Scalable Hashing-Based Network Discovery

Tara Safavi
Computer Science and Engineering
University of Michigan
Ann Arbor, MI

Chandra Sripada
Psychiatry and Philosophy
University of Michigan
Ann Arbor, MI

Danai Koutra
Computer Science and Engineering
University of Michigan
Ann Arbor, MI

***Abstract*—Discovering and analyzing networks from non-network data is a task with applications in fields as diverse as neuroscience, genomics, energy, economics, and more. In these domains, networks are often constructed out of multiple time series by computing measures of association or similarity between pairs of series. The nodes in a discovered graph correspond to time series, which are linked via edges weighted by the association scores of their endpoints. After graph construction, the network may be thresholded such that only the edges with stronger weights remain and the desired sparsity level is achieved.**

While this approach is feasible for small datasets, its quadratic time complexity does not scale as the individual time series length and the number of compared series increase. Thus, to avoid the costly step of building a fully-connected graph before sparsification, we propose a fast network discovery approach based on probabilistic hashing of randomly selected time series subsequences. Evaluation on real data shows that our methods construct graphs nearly 15 times as fast as baseline methods, while achieving both network structure and accuracy comparable to baselines in task-based evaluation.

I. Introduction

Prevalent among data in the natural, social, and information sciences are graphs or networks, which are abstract data structures consisting of entities (nodes) and connections among those entities (edges). In some cases, graphs are directly observed, as in the well-studied example of social networks, where nodes represent users and edges represent a variety of user interactions such as friendship, likes, or comments. However, graphs may also be constructed from non-network data, a task of interest in domains such as neuroscience [7][11], finance [24], and transportation [29], where practitioners seek to represent similarity or correlation among pairs of time series as network interactions in order to gain network-related insights from the resultant graphs.

Motivated by the growing need for scalable data analysis, we address the problem of efficient network discovery on *many* time series, which may be informally described as:

Problem 1 (Efficient network discovery: informal). *Given N time series $\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\}$, efficiently construct a sparse similarity graph which captures the strong associations (edges) between the time series (nodes).*

Traditional network discovery [7][21] on time series suffers from the simple but serious drawback of scalability. The established technique for building a graph out of N time series is to compare all pairs of series, forming a fully-connected graph where nodes are time series and edges are weighted proportionally to the computed similarity or association of the nodes they connect [13], with optional sparsification afterward to keep only the stronger associations. This all-pairs method is at *least* an $\Omega(N^2)$ operation depending on the complexity of the similarity measure, which makes the process computationally inefficient and wasteful on anything other than small datasets. For example, to generate even a small graph of 5000 nodes, about 12.5 million comparisons are required for a graph that may eventually lose most of its edges via thresholding before further analysis. Moreover, each comparison itself is at least linear in the time series length. For example, correlation is linear, and the dynamic time warping (DTW) distance measures are slower yet, adding an extra runtime factor as the series length increases.

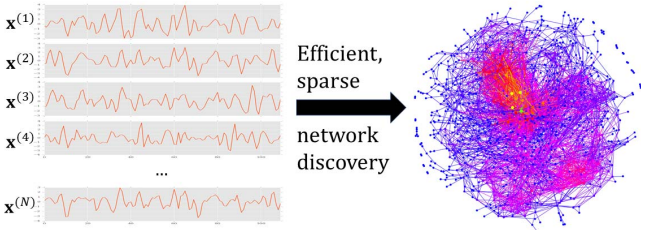


Fig. 1: Network discovery: a key to efficiency is to avoid the all-pairs similarity computations.

We propose to circumvent the bottleneck of the established network discovery approach, namely the all-pairs comparison, by introducing a new method based on locality-sensitive hashing (LSH) tailored to time series. Informally, we first compute a compact randomized signature for each time series, then hash all series with the same signature to the same "bucket" such that only the intra-bucket pairwise similarity scores need be computed. Our main contributions are as follows:

- *Novel distance measure*. Constrained by the theoretical requirements of LSH and motivated by the widespread use of correlation as an association measure, we propose a novel and intuitive time series distance metric, ABC, that captures *consecutively* matching approximate trends between time series. To the best of our knowledge, this is the first distance metric to do so.
- *Network discovery via hashing*. We introduce a new family of LSH hash functions, window sampling LSH, based on our proposed distance metric. We show how the false positive and negative rates of the randomized hashing

2374-8486/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDM.2017.50

process can be controlled in network construction.

- *Evaluation on real data.* We evaluate the efficiency and accuracy of our method. To evaluate accuracy, we rely on domain knowledge from neuroscience, an area of active research on discovered networks. The graphs built by our proposed method, ABC-LSH, are created up to 15× faster than baselines, while performing just as well in classification-based evaluation.

For reproducibility, we make the code available at https://github.com/tsafavi/hashing-based-network-discovery.

II. Related Work

Recently there has been significant interest in inferring sparse graphical models from multivariate data using lasso regularization (inverse covariance estimation) [15][30]. These approaches assume that the data follow a k-variate normal distribution $N(0, \Sigma)$, where k is the number of the parameters and Σ is the covariance of the distribution. Recent work focuses on scaling this approach, albeit with the same modeling assumptions. In our work, we tackle efficient network discovery without distributional assumptions.

In graph signal processing, "graph signals" are described as functions on the nodes in the graph, where high-dimensional data represent nodes and weighted edges connect nodes according to similarity [29]. While graph signal processing involves transforming time series signals to construct a weighted similarity graph, its focus is more on extending traditional signal processing tools to graphs—the filtering and transforming of the signals themselves—rather than the efficiency or evaluation of network discovery.

The problem of fast k-nearest neighbor graph construction has also been addressed. In a k-NN graph, each node is connected to the top k most similar other nodes in the graph. Some methods proposed to improve the quadratic runtime of traditional k-NN graph construction include local search algorithms and hashing [12] [31]. However, limiting the number of neighbors per node is an unintuitive task. Our method does not require a predetermined number of nearest neighbors, but rather lets the hashing process determine which pairs of nodes are connected. Relatedly, the set similarity self-join problem [8] seeks to identify all pairs of objects above a user-set similarity threshold. We argue that thresholding is neither intuitive nor informative in network discovery: we seek to analyze the resultant network's connectivity patterns and strengths without hard-to-define edge weight thresholds.

Locality-sensitive hashing (LSH) [3] has been successfully employed in various settings, including efficiently finding similar documents. Unlike general hashing, which aims to avoid collisions between data points, LSH *encourages* collisions between items such that, with high probability, only colliding elements are similar. In our domain, general methods of hashing time series have been proposed, although neither for the purpose of graph construction nor approximation of correlation between two time series [17] [20] [19]. Most recently, random projections of sliding windows on time series have been proposed for constructing approximate short signatures for hashing [23]. However, this approach uses dynamic time warping (DTW) as a measure of time series distance. While DTW has the advantage of potentially matching similarly-shaped time series out of phase in the time axis, it is not a *metric*, unlike our proposed measure, and therefore cannot be used for theoretical guarantees of false positives and negatives in hashing [26].

In summary, while fields tangential to our problem have been explored, some to a greater degree than others, to the best of our knowledge efficient network discovery on time series with hashing has not been explored.

TABLE I: Major symbols.

Symbol	Definition
$\mathbf{x}$	a time series, or a real-valued length-n sequence
$\mathbf{X}$	a set of N time series $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$
$\mathbf{b}(\mathbf{x})$	binary approximation of a time series $\mathbf{x}$
$S(\mathbf{x}, \mathbf{y})$	maximum ABC similarity between two sequences
p	number of agreeing "runs" between two sequences
k_i	length of the i-th agreeing "run" between two sequences
α	The parameter upon which ABC operates; α controls the emphasis on consecutiveness as a factor in similarity scoring
$\mathbf{F}$	LSH family of hash functions
k	length of a window or subsequence of $\mathbf{x}$
r	number of hash functions to AND with LSH
b	number of hash functions to OR with LSH

III. Proposed Approach

The problem we address is given as:

Problem 2 (Multiple time series to weighted graph). *Given N time series $\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\}$, construct a similarity graph where each node corresponds to a time series $\boldsymbol{x}^{(i)}$ and each edge is weighted according to the similarity of the nodes $(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$ that it connects.*

As previously stated, the traditional method of network discovery on time series is quadratic in the number of time series N, and its total complexity also depends on the chosen similarity measure. We thus propose a 3-step method, depicted in Figure 2, that avoids the costly all-pairs comparison step:

1) *Preprocess time series*. First, the input real-valued time series are approximated as binary sequences, following a representation that captures "trends".
2) *Hash binary sequences to buckets*. Next, the binary sequences are hashed to short, randomized signatures. To achieve this, we define a novel distance metric that is both theoretically eligible for LSH and qualitatively comparable to the commonly-used correlation measure.
3) *Compute intra-bucket pairwise similarity*. Each pair of time series that hash to the same signature, or bucket, are compared such that an edge is created between them.

The output of the process is a graph in which all pairs of time series that collide in any round of hashing are connected by an edge weighted according to their similarity. For reference, Table I defines the major symbols we use.

A. Time Series Representation

The first step in our pipeline converts raw, real-valued time series to binary, easily hashable sequences. In the literature,

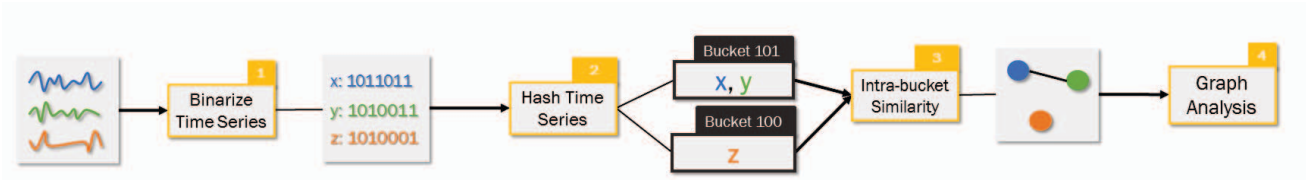


Fig. 2: Our proposal, ABC-LSH. The output of step 3 is a graph where edges are weighted according to node similarity.

several binarized representations of time series have been proposed [18] [4] [27]. The one we use has been called the "clipped" representation of a time series [27].

Definition 1 (Binarized representation of time series). *Given a time series $\boldsymbol{x} \in \mathbb{R}^n$, the binarized representation of the series $\mathbf{b}(\boldsymbol{x})$ replaces each real value x_i, $i \in [1, n]$, by a single bit such that $\mathbf{b}(x_i) = 1$ if x_i is above the series mean $\mu_{\boldsymbol{x}}$, and 0 otherwise.*

We choose this representation because it captures key fluctuations in the time series—which we want to compare, as correlation does—while approximating the time series in a manner that facilitates fast similarity search. In particular, LSH requires a method of computing hash signatures of reduced dimensionality from input data, which is dependent on which similarity or distance measure is chosen. In our proposals, binarizing the time series naturally gives way to the construction of short, representative hash signatures. As we show in the evaluation, these benefits outweigh the "loss" of information from converting to binary: we are able to approximate correlation well with just 1s and 0s.

B. ABC: Approximate Binary Correlation

Given binary sequences that approximate real-valued time series, we propose an intuitive and simple measure of similarity, and a complementary distance *metric*, that mirrors traditional correlation by observing matching consecutive fluctuations between pairs of series. The intuition is that capturing similarity via *pointwise* agreement—e.g., via Euclidean distance (ED) or other commonly used measures—can be misleading or ineffective. Agreement between two series in t randomly scattered timesteps may not capture overall similarity in trends, whereas two series following the exact same pattern of fluctuations in t *consecutive* timesteps are arguably more "associated". Figure 3 illustrates this with ED and shows the effectiveness of our proposed measure.

The intuition behind our proposed metric, ABC or Approximate Binary Correlation, is to count similar bits between two binarized time series **x** and **y**, while slightly exponentially *weighting* consecutively similar bits in **x** and **y** such that the longer the consecutive matching subsequences, which we call *runs* [5], the more similar **x** and **y** will be deemed. More formally, we define the similarity score as a summation of multiple geometric series, which elegantly captures this intuition. For some $0 < \alpha \ll 1$, ABC adds $(1+\alpha)^i$ to the total "similarity score" for every i-th consecutive bit of agreement between the two series, starting with $i = 0$ every time a new run begins. Thus, the similarity of two binarized series is a

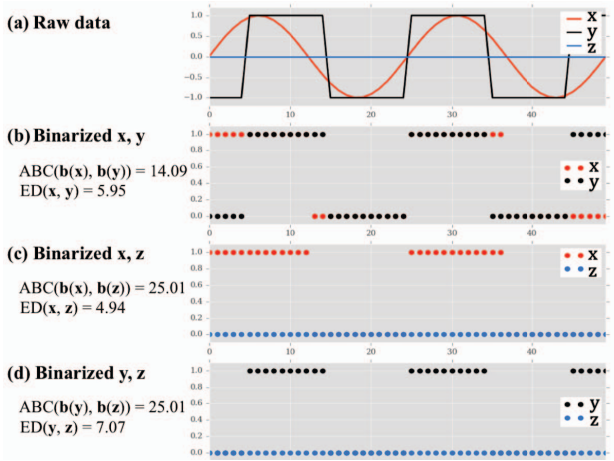


Fig. 3: Although time series x and y are more visually similar in (a) than the other pairs, Euclidean distance (ED) and other pointwise methods fail to reflect that, as x and z are assigned the lowest Euclidean distance (i.e., are the most similar with ED). By contrast, our ABC distance metric assigns the lowest distance to x and y. Moreover, it approximates correlation and admits LSH.

sum of $1 \leq p \leq \frac{n}{2}$ geometric series, each with a common ratio $r = (1+\alpha)$ and a length k_i where $k_1 + \ldots + k_i + \ldots + k_p \leq n$. A visual depiction of computing similarity is given in Figure 4.

The parameter α controls the emphasis on consecutiveness in similarity scoring. For example, choosing $\alpha = 0$ reduces the similarity score to the complement of Hamming distance, and increasing α both increases the maximum similarity score, as explained in the next paragraph, and the "gap" between pairs of series that match in long consecutive runs versus shorter runs (for example, agreement in every other timestep).

Definition 2 (ABC: Approximate Binary Correlation). *Given two binarized time series $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$, which have p matching consecutive subsequences i of length k_i, the ABC similarity is defined as*

$$s(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{p} \sum_{b=0}^{k_i} (1+a)^b = \frac{\sum_{i=1}^{p} (1+\alpha)^{k_i} - p}{\alpha} \qquad \text{(III.1)}$$

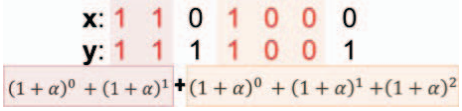


Fig. 4: The ABC similarity between x and y is the sum of the two geometric series that encode the length-2 and 3 runs.

where $\alpha \in (0, 1]$ controls the emphasis on consecutiveness: higher α, higher emphasis. Above we use that the sum of a geometric progression is $\sum_{b=0}^{n-1} r^b = \frac{1-r^n}{1-r}$ for $r \neq 1$.

The maximum possible ABC similarity for two identical series, which we denote $S(\mathbf{x}, \mathbf{y})$, is the sum of a geometric progression from 0 to $n-1$ with common ratio $(1+\alpha)$: $S(\mathbf{x}, \mathbf{y}) = \sum_{i=0}^{n-1} (1+\alpha)^i = \frac{(1+\alpha)^n - 1}{\alpha}$.

We derive the complementary *distance* score by simply subtracting from the maximum similarity. Thus, identical sequences will have a distance of 0, and sequences with no agreeing bits will have a distance of $S(\mathbf{x}, \mathbf{y})$:

$$d(\mathbf{x}, \mathbf{y}) = S(\mathbf{x}, \mathbf{y}) - s(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{p} (1+\alpha)^{k_i} + p - 1}{\alpha} \quad \text{(III.2)}$$

To the best of our knowledge, the ABC distance is the first metric that goes beyond pointwise comparisons and approximates the widely-used correlation well. As we explain in the following sections, the metric property is key for speeding up the network discovery problem.

C. Comparison to Correlation

One of our objectives in designing ABC is to assign similarity scores comparable to Pearson's correlation coefficient, since the latter is the most common association measure in network discovery on time series [7][21]. Intuitively, ABC is like correlation in that it assigns higher similarity to pairs of series following similar trends, which occurs when the series follow the same pattern over longer consecutive intervals. To confirm the intuition, we plotted Pearson's correlation coefficient r against ABC similarity s with $\alpha = .0001$ on all pairs of time series in 10 real brain time series datasets, each taken from the COBRE dataset described in Section IV. Two of these plots, shown as heatmaps, are given in Figure 5. We also performed linear regression on all (r, s) pairs, finding a strong correlation—on average, $r = .84$ with a p-value of 0—between the sets of similarity score pairs in all datasets.

However, one caveat is that ABC similarity as it is introduced here does not take into account inversely correlated relationships as Pearson's correlation coefficient does, so we consider Pearson's correlation coefficient on a scale of [-1, 1] and ABC similarity (normalized by dividing by $S(\mathbf{x}, \mathbf{y})$) on a scale of [0, 1]. In other words, many of the pairs with the lowest ABC score have a strong inverse correlation. We note that although we do not do so here, we could easily consider inverse correlations using ABC by computing similarity scores on both agreeing and *disagreeing* runs, or runs where the corresponding values between the series are opposite.

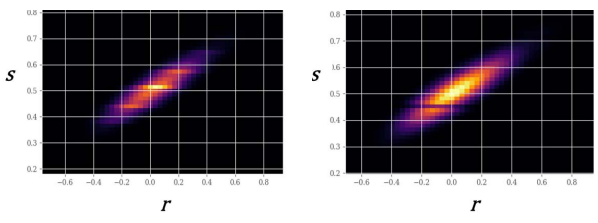


Fig. 5: Pearson's correlation coefficient (x-axis) versus ABC similarity (y-axis) scores for all pairs of time series in two unique time series matrices from the COBRE dataset. There is an evident and strong correlation between the scores, confirming that ABC is a good approximator of correlation.

D. Defining ABC's LSH Family

The final step in our network discovery process is to apply ABC similarity and its distance complement to LSH such that we can quickly identify and connect similar time series. In this section, we show that ABC can be applied to LSH. Satisfying the metric properties is an important first step for defining a distance measure's corresponding LSH family, as upholding these properties allows for guarantees of false positive and negative rates in hashing.

Definition 3 (Distance metric). *A distance metric is a distance measure that satisfies the following axioms [26]: (1) Identity: $d(\boldsymbol{x}, \boldsymbol{y}) = 0 \iff \boldsymbol{x} = \boldsymbol{y}$. (2) Non-negativity: $d(\boldsymbol{x}, \boldsymbol{y}) \geq 0$. (3) Symmetry: $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$. (4) Triangle inequality: $d(\boldsymbol{x}, \boldsymbol{y}) \leq d(\boldsymbol{x}, \boldsymbol{z}) + d(\boldsymbol{z}, \boldsymbol{y})$.*

Our main result is the following:

Theorem 1 (ABC is a metric). *The ABC distance in Eq.* (III.2) *is a metric. It satisfies all the axioms of a metric, including the triangle inequality.*

Proof. We give a sketch of the proof that ABC satisfies these properties, with necessary supporting proofs in the appendix.

(1) **Identity**: the distance $d(\mathbf{x}, \mathbf{y})$ is 0 when $p = 1$ and $k_1 = n$ (Eq. (III.2)): $\mathbf{x}$ and $\mathbf{y}$ have a single consecutive agreeing run of length n, which means that $\mathbf{x} = \mathbf{y}$. Likewise, when $\mathbf{x} = \mathbf{y}$, the number of agreeing runs p will be 1 and the length of the run will be n, so $d(\mathbf{x}, \mathbf{y}) = 0$.

(2) **Non-negativity**: if $\mathbf{x}$ and $\mathbf{y}$ are the same, $d(\mathbf{x}, \mathbf{y}) = 0$. Otherwise, the maximum similarity $S(\mathbf{x}, \mathbf{y})$ is greater than the maximum possible similarity between $\mathbf{x}$ and $\mathbf{y}$, as shown in the Appendix B1, so $d \geq 0$.

(3) **Symmetry**: the order in which we compare sequences does not change the distance.

(4) **Triangle inequality**: this is the most complex and difficult-to-satisfy property. For ABC, we prove it by induction and by considering different options for agreement between sequences. The details of our proof are given in Appendix B2. □

Proposed LSH Family $\mathbf{F_W}$. Although not all distance metrics have a corresponding LSH family, our ABC distance metric does. Formally an LSH family is defined as:

Definition 4 (Locality-sensitive family of hash functions). *Given some distance measure $d(\boldsymbol{x}, \boldsymbol{y})$ satisfying the distance metric axioms, a family of locality-sensitive hash functions $\boldsymbol{F} = (h_1(\boldsymbol{x}), \ldots, h_f(\boldsymbol{x}))$ is said to be (d_1, d_2, p_1, p_2)-sensitive if for every function $h_i(\boldsymbol{x})$ in $\boldsymbol{F}$ and two distances $d_1 < d_2$:*

(1) If $d(\boldsymbol{x},\boldsymbol{y}) \leq d_1$, then the probability that $h_i(\boldsymbol{x}) = h_i(\boldsymbol{y})$ is at least p_1. The higher the p_1, the lower the probability of false negatives.

(2) If $d(\boldsymbol{x},\boldsymbol{y}) \geq d_2$, then the probability that $h_i(\boldsymbol{x}) = h_i(\boldsymbol{y})$ is at most p_2. The lower the p_2, the lower the probability of false positives.

The simplest LSH family uses bit sampling [16] and applies to Hamming distance, which quantifies the number of differing components between two vectors. The bit-sampling LSH family $\mathbf{F}_\mathrm{H}$ over n-dimensional binary vectors consists of all functions that randomly select one of its n components or bits: $\mathbf{F}_\mathrm{H} = \{h : \{0,1\}^n \to \{0,1\} | h(\mathbf{x}) = x_i \text{ for } i \in [1,n]\}$. Under this family, $h_i(\mathbf{x}) = h_i(\mathbf{y})$ if and only if $x_i = y_i$, or in other words the bit at the i-th index of $\mathbf{x}$ is the same as the bit at the i-th index of $\mathbf{y}$. The $\mathbf{F}_\mathrm{H}$ family is a $(d_1, d_2, 1-\frac{d_1}{n}, 1-\frac{d_2}{n})$-sensitive family, based on Def. 4. In this definition, p_1 describes the probability of two vectors colliding when their distance is at most d_1 (i.e., $\mathbf{x}$ and $\mathbf{y}$ differ in at most d_1 bits). Thus, p_1 corresponds to the *complement* of the probability of the vectors *not* colliding or, equivalently, the probability of selecting one of the disagreeing bits out of the n total bits, $\frac{d_1}{n}$. $p_1 = 1 - \frac{d_1}{n}$; $p2$ is derived similarly.

We propose a new LSH family, $\mathbf{F}_\mathrm{W}$, which carries the consecutiveness intuition behind ABC. While the established LSH family on Hamming distance samples bits, our proposed LSH family $\mathbf{F}_\mathrm{W}$ extends this by consisting of randomly sampled *consecutive* subsequences, starting from the same index, for all binary sequences in the dataset. An example of how it works in practice is given in Figure 6.

Theorem 2 (Window sampling LSH family). *Given a window size k, our proposed family of hash functions $\boldsymbol{F}_W$ consists of $n-k+1$ hash functions:*

$$\boldsymbol{F}_W = \{h : \{0,1\}^n \to \{0,1\}^k | h(\mathbf{x}) = (x_i, \ldots, x_{i+k-1}), i \in [1, n-k+1]\}$$

Equivalently, $h_i(\boldsymbol{x}) = h_i(\boldsymbol{y})$ if and only if $(x_i, \ldots, x_{i+k-1}) = (y_i, \ldots, y_{i+k-1})$. The family $\boldsymbol{F}_W$ is $(d_1, d_2, 1-\alpha\frac{d_1}{(1+\alpha)^n-1}, 1-\alpha\frac{d_2}{(1+\alpha)^n-1})$-sensitive.

Proof. Let $s_1 = S(\mathbf{x},\mathbf{y}) - d_1$ be the complementary similarity score for d_1 and s_2 be the complementary similarity score for d_2. The probabilities p_1 and p_2 are derived by scaling d_1 and d_2 and taking their complement; in other words, they are found in the same way that Hamming distance is converted into a normalized similarity score for its corresponding LSH family. In our case, if we normalize both d_1 and d_2 by dividing by the maximum distance $S(\mathbf{x},\mathbf{y})$ we get $\frac{S(\mathbf{x},\mathbf{y})-s_1}{S(\mathbf{x},\mathbf{y})}$ and $\frac{S(\mathbf{x},\mathbf{y})-s_2}{S(\mathbf{x},\mathbf{y})}$, or equivalently $p_1 = \frac{s_1}{S(\mathbf{x},\mathbf{y})} = \alpha\frac{s_1}{(1+\alpha)^n-1}$ and $p_2 = \alpha\frac{s_2}{(1+\alpha)^n-1}$. □

E. ABC-LSH: Hashing Process for ABC

Given an LSH family, it is typical to construct new "amplified" families **F'** by the AND and OR constructions of **F** [26], which provide control of the false positive and negative rates in the hashing process. Concretely:

Definition 5 (AND and OR Construction). *Given a (d_1, d_2, p_1, p_2)-sensitive family of LSH functions $\boldsymbol{F} = (h_1(\boldsymbol{x}), \ldots, h_f(\boldsymbol{x}))$, the* **AND construction** *creates a new hash function $g(\boldsymbol{x})$ as a logical AND of r members of $\boldsymbol{F}$ such that $g(x) = \{h_i\}^r$ for each i chosen uniformly at random without replacement from $[1, f]$. Then we say $g(\boldsymbol{x}) = g(\boldsymbol{y})$ if and only if $h_i(\boldsymbol{x}) = h_i(\boldsymbol{y})$ for all $i \in [1, r]$. The* **OR construction** *does essentially the same as the AND operation, except we say that $g(\boldsymbol{x}) = g(\boldsymbol{y})$ if $h_i(\boldsymbol{x}) = h_i(\boldsymbol{y})$ for any $i \in [1, b]$.*

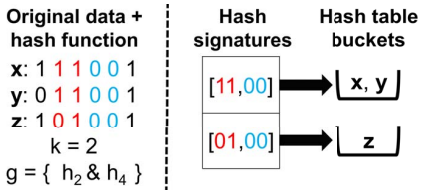


Fig. 6: The hash function g ANDs the second and fourth hash functions (h_2 and h_4) in the LSH family $\mathbf{F}_\mathrm{W}$ with $k = 2$, so the hash signatures are the concatenation (i.e. the AND) of the length-2 windows starting from bits two and four of each sequence.

The AND operation turns any locality-sensitive family **F** into a new $(d_1, d_2, {p_1}^r, {p_2}^r)$-sensitive family **F'**, whereas the OR operation turns the same family **F** into a $(d_1, d_2, 1-(1-p_1)^b, 1-(1-p_2)^b)$-sensitive family. To leverage the benefits of these constructions, we incorporate both of them in the hashing process of our proposed network discovery approach. Let k be the length of the sampled window, and r, b the number of hash functions used in the AND and OR operations, respectively.

AND operation. For the AND operation, we have a single hash table and a hash function $g(x) = \{h_i\}^r$, where each h_i is chosen uniformly at random from $\mathbf{F}_\mathrm{W}$. We compute a hash signature for each data point $\mathbf{x}^{(i)}$ as a concatenation of the r length-k potentially overlapping windows starting at index i for all $h_i \in g$. The algorithm returns all of the buckets of the single hash table created so that all pairs within each bucket can be compared. An example round of hashing with the AND construction is depicted in Figure 6.

OR operation. For the OR operation, we create b hash tables. In b rounds, we compute a hash signature for each data point, where the signature is a length-k window starting at index i for some $h_i \in \mathbf{F}_\mathrm{W}$ chosen independently and randomly at uniform. Thus each data point is hashed b times, and the algorithm returns the union of all buckets for the b hash tables such that all pairs within all buckets can be compared.

Parameter Setting. As we show in the experiments, parameter setting is quite intuitive based on the desired properties of the constructed graph. Adjusting the parameters controls false positive and negative rates. The higher the r, the lower the probability of false positives; the higher the b, the lower the probability of false negatives. The union of AND/OR constructions form an *S-curve*, which is explained in more detail in [26]. In all cases—AND, OR, both, or neither—all pairs $(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ that hash to the same bucket in any of the constructed tables are compared pairwise, and an edge between $(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ with the computed similarity between the two sequences is added to the graph G. Notably, the independence of both the intra-bucket comparison step and the OR construction

make our algorithm easily and embarrassingly parallelizable, although we do not explore parallelization here.

There are no strict runtime guarantees we can make with LSH, since every series could hash to a unique bucket or the same bucket depending on the nature of the data. However, while in theory graph construction *could* be quadratic if all time series are nearly identical, in practice most datasets—even those with series that are on average correlated—will result in sub-quadratic graph construction time, as we show in the experiments.

F. Network Discovery: Putting Everything Together

Having introduced a new distance metric and its corresponding LSH family with the necessary theoretical background, we turn back to our original goal: scalable network discovery, as depicted at a high level in Figure 2. Given N time series, each of length n, we convert all data to binary following the representation in Section III-A, then follow the steps in Algorithm 1, using LSH as described in Section III-E.

Algorithm 1: Window LSH to graph

Input : A set of N binarized time series $\mathbf{X}$;
k: length of the window to sample;
r: # of windows per time series signature;
b: # of hash tables to construct

Output: A weighted graph $G = (V, E)$ where each node $\mathbf{x}^{(i)} \in V$ is a time series and each edge $(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \in E$ has weight $s(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$

$G \leftarrow$ GRAPH
$buckets \leftarrow$ LSH-AND$(\mathbf{X}, k, r) \cup$ LSH-OR$(\mathbf{X}, k, b)$
for $bucket \in buckets$ **do**
  for $(\mathbf{x}^{(i)},\mathbf{x}^{(j)}) \in bucket$ **do**
    $weight \leftarrow S(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$
    G.ADDEDGE$(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}, weight)$
  end
end
return G

IV. EVALUATION

In our evaluation, we strive to answer the following questions by applying our method to several real and synthetic datasets: (1) How efficient is our approach and how does it compare to baseline approaches? (2) How do our output graphs perform in real applications, such as classification of patient and healthy brains? (3) How do the network discovery runtime and network properties change for varying parameters of ABC-LSH? We answer the first two questions in the following subsections and the last in Appendix A.

A. Data

We used several datasets from different domains, since network discovery is relevant in a variety of settings.

Brain data. In recent years, psychiatric and imaging neuroscience have shifted away from the study of segregated or localized brain functions toward a dynamic network perspective of the brain, where statistical dependencies and correlations between activity in brain regions can be modeled as a graph [14]. One well-known data collection procedure from which the brain's network organization may be modeled is resting-state fMRI, which tracks temporal and spatial oscillations of activity in neuronal ensembles [25]. Key objectives of resting-state fMRI are to elucidate the network mechanisms of mental disorders and to identify diagnostic biomarkers—objective, quantifiable characteristics that predict the presence of a disorder—from brain imaging scans [6]. Indeed, a fundamental hypothesis in the science of functional connectivity is that cognitive dysfunction can be illustrated and/or explained by a disturbed functional organization.

For our evaluation of brain graphs, we used two publicly available datasets, given in Table II.

TABLE II: Brain data.

Dataset	Description	Labeled?
COBRE [2]	Resting-state fMRI from 147 subjects: 72 patients with schizophrenia and 75 healthy controls. Each subject is associated with 1166 time series (brain regions) measured for around 100 timesteps, with some variation. For those pairs of time series with unequal lengths, we take the minimum of the lengths.	✔ Control vs. Patient
Penn [1][28]	Resting-state fMRI from 519 subjects. Each subject's is associated with 3789 time series measured across 110 timesteps.	✘

Both datasets were subject to a standard preprocessing pipeline, including 1) linear detrending, or removal of low-frequency signal drift; 2) removal of nuisance effects by regression; 3) band-pass filtering, or rejection of frequencies out of a certain range; and 4) censoring or removal of timesteps with high framewise motion.

For all graphs constructed with pairwise comparison, we held $\theta = 0.6$. For those graphs constructed with ABC, we held $\alpha = .0001$. We constructed graphs with the all-pairs method for COBRE and Penn using the absolute value of Pearson's correlation coefficient as our baseline, as is typical in functional connectivity studies [14]. We also constructed graphs on the same datasets using ABC for both the all-pairs and LSH construction methods. For LSH, we set the window size $k = 10$, following a rule of thumb of keeping the window size roughly $\sqrt{n}$. We set $d_1 = 10$ and $d_2 = 95$, the number of OR constructions $b = 8$, and did not use the AND construction (i.e., $r = 1$) such that we achieved $p_1 = .9999$ and $p_2 = .36$ so as to avoid false negatives for the purpose of comprehensive nearest-neighbors search: we were guaranteed with a 99.99% probability that the ABC distance between any colliding series $\mathbf{x}$ and $\mathbf{y}$ was less than or equal to 10 (out of on average length-100 series).

We acknowledge that our brain datasets are relatively small, as most studies in functional connectivity have used small datasets. Therefore, to compare scalability on larger datasets—both in terms of *more* time series and *longer* time series—we studied graph construction on a generated synthetic dataset as well as a real dataset taken from the UCR Time Series archive.

Synthetic data. Synth-Penn consists of 20,000 length-100 time series that were either taken directly from a single brain in the Penn dataset or else were randomly selected

phase-shifted versions of time series for the same brain in the Penn dataset. The average absolute correlation in Synth-Penn was $|r| = .22$. We used the same LSH parameters as with the real brain graphs. For the purpose of evaluating scalability as the number of time series grows, we generated graphs out of the first 1000, 2000, 5000, and 10,000 series before using the full dataset.

Star Light Data. We also used the StarLightCurves dataset, which is the largest time series dataset from the UCR Time Series archive. It comprises 9236 time series each of length 1024 [9], with an average absolute correlation of $|r| = .577$. Again, for the purpose of evaluating scalability as the number of time series grows, we constructed graphs out of the first 1000, 2000, and 5000 time series as well as the full dataset. For LSH, we set $k = 64$, $d_1 = 64$, and $d_2 = 960$. Then, to avoid both false positives and false negatives with a high probability—and to construct sparser graphs, since our parameter setting for the brain graphs admitted false positives at a higher rate—we chose $r = 6$, and $b = 2$ for false positive and negative rates of less than .01. For detailed discussion on parameter tuning of ABC-LSH (r, b, k), we refer the reader to Appendix A.

B. Scalability Analysis

We ran all experiments and evaluation on a single Ubuntu Linux server with 12 processors and 256 GB of RAM, written in single-threaded Python 3.

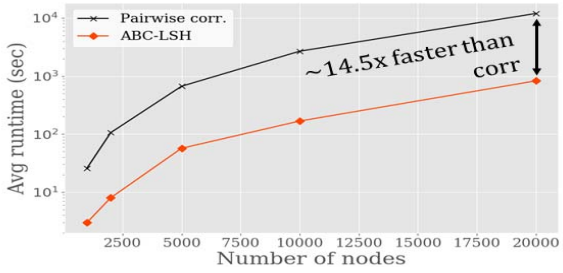
Brain data. An overview of the overall graph construction runtime for all subjects in the COBRE and Penn datasets is given in Table III. Since LSH is randomized, we averaged runtimes for LSH-generated graphs over three trials. We find that though these data are relatively small, meaning that all-pairs comparison is not particularly costly, graph generation with LSH was still an order of magnitude faster. For example, generating all Penn graphs took over 36 hours (on average, 5 minutes/graph) with the pairwise technique, whereas generating the same subject brains with LSH took around 5.5 hours (on average, 38 seconds/graph).

TABLE III: Runtime comparison between pairwise correlation and ABC-LSH for the two brain datasets.

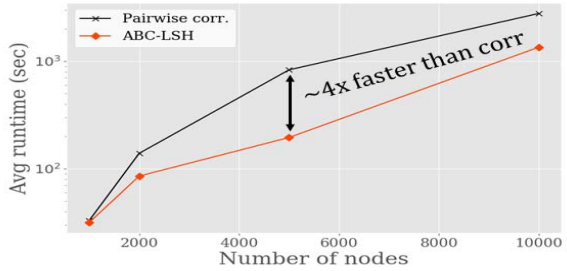
Dataset	Pairwise (sec)	LSH (sec)
COBRE	3,969	**441**
Penn	131,307	**19,722**

Synthetic data. The differences in graph construction runtime grew more pronounced as the size of the data increased. For example, with $N = 20,000$, pairwise construction for a single graph took over 3 hours, whereas LSH construction took on average 13 minutes, as shown in Figure 7a.

Star Light Data. While LSH is still faster than pairwise comparison on the StarLightCurves dataset, we find that it is in general slower than with the brain data. This result is not surprising: the number of comparisons performed by LSH depends heavily on the nature of the data and the average similarity of points in the dataset. The average correlation between time series in StarLightCurves is much higher than that of Synth-Penn, and as such more time series hash to the same bucket. Overall, though, it still outperforms all-pairs comparison at $2-4\times$ faster, as shown in Figure 7b.



(a) Synth-Penn **data**



(b) StarLightCurves **data**

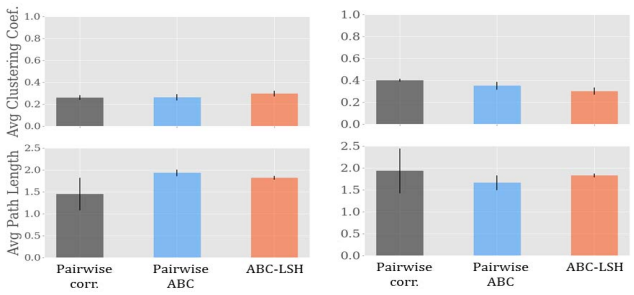
Fig. 7: ABC-LSH vs. pairwise correlation: average runtime w.r.t. the number of nodes. ABC-LSH is up to 14.5× faster. Runtime, plotted on the y-axes, is in log scale.

C. Evaluation of Output Graphs

Our goal is not only to scale the process of network discovery but to construct graphs that are as useful to practitioners as the traditional pairwise-built graphs. For evaluation, we focused on the brain data, as the neuroscience and neuroimaging communities are rich with functional connectivity studies on brain graphs discovered from resting-state fMRI.

Qualitative analysis. Two properties that are often studied in constructed functional networks are the graph clustering coefficients and average path lengths [10], which are hypothesized to encode physical meaning on communication between brain regions. We computed these statistics on all output brain graphs for the COBRE and Penn datasets generated by the pairwise correlation, pairwise ABC, and ABC-LSH methods, and plotted the averages and standard deviations per dataset and method in Figure 8. We find that the averages for both ABC pairwise and ABC-LSH are good approximations of the pairwise correlation baseline.

Task-based evaluation. Since we do not have ground-truth networks for our datasets, we evaluated the quality and distinguishability of the constructed brain networks by performing classification on the output COBRE brain graphs, for which we have healthy/schizophrenic labels. We use graph properties commonly computed on functional networks as features [7]: each output graph is represented by a feature vector of [density, average weighted degree, average clustering coefficient, modularity, average path length]^T.



(a) COBRE **(b)** Penn

Fig. 8: Average clustering coefficient and path length for the generated brain graphs. The averaged properties of the graphs generated by the ABC pairwise and ABC-LSH methods are very close to those of the baseline method, pairwise correlation.

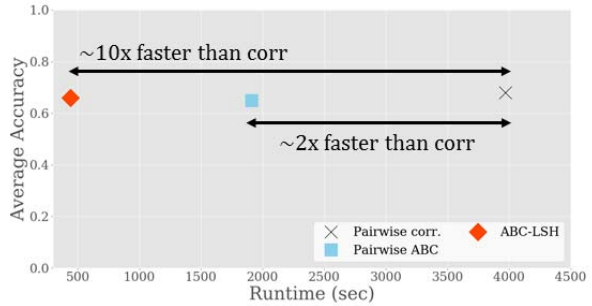


Fig. 9: (Top left is best.) Average classification accuracy versus runtime for pairwise correlation, pairwise ABC, and ABC-LSH.

We followed the generated graph classification setup in [22], training a logistic regression classifier on the graph feature vectors. Per method, we performed a grid search over $\{.01, .1, 1, 10, 100, 1000, 10000\}$ for the regularization parameter C to identify the best setting. We performed classification with stratified 10-fold cross-validation on each method and report the average scores for accuracy, precision, and recall in Table IV, with accuracy versus graph construction runtime is plotted in Figure 9. Unlike the pronounced differences we observe in runtime, classification scores per metric are in the same range for all three methods, with a mere .02 average difference between pairwise correlation and ABC-LSH.

TABLE IV: Classification scores on the output COBRE **brain graphs using pairwise correlation, pairwise ABC, and ABC-LSH.**

Method	C	Metric Score		
		Accuracy	Precision	Recall
Pairwise corr.	1	.68	.66	.80
Pairwise ABC	10	.65	.63	.76
ABC-LSH	10000	.66	.63	.76

V. Conclusion

In this work we motivate the problem of efficient network discovery on time series, drawing in particular from the vibrant research area of functional connectivity. To scale the existing quadratic-time methods, we propose ABC-LSH, a 3-step approach that approximates time series, hashes them via window sampling, and builds a graph based on the results of hashing. This method is based on our novel and intuitive distance measure, ABC, which approximates correlation by emphasizing consecutiveness in trends, as well as on a new window-sampling LSH family. We apply our work to several datasets and show that, with the right choice of parameters on our real and synthetic datasets, ABC-LSH graph construction is up to $15\times$ faster than all-pairs graph construction, while maintaining accuracy in terms of output graph properties and classification power. We believe our work opens up many possibilities for future study at the intersection of network discovery, hashing, and time series, which will impact a variety of domains as data are continuously generated at ever-increasing scale.

Acknowledgment

This material is based upon work supported by the University of Michigan.

References

[1] Philadelphia Neurodevelopmental Cohort. http://www.med.upenn.edu/bbl/philadelphianeurodevelopmentalcohort.html.
[2] Center for Biomedical Research Excellence. http://fcon_1000.projects.nitrc.org/indi/retro/cobre.html, 2012.
[3] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, January 2008.
[4] Yosef Ashkenazy, Plamen Ch. Ivanov, Shlomo Havlin, Chung-K. Peng, Ary L. Goldberger, and H. Eugene Stanley. Magnitude and Sign Correlations in Heartbeat Fluctuations. *Physical Review Letters*, 86(9):1900–1903, 2001.
[5] Narayanaswamy Balakrishnan and Markos Koutras. *Runs and Scans With Applications*. Wiley, 2002.
[6] Danielle Bassett and Ed Bullmore. Human Brain Networks in Health and Disease. *Current Opinion in Neurology*, 22(4):340–347, 2009.
[7] Ed Bullmore and Olaf Sporns. Complex Brain Networks: Graph Theoretical Analysis of Structural and Functional Systems. *Nature Reviews Neuroscience*, 10(3):186–198, 2009.
[8] Surajit Chaudhuri, Venkatesh Ganti, and Raghav Kaushik. A primitive operator for similarity joins in data cleaning. In *Proceedings of the 22Nd International Conference on Data Engineering*, ICDE '06, 2006.
[9] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The UCR Time Series Classification Archive. www.cs.ucr.edu/~eamonn/time_series_data/, 2015.
[10] Zhengjia Dai and Yong He. Disrupted Structural and Functional Brain Connectomes in Mild Cognitive Impairment and Alzheimer's Disease. *Neuroscience Bulletin*, 30(2):217–232, 2014.
[11] Ian Davidson, Sean Gilpin, Owen Carmichael, and Peter Walker. Network discovery via constrained tensor analysis of fmri data. In *KDD*, pages 194–202, 2013.
[12] Wei Dong, Charikar Moses, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. pages 577–586, 2011.
[13] Leonardo N. Ferreira and Liang Zhao. Time Series Clustering via Community Detection in Networks. 2015.
[14] Karl J. Friston. Functional and Effective Connectivity: A Review. *Brain Connectivity*, 1(1):13–36, 2011.
[15] David Hallac, Youngsuk Park, Stephen Boyd, and Jure Leskovec. Network Inference via the Time-Varying Graphical Lasso. In *KDD*, 2017.
[16] Piotr Indyk and Rajeev Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *STOC*, pages 604–613, 1998.
[17] David C. Kale, Dian Gong, Zhengping Che, Yan Liu, Gerard Medioni, Randall Wetzel, and Patrick Ross. An Examination of Multivariate Time Series Hashing with Applications to Health Care. In *ICDM*, pages 260–269, 2014.

[18] Eamonn Keogh and Michael Pazzani. An Indexing Scheme for Fast Similarity Search in Large Time Series Databases. In *SSDM*, pages 56–, 1999.
[19] Yongwook Bryce Kim and Una-May Hemberg, Erik O'Reilly. Stratified Locality-Sensitive Hashing for Accelerated Physiological Time Series Retrieval. In *EMBC*, 2016.
[20] Yongwook Bryce Kim and Una-May O'Reilly. Large-Scale Physiological Waveform Retrieval Via Locality-Sensitive Hashing. In *EMBC*, pages 5829–5833, 2015.
[21] Chia-Tung Kuo, Xiang Wang, Peter Walker, Owen Carmichael, Jieping Ye, and Ian Davidson. Unified and Contrasting Cuts in Multiple Graphs: Application to Medical Imaging Segmentation. In *KDD*, pages 617–626, 2015.
[22] John Boaz Lee, Xiangnan Kong, Yihan Bao, and Constance Moore. Identifying Deep Contrasting Networks from Time Series Data: Application to Brain Network Analysis. 2017.
[23] Chen Luo and Anshumali Shrivastava. SSH (Sketch, Shingle, and Hash) for Indexing Massive-Scale Time Series. In *NIPS Time Series Workshop*, 2016.
[24] Jukka-Pekka Onnela, Kimmo Kaski, and Jnos Kertsz. Clustering and Information in Correlation Based Financial Networks. *European Physical Journal B*, 38:353–362, 2004.
[25] Hae-Jeong Park and Karl Friston. Structural and Functional Brain Networks: From Connections to Cognition. *Science*, 342(6158), 2013.
[26] Anand Rajaraman and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA, 2011.
[27] Chotirat Ratanamahatana, Eamonn Keogh, Anthony J. Bagnall, and Stefano Lonardi. A Novel Bit Level Time Series Representation with Implication of Similarity Search and Clustering. In *PAKDD*, pages 771–777, 2005.
[28] T.D. Satterthwaite, M.A. Elliott, K Ruparel, J. Loughead, K. Prabhakaran, M.E. Calkins, R. Hopson, C. Jackson, J. Keefe, M. Riley, F.D. Mentch, P. Sleiman, R. Verma, C. Davatzikos, H. Hakonarson, R.C. Gur, and R.E. Gur. Neuroimaging of the Philadelphia Neurodevelopmental Cohort. *NeuroImage*, 86:544–553, 2014.
[29] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.
[30] Sen Yang, Qian Sun, Shuiwang Ji, Peter Wonka, Ian Davidson, and Jieping Ye. Structural graphical lasso for learning mouse brain connectivity. In *KDD*, pages 1385–1394, 2015.
[31] Yan-Ming Zhang, Kaizhu Huang, Guanggang Geng, and Cheng-Lin Liu. Fast knn graph construction with locality sensitive hashing. In *ECML PKDD*, pages 660–674, 2013.

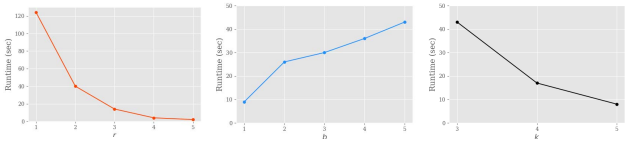
APPENDIX

A. Appendix 1: Sensitivity Analysis of ABC-LSH

Runtime. Since the number of time series compared in ABC-LSH depends on the length of hash signatures r, the number of hash tables constructed b, and the window length k, we generated a single brain graph from a COBRE subject with a variety of LSH settings using ABC-LSH and $\alpha = .0001$:

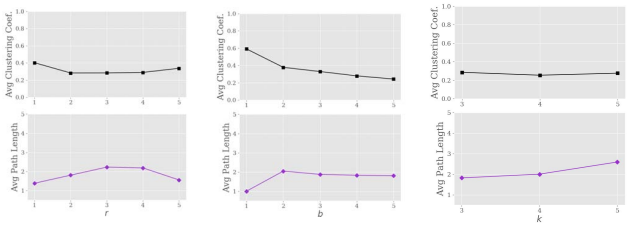
1) $r \in [1, 5]$, holding $b = 4$ and $k = 3$;
2) $b \in [1, 5]$, holding $r = 2$ and $k = 3$;
3) $k \in [3, 5]$, holding $r = 2$ and $b = 4$;

The runtime trends are plotted in Figure 10. As expected, increasing r and/or k reduced graph construction runtime,



(a) Increasing r. **(b) Increasing b.** **(c) Increasing k.**

Fig. 10: ABC-LSH graph construction time vs. parameters.



(a) Varying r. **(b) Varying b.** **(c) Varying k.**

Fig. 11: Clustering coeff. and avg. path length for varying the ABC-LSH parameters r, b, and k on a single COBRE subject.

whereas increasing b increased graph construction runtime. For r: the longer the length of the hash signature, the more unlikely that two time series **x** and **y** will have the same signature, so runtime decreases as r increases. For b: the more hash tables created, the more "chances" two time series $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ get to collide, so runtime increases as b increases. For k: the longer the window length, the more unlikely that two time series windows $(x_i, \ldots, x_{i+k-1})$ and $(y_i, \ldots, y_{i+k-1})$ will match exactly, so runtime decreases as k increases.

Network Properties. As we mentioned earlier, two properties that are often studied in functional networks are the clustering coefficient and average path length [10]. We computed the differences in these properties among the generated graphs by varying the parameters of ABC-LSH as before. The changes in properties as the parameters vary are plotted in Figure 11. The trends for average clustering coefficient and average path length are quite stable: barring small fluctuations, the average path lengths stay short (< 2.5) and the average clustering coefficients hover around .4 - .6. These network properties of the ABC-LSH networks are robust to parameter changes.

B. Appendix 2: Metric Proof of ABC

Here we give the theoretical foundation that underlies our time series distance proposal, ABC. We show that it satisfies the metric properties and is thus eligible for LSH.

1) Properties of Agreeing Runs: We first study the relationship between p, the number of agreeing runs between **x** and **y**, and the maximum value of $k_1 + \ldots + k_p$, the lengths of the p agreeing runs.

Lemma 1 (Maximum sum of lengths of p runs $k_1, \ldots, k_p$). *Given **x**, **y** $\in \{0,1\}^n$ with p agreeing runs, each of length k_i, the maximum sum of their lengths follows a linearly decreasing relationship, as $\sum_{i=1}^{p} k_i = n - p + 1$.*

This follows because the greater the number of runs, the more bits that must "disagree" in order to separate those runs.

Lemma 2 (Maximum ABC similarity). ***Given x**, **y** $\in \{0,1\}^n$ with p agreeing runs, each of length k_i, **x** and **y** have maximum ABC similarity when they agree on (without loss of generality, the first) $p-1$ runs of length $k_1 = \ldots = k_{p-1} = 1$ and one run of length $k_p = n - 2p + 2$.*

Proof. By induction on p. We omit the proof for brevity. □

Definition 6 (**Maximum ABC similarity given** p)**.** *Based on Theorem 2, the max ABC similarity $S(\boldsymbol{x},\boldsymbol{y})_p$ between two binary time series $\boldsymbol{x}$ and $\boldsymbol{y}$ with p agreeing runs, is*

$$S(\boldsymbol{x},\boldsymbol{y})_p = (p-1) + \frac{(1+\alpha)^{n-2p+2}-1}{\alpha} \tag{A.1}$$

Likewise, the minimum ABC distance between two binary sequences $\boldsymbol{x}$ and $\boldsymbol{y}$ given a p is

$$min_p[d(\boldsymbol{x},\boldsymbol{y})] = \frac{(1+\alpha)^n - (1+\alpha)^{n-2p+2}}{\alpha} - p + 1 \tag{A.2}$$

2) Proving the Triangle Inequality: As stated in Section III, our main result that enables scalable network discovery is that ABC is a metric satisfying the triangle inequality.

Proof. We prove by induction on n, the length of the binary sequences compared.

• **Base case**: $n = 1$. **x**, **y**, and **z** are the same: $d(\mathbf{x},\mathbf{y}) = d(\mathbf{x},\mathbf{z}) = d(\mathbf{z},\mathbf{y}) = 0$; **x** and **y** are the same, **z** is different: $d(\mathbf{x},\mathbf{y}) = 0$, $d(\mathbf{x},\mathbf{z}) = 1$, and $d(\mathbf{z},\mathbf{y}) = 1$; **x** and **z** are the same, **y** is different: $d(\mathbf{x},\mathbf{y}) = 1$, $d(\mathbf{x},\mathbf{z}) = 0$, and $d(\mathbf{z},\mathbf{y}) = 1$; **y** and **z** are the same, **x** is different: $d(\mathbf{x},\mathbf{y}) = 1$, $d(\mathbf{x},\mathbf{z}) = 1$, and $d(\mathbf{z},\mathbf{y}) = 0$.

• **Inductive step**: Assume that $d(\mathbf{x},\mathbf{y}) \leq d(\mathbf{x},\mathbf{z}) + d(\mathbf{z},\mathbf{y})$ for some value of $n > 1$. We show that this implies that the inequality holds for binarized series of length $n+1$, which are constructed by adding a single bit to the end of each of **x**, **y**, and **z** to create $\mathbf{x}'$, $\mathbf{y}'$, and $\mathbf{z}'$, respectively. We begin with basics and setup. First, we denote the distance between **x** and $\mathbf{y} \in \{0,1\}^n$ as $d(\mathbf{x},\mathbf{y})$:

$$d(\mathbf{x},\mathbf{y}) = \frac{(1+\alpha)^n - (1+\alpha)^{k_1} - \ldots - (1+\alpha)^{k_p} + p - 1}{\alpha}$$

and the distance between $\mathbf{x}'$ and $\mathbf{y}' \in \{0,1\}^{n+1}$ as $d'(\mathbf{x}',\mathbf{y}')$:

$$d'(\mathbf{x}',\mathbf{y}') = \frac{(1+\alpha)^{n+1} - (1+\alpha)^{k_1} - \ldots - (1+\alpha)^{k_{p'}} + p' - 1}{\alpha}$$

Here, p' can either be p or $p+1$: $p' = p$ in the case that the $n+1$-th bit either appends onto an existing run between **x** and **y**, or else disagrees between the two sequences, and $p' = p+1$ in the case that the $n+1$-th bit creates a new run of length 1 between **x** and **y**. We now examine how distance between **x** and **y** changes by adding a single bit to the end of **x** and **y**: in other words, moving from n to $n+1$. We denote this change in distance $\Delta_{(i)}$ for $i = 1, 2,$ or 3.

Case 1: The $n+1$-th bits of **x** and **y** agree, creating a new run of length one between the sequences. Here $p' = p+1$, so $k_{p'} = 1$ and $(1+\alpha)^{k_{p'}} = (1+\alpha)$. In this case, $\Delta_{(1)} = d'(\mathbf{x}',\mathbf{y}') - d(\mathbf{x},\mathbf{y}) = (1+\alpha)^n - 1$.

Intuitively this result means that the maximum similarity $S_G(\mathbf{x},\mathbf{y})$ increases by $(1+\alpha)^n$, and from this we subtract a new agreeing run of length 1. In other words, we subtract $(1+\alpha)^0$ from the new maximum similarity since the exponent of a new run always begins at 0. Thus, overall the distance changes by $(1+\alpha)^n - (1+\alpha)^0 = (1+\alpha)^n - 1$.

Case 2: The $n+1$-th bits of **x** and **y** agree, adding or appending onto an existing run of length k_p between the sequences. Here $p' = p$ and $k_{p'} = k_p + 1$, so $(1+\alpha)^{k_{p'}} = (1+\alpha)^{k_p+1}$. Then $\Delta_{(2)} = d'(\mathbf{x}',\mathbf{y}') - d(\mathbf{x},\mathbf{y}) = (1+\alpha)^n - (1+\alpha)^{k_p}$.

Case 3: The $n+1$-th bits of **x** and **y** disagree. Here $p' = p$ and $k_{p'} = k_p$. Then, $\Delta_{(3)} = d'(\mathbf{x}',\mathbf{y}') - d(\mathbf{x},\mathbf{y}) = (1+\alpha)^n$.

With this setup, we enumerate several of the possible cases that can occur when we add a single bit to **x**, **y**, and **z** to obtain $\mathbf{x}'$, $\mathbf{y}'$, and $\mathbf{z}'$, omitting repetitive cases for brevity. Let **n** stand for "new run" (case **(1)** above with $\Delta_{(1)}$), **a** stand for "append to existing run" (case **(2)** with $\Delta_{(2)}$), and **d** stand for "disagree" (case **(3)** with $\Delta_{(3)}$). There are $3 \times 3 \times 3$ length-3 permutations with repetition of **n**, **a**, and **d** for three series $\mathbf{x}'$, $\mathbf{y}'$, and $\mathbf{z}'$, although not all are possible in practice: in fact, only 10 out of the 27 cases are feasible. Below we enumerate some of the cases and either explain why the case is impossible or else show that the triangle inequality holds.

• **ddd**: Not possible: the $n+1$-th bit can be 0 or 1; since there are three binarized series $\mathbf{x}'$, $\mathbf{y}'$, and $\mathbf{z}'$, by the pigeonhole principle at least two of them must agree in that bit. Similar arguments show the impossibility of the following cases: **daa**, **dan**, **dnn**, **dna**, **ada**, **adn**, **ndn**, **nda**, **and**, **aad**, **nad**, **nnd**, **aan**, **naa**, **ana**, **nnn**.

• **dda**: The distance between **x** and **y** changes by $\Delta_{(3)} = (1+\alpha)^n$, as does the distance between **x** and **z**. The distance between **z** and **y** changes by $\Delta_{(2)} = (1+\alpha)^n - (1+\alpha)^{k_p}$ where k_p is the length of the last run between **z** and **y**. We have $(1+\alpha)^n \leq (1+\alpha)^n + (1+\alpha)^n - (1+\alpha)^{k_p}$, or $0 \leq (1+\alpha)^n - (1+\alpha)^{k_p}$. Since $k_p \leq n$, the inequality holds. Symmetric case: **dad**.

• **ddn**: The distance between **x** and **y** changes by $\Delta_{(3)} = (1+\alpha)^n$, as does the distance between **x** and **z**. The distance between **z** and **y** changes by $\Delta_{(1)} = (1+\alpha)^n - 1$. We have $(1+\alpha)^n \leq (1+\alpha)^n + (1+\alpha)^n - 1$, or $0 \leq (1+\alpha)^n - 1$. Since $n > 0$, the inequality holds. Symmetric case: **dnd**.

• **add**: The distance between **x** and **y** changes by $\Delta_{(2)} = (1+\alpha)^n - (1+\alpha)^{k_p}$ where k_p is the length of the last run between **x** and **y**. The distance between **x** and **z** changes by $\Delta_{(3)} = (1+\alpha)^n$, as does the distance between **z** and **y**. We have $(1+\alpha)^n - (1+\alpha)^{k_p} \leq (1+\alpha)^n + (1+\alpha)^n$, or $-(1+\alpha)^{k_p} \leq (1+\alpha)^n$. The right-hand side must be positive, so the inequality holds.

• **nan**: The distance between **x** and **y** changes by $\Delta_{(1)} = (1+\alpha)^n$, as does the distance between **z** and **y**. The distance between **x** and **z** changes by $\Delta_{(2)} = (1+\alpha)^n - (1+\alpha)^{k_p}$, where k_p is the length of the lst run between **x** and **y**. Symmetric case: **nna**.

• **ndd**: The distance between **x** and **y** changes by $\Delta_{(1)} = (1+\alpha)^n - 1$. The distance between **x** and **z**, as well as the distance between **z** and **y**, changes by $(1+\alpha)^n$. We have $(1+\alpha)^n - 1 \leq (1+\alpha)^n + (1+\alpha)^n$, or $0 \leq (1+\alpha)^n + 1$, so the inequality holds.

• **ann**: The distance between **x** and **y** changes by $\Delta_{(3)} = (1+\alpha)^n - (1+\alpha)^{k_p}$, where k_p is the length of the last run between **x** and **y**. The distance between **x** and **z** changes by $\Delta_{(1)} = (1+\alpha)^n - 1$, as does the distance between **z** and **y**. We have $(1+\alpha)^n - (1+\alpha)^{k_p} \leq (1+\alpha)^n - 1 + (1+\alpha)^n - 1$, or equivalently $0 \leq (1+\alpha)^n + (1+\alpha)^{k_p} - 2$. $(1+\alpha)^n + (1+\alpha)^{k_p} \leq 2$, since each term is at least 1, so the inequality holds.

• **aaa**: Let the length of last run between **x** and **y** be denoted k_{p1} and the length of the last run between **x** and **z** be noted k_{p2}. Then the distance between **x** and **y** increases by $\Delta_{(2)} = (1+\alpha)^n - (1+\alpha)^{k_{p1}}$, and the distance between **x** and **z** increases by $\Delta_{(2)} = (1+\alpha)^n - (1+\alpha)^{k_{p2}}$. Since the new bit has added onto an existing run between **x** and **y** of length k_{p1}, and an existing run between **x** and **z** of length k_{p2}, the longest the last run between **z** and **y** can be is $\min[k_{p1}, k_{p2}]$. For example, if the run between **x** and **y** is 3 bits and the run between **x** and **z** is 5 bits, the run between **y** and **z** can be at most 3 bits. Therefore, the distance between **z** and **y** changes by $\Delta_{(2)} = (1+\alpha)^n - (1+\alpha)^{\min[k_{p1},k_{p2}]}$.

If $k_{p1} < k_{p2}$, the inequality becomes $(1+\alpha)^n - (1+\alpha)^{k_{p1}} \leq (1+\alpha)^n - (1+\alpha)^{k_{p2}} + (1+\alpha)^n - (1+\alpha)^{k_{p1}}$, or equivalently $0 \leq (1+\alpha)^n - (1+\alpha)^{k_{p2}}$. Since $k_{p2} \leq n$, the inequality holds. If $k_{p1} > k_{p2}$, the inequality becomes $(1+\alpha)^n - (1+\alpha)^{k_{p1}} \leq (1+\alpha)^n - (1+\alpha)^{k_{p2}} + (1+\alpha)^n - (1+\alpha)^{k_{p2}}$, or equivalently $0 \leq (1+\alpha)^n - (1+\alpha)^{k_{p2}} + [(1+\alpha)^{k_{p1}} - (1+\alpha)^{k_{p2}}]$. The bracketed term must be greater than 0 because $k_{p1} > k_{p2}$. Since $(1+\alpha)^n - (1+\alpha)^{k_{p2}}$ because $k_{p2} \leq n$, we get a sum of two terms that are each greater than or equal to 0, so the inequality holds. □