the hypergraph representation, items of one category (e.g., social media users) are represented as nodes and share a hyperedge if they were each involved in an interaction event with the same item of the other category (e.g., two users who commented on the same thread) within some specified time window.

In some applications of interest, there is a physically meaningful time window of interest for the temporal hypergraph snapshots. For example, in epidemiology one may want to set the timescale of colocation in human mobility data to be on the order of days in order to capture possible transmission risk from infected individuals that visited a given location. In this paper we are interested in situations where the timescale of interest is not clear ahead of time, and one must infer the characteristic time windows based on structural regularities in the event data itself. Such a need arises, for example, when identifying seasonality or anomalies in application areas without clear physical timescales such as certain online shopping behaviors [6] or vulnerabilities in cybersystems [7], as well as in exploratory machine learning analyses of geolocalized events in urban planning [8] and ecology [9].

Existing methods for constructing networks or hypergraphs from temporal data often require each temporal event to have some nonzero duration (such representations are also called "interval graphs") [10–12], but event time intervals can be hard or impossible to infer from many data generating sources, including social media check-ins, online purchases, and plant-pollinator interactions. Other works choose uniform, predefined time windows for event aggregation [13], but the precise window size chosen for temporal network aggregation can have a sizable impact on a wide variety of structural and dynamical characteristics including clustering and other centrality measures [14,15], latent node geometries [16], consensus dynamics [17], controllability [18], epidemic spreading [19], and ecological processes [20]. Uniform time windows may also fail to capture the "bursty" dynamics of temporal network interactions, in which many events occur within short time periods that are separated by long time periods of inactivity [11,21,22].

A natural way to construct hypergraph snapshots from temporal event data that overcomes these problems is to identify time windows within which the events exhibit

[*]alec.w.kirkley@gmail.com

significant shared structure. Such structural regularities can be readily identified using information theory, which allows us to quantify the level of data compression we can achieve by exploiting these regularities to transmit the data to a receiver. Among all hypergraph representations of the event data, those that better encapsulate structural regularities in the data will result in better compression from an information theoretic perspective, which can be operationalized using the Minimum Description Length (MDL) principle [23]. The MDL principle states that the best model among a set of competing models for a given data set is the one that can describe the data using the fewest symbols by exploiting its structural regularities [24]. The MDL principle is a powerful first-principles framework for model selection which has been employed in a range of graph mining and network science applications including community detection [25–27], significant subgraph identification [28–32], and graph summarization [33–35].

A few existing works have examined the aggregation of temporal networks with nontrivial edge structure into representative snapshots of varying duration. In the method of Masuda and Holme [36], time is discretized into small time steps and unipartite interactions that occur within each time step are aggregated into network snapshots. Then, a distance matrix is computed among these high-resolution network snapshots using any user-specified network distance measure, and the snapshots are clustered using a hierarchical clustering algorithm to give a coarse-grained representation of the data. This method is similar in spirit to that of De Domenico *et al.* [37], which aggregates multilayer networks (which may or may not represent temporal snapshots) using a spectral distance between network layers. Kirkley *et al.* [38] also approach this problem but using a nonparametric MDL approach that is motivated by the exploitation of shared edges in these layers. These methods all differ from the one proposed in this paper in a few crucial ways.

First, and most importantly, the methods in Refs. [36–38] require the fundamental measured network units (i.e., the disaggregated network layers) to have nontrivial global structure—in other words, more than just a single isolated interaction event (edge) at a given instant in time—in order to compute the network distances and clustering criteria of interest. These methods therefore require an input data set consisting of preaggregated individual isolated events into coarser network snapshots to detect any signal of cohesion among the data points, which is precisely the problem studied in this paper. Second, these methods do not specifically exploit node-level structure (e.g., degrees of each node set in a bipartite representation) for compression, making them unsuitable for handling hypergraph data in which edges are shared among multiple nodes. These distinctions are critical in applications where distinct interaction events are rarely repeated. For example, in recommendation systems, a user may rarely ever consume the same product twice, which results in no meaningful shared structure among data points for the methods of Refs. [36–38] to detect. In contrast, by aiming to compress interaction event data through the exploitation of repeated items in each category (e.g., users and products) individually, the method of this paper can find meaningful hypergraph structure in high-resolution temporal streams of interaction events.

In this paper we first derive an objective function which computes the description length of a temporal interaction event data set under a three-part encoding that exploits structural regularities and temporal localization in the events while using a temporal hypergraph representation of the data as an intermediate step. We develop an exact dynamic programming algorithm that minimizes this description length objective to find the MDL-optimal configuration of temporal hypergraph snapshots associated with the event data set. To improve scalability for larger data sets, we also develop a fast approximate greedy algorithm to minimize our objective. Our methods are then applied in a variety of experiments involving real and synthetic data sets to demonstrate their utility and performance. We first examine the ability for these algorithms to reconstruct planted hypergraphs in synthetic data, finding that they can recover the planted structure with high accuracy even in the presence of considerable noise. Then we apply our methods to a longitudinal location-based social network (LBSN) data set of check-ins to various locations by app users, finding that we can compress this data to automatically extract meaningful regularities in these human mobility patterns.

II. METHODS

A. Temporal hypergraph binning from bipartite event data

Suppose we are given a data set of N data points ("interaction events") $\mathcal{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$, where each data point $\boldsymbol{x}_i = (s_i, d_i, t_i)$ consists of a source item s_i of one category, a destination item d_i of a different category, and a time t_i when the event involving the source s_i and destination d_i occurred. For simplicity we can assume $\mathcal{X}$ has been ordered in time (i.e., $t_i < t_{i+1}$ for $i = 1, \dots, N-1$), so that the entire time period of interest is $[t_1, t_N]$. We can also assume that the sources $\mathcal{S}$ and destinations $\mathcal{D}$ form disjoint sets of sizes $|\mathcal{S}| = S$ and $|\mathcal{D}| = D$, respectively, and that we are interested in understanding the interactions among items in only one set (e.g., $\mathcal{S}$) as mediated by the events in $\mathcal{X}$. This set comprises the nodes of the hypergraph representation we will construct. Figure 1(a) shows an example of an event data set $\mathcal{X}$ consisting of $N = 10$ events with sources $\mathcal{S} = \{1, 2, 3, 4\}$, destinations $\mathcal{D} = \{A, B, C\}$, and $T = 12$ time steps of size Δt with which we discretize the event times $\{t_i\}$ (see Sec. II B for further details).

Data in this form occurs in a wide variety of applications. Take as an example human mobility data $\mathcal{X}$, where an event $\boldsymbol{x}_i = (s_i, d_i, t_i)$ represents the presence of individual s_i at location d_i at time t_i—we will study this example in more detail in Sec. III using location-based social network data. In this case, for applications across epidemic modeling [5], sociology [39], and urban planning [40], one may be interested in the colocation patterns among individuals in $\mathcal{S}$. Alternatively, in recommendation systems applications, purchasing data often consists of events in which a user s_i purchases a product d_i at time t_i, and correlations among user purchasing behavior can be used for effective advertising of new products [41].

A natural representation of the event data $\mathcal{X}$ in these and similar settings is as a set of hypergraph snapshots $\mathcal{G} = \{\boldsymbol{G}^{(1)}, \dots, \boldsymbol{G}^{(K)}\}$ corresponding to consecutive

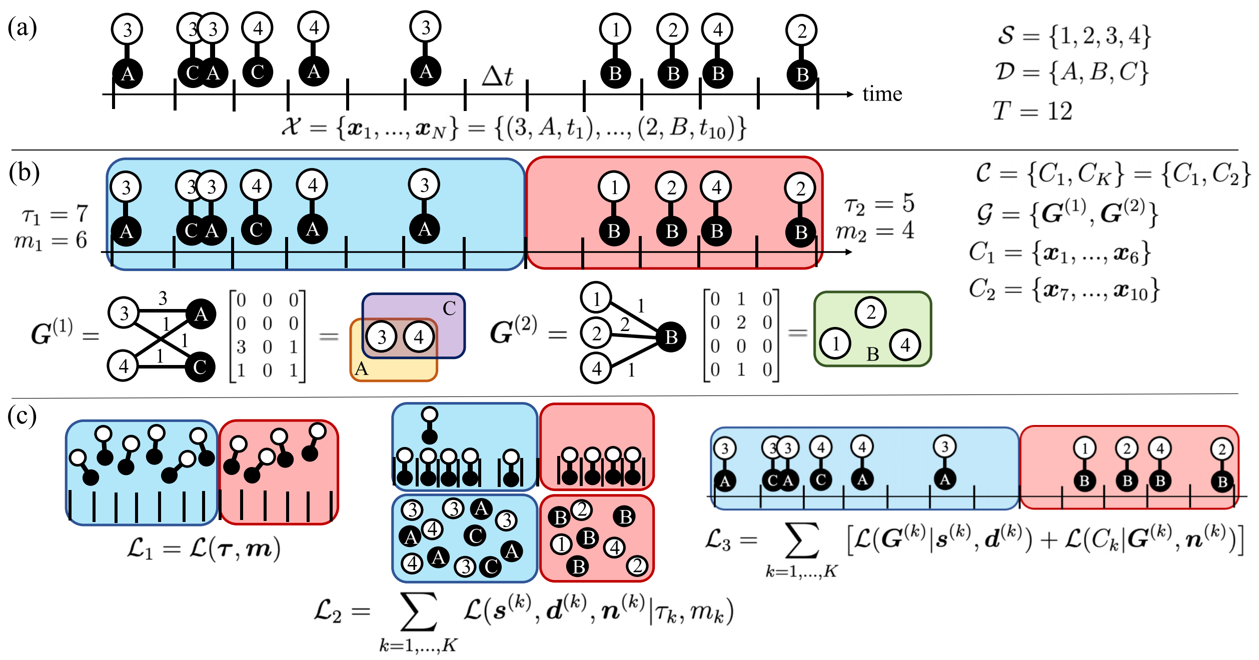

(a)
time
Δt
$\mathcal{X} = \{x_1, ..., x_N\} = \{(3, A, t_1), ..., (2, B, t_{10})\}$
$\mathcal{S} = \{1,2,3,4\}$
$\mathcal{D} = \{A,B,C\}$
$T = 12$
(b)
$\tau_1 = 7$
$m_1 = 6$
$\tau_2 = 5$
$m_2 = 4$
$\mathcal{C} = \{C_1, C_K\} = \{C_1, C_2\}$
$\mathcal{G} = \{G^{(1)}, G^{(2)}\}$
$C_1 = \{x_1, ..., x_6\}$
$C_2 = \{x_7, ..., x_{10}\}$
$G^{(1)} =$
$G^{(2)} =$
(c)
$\mathcal{L}_1 = \mathcal{L}(\tau, m)$
$\mathcal{L}_2 = \sum_{k=1,...,K} \mathcal{L}(s^{(k)}, d^{(k)}, n^{(k)} | \tau_k, m_k)$
$\mathcal{L}_3 = \sum_{k=1,...,K} [\mathcal{L}(G^{(k)} | s^{(k)}, d^{(k)}) + \mathcal{L}(C_k | G^{(k)}, n^{(k)})]$

FIG. 1. Diagram of hypergraph binning method. (a) Data set $\mathcal{X}$, consisting of $N = 10$ events $[x_i = (s_i, d_i, t_i)]$ involving a "source" $s_i \in \mathcal{S}$ and "destination" $d_i \in \mathcal{D}$ interacting at time t_i. $\mathcal{X}$ may, for example, be used to examine colocation patterns from user-location data or copurchasing patterns among consumers in recommendation systems analysis. Time is discretized into T time steps to allow for data compression at a desired temporal resolution $\Delta t = (t_N - t_1)/T$. (b) Hypergraphs $\mathcal{G} = \{G^{(1)}, G^{(2)}\}$ extracted from partitioning the events $\mathcal{X}$ into $K = 2$ clusters $\mathcal{C} = \{\{x_1, \dots, x_6\}, \{x_7, \dots, x_{10}\}\}$ with localized activity patterns. The inferred weighted hypergraphs $G^{(k)}$ are shown in both their incidence (bipartite) representation and their standard representation, with sources s mapped to nodes and destinations d mapped to hyperedges. (c) Three-stage information transmission process used to design a minimum description length objective [Eq. (12)] to infer the hypergraphs $\mathcal{G}$ from event data $\mathcal{X}$. The data $\mathcal{X}$ are transmitted at increasing levels of granularity, and the optimal hypergraphs $\mathcal{G}$ (constructed using clusters $\mathcal{C}$ of events) are selected as those that minimize the description length of the transmission process.

nonoverlapping time intervals $\{[t^{(k)}_{\min}, t^{(k)}_{\max}]\}_{k=1}^{K}$ that partition the time interval $[t_1, t_N]$, and where node $s \in \mathcal{S}$ participates in a hyperedge labeled by $d \in \mathcal{D}$ within hypergraph $G^{(k)}$ if and only if s is involved in an event with d in the time interval $[t^{(k)}_{\min}, t^{(k)}_{\max}]$. Here we allow for a node s to be repeated any number of times within a hyperedge d to signal multiple events involving $\{s, d\}$ in a given time window. For maximum generality, we also allow for $G^{(k)}$ to have self-loops (hyperedges with a single node) as well as multiedges (distinct labeled hyperedges containing the same set of nodes). In other words, $G^{(k)}$ is not necessarily a simple hypergraph. The hypergraph representation $G^{(k)}$ captures all the (potentially indirect) interactions among items in $\mathcal{S}$ that occur via their interactions with items in $\mathcal{D}$ during the time period $[t^{(k)}_{\min}, t^{(k)}_{\max}]$ and can be analyzed using the wealth of newly available tools for higher-order networks [1].

For simplicity of presentation, we will write the hypergraph snapshot $G^{(k)}$ in its weighted bipartite ("incidence") representation $G^{(k)} = \{(s, d, G^{(k)}_{sd})\}_{s,d=1}^{S,D}$, where

$$G^{(k)}_{sd} = \sum_{(s_i,d_i,t_i)\in\mathcal{X}} \mathbb{1}_{t_i\in[t^{(k)}_{\min},t^{(k)}_{\max}]}\delta_{s_i,s}\delta_{d_i,d} \tag{1}$$

is the number of events involving the node s and the hyperedge d within the kth time window. This representation also naturally permits a symmetric treatment of the source set $\mathcal{S}$ and destination set $\mathcal{D}$, in case one is interested in the dual hypergraph representation with node set $\mathcal{D}$ and hyperedge set $\mathcal{S}$—for example, to examine similarities among products rather than users.

To construct a series of hypergraphs $\mathcal{G} = \{G^{(k)}\}_{k=1}^{K}$ of the form in Eq. (1) from event data $\mathcal{X}$, one only needs to make two choices:

(1) The number of temporal snapshots ("bins") K.

(2) The consecutive nonoverlapping time intervals $\{[t^{(k)}_{\min}, t^{(k)}_{\max}]\}_{k=1}^{K}$. Equivalently, in discretized time (see Sec. II B), one just needs to specify the integer-valued interval widths $\boldsymbol{\tau} = \{\tau_1, \dots, \tau_K\}$, where $\tau_k\Delta t = t^{(k)}_{\max} - t^{(k)}_{\min}$ and $\sum_{k=1}^{K} \tau_k = T$ is the total number of time steps in our discretization.

The integer-valued widths $\boldsymbol{\tau}$ alone fully specify the intervals $\{[t^{(k)}_{\min}, t^{(k)}_{\max}]\}_{k=1}^{K}$ in discretized time because of the consecutive, nonoverlapping nature of the intervals discussed above. For this reason, we will refer to $\boldsymbol{\tau}$ as the "binning" of the event data $\mathcal{X}$.

Any binning $\boldsymbol{\tau}$—a partition of (discrete) time—induces a partition of the events in $\mathcal{X}$, which we denote with $\mathcal{C} = \{C_1, \dots, C_K\}$. C_k, which we call the kth "event cluster," is the set of data points $x_i = (s_i, d_i, t_i)$ such that $t_i \in [t^{(k)}_{\min}, t^{(k)}_{\max}]$. From C_k we can construct the kth hypergraph snapshot $G^{(k)}$ using

$$G^{(k)}_{sd} = \sum_{(s_i,d_i,t_i)\in C_k} \delta_{s_i,s}\delta_{d_i,d}. \tag{2}$$

We denote the number of events in C_k with m_k, and the vector of sizes for all clusters in $\mathcal{C}$ as $\boldsymbol{m}$. In this way, the integer vector $\boldsymbol{\tau} = \{\tau_1, \dots, \tau_K\}$ indicates the sizes of the K snapshots in terms of time steps Δt, and the integer vector $\boldsymbol{m} = \{m_1, \dots, m_K\}$ indicates the sizes of the snapshots in terms of the number of events $\boldsymbol{x} \in \mathcal{X}$ they contain. Note that there may be multiple binnings $\boldsymbol{\tau}$ that induce the same event partition $\mathcal{C}$, since any "empty" time steps Δt (i.e., time steps in which no events occur) at the boundary of a snapshot can be moved to an adjacent snapshot without changing the number of events occurring in each snapshot.

In Fig. 1(b) we show a binning of an event data set $\mathcal{X}$ into $K = 2$ bins of widths $\boldsymbol{\tau} = \{\tau_1, \tau_2\} = \{7, 5\}$, which induces an event partition $\mathcal{C} = \{\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_6\}, \{\boldsymbol{x}_7, \dots, \boldsymbol{x}_{10}\}\}$ and hypergraphs $\boldsymbol{G}^{(1)} = \{(3, A, 3), (3, C, 1), (4, A, 1), (4, C, 1)\}$, $\boldsymbol{G}^{(2)} = \{(1, B, 1), (2, B, 2), (4, B, 1)\}$. We show each hypergraph in both its bipartite incidence representation (along with its incidence matrix), as well as in its representation with nodes in $\mathcal{S} = \{1, 2, 3, 4\}$ and hyperedges in $\mathcal{D} = \{A, B, C\}$.

B. Minimum description length binning objective

The method we present in this paper provides a principled nonparametric solution to identify hypergraph snapshots $\mathcal{G}$ of any event data set $\mathcal{X}$ using the MDL principle from information theory, which states that the best model among a set of candidate models is the one that provides the best compression (shortest description) of a data set [23,24]. We do this by constructing a three-part encoding that allows us to gradually transmit the data $\mathcal{X}$ at increasing levels of granularity, with the hypergraphs $\mathcal{G}$ transmitted as an intermediate step in the process. The less information this transmission process requires, the more the hypergraph binning process has compressed the data $\mathcal{X}$ by capturing its statistical regularities, and the better the representation $\mathcal{G}$. The hypergraphs $\mathcal{G}$ that result in the most efficient lossless transmission of the data set $\mathcal{X}$ to a receiver (i.e., the lowest description length) give an MDL-optimal temporal hypergraph representation of $\mathcal{X}$.

In order to construct a lossless MDL objective, we need to discretize the relevant time interval $[t_1, t_N]$ into small, uniform time steps of size $\Delta t = (t_N - t_1)/T$, where T is the number of time steps. The parameter T is technically a free parameter of the method to be chosen by the user, but we show empirically in Sec. III that it has little to no impact on inference results. For this reason we consider the proposed method to be nonparametric since it has no parameters that require tuning by the user other than T, which can be set arbitrarily based on computational limitations (we discuss the time complexity of our methods in Sec. II C). Given the discretization of time into intervals of width Δt, we preprocess the data $\mathcal{X}$ by rounding each t_i to the value of the closest time step, which will incur an error of at most $\Delta t/2$ for each t_i and can potentially permit multiple events to occur simultaneously in the same time step. By discretizing time, we can then proceed with developing a lossless transmission scheme that results in perfect reconstruction of the discretized data $\mathcal{X}$, and whose information content is computed using discrete combinatorial structures. However, due to the rounding, we are in effect performing *lossy* compression with maximum distortion $\Delta t/2$ in the time values reconstructed by a receiver.

With this discretization in place, we can construct our MDL objective for communicating $\mathcal{X}$ using the hypergraphs $\mathcal{G}$ as an intermediate step. The fundamental mechanism behind our encoding is that we can obtain compression of event data $\mathcal{X}$ using hypergraphs $\mathcal{G}$ that are localized in time as well as with respect to sources s and destinations d. This is made possible by our encoding exploiting the redundancies in the events $\boldsymbol{x}_i$ that take place within the event clusters $\mathcal{C}$ corresponding to these hypergraphs. This localization is also consistent with previous findings that bipartite graphs and hypergraphs display heavy-tailed (hyper-)degrees [42–44], as well as "burstiness" in time [11,21,22].

Suppose we want to transmit the (temporally discretized) data set $\mathcal{X}$ to a receiver. We will assume that the number of events N, the number of discrete time steps T, the number of sources (nodes) S, and the number of destinations (hyperedges) D are known by the receiver. These are all integer constants and are of comparatively negligible information cost to transmit, so we can safely ignore them in our formulation. Suppose now that we do not use any intermediate steps in our transmission process that exploit event redundancies, and instead choose to communicate the data $\mathcal{X}$ directly to the receiver as a set of completely independent events. The receiver knows there N events $\boldsymbol{x}_i = (s_i, d_i, t_i)$, each with S possible sources, D possible destinations, and T possible time steps. Therefore, there are $(SDT)^N$ possible configurations of the data $\mathcal{X}$, and so to specify to the receiver in binary which particular configuration corresponds to our data set, we need to send a message of length up to approximately

$$\mathcal{L}_0 = \log[(SDT)^N] = N \log(SDT) \tag{3}$$

bits, where we have used the notation $\log \equiv \log_2$ for brevity. The quantity $\mathcal{L}_0$ is referred to as the description length of the data set $\mathcal{X}$ under the naïve one-level encoding scheme we have devised, which only uses the global information $\{N, T, S, D\}$ to constrain the space of possible data sets $\mathcal{X}$.

A smarter way to transmit the data $\mathcal{X}$ that exploits the redundancies we seek in our hypergraph representation $\mathcal{G}$ is to transmit $\mathcal{X}$ to the receiver in three stages by passing binary messages that communicate information at increasing levels of granularity and successively constrain the space of possible data sets $\mathcal{X}$ until there is only one remaining possibility. Each step in our transmission scheme requires an information content (e.g., description length) given by the logarithm of the number of possible message configurations, as in Eq. (3). We will assume that the number of data bins (i.e., hypergraphs and event clusters) K is known by the receiver and can ignore its information along with the other constants above. Crucially, although K is assumed known by the receiver, it remains a free variable in our description length optimization process, as we will see in Sec. II C. Our transmission process proceeds as follows:

(1) Transmit aggregate cluster-level information (event cluster sizes $\boldsymbol{m}$ and bin widths $\boldsymbol{\tau}$):

(a) To transmit $\boldsymbol{\tau} = \{\tau_k\}_{k=1}^{K}$, we must specify K positive integers that sum to T in a particular order—also known as a "K composition" of T. There are $\binom{T-1}{K-1}$ ways to do this, which can be shown using the classic "stars and bars" method from combinatorics [45]. Therefore, specifying the particular K composition corresponding to $\boldsymbol{\tau} = \{\tau_k\}_{k=1}^K$ requires $\log \binom{T-1}{K-1}$ bits of information.

(b) Similarly, $\boldsymbol{m} = \{m_k\}_{k=1}^K$ requires $\log \binom{N-1}{K-1}$ bits of information to specify, as it consists of K positive integers that sum to N.

The total information content of this first stage is therefore given by the sum of these two contributions:

$$\mathcal{L}_1 = \mathcal{L}(\boldsymbol{\tau}, \boldsymbol{m}) = \log \binom{T-1}{K-1} + \log \binom{N-1}{K-1}. \tag{4}$$

(2) Transmit detailed cluster-level information (counts of sources, destinations, and timestamps for each event cluster C_k):

(a) The number of instances (bipartite degree) of each source in event cluster C_k is stored in the vector $\boldsymbol{s}^{(k)} = \{s_r^{(k)}\}_{r=1}^S$, with $s_r^{(k)}$ the number of occurrences of source r in event cluster C_k. There are $\left(\!\binom{S}{m_k}\!\right)$ possible ways to assign each of the S sources a non-negative integer degree value such that the sum of the degrees is m_k, where $\left(\!\binom{y}{x}\!\right) = \binom{x+y-1}{y-1}$ is the multiset coefficient counting the number of ways to assign x objects to y distinct bins while allowing bins to be empty. (This result can also be found using the stars and bars argument.) Therefore, transmitting the particular counts $\boldsymbol{s}^{(k)} = \{s_r^{(k)}\}_{r=1}^S$ requires $\log\left(\!\binom{S}{m_k}\!\right)$ bits of information.

(b) The number of instances (bipartite degree) of each destination in event cluster C_k is stored in the vector $\boldsymbol{d}^{(k)} = \{d_r^{(k)}\}_{r=1}^D$, with $d_r^{(k)}$ the number of occurrences of destination r in event cluster C_k. Transmitting these counts requires $\log\left(\!\binom{D}{m_k}\!\right)$ bits of information.

(c) The number of events $\boldsymbol{x}_i$ in event cluster C_k that occur at each discrete time step within the temporal boundaries of the cluster is stored in the vector $\boldsymbol{n}^{(k)} = \{n_t^{(k)}\}_{t=1}^{\tau_k}$. Here $n_t^{(k)}$ the number of events within event cluster C_k that fall into the tth time step within the cluster's boundary (there are τ_k time steps to choose from). Transmitting these counts requires $\log\left(\!\binom{\tau_k}{m_k}\!\right)$ bits of information.

The total information content of this second stage is therefore given by the sum of these three contributions for each cluster C_k:

$$\mathcal{L}_2 = \sum_{k=1}^{K} \mathcal{L}(\boldsymbol{s}^{(k)}, \boldsymbol{d}^{(k)}, \boldsymbol{n}^{(k)} | \tau_k, m_k), \tag{5}$$

$$= \sum_{k=1}^{K} \left[\log \left(\!\binom{S}{m_k}\!\right) + \log \left(\!\binom{D}{m_k}\!\right) + \log \left(\!\binom{\tau_k}{m_k}\!\right) \right]. \tag{6}$$

(3) Transmit the events $\boldsymbol{x}_i = (s_i, d_i, t_i)$ within each cluster C_k, which fully specifies $\mathcal{X}$: We have the following three constraints based on previously transmitted information:

$$\sum_{r=1}^{S} s_r^{(k)} = m_k, \tag{7}$$

$$\sum_{r=1}^{D} d_r^{(k)} = m_k, \tag{8}$$

$$\sum_{t=1}^{\tau_k} n_t^{(k)} = m_k. \tag{9}$$

Therefore, the number of non-negative integer-valued three-dimensional (3D) tensors with margins defined by $\boldsymbol{s}^{(k)}, \boldsymbol{d}^{(k)}, \boldsymbol{n}^{(k)}$ is the number of possibilities for $\mathcal{X}$, and the logarithm of this quantity is the information content of this last step. However, this quantity itself is difficult to compute, so we can break up this last step into two stages, one of which involves the hypergraphs we are looking for:

(a) Transmit the hypergraph $\boldsymbol{G}^{(k)}$ given the bipartite degree constraints $\boldsymbol{s}^{(k)}, \boldsymbol{d}^{(k)}$. This requires $\log \Omega(\boldsymbol{s}^{(k)}, \boldsymbol{d}^{(k)})$ bits of information, where $\Omega(\boldsymbol{s}^{(k)}, \boldsymbol{d}^{(k)})$ is the number of non-negative integer-valued matrices with margins $\boldsymbol{s}^{(k)}, \boldsymbol{d}^{(k)}$. $\log \Omega(\boldsymbol{s}^{(k)}, \boldsymbol{d}^{(k)})$ is in general difficult to compute exactly, but can be approximated in order $O(S + D)$ time using the effective columns approximation in Ref. [46]. Here we use $O(\cdot)$ to indicate "Big O" notation.

(b) Transmit the final data points $\boldsymbol{x}_i = (s_i, d_i, t_i)$ in cluster C_k given the hypergraph $\boldsymbol{G}^{(k)}$ and the time step counts $\boldsymbol{n}^{(k)}$. Transmitting these requires $\log \Omega(\boldsymbol{G}^{(k)}, \boldsymbol{n}^{(k)})$ bits of information, where $\Omega(\boldsymbol{G}^{(k)}, \boldsymbol{n}^{(k)})$ is the number of non-negative integer-valued matrices with margins $\boldsymbol{n}^{(k)}$ and $\{G_{sd}^{(k)}\}_{s,d=1}^{S,D}$. Using the approximation in Ref. [46], $\log \Omega(\boldsymbol{G}^{(k)}, \boldsymbol{n}^{(k)})$ can be estimated in order $O(SD + \tau_k)$ time.

The total information content of this third stage is therefore given by the sum of these two contributions for each cluster C_k:

$$\mathcal{L}_3 = \sum_{k=1}^{K} [\mathcal{L}(\boldsymbol{G}^{(k)} | \boldsymbol{s}^{(k)}, \boldsymbol{d}^{(k)}) + \mathcal{L}(C_k | \boldsymbol{G}^{(k)}, \boldsymbol{n}^{(k)})], \tag{10}$$

$$= \sum_{k=1}^{K} [\log \Omega(\boldsymbol{s}^{(k)}, \boldsymbol{d}^{(k)}) + \log \Omega(\boldsymbol{G}^{(k)}, \boldsymbol{n}^{(k)})]. \tag{11}$$

Summing the description length of each stage, we have a total description length of

$$\mathcal{L}_{\text{total}}(\mathcal{X}, \boldsymbol{\tau}) = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3, \tag{12}$$

where we have explicitly noted the functional dependence of $\mathcal{L}$ on the binning $\boldsymbol{\tau}$, since the description length of the data $\mathcal{X}$ under our transmission scheme—including the hypergraphs $\mathcal{G}$—can be computed when $\boldsymbol{\tau}$ is known. Figure 1(c) shows a schematic of the three stages described above, for the event data set in Fig. 1(a). In the next section we describe how to minimize the description length of Eq. (12) over all binnings $\boldsymbol{\tau}$ to find the MDL set of hypergraphs $\mathcal{G}$ that best summarize the temporal event data $\mathcal{X}$.

We note that the transmission scheme described above and its resulting description length can also be motivated from a (microcanonical) Bayesian generative model for temporal event data, with hierarchical uniform priors corresponding to each transmission step and where description lengths are negative log-probabilities. In other words, each step in the transmission corresponds to a uniform prior over valid configurations given the constraints previously transmitted. For example, the very first transmission step corresponds to drawing K bin widths $\{\tau_1, \ldots, \tau_K\}$ uniformly at random given the constraint that they must sum to T. And the final transmission step corresponds to drawing the event data from a uniform distribution over all valid event configurations given the weighted hypergraph representation and temporal clustering patterns.

This model—possessing conceptual similarities to a configuration model for bipartite graph data where networks are generated uniformly at random given degree constraints on both node sets—is only one possible way to generate temporal hypergraphs, but it is arguably the simplest method that can meaningfully capture statistical regularities in event data when applied to discrete time periods that contain one or zero events (which is the case for $T \to \infty$). Optimization over our objective corresponds to maximum *a posteriori* estimation in the corresponding Bayesian model, and we discuss how to do this in the next section.

C. Optimization and model selection

The description length of Eq. (12) amounts to a one-dimensional clustering objective over binnings $\boldsymbol{\tau}$. Therefore, if our objective consisted of independent terms for each cluster (akin to the objective for K-means clustering), then we could optimize it exactly using a dyamic programming approach [38,47–49].

Only the first term in Eq. (12) couples clusters together, but in the regime we are interested, we have $K \ll T, N$ and we can rewrite Eq. (12) as (up to irrelevant constant factors)

$$\mathcal{L}_{\text{total}}(\mathcal{X}, \boldsymbol{\tau}) = \sum_{k=1}^{K} \mathcal{L}^{(k)}_{\text{cluster}}, \tag{13}$$

where

$$\begin{aligned}\mathcal{L}^{(k)}_{\text{cluster}} &= \log(N-1)(T-1)\\ &\quad + \log \left(\!\!\binom{S}{m_k}\!\!\right)\left(\!\!\binom{D}{m_k}\!\!\right)\left(\!\!\binom{\tau_k}{m_k}\!\!\right)\\ &\quad + [\log \Omega(\boldsymbol{s}^{(k)}, \boldsymbol{d}^{(k)}) + \log \Omega(\boldsymbol{G}^{(k)}, \boldsymbol{n}^{(k)})]. \end{aligned} \tag{14}$$

We can now minimize our MDL objective in Eq. (13) using a dynamic program. The key intuition behind this is that since the objective in Eq. (13) is a sum of independent terms over clusters in one dimension, its minimum over the first j time steps—i.e., the optimal binning $\boldsymbol{\tau}$ restricted to these first j time steps—must consist of the optimal binning up to some time step $i \in \{1, \dots, j\}$ (excluding the ith time step) plus a final cluster of time steps $i, \dots, j$. In other words, for all $j \in [1, T]$ we have

$$\mathcal{L}^{(j)}_{\text{MDL}} = \min_{i \in [1,j]} \left\{\mathcal{L}^{(i-1)}_{\text{MDL}} + \mathcal{L}^{([i,j])}_{\text{cluster}}\right\}, \tag{15}$$

where $\mathcal{L}^{(j)}_{\text{MDL}}$ is the minimum value of Eq. (13) when we include only the first j timestamps Δt, and $\mathcal{L}^{([i,j])}_{\text{cluster}}$ is the cluster-level description length of Eq. (14) evaluated at the cluster containing consecutive time steps $\{i, \dots, j\}$. Setting $\mathcal{L}^{(0)}_{\text{MDL}} = 0$ and iterating over all $j \in [1, T]$, we find the minimum of the description length in Eq. (13), giving us the optimal binning $\boldsymbol{\tau}$ for the data $\mathcal{X}$ according to the MDL principle.

In addition to finding the exact optimum over binnings $\boldsymbol{\tau}$, this approach has the advantage of automatically selecting the optimal number of bins K, since the entire unconstrained space of binnings $\boldsymbol{\tau}$ is explored by the algorithm. The objective function in Eq. (13) will naturally penalize high values of K since we will waste information to describe the clusters and increase the total description length if K is too high. On the other hand, Eq. (13) will also naturally penalize values of K that are too low, since we will waste information describing the events within each cluster if they are too heterogeneous and/or spread over too large a time period. The MDL-optimal binning $\boldsymbol{\tau}$ therefore balances the information required to describe the clusters and the information required to describe the data within each cluster by selecting an appropriate number of clusters K using the data itself.

To quantify the extent to which our method has achieved compression over a naïve one-level encoding, we could take the ratio of the optimal description length $\mathcal{L}_{\text{MDL}}$ from Eq. (13) with the description length of Eq. (3). However, in our case it is of more interest to determine how much of a compression gain we achieve when we use an optimal configuration of multiple hypergraphs to summarize the temporal event data $\mathcal{X}$, versus using only a single hypergraph that aggregates all the events together. We therefore construct an inverse compression ratio η which computes our compression gain as

$$\eta = \frac{\mathcal{L}_{\text{MDL}}}{\mathcal{L}(K=1)}, \tag{16}$$

where $\mathcal{L}(K=1)$ is the description length of Eq. (13) when all events are put into a single event cluster. A value $\eta = 1$ implies that the event data set $\mathcal{X}$ is not compressible using multiple hypergraphs, while $\eta \ll 1$ implies that the event data set $\mathcal{X}$ can be greatly compressed using a representation of multiple hypergraphs.

Evaluating $\mathcal{L}^{([i,j])}_{\text{cluster}}$ requires the evaluation of constant-time terms and two $\log \Omega$ terms which have nontrivial time complexity. Computing $\log \Omega(\boldsymbol{s}^{(k)}, \boldsymbol{d}^{(k)})$ requires $O(S+D)$ operations once the margin counts $\boldsymbol{s}^{(k)}, \boldsymbol{d}^{(k)}$ are known, and these margin counts require $O(m_k)$ operations to compute since we must iterate through the events in the cluster C_k. If the events are roughly evenly spaced in time, then we would expect that $m_k \approx N(j-i)/T$ events would occur within the interval $[i, j]$, and the total complexity of evaluating the term $\log \Omega(\boldsymbol{s}^{(k)}, \boldsymbol{d}^{(k)})$ would be $O(S+D+N(j-i)/T)$. Meanwhile, evaluating the term $\log \Omega(\boldsymbol{G}^{(k)}, \boldsymbol{n}^{(k)})$ will have complexity $O(SD + \tau_k)$, with $\tau_k = j - i$ in this case. Combining these operations gives a total complexity of approximately $O(N(j-i)/T + SD + (j-i))$ for evaluating $\mathcal{L}^{([i,j])}_{\text{cluster}}$. Iterating over all j and i in the recursion then gives a total complexity of roughly $O[(SD + N + T)T^2]$ for the dynamic programming algorithm using a naïve implementation.

However, we can speed up the method in the regime $SD, N \ll T$ by saving the $\mathcal{L}^{([i,j])}_{\text{cluster}}$ values as they are computed, requiring order $O(T^2)$ space. $\mathcal{L}^{([i,j])}_{\text{cluster}}$ can be computed from $\mathcal{L}^{([i-1,j])}_{\text{cluster}}$ in constant time if time step i has no events. Similarly, $\mathcal{L}^{([i,j])}_{\text{cluster}}$ can be computed from $\mathcal{L}^{([i,j-1])}_{\text{cluster}}$ in constant time if time step j has no events. Thus, when we loop over $j \in [1, T]$ and $i \in [1, j]$, we will only have to perform an $O[N(j-i)/T + SD + (j-i)]$ evaluation of $\mathcal{L}^{([i,j])}_{\text{cluster}}$ when both cells i and j contain an event. Otherwise, we perform a constant-time update using $\mathcal{L}^{([i-1,j])}_{\text{cluster}}$ or $\mathcal{L}^{([i,j-1])}_{\text{cluster}}$ (depending on which endpoint does not have any event).

If $T \gg N$ for N constant—many temporal grid cells have no event—then we have roughly N unique cells containing events, resulting in $\binom{N}{2} \ll \binom{T}{2}$ pairs $\{i, j\}$ for which both the interval endpoints of $[i, j]$ contain an event, and for which we must perform the entire $O[N(j-i)/T + SD + (j-i)]$ evaluation of $\mathcal{L}_{\text{cluster}}^{([i,j])}$. Thus, a vanishingly small fraction of pairs $\{i, j\}$ require these nontrivial evaluations, so the $O(T^2)$ evaluations will require roughly $O(T^2)$ total runtime to compute if $SD, N \ll T$ with SD and N constant with respect to T.

When the above scaling assumptions for SD and N are not met, the $O(T^2)$ approximate total runtime of the dynamic programming method can break down. This is because the $O(N^2)$ nontrivial evaluations of complexity $O[N(j-i)/T + SD + (j-i)]$ for $\mathcal{L}_{\text{cluster}}^{([i,j])}$ become important.

Despite this speed-up, the time complexity of our exact dynamic programming solution may be too high for practical applications where SD or N are comparable to T, or where we require high values of T for sufficient temporal resolution. In such cases we can use a greedy heuristic optimization method where we start with all time steps $i = 1, \dots, T$ in their own cluster and iteratively merge the pair of time steps that gives the greatest decrease to the description length in Eq. (13). We save the description length changes induced by all proposed merges (including those that were suboptimal) and perform greedy merges until all time steps are in a single cluster. We then pick the value of K for which the total description length was minimized over our set of merges. This greedy optimization method is not guaranteed to find the exact optimum, but it has a time complexity that is nearly $O(T)$ in practice, as for each $K = T, \dots, 1$ we will only have to update the $\log \Omega$ terms for two merge pairs (those involving adjacent clusters to the one most recently merged).

In Sec. III A and the Appendix we run numerical experiments computing the runtimes and inverse compression ratios achieved by the two algorithms on synthetic and real data sets, respectively, finding that—in the set of examples we have examined—this greedy method achieves MDL values that are nearly optimal but with much faster runtimes than the dynamic programming approach. However, one can never conclusively state that a greedy approximate method such as the one described above will be nearly optimal in a broader range of real applications without formal mathematical proof—indeed, approximate methods may perform quite poorly in practice for high-dimensional clustering problems [50].

III. RESULTS

A. Reconstruction of synthetic data

To examine the performance of the algorithms presented in Sec. II C, we can generate synthetic data consisting of planted event clusters $\mathcal{C}$ with binnings $\boldsymbol{\tau}$ and test the ability for these algorithms to recover the planted clusters at various levels of injected noise. We generated synthetic data sets with $N \in [200, 500, 1000]$, $T \in [50, 500]$, $K \in [2, 5, 10]$, and $S = D = 5$ (the results did not depend on S and D) in order to examine a range of model settings for the reconstruction tests.

The synthetic event clusters $\mathcal{C}$ are generated by first drawing a partition of the N events and T time steps into K sets uniformly at random, then drawing the time step counts $\boldsymbol{n}^{(k)}$ uniformly at random within each cluster. To control the level of heterogeneity across the K synthetic event clusters—which in turn controls the level of noise in the partition of the events, and consequently the reconstruction difficulty—our synthetic model includes a parameter $\gamma \geqslant 0$ which determines the localization of the edges (s, d) within hypergraph $\boldsymbol{G}^{(k)}$ on sources $s \in \mathcal{S}$ and $d \in \mathcal{D}$. More specifically, for each cluster C_k we independently generate the bipartite degrees $\boldsymbol{s}^{(k)}$ and $\boldsymbol{d}^{(k)}$ from a Dirichlet-multinomial distribution with m_k trials and concentration parameter $\gamma \mathbf{1}$, then draw the bipartite graph $\boldsymbol{G}^{(k)}$ at random from the set of non-negative integer matrices with row and column sums $\boldsymbol{s}^{(k)}$ and $\boldsymbol{d}^{(k)}$ using the algorithm in Ref. [51]. This will create more localized bipartite degree distributions within hypergraph $\boldsymbol{G}^{(k)}$ [thus, more localized edge weights $G_{sd}^{(k)}$] and a higher variance across clusters as $\gamma \to 0$. The concentration parameter γ thus serves as a tunable parameter that determines the distinguishability of the generated synthetic clusters, with $\gamma \to 0$ increasing the distinguishability of the clusters (e.g., increasing the signal-to-noise ratio).

In Fig. 2 we plot the results of our synthetic reconstruction experiments. In Fig. 2(a) we show the inverse compression ratio η [Eq. (16)] vs the logarithm of the planted heterogeneity γ for $N \in [200, 500, 1000]$, averaged over 30 simulations at each combination of K and T. Error bars represent three standard errors in the mean values estimated from the simulations, and the solid and dotted curves correspond to the dynamic programming and greedy algorithms described in Sec. II C, respectively. We set $\Delta t = 1$ for the reconstruction simulations. We can see that as the noise level γ increases from $\gamma = 10^{-3}$ to $\gamma = 1$, the synthetic event clusters $\mathcal{X}$ become less and less compressible, and that more events N results in better compression at all noise levels due to additional statistical evidence for the structure of each cluster. These results indicate that substantial data compression is possible using our algorithm, even at relatively high noise levels. We can also see very similar compression performance between the exact and greedy algorithms, indicating that, for these examples, the greedy method is achieving near-optimal compression at much lower computational cost than the dynamic programming method.

As there are multiple binnings $\boldsymbol{\tau}$ that could correspond to any given set of event clusters $\mathcal{C}$—any binning that preserves the event partition while shifting the cluster boundaries in time—there is always a high level of uncertainty in reconstruction of $\boldsymbol{\tau}$, even with perfectly distinguishable event clusters $\mathcal{C}$. We therefore quantify the reconstruction accuracy in our simulations by representing an event partition $\mathcal{C}$ as a 1D partition of the temporally ordered event indices $\{1, \dots, N\}$, then compute a mutual information measure between the 1D partitions corresponding to the planted clusters $\mathcal{C}_{\text{pl}}$ and the clusters $\mathcal{C}_{\text{in}}$ inferred by our algorithm. However, standard mutual information-based measures [52] are poorly suited for contiguous, low-dimensional partitions such as the ones we compare here, because they compute the partition similarity relative to an unconstrained (e.g., not necessarily contiguous) space of partitions that is much larger in size and much less structured than the space of contiguous partitions [38,53]. This results in artificially inflated values of the mutual information between contiguous partitions that may have little correlation other than that induced by their contiguity.

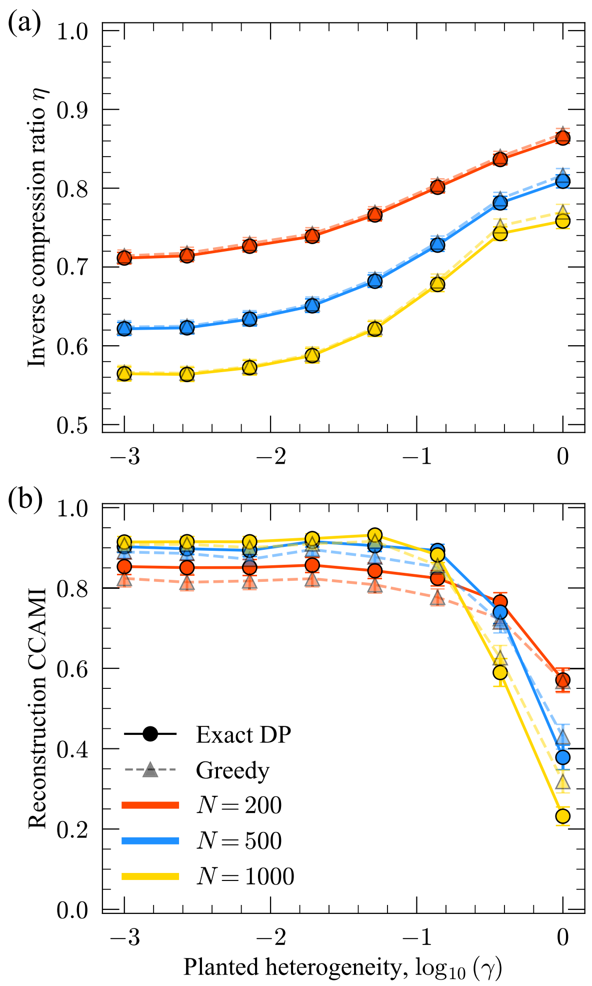


FIG. 2. Synthetic reconstruction performance. (a) Average inverse compression ratio [Eq. (16)] versus the logarithm of the planted level of cluster heterogeneity γ, for $N \in \{200, 500, 1000\}$. The exact dynamic programming algorithm results are shown with solid lines and circular markers, while the greedy algorithm results are shown with dotted lines and triangular markers. (b) Reconstruction accuracy, as quantified by the contiguity-corrected AMI [CCAMI, Eq. (17)], over the same set of experiments. Averages for each panel are taken over 30 simulations with the parameters $\{S, D, K, T\}$ described in Sec. III A, and error bars represent 3 standard errors in the mean.

With this in mind, here we construct a contiguity-corrected adjusted mutual information (CCAMI) to compute the similarity between the event clusters $\mathcal{C}_{\mathrm{pl}}$ and $\mathcal{C}_{\mathrm{in}}$, which is given by

$$\mathrm{CCAMI}(\mathcal{C}_{\mathrm{pl}}, \mathcal{C}_{\mathrm{in}}) = \frac{\mathrm{MI}(\mathcal{C}_{\mathrm{pl}}, \mathcal{C}_{\mathrm{in}}) - \langle \mathrm{MI}(\mathcal{C}_{\mathrm{pl}}, \mathcal{C}_{\mathrm{in}}) \rangle_c}{\max\{H(\mathcal{C}_{\mathrm{pl}}), H(\mathcal{C}_{\mathrm{in}})\} - \langle \mathrm{MI}(\mathcal{C}_{\mathrm{pl}}, \mathcal{C}_{\mathrm{in}}) \rangle_c}, \tag{17}$$

where

$$\mathrm{MI}(\mathcal{C}_{\mathrm{pl}}, \mathcal{C}_{\mathrm{in}}) = H(\mathcal{C}_{\mathrm{pl}}) + H(\mathcal{C}_{\mathrm{in}}) - H(\mathcal{C}_{\mathrm{pl}}, \mathcal{C}_{\mathrm{in}}) \tag{18}$$

is the standard Shannon mutual information between partitions $\mathcal{C}_{\mathrm{pl}}$ and $\mathcal{C}_{\mathrm{in}}$, $H(\mathcal{C})$ is the Shannon entropy of the cluster sizes in partition $\mathcal{C}$, and $H(\mathcal{C}_{\mathrm{pl}}, \mathcal{C}_{\mathrm{in}})$ is the Shannon entropy of the joint partition defined by the overlap of the clusters in $\mathcal{C}_{\mathrm{pl}}$ and $\mathcal{C}_{\mathrm{in}}$. $\langle \mathrm{MI}(\mathcal{C}_{\mathrm{pl}}, \mathcal{C}_{\mathrm{in}}) \rangle_c$ is the expected value of this mutual information over all possible contiguous partitions with the same numbers of groups as $\mathcal{C}_{\mathrm{in}}$ and $\mathcal{C}_{\mathrm{pl}}$. Equation (17) quantifies how much information is shared between the planted event partition $\mathcal{C}_{\mathrm{pl}}$ and our inferred event partition $\mathcal{C}_{\mathrm{in}}$, relative to all pairs of contiguous partitions with the same numbers of clusters. In practice, $\langle \mathrm{MI}(\mathcal{C}_{\mathrm{pl}}, \mathcal{C}_{\mathrm{in}}) \rangle_c$ is difficult to compute analytically, so we estimate it using an average over 100 random draws of partitioning the N events into $|\mathcal{C}_{\mathrm{in}}|$ and $|\mathcal{C}_{\mathrm{pl}}|$ contiguous clusters.

In Fig. 2(b) we show the reconstruction accuracy of our experiments as a function of the noise level γ, with the same parameter settings as in Fig. 2(a). Consistent with the improved compression at lower γ seen in Fig. 2(a), we can see better reconstruction accuracy as γ decreases and for synthetic data sets with a greater number of events N when $\gamma \leqslant 10^{-1}$. When the noise level increases to $\gamma > 10^{-1}$, we observe a sharper drop in reconstruction accuracy for greater N, likely as a result of finite-size smoothing in phase transition-like behavior for the model detectability [54]. We can also see that the exact and greedy algorithms have a non-negligible performance distinction with respect to reconstruction accuracy—as opposed to compression, as shown in Fig. 2(a)—since in the low-noise regime the CCAMI values are noticeably lower for the greedy method than the exact method. This relative performance discrepancy for the greedy algorithm in Fig. 2(a) and Fig. 2(b) indicates that in some cases good data compression can be achieved for a variety of different partitions of the events in $\mathcal{X}$.

To verify the time complexities estimated in Sec. II C, in Fig. 3(a) we plot the average run time for the dynamic programming algorithm (left axis) and greedy algorithm (right axis) as a function of the number of time steps T, for the reconstruction simulations in Fig. 2. Along with these points we plot the regression lines obtained from fits of the form $\log(\mathrm{Runtime}) = \beta_1 \log(T) + \beta_2$, which are labeled with their least-squares estimates for the exponent $\hat{\beta}_1$. We can see that the dynamic program has a run time that scales approximately like $O(T^2)$, and that the greedy algorithm has a run time that is slightly worse than linear in the number of time steps T, while the absolute run times for the greedy algorithm are much faster than those for the dynamic program.

In Fig. 3(b), we plot the average inverse compression ratio η of the reconstruction experiments as a function of the number of time steps T. We can see that the compression is essentially identical for all values of T, up to fluctuations. This indicates that the results of our algorithm are independent of the specific choice of temporal resolution T as long as it is in a reasonable range (roughly at least on the order of the number of events N) that does not merge large time periods into single time steps.

B. Foursquare check-ins in NYC neighborhoods

To examine the performance of our method on real event data $\mathcal{X}$, we apply the algorithms of Sec. II C to a data set of FourSquare check-ins in New York City collected from April 2012 to February 2013 [55,56]. In this data set, each check-in event $\boldsymbol{x}_i = (s_i, d_i, t_i) \in \mathcal{X}$ denotes a FourSquare check-in by

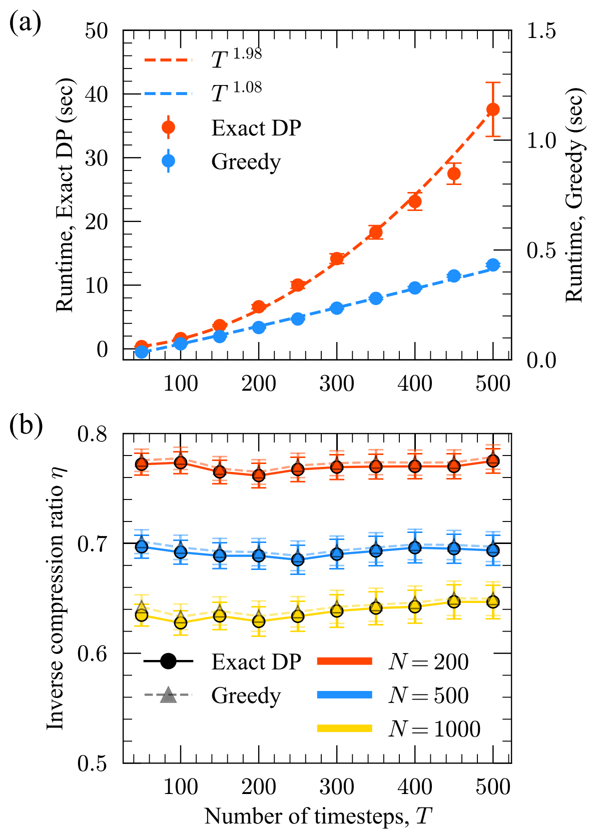


FIG. 3. Reconstruction parameter sensitivities. (a) Average run time (in seconds) of reconstruction experiments (Fig. 2) versus number of time steps T, for both algorithms described in Sec. II C. The performance of the exact dynamic programming algorithm is shown on the left axis, while that of the greedy algorithm is shown on the right axis. Regression lines of the form $\log(\text{Runtime}) = \beta_1 \log(T) + \beta_2$, labeled with their least-squares estimates for the exponent $\hat{\beta}_1$, are shown as dotted lines. (b) Inverse compression ratio η [Eq. (16)] versus T for the experiments conducted at different values of N. Averages for each panel are taken over 30 simulations with each combination of the parameters $\{S, D, K, \gamma\}$ described in Sec. III A [the averages in panel (a) also allow N to vary], and error bars represent 3 standard errors in the mean.

a user s_i at venue d_i at time t_i. LBSN data of this form are often used in urban planning, epidemiology, and sociology to understand human mobility colocation patterns [5,57,58], where users $s, s', s'', \ldots \in \mathcal{S}$ are colocated if they check in at the same venue $d \in \mathcal{D}$ within some predefined time window. We can use the MDL method described in Sec. II to automatically extract a set of representative hypergraphs $\mathcal{G}$ from the LBSN check-in data $\mathcal{X}$ that capture homogeneous user activity patterns at different points in time. This allows us to, for instance, perform market segmentation to identify users $s \in \mathcal{S}$ with similar consumption patterns at different points in the year, or to identify seasonality in the congestion patterns at different venues.

To preprocess the FourSquare check-in data for analysis, we used neighborhood boundary shapefiles [59] to map the (latitude, longitude) pairs of the check-ins to neighborhoods in NYC. We kept only the 1000 users and venues in the data set with the most check-ins and neighborhoods with at least 100 check-ins over the 10 month period, with the aim of reducing biases and noise from users and venues with very infrequent app usage. The final data set used in the analysis had $N = 64\,366$ events and $S = D = 1000$ users and venues spread across 91 neighborhoods.

In Fig. 4 we show the results of applying our exact dynamic programming method for hypergraph binning to the check-ins for each neighborhood separately. This neighborhood-level analysis allows us to more easily visualize the inferred hypergraphs as well as perform cross-sectional comparisons across the neighborhoods regarding their event homogeneity, temporal burstiness, and compressibility. In our inference we set $\Delta t = 1$ day. In Fig. 4(a) we show the inferred hypergraphs $\mathcal{G} = \{\boldsymbol{G}^{(1)}, \boldsymbol{G}^{(2)}, \boldsymbol{G}^{(3)}, \boldsymbol{G}^{(4)}\}$ for the neighborhood (Bay Terrace, Queens) for which our method resulted in the highest level of data compression ($\eta = 0.68$). The hypergraphs $\boldsymbol{G}^{(k)}$, which are ordered chronologically left to right, are shown in their incidence representation, with the width of edge (s, d) proportional to the edge weight $G_{sd}^{(k)}$ which counts the number of events that contain user s and venue d in the time period corresponding to hypergraph $\boldsymbol{G}^{(k)}$. Source and destination nodes in this representation are scaled in size proportionally to their weighted bipartite degrees (e.g., frequency of occurrence in events within the time period) and labeled by unique user and venue ids, respectively, for each neighborhood.

We can see that the four inferred hypergraphs in Fig. 4(a) are very structurally distinct from one another. In the first time period inferred by our method (from the start of the study until July 9), corresponding to hypergraph $\boldsymbol{G}^{(1)}$, we observe that a large portion of the activity was dominated by user "u89," who visited venues "v2" through "v7" as well as "v9," the last of which was also visited by the rest of the users except "u31." User "u31" also made a substantial number of check-ins during this period but only to venue "v8." The inclusion of these two distinct activity patterns (check-ins by "u89" and "u31") in the hypergraph $\boldsymbol{G}^{(1)}$ is a result of both users performing their check-ins consistently over the time period corresponding to the first hypergraph. In the second hypergraph (corresponding to the time period July 10–October 26), we can see that user "u31" is still making consistent check-ins at venue "v8," but that user "u89" is no longer making check-ins. There is also turnover in the other users and venues. In the third hypergraph (corresponding to October 27–November 1) we see a very different check-in activity pattern, which corresponds to the tropical storm Hurricane Sandy hitting New York City. Here we see many users (too many to be labeled in the figure) check-ing in at location "v1," Throgs Neck Bridge between Queens and the Bronx, likely signaling evacuation and return to the city. In the fourth hypergraph (corresponding to November 2 through the end of the study period), we see a return to normal with a somewhat similar activity structure as in the second hypergraph, where most check-ins are performed by users "u31" and "u136" at venues 'v8' and "v9," respectively. The check-in data for this neighborhood is easily compressed using our method due to these four very distinct periods of high localization of the events onto a few users and venues.

In contrast, in Fig. 4(b), we see a very different story for Melrose in the Bronx. Here we see that the event data was optimally compressed into a single hypergraph

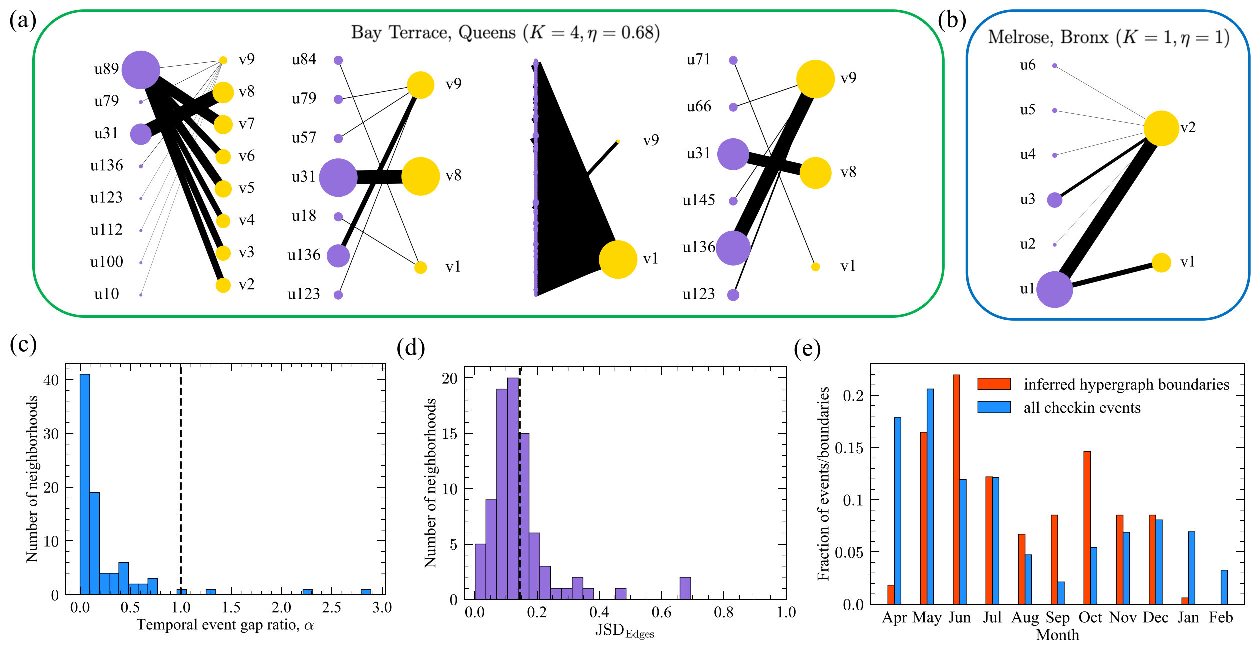


FIG. 4. FourSquare check-ins in NYC neighborhoods. The data set, which aggregated check-ins from April 2012 to February 2013 in New York City [55,56], consists of events $\boldsymbol{x}_i = (s_i, d_i, t_i) \in \mathcal{X}$ that denote a FourSquare check-in by a user s_i at venue d_i at time t_i. (a) Inferred hypergraphs for the Bay Terrace neighborhood, for which our method resulted in the highest level of data compression ($\eta = 0.68$). The hypergraphs are ordered chronologically left to right and shown in their incidence representation, with the width of edge (s, d) proportional to the edge weight $G^{(k)}_{sd}$ which counts the number of events that contain user s and venue d in the time window. Source and destination nodes are scaled proportionally to their frequency of occurrence and labeled by unique user and venue ids, respectively, for each neighborhood. (b) Inferred hypergraph for Melrose, for which our method resulted in the lowest level of data compression ($\eta = 1$). (c) Histogram of the temporal event gap ratio α [Eq. (19)] for all neighborhoods with $K > 1$. (d) Histogram of the edge Jensen-Shannon divergence $\mathrm{JSD}_{\mathrm{Edges}}$ [Eq. (23)] for all neighborhoods with $K > 1$, with mean indicated using the dotted black line. (e) Fraction of all events and inferred temporal bin boundaries that took place within each month, across all neighborhoods.

(i.e., $K = \eta = 1$), for which check-in activity is dominated by user "u1" and venue "v2." (Note that these are not the same as user "u1" and venue "v2" in Bay Terrace, since these abbreviated user and venue IDs were generated separately for the two neighborhoods in the figure.) The neighborhood-level set of events for Melrose is incompressible using multiple hypergraphs, since it does not have multiple distinct periods of activity, instead exhibiting consistent check-ins by one user at one venue.

To quantify the extent of temporal localization in the hypergraphs inferred with our method, we define a temporal event gap ratio α as the ratio of the median interevent time within clusters to the median interevent time between clusters, or

$$\alpha = \frac{\mathrm{median}\big(\{t_{i+1} - t_i | c_i = c_{i+1}\}_{i=1}^{N-1}\big)}{\mathrm{median}\big(\{t_{i+1} - t_i | c_i \neq c_{i+1}\}_{i=1}^{N-1}\big)}, \tag{19}$$

where $c_i \in \{1, \ldots, K\}$ is the event cluster index of the ith event $\boldsymbol{x}_i$. $\alpha < 1$ when the events within the event clusters tend to be more localized in time than the events on the borders of the clusters, and $\alpha > 1$ when the opposite is true. In Fig. 4(c), we plot a histogram of the ratio α for all neighborhoods analyzed that had an inferred $K > 1$. We can see that the inferred hypergraphs in all but four neighborhoods had events that were more temporally localized than the pairs of events that transitioned between hypergraphs ($\alpha < 1$), indicating that our method is identifying periods of temporally localized activity in the LBSN data.

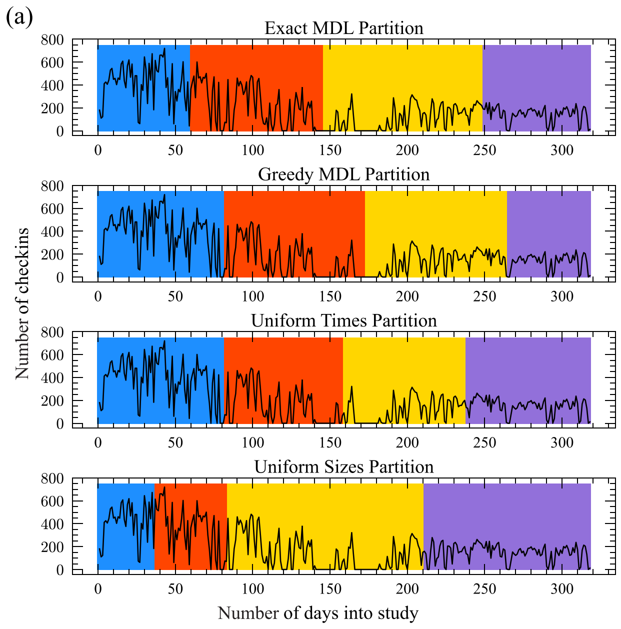
To examine the localization of inferred hypergraphs on sources and destinations relative to the overall localization in the data set $\mathcal{X}$, we compute the expected reduction in uncertainty for predicting the source and destination (s, d) of a randomly chosen edge in $\mathcal{G}$ versus the fully aggregated hypergraph $\boldsymbol{G}_0 = \bigcup_{k=1}^{K} \boldsymbol{G}^{(k)}$. If the edges (s, d) in each hypergraph $\boldsymbol{G}^{(k)}$ are much more highly localized on source-destination pairs than in the overall data $\mathcal{X}$, then it is substantially easier to predict the label (s, d) of a randomly chosen edge in $\mathcal{G}$ than in $\boldsymbol{G}_0$. The reduction in our predictive uncertainty in going from $\boldsymbol{G}_0$ to $\mathcal{G}$ can be quantified by the generalized Jensen-Shannon divergence [60,61], given by

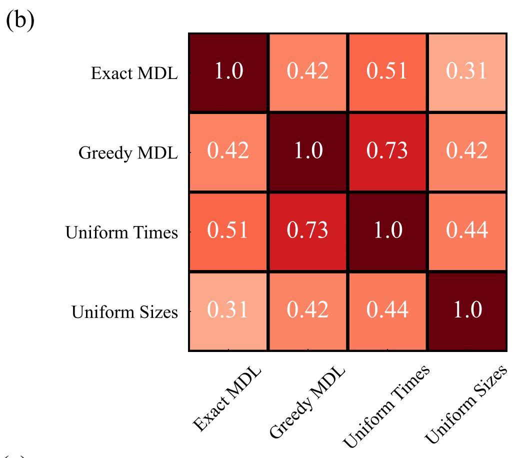
$$\mathrm{JSD}_{\mathrm{Edges,\ unnormalized}} = H(\boldsymbol{G}_0) - \sum_{k=1}^{K} \frac{m_k}{N} H(\boldsymbol{G}^{(k)}), \tag{20}$$

where

$$H(\boldsymbol{G}_0) = -\sum_{s,d=1}^{S,D} \left[\frac{\sum_{k=1}^{K} G^{(k)}_{sd}}{N}\right] \log\left[\frac{\sum_{k=1}^{K} G^{(k)}_{sd}}{N}\right], \tag{21}$$

$$H[\boldsymbol{G}^{(k)}] = -\sum_{s,d=1}^{S,D} \left[\frac{G^{(k)}_{sd}}{m_k}\right] \log\left[\frac{G^{(k)}_{sd}}{m_k}\right], \tag{22}$$





(c)

Partitioning method	Number of events (by cluster)	$\mathrm{JSD}_{\mathrm{Edges}}$	α
Exact MDL	[25230, 18213, 11000, 9923]	0.0648	0.0012
Greedy MDL	[31971, 12838, 12367, 7190]	0.0627	0.0003
Uniform Times	[31971, 11926, 8521, 11948]	0.0634	0.0025
Uniform Sizes	[16092, 16092, 16091, 16091]	0.0661	4.2692

FIG. 5. FourSquare check-ins across all of NYC. (a) Binnings obtained when applying the exact dynamic programming method (top plot) and greedy agglomerative method (second plot) of Sec. II C to the set of check-ins aggregated across all neighborhoods in NYC, with the number of check-ins for each day of the study plotted as a solid black line underneath. The $K = 4$ temporal bins inferred by each of these algorithms are highlighted. The bottom two plots show the partitions obtained by naïvely partitioning the events into $K = 4$ time windows of equal duration and into time windows with an equal number of events (third and fourth plots, respectively). (b) CCAMI matrix among all pairs of the four partitions shown in panel (a). (c) Table of summary statistics for the partitions in panel (a).

are the Shannon entropies of the edges in the aggregated graph and inferred hypergraphs, respectively. One can show that Eq. (20) is bounded below by 0 (due to the concavity of entropy) and above by $H(\boldsymbol{G}_0)$, so we can rescale the JSD to [0,1] by dividing by this upper bound, thus

$$\mathrm{JSD}_{\mathrm{Edges}} = 1 - \frac{1}{H(\boldsymbol{G}_0)} \sum_{k=1}^{K} \frac{m_k}{N} H(\boldsymbol{G}^{(k)}). \tag{23}$$

Equation (23) tells us the fraction of information (in terms of predictive power for the edge labels) we lose by using the aggregated hypergraph $\boldsymbol{G}_0$ instead of the cluster-level hypergraphs $\mathcal{G} = \{\boldsymbol{G}^{(k)}\}_{k=1}^{K}$. A value of $\mathrm{JSD}_{\mathrm{Edges}} \approx 0$ indicates little edge localization within the clusters relative to the overall data set, and a value of $\mathrm{JSD}_{\mathrm{Edges}} \approx 1$ indicates high edge localization within the clusters relative to the overall data set.

In Fig. 4(d), we plot a histogram of $\mathrm{JSD}_{\mathrm{Edges}}$ for all neighborhoods analyzed that had an inferred $K > 1$, with the distribution mean of $\mathrm{JSD}_{\mathrm{Edges}} \approx 0.15$ indicated with the vertical black line. The mean JSD value of 0.15 indicates a relatively high average level of localization among the edges (s, d) within each hypergraph. We can also observe that all but 5 neighborhoods had an information gain of at least 5% relative to the overall data $\mathcal{X}$, indicating non-negligible localization of the sources or destinations in the hypergraphs inferred with our method.

Finally, in Fig. 4(e) we plot a histogram showing the fraction of all events that took place within each month (blue) and the fraction of inferred temporal snapshot boundaries that took place within each month (red). Here we can observe substantial differences in these distributions, indicating that the inferred boundaries are to some extent negatively correlated with the temporal density of events. For example, there is a sizable drop in event frequency from May to June, and we see a spike in the number of inferred temporal boundaries in June, suggesting that the drop in events provided sufficient statistical evidence for the formation of a new cluster boundary in the information theoretically optimal binnings. We also see large discrepancies for September and October, where there are comparatively many boundaries but few events. This may be correlated with the uptick in the overall density of events, seasonal fluctuations in consumer behavior, and Hurricane Sandy (for October).

In the Appendix, we run additional tests using the neighborhood-level event data, in order to understand the discrepancies between the exact dynamic programming algorithm and the fast greedy agglomerative algorithm of Sec. II C when applied to this data set.

One can also examine the fluctuations in large-scale check-in patterns at the level of the entire city by aggregating the events over all neighborhoods. In the top two panels of Fig. 5(a), we show the binnings obtained when applying our exact dynamic programming method and greedy agglomerative method to the aggregated data set representing the top 1000 users and venues in the FourSquare data set across all neighborhoods. Different colors distinguish the $K = 4$ different temporal bins obtained by each of these algorithms, and the number of check-ins for each day of the study is plotted as a solid black line.

In the bottom two plots of Fig. 5(a), we show the partitions obtained using naïve binning heuristics with the same number of clusters ($K = 4$)—a partition of the events into time windows of equal duration (third plot), and a partition of the events into time windows with an equal number of events (fourth plot). We can see substantial shifts in the cluster boundaries for these methods relative to the MDL-based methods, despite the (relatively strong) constraint of fixing the value of K. The MDL-based methods have inferred boundaries in the gaps with sparse event data around days 145 (exact method) and 162 (greedy method), due to their focus on temporally localized clusters of events, while the other methods have boundaries that are uncorrelated with temporal event density. However, a binning heuristic that only looks for temporal gaps in events will also fail to reproduce the partitions obtained by the MDL methods, since we see some sizable gaps included within the inferred clusterings of both methods, where there is enough statistical evidence for the algorithm to create a contiguous cluster due to localization of the events on sources and destinations.

In Fig. 5(b), we plot a matrix of the CCAMI values between each pair of partitions among the four shown in Fig. 5(a). We observe that, despite the apparent visual similarity of some pairs of partitions, the information shared between many pairs is only modestly more than what one would expect in random partitions of the time interval into $K = 4$ clusters. We also can see that—despite having a description length ($\eta_{\text{greedy}} = 0.758$) which is comparable to the description length of the optimal partition obtained by the exact dynamic programming algorithm ($\eta_{\text{exact}} = 0.750$)—the greedy partition is actually quite different than the true MDL-optimal partition when considering the strong constraints imposed by contiguity (CCAMI $= 0.42$). In fact, the greedy MDL partition is less similar to the true MDL-optimal partition than the partition obtained by simply splitting the interval into $K = 4$ windows of equal duration. This highlights the importance of our exact dynamic programming solution, and is consistent with findings in network community detection that identify high levels of degeneracy in the near-optimal partitions of networks [27,50,62,63].

We can also compute the inverse compression ratio of Eq. (16) for the baseline partitions by plugging these partitions directly into the objective in Eq. (12). We find inverse compression ratios of 0.764 and 0.778, respectively, for the partitions whose clusters are uniform in time and the number of events, respectively, which correspond to states that are roughly 2^{600} and 2^{1800} times worse than the greedy solution (which is 2^{700} times worse than the exact optimum) in terms of relative posterior probability.

In Fig. 5(c), we plot summary statistics of the inferred hypergraphs using each binning method. Mirroring Fig. 5(a) we can see high variability in the number of events within the clusters across the four partitions. We can also see that the exact MDL approach has the best balance of edge localization ($\text{JSD}_{\text{Edges}} = 0.0648$) and temporal localization ($\alpha = 0.0012$) among its inferred event clusters. While it is only the second best method regarding each metric individually—e.g., it has the second-highest $\text{JSD}_{\text{Edges}}$ value and the second lowest α value—the top performers in $\text{JSD}_{\text{Edges}}$ (Uniform Sizes) and α (Greedy MDL) are the worst performers regarding α and $\text{JSD}_{\text{Edges}}$, respectively.

IV. CONCLUSION

In this paper we develop a nonparametric approach for inferring representative hypergraph snapshots from temporal event data based on the MDL principle. Our approach considers the problem of transmitting the data to a receiver in multiple stages of increasing granularity, with the hypergraph snapshots as an intermediate step. The configuration of hypergraphs that minimizes the description length of this transmission process is then selected as the MDL-optimal hypergraph representation of the data. Our method automatically performs model selection for the number and composition of the hypergraphs with no parameter tuning. We employ an exact dynamic programming algorithm to identify the hypergraphs that minimize our description length objective in a runtime that scales quadratically with the number of discrete time steps in the limit of high temporal resolution when the number of nodes and events is comparatively small. However, the exact approach may fail to scale to larger data sets where this condition is not met, so we also develop a fast greedy agglomerative algorithm that achieves near-optimal configurations with substantially reduced run times in the examples studied. We demonstrate that our methods are able to consistently reconstruct synthetic data with planted hypergraph structure even with appreciable noise, and can reveal meaningful representative structures in real location-based social network data to understand human mobility patterns.

There are a number of ways our methods can be extended in future work. In this paper we explore a data encoding that exploits redundancy provided by degree heterogeneity in the incidence representations of the representative hypergraphs within the data, but one can in principle exploit other structure as well to develop efficient encodings. In a Bayesian framing of the hypergraph inference problem, these alternative encodings would correspond to generative models other than the configuration-style model corresponding to the encoding of this paper. Alternative structure that could be potentially exploited for improved compression may include community structure, transitivity, or overlaps among hyperedges. One can also impose asymmetry in the encoding between the source and destination nodes to more clearly highlight the desired hypergraph structure, or incorporate other relevant metadata on the edges such as weights or temporal duration of events. Finally, there is no guarantee that the greedy method for minimizing the description length will be near-optimal in all problem settings, so a more comprehensive evaluation of this method in other applications or a mathematical proof of its approximation capabilities is important for future work.

Code for the algorithms presented in this paper is available [64].

ACKNOWLEDGMENTS

This research was supported in part by the HKU-100 Start Up Grant and the HKU Institute of Data Science Research Seed Fund. The author thanks S. Feng for helpful discussions.

APPENDIX : ADDITIONAL TESTS WITH NYC CHECK-INS DATASET

In this Appendix we plot the results of a variety of tests to compare the performance of the exact dynamic programming

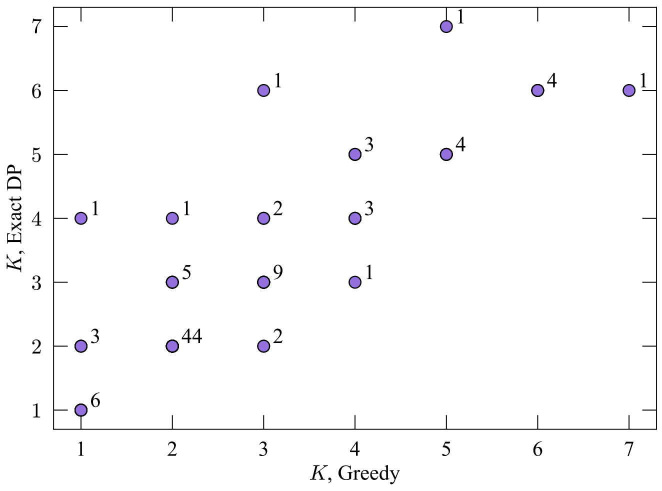


FIG. 6. Number of clusters inferred by the exact dynamic programming algorithm (y axis) and the greedy algorithm (x axis). Data point labels indicate the number of neighborhoods with the given combination ($K_{\text{greedy}}, K_{\text{exact}}$).

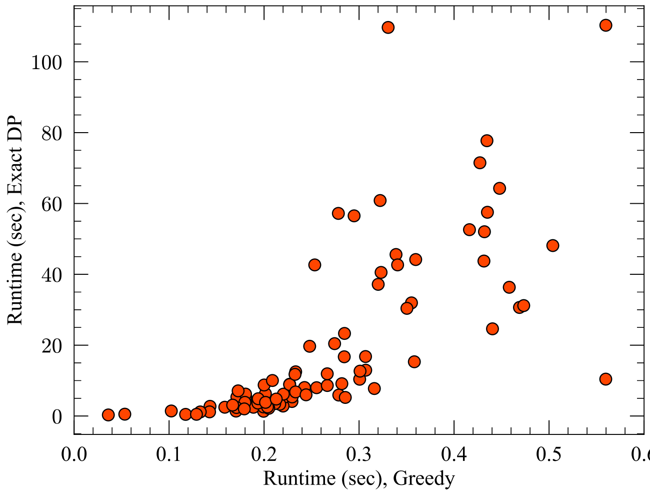


FIG. 7. Run time of the exact dynamic programming algorithm (y axis) and the greedy algorithm (x axis) for each neighborhood in the study.

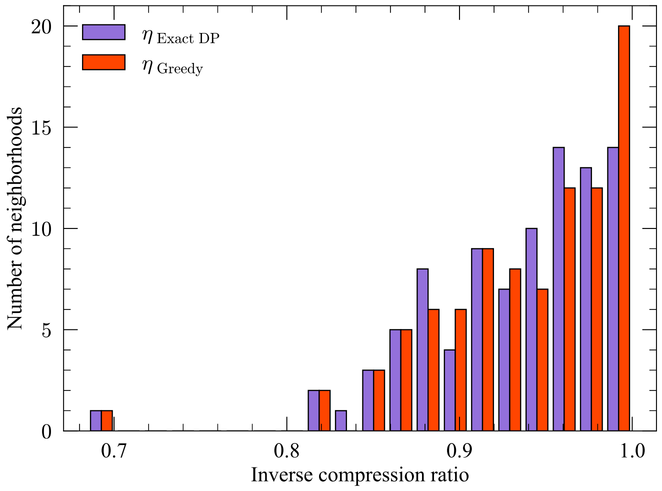


FIG. 8. Histograms of the compression ratio η [Eq. (16)] for the event data in each neighborhood in the study, after compression with the exact method and greedy method.

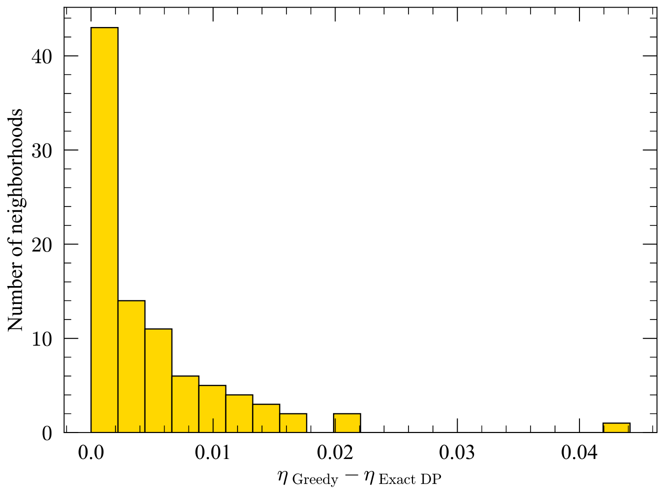


FIG. 9. Histograms of the differences in the compression ratio η [Eq. (16)] between the exact method and greedy method, for each neighborhood in the study.

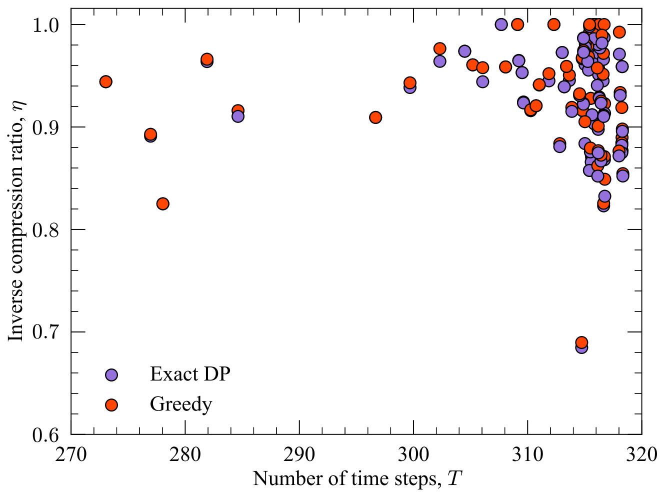


FIG. 10. Inverse compression ratio η versus number of time steps T for each neighborhood in the study.

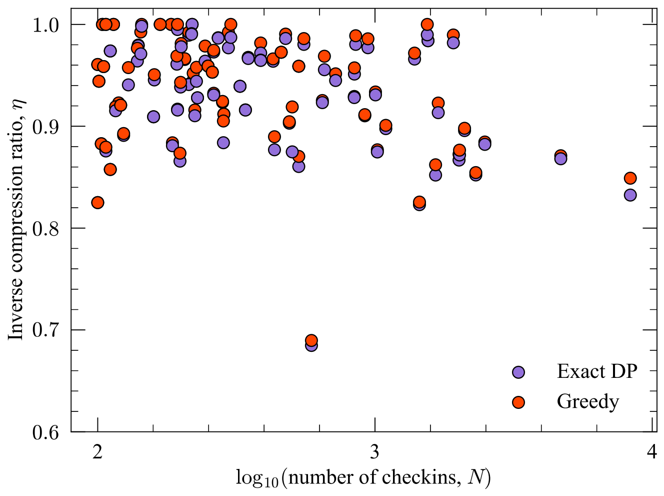


FIG. 11. Inverse compression ratio η versus the logarithm of the number of check-ins N for each neighborhood in the study.

TABLE I. Neighborhood-level check-in data-set size details (0–45).

No.	Neighborhood	N	S	D
0	Arverne	103	1	2
1	Astoria	852	65	16
2	Battery Park City	184	38	2
3	Bay Terrace	590	145	9
4	Bedford-Stuyvesant	1553	31	21
5	Bensonhurst	186	11	3
6	Boerum Hill	159	7	3
7	Borough Park	213	14	4
8	Brighton Beach	429	63	4
9	Bronxdale	100	9	2
10	Brooklyn Heights	206	9	3
11	Brownsville	434	14	7
12	Bull’s Head	229	6	6
13	Canarsie	491	13	9
14	Central Park	531	189	8
15	Chelsea	4678	654	60
16	Civic Center	219	26	3
17	Co-op City	124	12	2
18	College Point	459	28	9
19	Concourse	341	169	2
20	Concourse Village	194	24	4
21	Coney Island	114	5	2
22	Country Club	140	2	2
23	Crown Heights	101	8	3
24	Cypress Hills	282	25	3
25	DUMBO	119	21	3
26	Ditmars Steinway	296	7	4
27	Douglaston	193	5	3
28	Downtown Brooklyn	387	47	8
29	East Elmhurst	198	6	5
30	East Flatbush	224	40	1
31	East Harlem	843	62	14
32	East New York	1389	6	19
33	East Village	1691	398	30
34	Elmhurst	218	48	4
35	Financial District	1447	218	23
36	Flatbush	349	21	7
37	Flatiron District	2310	298	27
38	Flushing	721	40	14
39	Flushing Meadows Corona Park	221	90	3
40	Fordham	139	15	3
41	Forest Hills	213	22	4
42	Fort Greene	227	83	4
43	Gowanus	194	28	3
44	Gramercy	1094	266	12
45	Gravesend	285	12	5

TABLE II. Neighborhood-level check-in data-set size details (46–90).

No.	Neighborhood	N	S	D
46	Greenpoint	111	29	2
47	Greenwich Village	842	252	16
48	Harlem	1544	76	27
49	Hell’s Kitchen	2490	411	37
50	Inwood	262	13	5
51	Jamaica	657	84	7
52	Jamaica Estates	105	9	2
53	John F. Kennedy International Airport	919	315	4
54	Kips Bay	503	163	9
55	LaGuardia Airport	646	259	3
56	Long Island City	943	68	16
57	Longwood	284	9	8
58	Lower East Side	554	107	11
59	Melrose	168	6	2
60	Middle Village	104	1	1
61	Midtown	8338	687	126
62	Midwood	107	5	2
63	Mill Basin	107	21	3
64	Morningside Heights	116	12	2
65	Murray Hill	476	18	11
66	NoHo	244	39	5
67	Norwood	200	13	3
68	Pelham Bay	100	9	2
69	Prospect Heights	129	73	2
70	Prospect Park	272	71	3
71	Queens Village	111	2	2
72	Richmond Hill	121	8	2
73	Roosevelt Island	160	39	2
74	Sheepshead Bay	531	57	10
75	SoHo	1654	181	29
76	South Ozone Park	387	3	6
77	South Slope	143	8	3
78	St. George	199	37	2
79	Stapleton	144	3	2
80	Sunset Park	326	16	8
81	Theater District	2104	415	33
82	Tribeca	302	27	7
83	Unionport	250	12	4
84	Upper East Side	2010	264	37
85	Upper West Side	2018	283	35
86	Washington Heights	1004	28	16
87	West Village	1020	199	22
88	Whitestone	259	5	6
89	Williamsburg	1917	139	31
90	Woodside	262	49	5

algorithm and greedy algorithm of Sec. II C when applied to the FourSquare check-ins data set of Sec. III B.

(1) Fig. 6 plots the number of clusters inferred by the exact dynamic programming algorithm (y-axis) and the greedy algorithm (x-axis). Data point labels indicate the number of neighborhoods with the given combination ($K_{\text{greedy}}, K_{\text{exact}}$).

(2) Fig. 7 plots the run time of the exact dynamic programming algorithm (y-axis) and the greedy algorithm (x-axis) for each neighborhood in the study.

(3) Fig 8 plots histograms of the compression ratio η [Eq. (16)] for the event data in each neighborhood in the study, after compression with the exact method and greedy method.

(4) Fig. 9 plots histograms of the differences in the compression ratio η [Eq. (16)] between the exact method and greedy method, for each neighborhood in the study.

(5) Fig. 10 plots the inverse compression ratio η versus the number of time steps T for each neighborhood in the study.

(6) Fig. 11 plots the inverse compression ratio η versus the logarithm of the number of checkins N for each neighborhood in the study.

We also report summary statistics of the 91 neighborhoods studied in Tables I and II.

[1] F. Battiston, G. Cencetti, I. Iacopini, V. Latora, M. Lucas, A. Patania, J.-G. Young, and G. Petri, Networks beyond pairwise interactions: Structure and dynamics, Phys. Rep. **874**, 1 (2020).

[2] F. Battiston, E. Amico, A. Barrat, G. Bianconi, G. Ferraz de Arruda, B. Franceschiello, I. Iacopini, S. Kéfi, V. Latora, Y. Moreno *et al.*, The physics of higher-order interactions in complex systems, Nat. Phys. **17**, 1093 (2021).

[3] Z. Li, C. Huang, L. Xia, Y. Xu, and J. Pei, Spatial-temporal hypergraph self-supervised learning for crime prediction, in *Proceedings of the IEEE 38th International Conference on Data Engineering (ICDE '22)* (IEEE, Los Alamitos, CA, 2022), pp. 2984–2996.

[4] N. Proferes, N. Jones, S. Gilbert, C. Fiesler, and M. Zimmer, Studying reddit: A systematic overview of disciplines, approaches, methods, and ethics, Soc. Media Soc. **7**, 20563051211019004 (2021).

[5] A. Antelmi, G. Cordasco, C. Spagnuolo, and V. Scarano, A design-methodology for epidemic dynamics via time-varying hypergraphs, in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems* (ACM, New York, 2020), pp. 61–69.

[6] Y. Kim and R. Krishnan, On product-level uncertainty and online purchase behavior: An empirical analysis, Manag. Sci. **61**, 2449 (2015).

[7] M. E. Eren, J. S. Moore, and B. S. Alexandro, Multi-dimensional anomalous entity detection via Poisson tensor factorization, in *Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI '20)* (IEEE, Los Alamitos, CA, 2020), pp. 1–6.

[8] C. Huang, D. Wang, and J. Tao, An unsupervised approach to inferring the localness of people using incomplete geotemporal online check-in data, ACM Trans. Intell. Syst. Technol. **8**, 1 (2017).

[9] B. Balmaki, M. A. Rostami, T. Christensen, E. A. Leger, J. M. Allen, C. R. Feldman, M. L. Forister, and L. A. Dyer, Modern approaches for leveraging biodiversity collections to understand change in plant-insect interactions, Front. Ecol. Evol. **10**, 924941 (2022).

[10] P. Holme and J. Saramäki, Temporal networks, Phys. Rep. **519**, 97 (2012).

[11] G. Cencetti, F. Battiston, B. Lepri, and M. Karsai, Temporal properties of higher-order interactions in social networks, Sci. Rep. **11**, 7028 (2021).

[12] A. Myers, C. Joslyn, B. Kay, E. Purvine, G. Roek, and M. Shapiro, Topological analysis of temporal hypergraphs, in *Proceedings of the 18th International Workshop on Algorithms and Models for the Web Graph (WAW '23)* (Springer, Berlin, 2023), pp. 127–146.

[13] G. Lee and K. Shin, Temporal hypergraph motifs, Knowl. Inf. Syst. **65**, 1549 (2023).

[14] D. Taylor, M. A. Porter, and P. J. Mucha, Supracentrality analysis of temporal networks with directed interlayer coupling, in *Temporal Network Theory* (Springer, Cham, 2019), pp. 325–344.

[15] I. Amburg, N. Veldt, and A. Benson, Clustering in graphs and hypergraphs with categorical edge labels, in *Proceedings of the Web Conference* (2020), pp. 706–717.

[16] S. Huang, Z. Bao, G. Li, Y. Zhou, and J. S. Culpepper, Temporal network representation learning via historical neighborhoods aggregation, in *Proceedings of the IEEE 36th International Conference on Data Engineering (ICDE)* (IEEE, Los Alamitos, CA, 2020), pp. 1117–1128.

[17] L. Neuhäuser, R. Lambiotte, and M. T. Schaub, Consensus dynamics on temporal hypergraphs, Phys. Rev. E **104**, 064305 (2021).

[18] A. Li, S. P. Cornelius, Y.-Y. Liu, L. Wang, and A.-L. Barabási, The fundamental advantages of temporal networks, Science **358**, 1042 (2017).

[19] E. Valdano, L. Ferreri, C. Poletto, and V. Colizza, Analytical computation of the epidemic threshold on temporal networks, Phys. Rev. X **5**, 021005 (2015).

[20] B. Schwarz, D. P. Vázquez, P. J. CaraDonna, T. M. Knight, G. Benadi, C. F. Dormann, B. Gauzens, E. Motivans, J. Resasco, N. Blüthgen *et al.*, Temporal scale-dependence of plant–pollinator networks, Oikos **129**, 1289 (2020).

[21] V. Nicosia, J. Tang, C. Mascolo, M. Musolesi, G. Russo, and V. Latora, Graph metrics for temporal networks, in *Temporal Networks* (Springer, Berlin, Heidelberg, 2013), pp. 15–40.

[22] Y. Zha, T. Zhou, and C. Zhou, Unfolding large-scale online collaborative human dynamics, Proc. Natl. Acad. Sci. USA **113**, 14627 (2016).

[23] J. Rissanen, Modeling by the shortest data description, Automatica **14**, 465 (1978).

[24] P. D. Grünwald and A. Grünwald, *The Minimum Description Length Principle* (MIT Press, Cambridge, MA, 2007).

[25] M. Rosvall and C. T. Bergstrom, An information-theoretic framework for resolving community structure in complex networks, Proc. Natl. Acad. Sci. USA **104**, 7327 (2007).

[26] T. P. Peixoto, Hierarchical block structures and high-resolution model selection in large networks, Phys. Rev. X **4**, 011047 (2014).

[27] A. Kirkley and M. E. J. Newman, Representative community divisions of networks, Commun. Phys. **5**, 40 (2022).

[28] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos, Vog: Summarizing and understanding large graphs, in *Proceedings of the SIAM international conference on data mining*, (SIAM, Philadelphia, PA, 2014), pp. 91–99.

[29] A. E. Wegner, Subgraph covers: An information-theoretic approach to motif analysis in networks, Phys. Rev. X **4**, 041026 (2014).

[30] P. Bloem and S. de Rooij, Large-scale network motif analysis using compression, Data Min. Knowl. Discov. **34**, 1421 (2020).

[31] J.-G. Young, G. Petri, and T. P. Peixoto, Hypergraph reconstruction from network data, Commun. Phys. **4**, 135 (2021).

[32] G. Bouritsas, A. Loukas, N. Karalias, and M. Bronstein, Partition and code: learning how to compress graphs, Proc. Adv. Neur. Inf. Process. Syst. **34**, 18603 (2021).

[33] J. Feng, X. He, B. Konte, C. Böhm, and C. Plant, Summarization-based mining bipartite graphs, in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Association for Computing Machinery, New York, NY, 2012), pp. 1249–1257.

[34] H. Zhou, S. Liu, K. Lee, K. Shin, H. Shen, and X. Cheng, Dpgs: Degree-preserving graph summarization, in *Proceedings of the SIAM International Conference on Data Mining (SDM '21)*, (SIAM, Philadelphia, PA, 2021), pp. 280–288.

[35] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos, Summarizing and understanding large graphs, Stat. Anal. Data Min. **8**, 183 (2015).

[36] N. Masuda and P. Holme, Detecting sequences of system states in temporal networks, Sci. Rep. **9**, 1 (2019).

[37] M. De Domenico, V. Nicosia, A. Arenas, and V. Latora, Structural reducibility of multilayer networks, Nat. Commun. **6**, 6864 (2015).

[38] A. Kirkley, A. Rojas, M. Rosvall, and J.-G. Young, Compressing network populations with modal networks reveals structural diversity, Commun. Phys. **6**, 148 (2022).

[39] S. Gurukar, B. Boettner, C. Browning, C. Calder, and S. Parthasarathy, Leveraging network representation learning and community detection for analyzing the activity profiles of adolescents, Appl. Netw. Sci. **7**, 27 (2022).

[40] W. Yu, Spatial co-location pattern mining for location-based services in road networks, Expert Syst. Appl. **46**, 324 (2016).

[41] X. Zheng, Y. Luo, L. Sun, X. Ding, and J. Zhang, A novel social network hybrid recommender system based on hypergraph topologic structure, World Wide Web **21**, 985 (2018).

[42] E. Burgos, H. Ceva, L. Hernández, R. P. Perazzo, M. Devoto, and D. Medan, Two classes of bipartite networks: Nested biological and social systems, Phys. Rev. E **78**, 046113 (2008).

[43] G. Lee, M. Choe, and K. Shin, How do hyperedges overlap in real-world hypergraphs? Patterns, measures, and generators, in *Proceedings of the Web Conference* (2021), pp. 3396–3407.

[44] Z.-K. Zhang and C. Liu, A hypergraph model of social tagging networks, J. Stat. Mech: Theory Exp. (2010) P10005.

[45] W. Feller, *An Introduction to Probability Theory and Its Applications* (Wiley, Hoboken, NJ, 1950).

[46] M. Jerdee, A. Kirkley, and M. Newman, Improved estimates for the number of non-negative integer matrices with given row and column sums, Proc. R. Soc. A. **480**, 20230470 (2024).

[47] B. Jackson, J. D. Scargle, D. Barnes, S. Arabhi, A. Alt, P. Gioumousis, E. Gwin, P. Sangtrakulcharoen, L. Tan, and T. T. Tsai, An algorithm for optimal partitioning of data on an interval, IEEE Sign. Process Lett. **12**, 105 (2005).

[48] R. Bellman, *Dynamic Programming* (Princeton University Press, Princeton, NJ, 1957).

[49] A. Patania, A. Allard, and J.-G. Young, Exact and rapid linear clustering of networks with dynamic programming, Proc. R. Soc. A. **479**, 20230159 (2023).

[50] S. Aref, M. Mostajabdaveh, and H. Chheda, Heuristic modularity maximization algorithms for community detection rarely return an optimal partition or anything similar, in *Lecture Notes in Computer Science*, edited by J. Mikyka, C. de Mulatier, M. Paszynski, V. V. Krzhizhanovskaya, J. J. Dongarra, P. M. Sloot (Springer, Cham, 2023), Vol. 10476, pp. 612–626.

[51] W. Patefield, Algorithm as 159: An efficient method of generating random r $\times$ c tables with given row and column totals, J. R. Stat. Soc. C **30**, 91 (1981).

[52] N. X. Vinh, J. Epps, and J. Bailey, Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance, J. Mach. Learn. Res. **11**, 2837 (2010).

[53] A. Kirkley, Spatial regionalization based on optimal information compression, Commun. Phys. **5**, 249 (2022).

[54] F. Ricci-Tersenghi, G. Semerjian, and L. Zdeborová, Typology of phase transitions in bayesian inference problems, Phys. Rev. E **99**, 042109 (2019).

[55] D. Yang, D. Zhang, V. W. Zheng, and Z. Yu, Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNS, IEEE Trans. Syst. Man Cybernet.: Syst. **45**, 129 (2014).

[56] Foursquare—NYC and Tokyo check-ins, https://www.kaggle.com/datasets/chetanism/foursquare-nyc-and-tokyo-checkin-dataset?resource=download.

[57] Z. Chen, S. Kelty, A. G. Evsukoff, B. F. Welles, J. Bagrow, R. Menezes, and G. Ghoshal, Contrasting social and non-social sources of predictability in human mobility, Nat. Commun. **13**, 1922 (2022).

[58] C. Kadar, J. Iria, and I. P. Cvijikj, Exploring foursquare-derived features for crime prediction in new york city, KDD-Urban Computing WS **16**, 10 (2016).

[59] Pediacities-NYC-neighborhoods, https://data.beta.nyc/dataset/pediacities-nyc-neighborhoods/resource/35dd04fb-81b3-479b-a074-a27a37888ce7.

[60] F. Nielsen, On a generalization of the Jensen–Shannon divergence and the Jensen–Shannon centroid, Entropy **22**, 221 (2020).

[61] A. Kirkley, Information theoretic network approach to socioeconomic correlations, Phys. Rev. Res. **2**, 043212 (2020).

[62] B. H. Good, Y.-A. de Montjoye, and A. Clauset, Performance of modularity maximization in practical contexts, Phys. Rev. E **81**, 046106 (2010).

[63] T. P. Peixoto, Revealing consensus and dissensus between network partitions, Phys. Rev. X **11**, 021003 (2021).

[64] https://github.com/aleckirkley/hypergraph-binning.