Adaptive Granularity in Time Evolving Graphs as Tensors

Ravdeep Pasricha
University of California Riverside
rpasr001@ucr.edu

Ekta Gujral
University of California Riverside
egujr001@ucr.edu

Evangelos E. Papalexakis
University of California Riverside
epapalex@cs.ucr.edu

ABSTRACT

Data collected at very frequent intervals is usually extremely sparse and has no structure that is exploitable by modern tensor decomposition algorithms. Thus the utility of such tensors is low, in terms of the amount of interpretable and exploitable structure that one can extract from them. In this paper, we introduce the problem of finding a tensor of adaptive aggregated granularity that can be decomposed to reveal meaningful latent concepts (structures) from datasets that, in their original form, are not amenable to tensor analysis. Such datasets fall under the broad category of sparse point processes that evolve over space and/or time. To the best of our knowledge, this is the first work that explores adaptive granularity aggregation in tensors. Furthermore, we formally define the problem and discuss what different definitions of "good structure" can be in practice, and show that optimal solution is of prohibitive combinatorial complexity. Subsequently, we propose an efficient and effective greedy algorithm which follows a number of intuitive decision criteria that locally maximize the "goodness of structure", resulting in high-quality tensors. We evaluate our method on synthetic and semi-synthetic data where ground truth is known. Our proposed method constructs tensors that have very high structure quality.

KEYWORDS

Tensor analysis, unsupervised learning, time evolving graphs

1 INTRODUCTION

In the age of big data, applications deal with data collected at very fine-grained time intervals. In many real world applications, the data collected spans long periods of time and can be extremely sparse. For instance, a time-evolving social network that records interactions of users every second results in a very sparse adjacency matrix if observed at that granularity. Similarly, in spatio-temporal data, if one considers GPS data over time, discretizing GPS coordinates based on the observed granularity can lead to very sparse data which may not contain any visible and useful structure. How

can we find meaningful and actionable structure in these types of data? A great deal of such datasets are multi-aspect in nature and hence can be modeled using tensors. For instance, a three-mode tensor can represent a time-evolving graph capturing user-user interactions over a period of time, measuring crime incidents in a city community area over a period of time [1, 17], or measuring traffic patterns [21]. Tensor decomposition has been used in order to extract hidden patterns from such multi-aspect data [10, 13, 16]. However the degree of sparsity in the tensor, which is a function of the granularity in which the tensor is formed, significantly affects the ability of the decomposition to discover "meaningful" structure in the data.

Consider a dataset which can be modeled as three-mode tensor, where the third mode is temporal as shown in Figure 1. If the granularity of the temporal mode is too fine (in milliseconds or seconds), one might end up with a tensor that is extremely long on the time mode and where each instance of time has very small number of entries. This results in a extremely sparse tensor, which typically is of very high rank, and which usually has no underlying exploitable structure for widely popular and successful tensor decomposition algorithms [10, 13, 16]. However, as we aggregate data points over time, exploitable structure starts to appear (where-by "exploitable" we define the kind of low-rank structure that a tensor decomposition can successfully model and extract). In this paper we set out to explore what is the best such data-driven aggregation of a tensor which leads to better, exploitable, and interpretable structure, and how this fares against the traditional alternative of selecting a fixed interval for aggregation.

As far as tackling the problem above, there is considerable amount of work that focuses on a special case, that of aggregating edges of a time evolving graph into "mature" adjacency matrices based on certain graph properties [18–20]. In our work, however, we address the problem in more general terms, where the underlying data can be any point process that is observed over time and/or space, and where the aggregation/discretization of the corresponding dimensions directly affects our ability to extract interpretable patterns via tensor decomposition. Effectively, as shown in Figure 1, in this paper we work towards automating the data aggregation starting from raw data into a well-structured tensor.

Our contributions in this work are as follows:

- **Novel Problem Formulation**: We formally define the problem of optimally aggregating a tensor, which is formed from raw sparse data in their original level of aggregation, into a tensor with exploitable and interpretable structure. We further show that solving this problem optimally is computationally intractable. To the best of our knowledge, this paper is the first to tackle this problem in its general form, and we view our formulation as the first step towards automating the process of creating well-behaved tensor datasets.
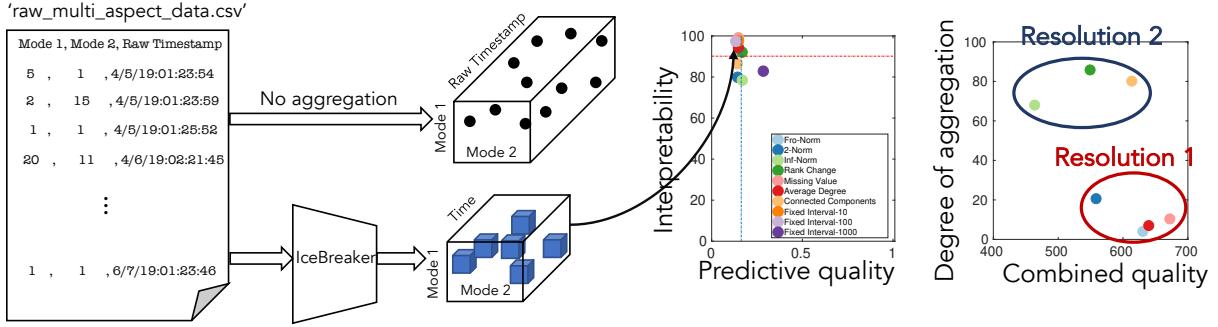
Figure 1: Starting from raw CSV files, IceBreaker discovers a tensor that has good structure (under various measures of quality, including interpretability and predictive quality), outperforming traditional fixed aggregation heuristics. Furthermore, IceBreaker using various notions of locally optimal structure, discovers different resolutions in the data.

- **Practical Algorithm**: We propose a practical, efficient, and effective algorithm that is able to produce high-quality tensors from raw data without incurring the combinatorial cost of the optimal solution. Our proposed method follows a greedy approach, where at each step we decide whether different "slices" of the tensor are aggregated based on a variety of intuitive functions that characterize the "goodness of structure" locally.
- **Experimental Evaluation**: We extensively evaluate our proposed method on synthetic and semi-synthetic data, where ground truth is known, where we use popular heuristic measures of structure goodness to measure success.
- **Reproducibility**: We make our implementation publicly available at link[1] in order to encourage reproducibility of our results.

2 PROBLEM FORMULATION

2.1 Tensor Definition and Notations

Tensors are multi-dimensional extensions of matrices, and tensor decompositions are a class of methods that extract latent structure from tensor datasets by extending techniques such as Principal Component Analysis and Singular Value Decomposition. The different "dimensions" of a tensor are usually referred to as "modes". In this paper, we focus on the CANDECOMP/PARAFAC (henceforth refered to as CP for brevity) decomposition [9], which is the "rank decomposition" of a tensor, i.e., the decomposition of an arbitrary tensor into a sum of R rank-one tensors. Mathematically, for a three-mode

tensor $\underline{\mathbf{X}}$, the CP decomposition is $\underline{\mathbf{X}} \approx \sum_{r=1}^{R} \mathbf{A}(:,r) \circ \mathbf{B}(:,r) \circ \mathbf{C}(:,r)$,

where $\circ$ is the generalized outer product. Matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are called "factor matrices", and each column corresponds to a latent pattern, directly relating an entity of the corresponding mode to a value that can be roughly construed as a soft clustering coefficient [5]. CP has arguably been the most popular tensor decomposition model in applications where the interest is to extract interpretable patterns for exploratory analysis, and thus, we adopt this decomposition model as our standard in this work. In the interest of space, we

---

[1]https://github.com/ravdeep003/adaptive-granularity-tensors

refer the reader to a number of available surveys [10, 13, 16]. We denote tensors as $\underline{\mathbf{X}}$ and matrices as $\mathbf{X}$, and we adopt Matlab-like notation for indexing.

2.2 Tensor decomposition quality

Unsupervised tensor decomposition, albeit very popular, poses a significant challenge: how can we tell whether a computed decomposition is of "high quality", and how can we go about defining "quality" in a meaningful way? Unfortunately, this happens to be a very hard problem to solve [14], and defining a new measure of quality is beyond the scope of this paper. However, there has been significant amount of work in that direction, which basically boils down to 1) model-based measures, where the quality is measured by how well a given decomposition represents the intrinsic hidden structure of the data, and 2) extrinsic measures, where the quality is measured by how well the computed decomposition factors perform in a predictive task.

In terms of model-based measures, the most straightforward one is the fit, i.e., how well does the decomposition approximate the data under the chosen loss function, in a *low rank*. Low rank is key, because the number of components (rank) has to be as small and compact as possible in order to lend itself to human evaluation and exploratory analysis. However, fit has been shown to be unstable and prone to errors especially in real and noisy data, thus the community has collectively turned its attention to more robust measures such as the Core Consistency Diagnostic (CORCONDIA for short) [3], which measures how well the computed factors obey the CP model.

On the other end of the spectrum, extrinsic quality measures are always tied to a predictive task. A popular such task is community detection, where the tensor consists of a multi-view or time-evolving graph, and the frontal slices contain adjacency matrices of that given graph; the task is to use the computed factors for the nodes as features for assigning them to a community label and subsequently measure the quality of that assignment with measures such as the Normalized Mutual Information (NMI) [4, 7, 8].

Both types of quality measure are heuristic and capture different elements of what an end-user would deem good in a set of decomposition factors. In this paper, we are going to use such popular

measures of quality in order to characterize the quality of a given tensor dataset $\underline{\mathbf{X}}$. In order to do so, we assume that we have a function $Q\left(\underline{\mathbf{X}}\right)$ which, optimizes the heuristic quality measure $q\left(\right)$ for a given tensor over all possible decomposition ranks R [2], i.e.,

$$Q_i\left(\underline{\mathbf{X}}\right) = \max_R q_i\left(\underline{\mathbf{X}}, \mathbf{A}, \mathbf{B}, \mathbf{C}\right)$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are the R-column factor matrices for $\underline{\mathbf{X}}$. Finally, a useful operation is the n-mode product, where a matrix $\mathbf{W}$ is multiplied by the n-th mode of a tensor (predicated on matching dimensions in the n-th mode of the tensor and the rows of the matrix), denoted as $\underline{\mathbf{X}} \times_n \mathbf{W}$. For instance, an $I \times J \times K$ tensor where $n = 3$ and $\mathbf{W}$ of size $K \times K^*$, the product $\underline{\mathbf{X}} \times_n \mathbf{W}$ multiplies all third mode slices of $\underline{\mathbf{X}}$ with $\mathbf{W}$ and results in a $I \times J \times K^*$ tensor.

2.3 The *Trapped Under Ice* problem

To give reader an intuition of the problem, consider an example of time-evolving graph which captures social activity over the span of some time. This example can be modeled as three-mode tensor $\underline{\mathbf{X}}$ of dimensions $I \times J \times K$ where "sender" and "receiver" are the first two modes, "time" being the third mode, and non-zero entry in the tensor represents communications between user at a particular time. If the time granularity is extremely fine-grained (milliseconds or seconds), there might be only handful of data points at a particular time stamp causing resulting tensor to be extremely sparse and to have a high tensor-rank as a result. In that case, $\underline{\mathbf{X}}$ might not have any interpretable low-rank structure that can be exploited by CP. In this example we assume that the third mode (time mode) is too fine-grained but in reality any mode (one or more) can be extremely fine grained. For example, in spatio-temporal data, where the first two modes are latitude and longitude and the third mode is time, all three modes can suffer from the same problem.

Given tensor $\underline{\mathbf{X}}$ which is created using the "raw" granularities, how does one find a tensor (say $\underline{\mathbf{Y}}$) which has better exploitable structure and hence can be decomposed into meaningful latent structure. This, is informally the *Trapped Under Ice* problem that we define here (which draws an analogy between the good structure that may exist within the data as being trapped under the ice and not visible by mere inspection). *Trapped Under Ice* has an inherent assumption that the mode in which we aggregate is ordered (e.g., representing time or space), thus permuting the third mode will lead to a different instance of the problem.

More formally we define our problem as follows:

Given a tensor $\underline{\mathbf{X}}$ of dimensions $I \times J \times K$ Find:
A tensor $\underline{\mathbf{Y}}$ of dimensions $I \times J \times K^*$ with $K^* \leq K$ such that

$$\max_{\mathbf{W}} Q\left(\underline{\mathbf{X}} \times_3 \mathbf{W}\right)$$

where Q is a measure of goodness and $\mathbf{W}(i, j) = 1$ if slice i in tensor $\underline{\mathbf{X}}$ is aggregated into slice j in the resulting tensor, otherwise $\mathbf{W}(i, j) = 0$.

At first glance, *Trapped Under Ice* might look like a problem amenable to dynamic programming, since it exhibits the optimal substructure property. However, it lacks the overlapping subproblems property: there are overlapping subproblems across the set of

different $\mathbf{W}$ matrices (e.g., two different matrices may have overlapping subproblems) but not within any single $\mathbf{W}$. Thus, we still have to iterate over 2^{K-1} $\mathbf{W}$'s refer subsection 2.4 for more details. **Structure of $\mathbf{W}$**: The matrix $\mathbf{W}$ has a special structure. Here we provide an example. Consider a three-mode tensor $\underline{\mathbf{X}}$ of dimensions $10 \times 10 \times 10$, with the third mode being the time mode. Suppose that the optimal level of aggregation for $\underline{\mathbf{Y}}$ is $K^* = 3$. In this case, $\mathbf{W}$ is of size 3×10 and an example of such matrix is:

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

This $\mathbf{W}$ aggregates first three slice of $\underline{\mathbf{X}}$ to form first slice of $\underline{\mathbf{Y}}$, next three to form the second slice and last four to form the third slice. No two $\mathbf{W}$ matrices will produce the same aggregation. They can have the same K^* but order of aggregation of slices will be different.

2.4 Solving *Trapped Under Ice* optimally is hard

Solving *Trapped Under Ice* optimally poses a number of hurdles. First and foremost, the hardness of the problem depends on the definition of function Q, and most reasonable and intuitive such definitions are very hard to optimize since they are non-differentiable, non-continuous, and not concave. So far, in the literature, to the best of our knowledge, there are only heuristics for this quality function. Even so, those heuristic functions can only be evaluated on a single already fully-aggregated tensor, not a partially aggregated version thereof. Thus, *Trapped Under Ice* can be only solved optimally via enumerating all admissible solutions and choosing the best. In order to conduct this enumeration, we need to calculate the cardinality of the set of all $\mathbf{W}$ for a given instance of the problem.

LEMMA 2.1. *For an instance of a problem with K initial slices, the cardinality of the set of all $\mathbf{W}$ is 2^{K-1}*

PROOF. To get K^* aggregated slices there are $\binom{K-1}{K^*-1}$ ways to choose each of them leading to a different $\mathbf{W}$. This is a number of ways that $K-1$ partition slots can be filled partitioned by K^*-1 blocks. In order to get the final number, we need to sum up over all potential K^*:

$$\sum_{K^*=0}^{K-1} \binom{K-1}{K^*} = 2^{K-1}$$

$\square$

Direct corollary of the above lemma is that solving optimally *Trapped Under Ice* requires calling the function Q $O\left(2^K\right)$ times, which is computationally intractable. There may be small room for improvement by exploiting special structure in the set of all $\mathbf{W}$, however, given discontinuities in our objective function Q, this is not be a feasible alternative either. In this paper we define proxy quality functions Q that lend themselves to partial evaluation on a partially aggregated solution, thus allowing for efficient algorithms Thus, in the next section we propose a fast greedy approach which locally optimizes different criteria quality.

3 PROPOSED METHOD

In this section, we propose our efficient and effective greedy algorithm called IceBreaker which takes a tensor $\underline{\mathbf{X}}$ as an input, which

---

[2]In practice, this is done over a small number of low ranks, since low-rank structure is desirable.

has been created directly from raw data, and has no exploitable structure. and returns a tensor $\underline{\mathbf{Y}}$ which maximizes the interpretable and exploitable structure. The basic idea behind IceBreaker is to make a linear pass on the mode for which the granularity is suboptimal, and using a number of intuitive and locally optimal criteria for goodness of structure (henceforth referred to as *utility functions*), we greedily decide whether a particular slice across that mode needs to be aggregated[3] into an existing slice or contains good-enough structure to stand on its own. IceBreaker can choose from a number of intuitive utility functions which are based on different definitions of good quality in matrices and graphs (in cases where we are dealing with underlying graph data).

3.1 The IceBreaker algorithm

Algorithm 1 gives a high level overview of IceBreaker . More specifically, the algorithm takes a three-mode tensor $\underline{\mathbf{X}}$ of dimension $I \times J \times K$ as an input and loops over all the K slices of tensor $\underline{\mathbf{X}}$. Two slices next to each other get aggregated into a single slice if a certain utility function *has stabilized*, i.e., if aggregating the two slices does not offer any additional utility (larger than a particular threshold), then the second slice should not be aggregated with the first, and should mark the beginning of a new slice.

Consider a three-mode tensor $\underline{\mathbf{X}}$ with time as third mode of dimension $I \times J \times K$ is ran through IceBreaker with a particular utility function. Our algorithm iterates over the time mode (K slices) and aggregates slices as decided by the utility function. IceBreaker is agnostic to utility function used. Let us consider a slice that has been aggregated into a single slice from indices i to $j - 1$ called previous slice and another aggregated slice from indices i to j called a candidate slice. Both previous and candidate slice are passed to utility function separately to obtain a value each called previous and current value respectively. These values are compared (line 5 in algorithm 1) to decide whether j^{th} slice is absorbed(line 6 in algorithm 1) into previous slice or previous slice has stabilized and entry is added in W to indicate which indices of tensor $\underline{\mathbf{X}}$ are aggregated together(line 8−9 in algorithm 1). Now j^{th} slice becomes the previous slice and aggregated slice of j and $j + 1$ become the candidate slice, the whole process is repeated until all the slices are exhausted.

Note that IceBreaker 's complexity is *linear* in terms of the slices K of the original tensor, and its overall complexity depends on the specific function used (which is called $O(K)$ times).

3.1.1 Utility functions: In the subsection, we summarize a number of intuitive utility functions that we are using in this work. This list is by no means exhaustive, and can be augmented by different functions (or function combinations) that capture different elements of what is good structure and can be informed by domain-specific insights.

(1) **Norm:** We use multiple norm types to find adaptive granularity of a tensor. For a given threshold, if rate of change of norm between previous and candidate slice is less than the threshold, candidate slice is not selected. Our assumption

---

[3]For the purposes of our work, we use matrix addition as aggregation of slice but this might not be the case and would depend on the problem domain. Other aggregation functions that can be used are OR, min, max, depending on the application domain (e.g., binary data).

Algorithm 1: IceBreaker

Input: Tensor $\underline{\mathbf{X}}$ of dimension $I \times J \times K$
Output: Tensor $\underline{\mathbf{Y}}$ of dimension $I \times J \times K_1$ and matrix $\mathbf{W}$ of size $K_1 \times K$
1: $i = 1; j = 2$
2: $previousValue = UtilityFunction(X(:,:,i))$
3: **while** $j \leq K$ **do**
4:     $currentValue = UtilityFunction(sum(X(:,:,i:j),3)$
5:     **if** $previousValue \lesseqgtr currentValue$ **then**
6:         $j = j+1$ {Aggregate Slice}
7:     **else**
8:         {Create a New Slice}
        Add a row in $\mathbf{W}$ with value as 1 for indices i to $j - 1$.
        {Update indices for next candidate slice}
9:         $i = j; j = j + 1$;
10:         $previousValue = UtilityFunction(X(:,:,i))$;
11:     **end if**
12: **end while**
13: $\underline{\mathbf{Y}} = \underline{\mathbf{X}} \times_3 \mathbf{W}$
14: return $\underline{\mathbf{Y}}$ and $\mathbf{W}$

---

in this case is no significant amount of information is being added to previous slice and is considered to have been stabilized. Matrix $\mathbf{W}$ is updated accordingly with indices of the previous slice (aggregated slices in previous slice). Otherwise the candidate slice is selected and the process continues until all the slices are exhausted. Different norms demonstrated in this work are Frobenius, 2-norm, and Infinity norm.

(2) **Matrix Rank:** In case of matrix rank, we focus on the 95% reconstruction rank, which is typically much lower than the full rank of the data, but captures the essence of the number of components within the slice. In this case, we consider previous slice to be stabilized if the matrix-rank of previous slice decreases by addition of new slice, no more slices are added and an entry in matrix $\mathbf{W}$ is added. We keep aggregating slices if the matrix-rank of the slice is increasing or remains constant.

(3) **Missing Value Prediction:** If a piece of data has good structure, when we hide a small random subset of the data, the remaining data can successfully reconstruct the hidden values, under a particular model that we have chosen. To this end, we employ a variant of matrix factorization based collaborative filtering [11] as a utility function to see how good is the aggregated matrix in predicting certain percent of missing values. This utility function takes percent of missing value as a parameter, hides those percent of non zeros values in the matrix. Our implementation of matrix factorization with Stochastic Gradient Descent tries to minimize the loss function: $\min_{\mathbf{U},\mathbf{V}} \sum_{i,j \in \Omega} \mathcal{RMSE}\left(\mathbf{A_{ij}} - \mathbf{U_{i,:}} \cdot \mathbf{V_{:,j}}\right)$ where $\mathbf{A}$ is a given slice, $\mathbf{U}, \mathbf{V}$ are factor matrices for a given rank (typically chosen using the same criterion as the matrix rank above), and Ω is the set of *observed* (i.e., non-missing) values. In order to create a balanced problem, since we are dealing with very sparse slices, we conduct *negative sampling* where we randomly sample as many zero entries as there are non-zeros in the slice, and this ends up being the Ω set of observed values.

(4) **Graph Properties:** There has been a significant amount of work in graph mining with respect to aggregation of temporal graph [18–20], taking inspiration from [18], we use the following functions:

(a) **Average degree:** Similar to norm, we consider a previous slice to stabilized if the rate of change of average degree between previous and candidate slice is less than the threshold.

(b) **Connected components:** We consider number of connected components that has more that one node. If the number of connected components remain the same we keep aggregating the slices and if the increases we consider it to stabilize.

4 EXPERIMENTAL EVALUATION

In this section we present a thorough evaluation of IceBreaker using variety of data, including synthetic and semi-synthetic data (where ground truth is known), and where we empirically evaluate our analysis using a number of criteria described in detail below. We implement our method in Matlab using tensor toolbox library [2]. For small variations in parameters to the utility functions we did not observe much deviations however we plan to investigate in greater detail the effects of various thresholds in an extended version.

4.1 Evaluation measures

When formulating the problem, we did not specify a quality function Q to be maximized, nor did we use such a function in our proposed method. The reason for that is because we reserve the use of different quality functions as a form of evaluation. In particular, we use the two following notions of quality:

- **CORCONDIA**: To evaluate the *interpretability* of the resulting tensor we employ Autoten [14] that given a tensor and some estimated tensor rank, returns a CORCONDIA score and low rank that provides best attainable tensor decomposition quality in a user-defined search space.

- **Community detection NMI**: We use NMI as a measure of predictive accuracy when the ground truth for the time-evolving data is available. To compute NMI score we perform CP decomposition on the resulting tensor with tensor rank provided by Autoten from above measure and then we use K-means clustering on the relevant factor matrices with number of communities in ground truth [6]. With that each node (row in the factor matrix) gets assigned to a single community we pass this result with the actual ground truth to NMI function to get NMI score.

We should note at this point that the two quality measures above are far from continuous and monotonic functions, thus we do not expect that our IceBreaker progresses the quality will monotonically increase. Thus, we calculate the quality for the final solution of IceBreaker , and we reserve investigating whether monotonic and well-behaved quality functions exist for future work.

4.2 Baseline methods

A naive way to find tensor $\underline{Y}$ can be by aggregating time mode based on some fixed intervals. If time granularity was in milliseconds, then combining one thousand slices to form slices of seconds granularity reducing the third dimension of tensor $\underline{X}$ from K to $K/1000$. This can be applied incrementally from seconds to minutes and so on to find a tensor which has some exploitable structure. We

compare the resulting tensor $\underline{Y}$ determined by IceBreaker against tensors constructed with fixed aggregations. For fixed aggregation we aggregate the temporal with window size of 10, 100 and 1000.

4.3 Performance for synthetic data with ground truth

Creating synthetic data: In order to create synthetic data we follow a two-step process: 1) we use an existing time-evolving tensor generator, proposed in [15], which creates a tensor that has a certain number of latent factors which appear, disappear, and reappear randomly over time, 2) subsequently, we simulate the *Trapped Under Ice* problem by taking every non-zero entry of that tensor and creating a new slice containing that entry (effectively setting K equal to the number of non-zero entries of the tensor). Furthermore, we modify the factor matrices of the first two modes, so that each row has only one non-zero entry, corresponding to a particular "cluster" it belongs to, thus enabling us to have ground truth labels for the first two modes of the data. We create two synthetic scenarios: 1) one where changes in the number of components happen on a fixed time scale (which is ideal for fixed aggregation) and 2) one where changes happen at randomized time windows, which is a more challenging case.

Results for synthetic data: In order to evaluate the performance of IceBreaker , we measure CORCONDIA and NMI on the two synthetic datasets, over 10 different runs. The leftmost[4] part of Figure 2 shows the results for the first dataset, where fixed aggregations outperform by a small margin the proposed method, a behavior which was expected, since the dataset has been created with a natural fixed size window of aggregation. The leftmost part of Figure 3 shows the results in the more challenging case, where the natural window of aggregation is randomized and variable. In this case, our proposed method works on par with the fixed aggregation methods. In both scenarios fixed aggregations performed well because of the nature of synthetic data which has very well defined latent structures repeated over period of time.
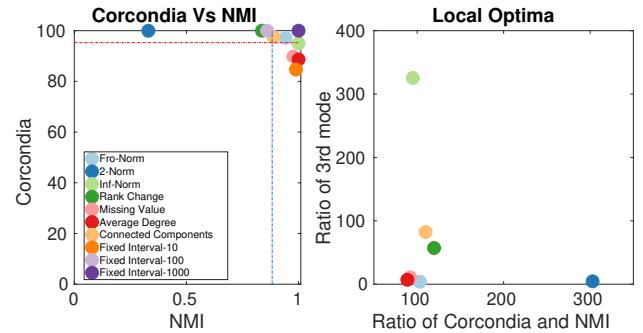

Figure 2: Corcondia vs NMI and Local Optima graph for Synthetic dataset-1

---

[4]Dotted line in the leftmost part of figures 2, 3, 4, and 5 are average value of the score of that experiment.
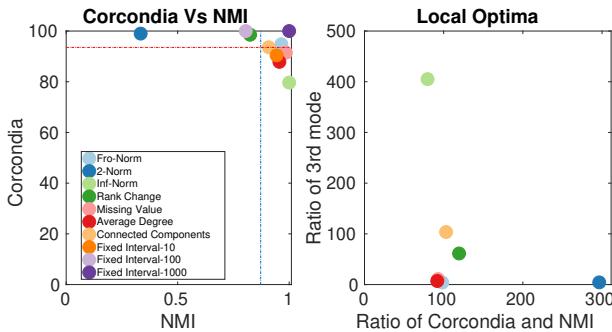
Figure 3: Corcondia vs NMI and Local Optima graph for Synthetic dataset-2

4.4 Performance for semi-synthetic data with ground truth

Creating semi-synthetic data: In addition to fully synthetic data, we also create semi-synthetic data, where the first two modes come from a real-world graph for which the ground truth labels are given, and the time mode is simulated according to the synthetic data generator used in [7]. The process followed is the same as in the synthetic case, where we take the non-zero elements and create a tensor with equal number of slices. The real-world graph we use for semi-synthetic data generation is the American Football used in [7]. Other dataset used is European Email from [12] where we create a tensor such that each non-zero entry corresponds to a slice.
Results for semi-synthetic data: The leftmost parts of Figures 4 and 5 show our results on the semi-synthetic data. We observe that, in general, IceBreaker outperforms the fixed aggregation methods, since IceBreaker results in high score of both CORCONDIA and NMI.
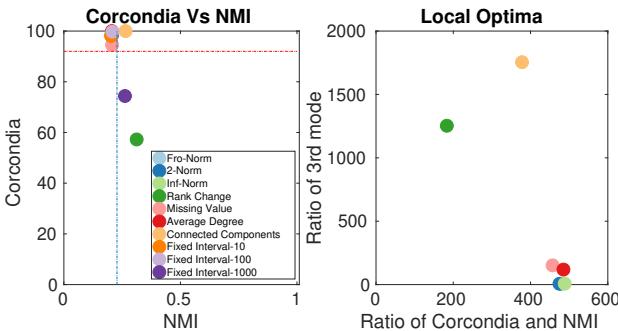

Figure 4: Corcondia vs NMI and Local Optima graph for Semi-synthetic American Football

4.5 Different local optima hint to multiple resolutions in time

IceBreaker , depending on what utility function it uses, converges to a different (locally) optimal solution, and all such solutions achieve roughly the same combined quality measure (ratio of COR-CONDIA and NMI in this case). Most interestingly, those different
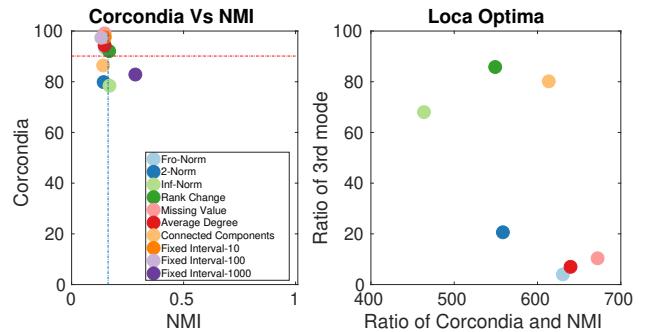

Figure 5: Corcondia vs NMI and Local Optima graph for Semi-synthetic European Email

local optima actually pertain to different levels of resolution for the aggregation. In the rightmost parts of Figures 2, 3, 4, and 5, we plot the degree of aggregation (measured by the ratio of $\frac{K}{K^*}$) versus the quality measure. We observe distinct clusters of solutions that achieve roughly the same quality, for vastly different aggregation levels, which points to the fact that those local optima reveal a multi-resolution view of the raw data.

5 RELATED WORK

To the best of our knowledge, this is the first attempt at formalizing and solving this problem, especially as it pertains in the tensor and multi-aspect data mining domain. Nevertheless, there has been significant amount of work on temporal aggregations in graphs [18–20] and in finding communities in temporal graphs [7]. The closest work to ours is [18], in which the authors looks at aggregating stream of temporal edges to produce sequence of structurally mature graphs based on a variety of network properties.

6 CONCLUSIONS

In this paper we are, to the best of our knowledge, the first to define and formalize the *Trapped Under Ice* problem in constructing a tensor from raw sparse data. We demonstrate that an optimal solution is intractable and subsequently proposed IceBreaker , a practical solution that is able to identify good tensor structure from raw data, and construct tensors from the same dataset that pertain to multiple resolutions. Our experiments demonstrate the merit of IceBreaker in discovering useful and high-quality structure, as well as providing tools to data analysts in automatically extracting multi-resolution patterns from raw multi-aspect data. In future work we will work towards extending IceBreaker in cases where more than one modes is *Trapped Under Ice* (naively one can apply IceBreaker to each mode sequentially, but this disregards joint variation across modes), and extend IceBreaker for higher-order tensors.

ACKNOWLEDGEMENTS

REFERENCES

[1] [n.d.]. Chicago Data Portal. https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2

[2] B.W. Bader and T.G. Kolda. 2007. Matlab tensor toolbox version 2.2. *Albuquerque, NM, USA: Sandia National Laboratories* (2007).

[3] Rasmus Bro and Henk AL Kiers. 2003. A new efficient method for determining the number of components in PARAFAC models. *Journal of chemometrics* 17, 5 (2003), 274–286.

[4] Evangelos E. Papalexakis, Leman Akoglu, and Dino Ienco. [n.d.]. Do more views of a graph help? community detection and clustering in multi-graphs. In *IEEE FUSION'13*.

[5] Evangelos E. Papalexakis, Nicholas D Sidiropoulos, and Rasmus Bro. 2013. From K-means to higher-way co-clustering: multilinear decomposition with sparse latent factors. *IEEE Transactions on Signal Processing* (2013).

[6] Sofia Fernandes, Hadi Fanaee-T, and João Gama. 2018. Dynamic graph summarization: a tensor decomposition approach. *Data Mining and Knowledge Discovery* 32, 5 (2018).

[7] Alexander Gorovits, Ekta Gujral, Evangelos E Papalexakis, and Petko Bogdanov. 2018. LARC: Learning Activity-Regularized Overlapping Communities Across Time. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[8] Ekta Gujral and Evangelos E Papalexakis. 2018. SMACD: Semi-supervised Multi-Aspect Community Detection. In *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM.

[9] R.A. Harshman. 1970. Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multimodal factor analysis. (1970).

[10] T.G. Kolda and B.W. Bader. 2009. Tensor decompositions and applications. *SIAM review* (2009).

[11] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.

[12] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. [n.d.]. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)* ([n. d.]).

[13] E.E. Papalexakis, C. Faloutsos, and N.D. Sidiropoulos. [n.d.]. Tensors for Data Mining and Data Fusion: Models, Applications, and Scalable Algorithms. *ACM Trans. on Intelligent Systems and Technology* ([n. d.]).

[14] Evangelos E Papalexakis. 2016. Automatic unsupervised tensor mining with quality assessment. In *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 711–719.

[15] Ravdeep Pasricha, Ekta Gujral, and Evangelos E Papalexakis. 2018. Identifying and alleviating concept drift in streaming tensor decomposition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 327–343.

[16] Nicholas D Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E Papalexakis, and Christos Faloutsos. [n.d.]. Tensor Decomposition for Signal Processing and Machine Learning. *IEEE Transactions on Signal Processing* ([n. d.]).

[17] Shaden Smith, Jee W. Choi, Jiajia Li, Richard Vuduc, Jongsoo Park, Xing Liu, and George Karypis. 2017. *FROSTT: The Formidable Repository of Open Sparse Tensors and Tools*. http://frostt.io/

[18] Sucheta Soundarajan, Acar Tamersoy, Elias B. Khalil, Tina Eliassi-Rad, Duen Horng Chau, Brian Gallagher, and Kevin Roundy. 2016. Generating Graph Snapshots from Streaming Edge Data. In *Proceedings of the 25th International Conference Companion on World Wide Web*. 2. https://doi.org/10.1145/2872518.2889398

[19] Rajmonda Sulo, Tanya Berger-Wolf, and Robert Grossman. 2010. Meaningful Selection of Temporal Resolution for Dynamic Networks. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs* (Washington, D.C.) *(MLG '10)*. ACM, 10.

[20] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S Yu. 2007. Graphscope: parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*.

[21] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* (2014).