On Compressing Weighted Time-evolving Graphs

Wei Liu[†], Andrey Kan[†], Jeffrey Chan[†], James Bailey[†], Christopher Leckie[†], Pei Jian[‡], and Ramamohanarao Kotagiri[†]
[†] Dept of Computing and Information Systems, The University of Melbourne, Australia
{wei.liu, akan, jeffrey.chan, baileyj, caleckie, kotagiri}@unimelb.edu.au
[‡] School of Computing Science, Simon Fraser University, Canada
jpei@cs.sfu.ca

ABSTRACT

Existing graph compression techniques mostly focus on static graphs. However for many practical graphs such as social networks the edge weights frequently change over time. This phenomenon raises the question of how to compress dynamic graphs while maintaining most of their intrinsic structural patterns at each time snapshot. In this paper we show that the encoding cost of a dynamic graph is proportional to the heterogeneity of a three dimensional tensor that represents the dynamic graph. We propose an effective algorithm that compresses a dynamic graph by reducing the heterogeneity of its tensor representation, and at the same time also maintains a maximum lossy compression error at any time stamp of the dynamic graph. The bounded compression error benefits compressed graphs in that they retain good approximations of the original edge weights, and hence properties of the original graph (such as shortest paths) are well preserved. To the best of our knowledge, this is the first work that compresses weighted dynamic graphs with bounded lossy compression error at any time snapshot of the graph.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining

Keywords

Dynamic graphs, graph compression, graph mining.

1. INTRODUCTION

An important intrinsic property of real graphs such as social networks is that the weights of their edges tend to continuously and irregularly change with time. The irregular changes of weights make the compression of dynamic graphs more challenging than that of static graphs due to the additional dimension of time. For example, the size of the publication network maintained in DBLP[1] keeps growing with time as new publications are being added to the repository. If one wants to construct a *who-published-with-whom* dynamic network, one would have to download a large amount of data from the server to uncover all the historical publications. An efficient method for compressing these dynamic graphs will not only save the cost of storage on the server side, but also reduce the communication cost needed for transmitting this information on the Internet.

Existing methods for compressing static graphs generally use two types of strategies: *(1)* removing edges to simplify the overall graph [4, 8], or *(2)* merging nodes that have similar properties (such as common neighbors) [5, 6]. While the existing methods compress a static graph from its "spatial" (nodes or edges) perspective, in this paper we propose to compress a dynamic graph from both the "spatial" and the "temporal" perspectives simultaneously.

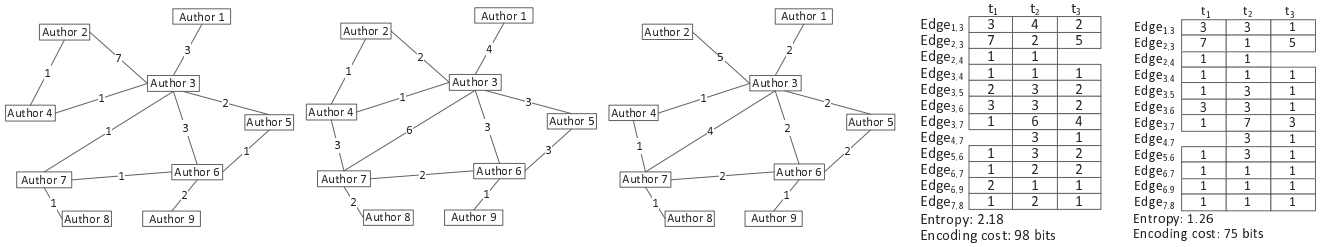
Static graphs are usually represented by their adjacency matrices. Similarly, we characterize a dynamic graph by a three dimensional (3D) tensor (i.e., a 3D array: $Vertices \times Vertices \times Time$), where an entry of this tensor is an edge weight at a certain time stamp. Since large dynamic graphs are mostly sparse, when using tensors to represent dynamic graphs, we only have to encode the *sparse version* of a tensor to encode the entire dynamic graph. The *sparse version* of a tensor is a set of *locations* and *values* of non-zero entries in the tensor. While the *locations* of non-zero entries can be efficiently encoded by (for example) run-length encoding, in this research we put our focus on reducing the cost of encoding the *values* of the tensor's non-zero entries.

For a small example, consider a co-authorship publication network that evolves in three time periods shown in Fig. 1(a) to 1(c), where edge weights are numbers of collaborated papers. The original cost of encoding this dynamic graph is shown in Fig. 1(d). For visualization purposes, in this figure we demonstrate the 3D $Vertices \times Vertices \times Time$ tensor by a 2D $Edges \times Time$ matrix. Our target is to merge certain subsets of weights across all dimensions of the tensor so that the overall encoding cost of the graph is minimized. Fig. 1(e) is an example of the re-weighted graph which has lower heterogeneity of edge weights and lower encoding cost.

Since the reduction of the encoding cost of a tensor is from the homogenization of subsets of weights, a major challenge of compressing dynamic graphs becomes how to select appropriate subsets of weights and unify them while maintaining desirable properties (such as the shortest paths) at each time snapshot. In this research we introduce the use of hierarchical clusters of edge weights to address the weight subset selection problem. The **main contributions** we make in this paper are as follows:

[1] `http://dblp.uni-trier.de`

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.



(a) Graph at time t_1. (b) Graph at time t_2. (c) Graph at time t_3. (d) Original weights. (e) Merged weights.

Figure 1: (a) to (c) show a synthetic co-authorship network evolving over three time periods. (d) and (e) give an example of the compression of this evolving graph by reducing the heterogeneity of weights across both "edge" and "time" dimensions. For visualization purposes, in this example we demonstrate the tensor representation of this dynamic graph by an $Edges \times Time$ matrix. The concrete approaches of computing the "entropy" and the "encoding cost" stated in (d) and (e) are introduced in Section 3.

- We propose to encode a dynamic graph by encoding its tensor representation, and compress the dynamic network by reducing the heterogeneity of the tensor.
- We design an effective compression algorithm (named `MaxErrComp`), which uses hierarchical clusters of edge weights to partition and merge weights across all dimensions of a tensor. This algorithm also guarantees a bound on the compression error of path weights at any time snapshot in the dynamic graph.
- We evaluate our method on two real networks, a co-authorship network and an email communication network, which demonstrates the advantage of our algorithm in both compressing dynamic graphs and preserving important aspects of their temporal properties.

2. PRELIMINARY DEFINITIONS

We define a weighted static graph by a triple $G = (V, E, w)$, where V is a set of vertices (nodes), $E \in V \times V$ is a set of edges, and $w(e) \rightarrow \mathbb{R}$ represents the set of non-negative weights assigned to all edges $e \in E$. We assign zero weights to edges that do not exist.

An essential property of a static graph is the connectivity between two nodes. The quantification of how closely two nodes are connected can be examined by the shortest path between them. We use the notation $u_i \overset{P}{\Rightarrow} u_j$ to represent a path from u_i and u_j (if the path exists). The weight of the shortest path (W_{SP}) among the two nodes u_i and u_j can be expressed as:

$$W_{SP}(u_i, u_j) = \begin{cases} \min_{u_i \overset{P}{\Rightarrow} u_j} \sum_e w(e), e \in P & \text{if } P \text{ exists} \\ +\infty & \text{otherwise} \end{cases} \quad (1)$$

We use the average of shortest paths over all pairs of connected nodes in a static graph to measure the connectivity of that graph. Denote by $\mathbb{P}$ the set of all shortest paths in a graph that exist (i.e., excluding paths of infinity weights defined in Eq. 1), the average shortest path weight ($AvgSP$) of a graph is defined by:

$$AvgSP(G) = \frac{1}{|\mathbb{P}|} \sum_{P \in \mathbb{P}, u \overset{P}{\Rightarrow} v} W_{SP}(u, v) \quad (2)$$

Now we give the definition of a dynamic graph:

Definition 1. *A time-evolving (aka dynamic) graph DG is a sequence of static graphs, $DG = \{G_1, G_2, ..., G_T\}$, where $G_t = (V, E, w(e,t))$, and $w(e,t)$ assigns a weight to an edge e ($e \in E$) at time stamp t ($1 \leq t \leq T$).*

Intuitively, when G_t is represented by its adjacency matrix of size $|V| \times |V|$, a concatenation of G_1 through G_T forms a tensor of size $|V| \times |V| \times |T|$. We denote the tensor of DG by $\mathcal{T}_{DG}$.

3. ENCODING DYNAMIC GRAPHS

The *locations* and *values* of non-zeros entries of $\mathcal{T}_{DG}$ respectively indicate the end-vertices and weights of all edges in DG. As a way of discretizing edge weights, we round values of these non-zeros entries to their closest integers (if they are not originally integers). These non-zeros entries are ordered by traversing along the time dimension first and then along the two vertex dimensions of the tensor. Similar to the principle of the run-length coding, the *locations* can be specified by the number of zeros in the tensor between adjacent non-zero entries, which makes the *location* information simply a sequence of integers (denoted by *Loc*). We concatenate *Loc* with the sequence of values to obtain a single string *LocVal*, and encode it by an entropy encoding method (e.g., arithmetic encoding [7]) as follows.

Let $Int(x)$ be a bit string that encodes a positive integer x, and $L(s)$ be the length of a bit string s. There are different possible implementations of $Int(x)$. We use the common variable length quantity (VLQ) format used in MIDI files, so $L(Int(x))$ is $8 \times \lceil \log_2(x+1)/7 \rceil$ bits. We first encode the tensor's dimensions, and then encode its entries, i.e., $DG = Int(|V|)Int(|T|)Meta(LocVal)LocVal$. The string $Meta(LocVal)$ contains all k unique integers x_i and their frequencies $freq_i$ ($1 \leq i \leq k$) from *LocVal*. Therefore $L(Meta(LocVal)) = \sum_{i=1}^{k} \big(L(Int(x_i)) + L(Int(freq_i))\big)$. The Shannon entropy estimates the lowest number of bits required for encoding a source string [1]. The entropy of *LocVal* is $H(LocVal) = \sum_{i=1}^{k} -\frac{freq_i}{|LocVal|} \log_2 \frac{freq_i}{|LocVal|}$. So in total, a dynamic graph DG in our representation requires $L(DG) = L(Int(|V|)) + L(Int(|T|)) + L(Meta(LocVal)) + H(LocVal)$ bits to be encoded. This graph encoding procedure is stated as "Encode($\mathcal{T}_{DG}$)" in the algorithm introduced in the next section.

Since the costs of the first and the second terms of $L(DG)$

Algorithm 1 `MaxErrComp`

Require: A dynamic graph $DG = \{G_1, G_2, ..., G_T\}$, and an error threshold ϵ.

1: Compute a hierarchical cluster tree *ClusterTree* on non-zero entries of DG's tensor representation $\mathcal{T}_{DG}$;
2: Initialize *cutoff* $\leftarrow 0$;
3: Compute original average shortest path weights: $OldSP_t \leftarrow AvgSP(G_t), \forall t \in [1, T]$;
4: **while** *cutoff* has not reached the root of *ClusterTree* **do**
5: Increase *cutoff* by 1;
6: Obtain weight clusters from *ClusterTree* by applying *cutoff*;
7: **for** each cluster of weights CL **do**
8: $CL_{old} \leftarrow CL$;
9: $w_i \leftarrow \text{round}(\text{mean}(CL)), \forall w_i \in CL$;
10: $NewSP_t \leftarrow AvgSP(G_t), \forall t \in [1, T]$;
11: **if** $\max_{1 \le t \le T} (\frac{|OldSP_t - NewSP_t)|}{OldSP_t}) > \epsilon$ **then**
12: Restore weights: $w_i \leftarrow (CL_{old})_i, \forall w_i \in CL$;
13: **return** Encode($\mathcal{T}_{DG}$);
14: **end if**
15: **end for**
16: **end while**
17: **return** Encode($\mathcal{T}_{DG}$);

are fixed, our method aims to reduce the cost of the third term $L(Meta)$ by generating fewer unique weight integers, and the fourth term $H(LocVal)$ by decreasing the heterogeneity of the bit string.

The heterogeneity of a tensor is straightforwardly minimized when all its non-zero entries are set to a constant value across different edges. From this perspective, the "average" of all weights of a dynamic graph, which we call the "average graph", provides a lower bound for the encoding cost of the dynamic graph. However, simply setting all weights to their average value loses the inherent variances and dynamics in all snapshots of the graph, which is detrimental to the analysis of the temporal behavior of graphs. Therefore, in the next section we propose an effective algorithm that finds appropriate tradeoffs between the original graph and the average graph, which reduces the encoding cost and preserves desirable temporal properties simultaneously.

4. THE `MAXERRCOMP` ALGORITHM

Our dynamic graph compression algorithm "merges" subsets of edge weights by assigning a common (average) weight to them. The subsets of weights being merged are selected using agglomerative hierarchical cluster trees [2] built on non-zero entries of the dynamic graph's tensor representation. We note that we only need to build the cluster tree on *unique* weight values (instead of all weight values), since identical weights are always clustered together without distance checking. In Alg. 1 we present the `MaxErrComp` algorithm which reduces the cost of encoding a dynamic graph, and at the same time provides error-bounded changes on average shortest path weights for all time snapshots of the dynamic graph.

The `MaxErrComp` algorithm runs in a greedy fashion, which gradually increases the clustering cutoff value to obtain clusters from the hierarchical cluster tree. The cutoff value is the maximum difference allowed among the weights within each cluster. The cost of encoding the dynamic graph is reduced by averaging the weights that belong to the same cluster (line 9 in Alg. 1). Similar to the original weights, we round new weights to their nearest integers. The algorithm checks the max error of average shortest path weights among all snapshots of the dynamic graphs at each iteration. As soon as the compression error reaches the threshold ϵ, the algorithm terminates the iterations and the weights that are changed in the last iteration roll back to their previous values (lines 11 to 14), so the compression error is always bounded by ϵ. If the dynamic graph does not change much across time or if the threshold ϵ is set to a high value, the program would run to the last line of Alg. 1 and potentially produces the average graph whose error could be lower than ϵ.

It can be observed from line 11 of Alg. 1 that our `MaxErrComp` algorithm can be *application-generic*: by changing the termination condition to other properties of graphs (such as changes to communities of vertices), one can straightforwardly generalize `MaxErrComp` to preserve other types of properties of a dynamic graph.

Compared to existing methods that compress static graphs (i.e., compress one snapshot at a time in a dynamic graph), a major advantage of `MaxErrComp` is that it takes into account both the spatial and the temporal information of a dynamic graph, and thus the cost of encoding all dimensions of the dynamic graph's tensor representation can be reduced.

4.1 A Baseline Algorithm

We present a baseline method (`MaxErrRandom`) that averages random partitions of edge weights. The `MaxErrRandom` algorithm is similar to the `MaxErrComp` algorithm in terms of their terminating conditions. The difference is that instead of choosing weights from cluster trees, `MaxErrRandom` randomly selects a partition of weights, and sets all weights in each partition to their average value. The number of partitions used in `MaxErrRandom` decreases by 1 at each iteration from the total number of edges until it reaches 1, or until the terminating condition is met.

5. EXPERIMENTS AND ANALYSIS

In our evaluations we include a recently proposed static-graph compression method [6], which we denote by `StatComp`. Since `StatComp` does not provide a strategy for compressing dynamic graphs, we measure the encoding costs of a dynamic graph compressed by `StatComp` using the cost of encoding all of its snapshots that are separately compressed by `StatComp`. In this way, we can evaluate the scope for compressing temporal information of dynamic graphs.

5.1 Data sets

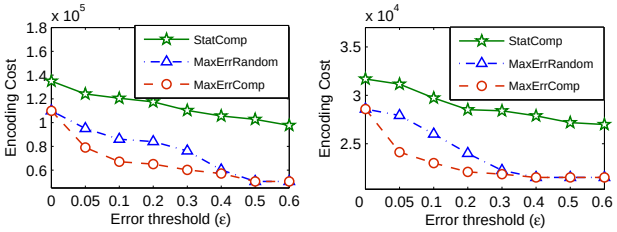
We use two types of dynamic graphs to validate our algorithm. The first is a co-authorship network extracted from the DBLP bibliography. We select co-authors who have data mining publications from year 2000 to 2011. We select data mining journals and conferences from the venues whose full book or proceeding titles contain the phrases "data mining" or "knowledge discovery", and only include authors who have at least 5 publications through the 12 years period. The second data set is extracted from the Enron email network [3], which provides email connections from senior executives to other employees in the Enron company. Statistics of these two datasets are shown in Table 1.

5.2 Results

We first evaluate our method under different settings of error thresholds. Comparisons on the performance of the `Max-`

Table 1: Statistics of data sets.

Data set	|V|	|E|	|T|	Type
DBLP	3282	55756	12 (years)	Undirected
Enron	2359	99742	28 (months)	Directed



(a) On DBLP data (b) On Enron data

Figure 2: Comparisons on the degree of compression, in terms of the reduction of encoding costs. Values correspond to "$\epsilon = 0$" in the *x-axis* are the encoding costs before lossy compression, and those correspond to "$\epsilon > 0$" are the encoding costs after compression.

Table 2: Comparisons on the quality of compression, in terms of the error on compressed average shortest path weight, at each snapshot of the DBLP data set.

Time	Compression error on path weights								
	$cr = .05$			$cr = 0.1$			$cr = 0.2$		
	M1	M2	M3	M1	M2	M3	M1	M2	M3
2000	0.13	0.67	0.39	1.84	7.89	6.67	10.53	25.4	18.9
2001	0.12	0.89	0.67	1.56	8.32	8.92	9.89	27.6	14.9
2002	0.13	0.78	1.42	2.79	9.64	3.29	8.56	18.9	15.6
2003	0.16	0.96	0.58	2.03	7.77	9.87	6.59	16.7	14.7
2004	0.17	0.68	0.67	1.49	9.93	4.69	8.97	15.2	19.8
2005	0.21	1.09	0.86	2.83	10.64	9.91	12.54	28.8	21.9
2006	0.25	1.18	1.54	1.87	9.36	11.20	17.71	29.4	24.1
2007	0.13	0.66	0.94	1.57	8.54	6.35	14.45	22.1	22.8
2008	0.14	0.89	1.88	2.36	11.81	10.33	13.36	27.6	19.3
2009	0.28	0.74	0.42	2.68	14.52	7.89	9.46	18.7	16.5
2010	0.14	0.85	0.52	1.59	8.97	6.29	7.27	15.5	18.2
2011	0.16	0.55	0.70	3.67	12.8	7.74	8.98	17.1	15.9
t-tests on $cr = .05$	Base	4E-8	3E-4						
t-tests on $cr = 0.1$				Base	5E-9	7E-6			
t-tests on $cr = 0.2$							Base	4E-7	1E-8

`ErrComp`, `MaxErrRandom` and `StatComp` methods are shown in Fig. 2. The scenario of "$\epsilon = 0$" on this figure represents the encoding cost of the original dynamic graph before lossy compression. We can observe that on a fixed error threshold, the cost of encoding a dynamic graph using `MaxErrComp` is always lower than that using `MaxErrRandom` before they reach a common stationary point (i.e., the average graph). This phenomenon demonstrates that on the same error threshold, the subsets of weights selected by hierarchical cluster trees are more effective in reducing the encoding costs of dynamic graphs. We can also observe that the encoding costs of the `StatComp` method are generally higher than those of the `MaxErrComp` and `MaxErrRandom` methods, and the compression rates (i.e., $\frac{\text{Cost on } \epsilon > 0}{\text{Cost on } \epsilon = 0}$) of `StatComp` are also the lowest among the three methods.

It is also important to examine whether the temporal properties of each time snapshot of the dynamic graphs are preserved after compression. For this purpose, we change the termination condition of Alg. 1 (line 12) to a compression ratio cr ($0 \leq cr \leq 1$): $\frac{Encode(\mathcal{T}_{CompressedDG})}{Encode(\mathcal{T}_{OriginalDG})} > cr$. Then we compare the compression error made on the average shortest path weight in every snapshot of the dynamic network, where the error is defined by $|1 - \frac{\text{Compressed avg. path weight}}{\text{Original avg. path weight}}|$. As shown in Table 2, the shortest path weights preserved by `MaxErrComp` are the most accurate among the three methods at all times. Due to space limits, in Table 2 we only present comparions of methods on the DBLP data set. Results obtained from the Enron data set lead to the same conclusions to those of the DBLP data. To confirm the superiority of `MaxErrComp` on compression quality, we apply paired t-tests on the errors made by the three compression methods, under the null hypothesis that their errors are not significantly different. From the low p-values in the bottom three rows of Table 2, we can confidently reject the null hypothesis. This suggests that under the same compression ratio, the compression quality preserved by `MaxErrComp` is significantly better than the other two methods.

6. CONCLUSIONS AND FUTURE WORK

We propose to encode a dynamic graph by encoding its tensor representation, and compress the dynamic graph by reducing the heterogeneity of the tensor. We have designed a compression algorithm that decreases the heterogeneity of the tensor entries by using hierarchical cluster trees built on the time-stamped edge weights. This algorithm is highly generic and can be easily generalized to preserve various properties of the original dynamic graph while compressing it. We have exemplified weights of shortest paths in all time snapshots of a dynamic graph as a desired property to maintain, and with this property we have empirically tested our method on the compression of a co-authorship network and an email communication network.

In future we would like to apply our method to numerical edge weights by using differential entropy.

Acknowledgements

This research was supported under Australian Research Council's Discovery Projects funding scheme (DP110102621).

7. REFERENCES

[1] T. Cover and J. Thomas. *Elements of Information.* John Wiley & Sons, New Jersey, 2006.

[2] A. Fernández and S. Gómez. Solving non-uniqueness in agglomerative hierarchical clustering. *Journal of Classification*, 25(1):43–65, 2008.

[3] B. Klimt and Y. Yang. Introducing the enron corpus. In *Proc of CEAS 2004.*

[4] H. Maserrat and J. Pei. Neighbor query friendly compression of social networks. In *Proc of KDD 2010.*

[5] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *Proc of SIGMOD 2008.*

[6] H. Toivonen, F. Zhou, A. Hartikainen, and A. Hinkka. Compression of weighted graphs. In *Proc of KDD 2011.*

[7] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Commun. ACM*, 30(6):520–540, 1987.

[8] F. Zhou, S. Malher, and H. Toivonen. Network simplification with minimal loss of connectivity. In *Proc of ICDM 2010.*