2010 International Conference on Advances in Social Networks Analysis and Mining

Using Vector Clocks to Visualize Communication Flow

Martin Harrigan
Complex & Adaptive Systems Laboratory,
University College Dublin, Ireland
Email: martin.harrigan@ucd.ie

***Abstract*—Given a dataset comprising a temporal sequence of communications between actors, how can we visualize the 'flow' of communication over time? Current practice transforms the dataset into a dynamic graph – vertices represent the actors and directed edges represent the communications. The directed edges are added and removed over time. There are then several approaches to visualizing dynamic graphs that optimize aesthetic criteria, most producing animated node-link diagrams. However, dynamic graphs are not the only way to model this problem. One alternative from the field of distributed computing is vector clocks. Recent work employed vector clocks to analyze communication flow in social networks with much effect, arguing that they provide new insights into the problem. In this paper, we use vector clocks as a basis for *visualizing* communication flow. We show that communication patterns, e.g., random, partitioned and core-periphery, are easily discernible in the resulting visualizations. We also argue that, in the cases where vector clocks are used to analyze communication flow, it is most natural to base the accompanying visualizations on vector clocks also.**

I. Introduction

A conventional metaphor for communication is that of messages 'flowing through conduits' [1]. This metaphor is particularly apt for electronic communication since the conduits, e.g., copper wires, TCP connections, email addresses (username@node.domain.ext.country-code) and hash-tags, are explicit. Furthermore, the temporal sequencing of communications through the conduits is often recorded allowing for analyses of communication flow. The metaphor extends nicely to other aspects of communication: starting at various places and times, communication flows through conduits in many directions at differing speeds, interfering with other communications when conduits meet, and growing and shrinking over time. Recent analyses of email and instant messaging datasets strengthen this metaphor when they discuss the 'topology of email networks and instant messaging systems' [2, 3], a 'flux of activation that concentrates and clusters into structures' [4], 'slow and fast conduits' [5] and 'transatlantic flows of communication' [6].

Communication flowing through conduits is a tempting metaphor to use when visualizing communication systems. However, it is important to distinguish between visualizations of the conduit system and visualizations of the communication flow. The conduit system (or at least an inference of it) can be visualized by constructing a weighted directed graph where the vertices represent the actors (conduit end points) and the weighted directed edges represent the conduits. The weight of an edge is determined by the number of communications flowing through the corresponding conduit within a given time-slice. We can vary the weights of the edges as we progress through the time-slices removing any edge whose weight becomes zero and creating any edge that does not already exist whose weight becomes greater than zero. There are then several approaches to visualizing dynamic graphs that optimize aesthetic criteria or minimize energy functions [7, 8, 9, 10].

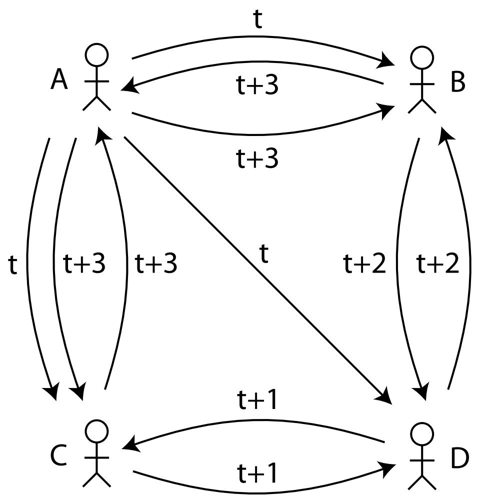


Figure 1. Four actors plan a meeting. Each communication is represented by a directed edge that is labeled with the time at which the communication was sent.

However, dynamic graphs are not the only way to model this problem. One alternative from the field of distributed computing is vector clocks [11, 12, 13]. A vector clock represents, for a given actor and time, when the actor last received a communication, either directly or indirectly, from every other actor at that time. Suppose, for example, that four actors, A, B, C and D, are planning a meeting (see Fig. 1). The planning starts, at time t, with A suggesting to everyone that they meet at the university. Later at time $t+1$, D discusses alternatives with C and they decide to meet at the library instead. Later still at time $t+2$, D discusses alternatives with B and they decide to meet at the park instead. D also informs B that he had previously agreed with C to meet at the library. When A checks with everyone again at time $t+3$ to make sure that they still agree to meet at the university, he gets mixed responses: C claims to have settled on the library with D; B claims to have settled on the park with D; and D cannot be reached. However, B knows that he agreed on the park with D after C agreed on the library with D. Therefore, A, B and C can safely meet D at the park without any further communication (assuming there was no confusion regarding the time of the meeting). This decision relied on B's knowledge of C's previous agreement with D. Vector clocks formalize this notion by maintaining for each actor the time of their most recent communication, either directly or indirectly, with every other actor. If we consider the communication metaphor once more, we can imagine each communication or wave pushing actors to different coordinates. Actors that reside at the same or similar coordinates have been pushed by the same or 'communicatively equivalent' waves. The distance a wave pushes an actor depends on the actor's previous coordinates and the 'communicative content' of the wave.

We use vector clocks as a basis for visualizing communication flow. We consider each vector clock to represent a point in a high-dimensional space. Each communication updates the vector clock of the actor receiving the communication. If the communication is two-way, then the vector clocks for the actors at either end of the communication are updated. We can compute the distance between two vector clocks using an appropriate measure. For each time-slice we construct a dissimilarity matrix representing the distances between all pairs of vectors clocks. We use multidimensional scaling (MDS) [14, 15] to produce an 2-dimensional visualization of the data points. We show that communication patterns, e.g., random, partitioned and core-periphery, are easily discernible in the resulting visualizations. Although we are primarily interested in visualizing communication flow between actors, the methodology is applicable to a wide range of discrete dynamical systems.

This paper is organized as follows. In Sect. II we review some work relating to vector clocks and temporal statistics of communication datasets. In Sect. III we review some definitions and an incremental algorithm for updating vector clocks for all vertices at all times. In Sect. IV we present our methodology. In Sect. V we apply our methodology to generated datasets and the VAST 2008 Challenge mobile phone call dataset [16]. We show that a variety of communication patterns are evident in the corresponding visualizations. Finally, we conclude in Sect. VI and discuss some future research.

II. Related Work

Vector clocks [12, 13] were proposed in the field of distributed computing to aid with the causal ordering of events. The ordering of events produced by a set of processes within a distributed system can be represented by a total ordering. Such an ordering requires that each event be authoritatively time-stamped. The time-stamp must be created by a central process or the processes must keep their individual clocks synchronized. However, if the events are the result of communication or message-passing only, then a partial ordering of events is more appropriate. The partial ordering can be derived from vector clocks without the need for a central process.

Kossinets et al. [5] employed vector clocks when analyzing social networks. They formulated the notion of temporal distance in a social network by measuring the minimum time required for information to spread from one actor to another. By examining the sequence of time-stamped communications in a social network, they derive potential conduits for communication flow. They argue that vector clocks provide greater insight into communication flow than, say, directed weighted graphs. They show, for example, that temporal distance can be used to discover 'network backbones'. Network backbones are sparse and try to balance two objectives – they include edges that connect distant parts of the network (long range edges) and edges that are local but in frequent use (short range edges that are highly embedded). Our contribution uses vector clocks as a basis for *visualizing* communication flow.

There are a variety of studies involving temporal statistics of communication datasets [17, 18, 19, 4, 20, 21]. However, an appropriate method of visualizing these datasets is lacking.

III. Definitions

During a time interval $[0,T]$, we have a set of vertices V and a set of *time-attributed directed edges* E. A time-attributed directed edge, or simply an edge, is an ordered triple $(u,v,t) \in E$ where $u,v \in V$ are the source and target respectively and t is the time attribute.

The instantaneous graph $G_t = (V, E_t)$ at time-slice t is the graph with vertex set V and edge set $E_t = \{(u,v) | (u,v,t) \in E\}$. The aggregated graph $G_{t_i,t_j} = (V, E_{t_i,t_j})$ between the times t_i and t_j inclusive where $t_i \le t_j$ is the graph with vertex set V and edge set $E_{t_i,t_j} = E_{t_i} \cup \ldots \cup E_{t_j}$. Both the instantaneous and aggregated graphs are directed multigraphs, i.e., there may be multiple directed edges between any pair of vertices. We note that $G_t = G_{t,t}$ and $G_{0,T} = (V,E)$. We can attach a time attribute to each directed edge in an aggregated graph to indicate the E_t it originated from. A time-respecting directed path [18] in an aggregated graph is a directed path in which the time-attributes along the directed edges ordered from the start to the end of the directed path are non-decreasing.

The vector clock $\phi_{v,t}$ of a vertex u at time-slice t with coordinates $\phi_{v,t}[u], u \in V$ is defined by:

$$\phi_{v,t}[u] = \begin{cases} t' & \text{if } u \rightsquigarrow v \text{ in } G_{t',t} \text{ and } t' \text{ is} \\ & \text{the largest such value} \\ \perp & \text{if } u \not\rightsquigarrow v \text{ in } G_{0,t} \end{cases} \tag{1}$$

where $u \rightsquigarrow v$ and $u \not\rightsquigarrow v$ indicate the presence and absence of a time-respecting directed path from u to v respectively. $\perp$ represents an undefined value and is treated as smaller than any other number in the comparisons below. In the context of a communicating system, $\phi_{v,t}[u]$ is the most recent time-slice on or before time-slice t at which v last received a communication either directly or indirectly from u.

An incremental algorithm to compute the vector clocks for all vertices at all times [12, 13] proceeds as follows. We initialize all of the vector clocks according to (1) by setting:

$$\phi_{w,0}[x] \leftarrow \begin{cases} 0 & \text{if } w = x \\ \perp & \text{otherwise} \end{cases} \tag{2}$$

for all $w, x \in V$. We process the edges in the order of their time attributes. The case of two or more edges having the same time attribute was not considered explicitly in earlier descriptions of the algorithm but will be considered in Sect. IV-A. For now we assume that all edges have distinct time attributes. We process an edge (u, v, t) by updating v's vector clock as follows:

$$\phi_{v,t}[x] \leftarrow \max(\phi_{u,t}[x], \phi_{v,t}[x]). \tag{3}$$

for all $x \in V$. Once we have processed all edges directed towards some vertex v at time-slice t according to (2) and (3), $\phi_{v,t}$ satisfies (1) for those values of v and t and all values of $u \in V$. Then, if $t \neq 0$ and t is different to the time attribute of the previously processed edge, we advance all of the vector clocks by setting:

$$\phi_{w,t}[x] \leftarrow \begin{cases} \phi_{w,t-1}[x] + 1 & \text{if } w = x \\ \phi_{w,t-1}[x] & \text{otherwise} \end{cases} \tag{4}$$

for all $w, x \in V$. If we only need to store the vector clocks for the current time-slice t then we can safely overwrite $\phi_{w,t-1}$ with $\phi_{w,t}$.

IV. Methodology

In this section we describe our methodology of producing a sequence of 2-dimensional visualizations, one per time-slice, to illustrate communication flow. Briefly, we use the algorithm described in Sect. III to update the vector clocks at each time-slice. We consider each vector clock to represent a point in $|V|$-dimensional space by replacing the $\perp$ values with 0. Each communication from a source to a target updates the vector clock of the target and moves the corresponding vertex closer to that of the source. Vertices move towards each other as their vector clocks become 'synchronized' and away from each other as their vector clocks become 'unsynchronized'. We compute the distances between all pairs of vector clocks using a distance metric and construct a distance or dissimilarity matrix. We produce a 2-dimensional visualization at each time-slice using an MDS algorithm.

We provide pseudo-code, `visualizeVectorClocks`, in Algorithm 1. In lines 2–4 we initialize the vector clocks according to (2). We use an outer loop to consider each time-slice t and an inner loop to consider each edge e present at time-slice t. In line 26 we compute the dissimilarity matrix and in line 27 we compute the coordinates for time-slice t using an MDS algorithm. We will discuss the specifics of these computations at the end of this section. In lines 28–31 we advance the vector clocks according to (4). We update the vector clocks in a process that is somewhat more complicated than (3) suggests. The reason for this is that two or more edges may have the same time attribute. We consider this case and explain the remaining lines of the algorithm below.

Algorithm 1: `visualizeVectorClocks`

```
   Input: $G_0, \ldots, G_T$ (instantaneous graphs)
   Output: $C_0, \ldots, C_T$ (2-dimensional coordinates)
1  $t \leftarrow 0$;
2  foreach $u \in V$ do
3      $\phi_{u,t} \leftarrow [0, \ldots, 0]$;
4      $\phi_{u,t}[u] \leftarrow 1$;
5  while $t \leq T$ do
6      $\phi_t \leftarrow [0, \ldots, 0]$;
7      $D_t \leftarrow$ SCC decomposition of $G_t$;
8      $T_t \leftarrow$ topological ordering of $D_t$;
9      foreach $e \in edges(D_t)$ in an order derived from $T_t$
       do
10         if $p$ has not been initialized then
11             $p \leftarrow target(e)$
12         if $p \neq target(e)$ then
13             if $p$ is a SCC then
14                 foreach $u \in vertices(SCC)$ do
15                     $\phi_t \leftarrow \max(\phi_t, \phi_{u,t})$
16                 foreach $u \in vertices(SCC)$ do
17                     $\phi_{u,t} \leftarrow \phi_t$
18             else
19                 $\phi_{p,t} \leftarrow \max(\phi_{p,t}, \phi_t)$;
20             $\phi_t \leftarrow [0, \ldots, 0]$;
21             $p \leftarrow target(e)$;
22         $u \leftarrow source(e)$;
23         if $u$ is a SCC then
24             $u \leftarrow$ any element of $vertices(u)$;
25         $\phi_t \leftarrow \max(\phi_t, \phi_{u,t})$;
26     $M_t \leftarrow$ dissimilarity matrix;
27     $C_t \leftarrow$
       coordinates computed by MDS algorithm for $M_t$;
28     $t \leftarrow t + 1$;
29     foreach $u \in V$ do
30         $\phi_{u,t} \leftarrow \phi_{u,t-1}$;
31         $\phi_{u,t}[u] \leftarrow \phi_{u,t}[u] + 1$;
32 return $C_0, \ldots, C_T$;
```

A. Vector Clock Updates

In Sect. III we reviewed an incremental algorithm to compute the vector clocks for all vertices at all times [12, 13]. We noted that the case of two or more edges having the same time attribute was not considered explicitly in earlier descriptions of the algorithm. The granularity of the time interval $[0, T]$ may be too coarse to determine the actual ordering of the communications. We consider this case now. Suppose we need to process $e_1 = (u_1, v_1, t)$ and $e_2 = (u_2, v_2, t)$. If u_1, v_1, u_2 and v_2 are distinct (see Fig. 2(a)) and there are no other edges

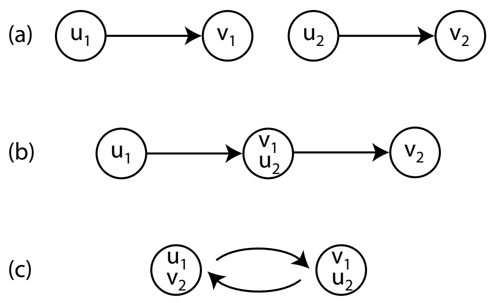


Figure 2. When two or more edges with the same time attribute form a directed path, the order in which we update the vector clocks becomes important.

with time attribute t then we can safely update the vector clocks $\phi_{v_1,t}$ and $\phi_{v_2,t}$ according to (3) in any order.

However, suppose $v_1 = u_2$ and $u_1 \neq v_2$ (see Fig. 2(b)). We identify two strategies: we can *optimistically* update the vector clocks $\phi_{v_1,t}$ and $\phi_{v_2,t}$ in that order; or, we can *pessimistically* update the vector clocks in the reverse order. After the update, $\phi_{v_2,t}$ differs in the two cases. Since vector clocks are already somewhat optimistic in the sense that the vector clock of a target has all of its elements increased to at least the values of the corresponding elements of the source, we employ the optimistic strategy. If we assume for the moment that the instantaneous graph G_t is a directed acyclic graph (DAG), that is, it does not contain any directed cycles, then the optimistic strategy requires that we update the vector clocks in an order determined by a *topological ordering* of the vertices of G_t. A topological ordering of a DAG is a total ordering of its vertices such that for every edge (u, v), $u < v$. In this way, the vector clock of a vertex will not be updated until the vector clocks of all vertices from which it could have received a communication, either directly or indirectly, are updated first. We note that in the pessimistic strategy we update the vector clocks in the reverse order.

Now, suppose $u_1 = v_2$ and $v_1 = u_2$. It is no longer clear which vector clock we need to update first. More generally, if G_t is not a DAG but a general directed graph, we first need to identify the *strongly connected components* of G_t. A strongly connected component (SCC) of a directed graph is a subgraph that is maximal with respect to edge inclusion and has a directed path between every pair of vertices. If $u_1 = v_2$ and $v_1 = u_2$ and there are no other edges with time attribute t then e_1 and e_2 form a SCC in G_t. We again use the optimistic strategy. We update the vector clock of each vertex in a SCC to be the 'supremum' vector clock, i.e., the vector clock whose elements are the element-wise maximum of the vector clocks of all of the vertices in the SCC. We substitute a single dummy vertex for the set of vertices in each SCC and set its vector clock to be the supremum vector clock for that SCC. We also add a directed loop to each of these dummy vertices to indicate that they are involved in a communication during time-slice t. This is the SCC decomposition of G_t. We process the remaining vertices for time-slice t using a topological ordering as above (ignoring the directed loops). At the end of time-slice t we substitute each dummy vertex with its original set of vertices. If the vector clock of a dummy vertex has been updated, we update the vector clocks of the original set of vertices similarly after the substitution.

If we consider again the pseudo-code for `visualize-VectorClocks`, line 7 computes the SCC decomposition of G_t. In line 8 we compute a topological ordering of this decomposition. Finally, in lines 12–21 we update the vector clocks of the target vertices according to (3) and the strategy above. ϕ_t represents a 'tentative' vector clock when computing the element-wise maximum of several vector clocks.

B. Dissimilarities and Multidimensional Scaling

In our methodology we compute a dissimilarity matrix by computing the distance between each pair of vector clocks using the Euclidean distance metric (`visualizeVector-Clocks`, line 26). When computing the distance between two vector clocks, we replace the $\perp$ values with 0. However, this may not be the most appropriate way of handling undefined values. This is also an important problem in the fields of case-based reasoning and machine learning where comparisons are made between partial problem descriptions and partial feature descriptions respectively. Bogaerts and Leake [22] and Mingers [23] devise a number of distance metrics for this problem. For the purposes of our visualizations (see Sect. V), the Euclidean distance metric suffices but we note that the distance metrics for partial descriptions and other measures, e.g., Pearson Correlation, are also applicable.

Once we have computed a dissimilarity matrix we need to visualize the (dis)similarities. Multidimensional scaling (MDS) is a collection of statistical techniques that are particularly useful for the geometric representation of objects [14]. They use measures of similarity (or dissimilarity) between objects in a given set to produce coordinates for the objects in such a way that similar objects are placed close together and dissimilar objects are placed far apart. In our methodology, we use a publicly available implementation of an MDS algorithm [24, 15] (`visualizeVectorClocks`, line 27). We note that even though the vector clocks are maintained between time-slices, the coordinates are computed for a given time-slice from scratch and are not based on the corresponding coordinates from the previous time-slice. This results in poor stability between successive layouts and is a drawback of our current implementation. However, an implementation of a dynamic MDS algorithm will solve the problem.

V. Datasets and Results

To support our methodology, we applied `visualize-VectorClocks` to five datasets. We generated four artificial datasets to illustrate the effect of underlying communication patterns on the resulting visualizations and we derived the fifth dataset from the mobile phone call data from the VAST 2008 Challenge [16].

A. The Datasets

We generated four datasets comprising temporal sequences of communications between 100 actors during a time interval $[0, 99]$. Each dataset had a distinct underlying communication pattern. The datasets were generated by fixing the set of possible communications and then selecting a communication from the set of possible communications at time-slice t with probability $p = 0.005$. The datasets were characterized by their sets of possible communications:

- **DS1**: All communications were possible.
- **DS2**: Communications were possible between every pair of actors in only one direction such that there were no directed cycles of communication.
- **DS3**: The actors were partitioned into four equal subsets and all intra-subset communications were possible.
- **DS4**: The actors were partitioned as in **DS3** and intra-subset communications were possible between every pair of actors in only one direction such that there were no directed cycles of communication.

We note that, in terms of sets of possible communications, **DS4** $\subset$ **DS2** $\subset$ **DS1** and **DS4** $\subset$ **DS3** $\subset$ **DS1**. The **VAST 2008** dataset [16] comprises mobile phone call records over a 10 day period between 400 unique mobile phones. We set each time-slice equal to one hour.

B. The Results

We applied `visualizeVectorClocks` to **DS1**, **DS2**, **DS3**, **DS4** and **VAST 2008**; the coordinates of the vertices during the 25th, 50th, 75th and 100th time-slices are shown in Table I. The underlying communication patterns are easily discernible. The visualizations for **DS1** reflect that all communications were possible and equally likely. The visualizations for **DS2** reflect the asymmetry and directionality in the set of possible communications. The visualizations for both **DS3** and **DS4** reflect the partitioning of the sets of possible communications. These are extreme examples but they show that visual artifacts only emerge when some pattern exists. For the **VAST 2008** dataset, the picture emerging is that of a core, within which communication frequently flows, and a periphery, within which communication rarely flows. We note that the visualizations are derived solely from the dissimilarities between the vector clocks, the dissimilarities between the vector clocks are derived solely from the values of the vector clocks, and finally, the values of the vector clocks are derived solely from the temporal sequence of communications.

VI. Conclusions

We have presented a novel methodology for visualizing communication flow. Given a temporal sequence of communications between actors, we use vector clocks as a basis for producing a sequence of visualizations that place actors who have received the same or 'communicatively-equivalent' communications close together and actors that have received largely different communications far apart. The visualizations are derived from the temporal sequencing of the communications rather than from the topology of the communication system. Additionally, the visualization for a given time-slice is derived not just from the communications that occurred during that time-slice but from all previous communications.

There are several directions for future research. We noted at the end of Sect. IV that both the measure of (dis)similarity between vector clocks and the choice of MDS algorithm need investigation. Each vector clock represents a point in a very high-dimensional space and the Euclidean distance metric suffers from 'the curse of dimensionality' [25]. We need to compare a variety of metrics and their resulting visualizations both analytically and experimentally. We also require a dynamic MDS algorithm to produce a gradually changing series of visualizations. The assumptions underling vector clocks are, in many cases, naïve when modeling communication. For example, each vector clock update assumes that all possible information is transferred during each communication. However, it seems plausible to extend vector clocks to model the attenuation of information, the bounded capacity of communication channels, etc. Finally, from a scalability perspective, the major bottlenecks in our methodology with respect to storage and running-time are the maintenance of the vector clocks and the computation of the dissimilarity matrix respectively. We are investigating both, e.g., Wang et al. [26], with a view to tackling larger datasets.

Acknowledgment

This research was supported by Science Foundation Ireland (SFI) Grant No. 08/SRC/I1407.

References

[1] G. Lakoff and M. Johnson, *Metaphors We Live By*. University Of Chicago Press, 1980.

[2] H. Ebel, L. Mielsch, and S. Bornholdt, "Scale-Free Topology of E-mail Networks," *Physical Review E*, vol. 66, no. 035103, 2002. [Online]. Available: arXiv:cond-mat/0201476 [cond-mat.dis-nn]

[3] R. Smith, "Instant Messaging as a Scale-Free Network," 2002. [Online]. Available: arXiv:cond-mat/0206378 [cond-mat.stat-mech]

[4] J. Eckmann, E. Moses, and D. Sergi, "Entropy of Dialogues Creates Coherent Structures in E-mail Traffic," *Proceedings of the National Academy of Sciences*, vol. 101, pp. 14 333–14 337, 2004.

[5] G. Kossinets, J. Kleinberg, and D. Watts, "The Structure of Information Pathways in a Social Communication Network," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08)*, Y. Li, B. Liu, and S. Sarawagi, Eds. Springer, 2008, pp. 435–443. [Online]. Available: arXiv:0806.3201 [physics.soc-ph]

[6] J. Leskovec and E. Horvitz, "Planetary-Scale Views on a Large Instant-Messaging Network," in *Proceeding of the 17th International Conference on World Wide Web (WWW'08)*. ACM, 2008, pp. 915–924.

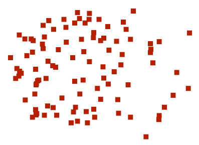
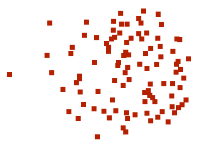
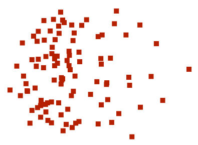
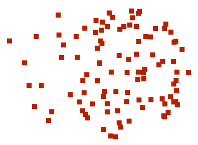
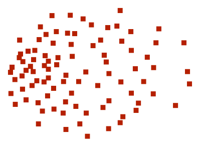
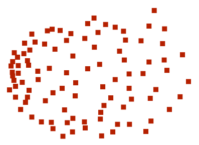
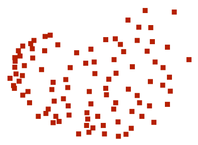
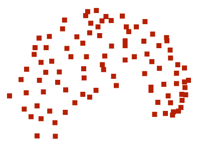
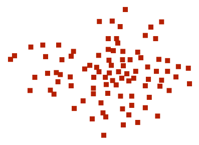
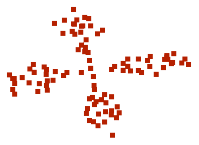
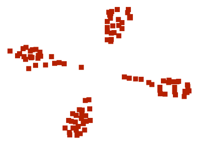
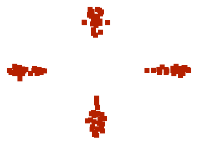
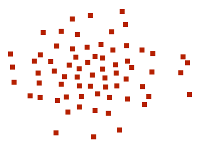
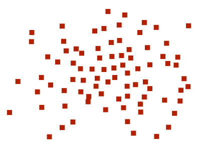
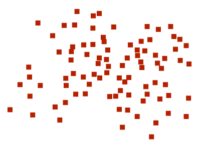
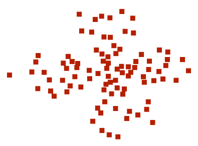
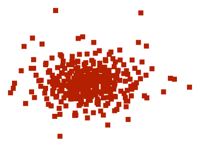
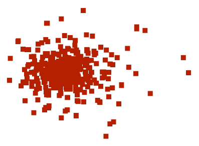
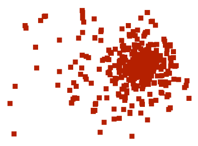
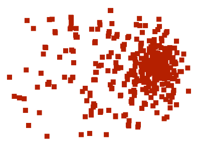
	$t = 24$	$t = 49$	$t = 74$	$t = 99$
DS1				
DS2				
DS3				
DS4				
VAST 2008				

Table I
EACH CELL CONTAINS A VISUALIZATION OF THE DISSIMILARITIES BETWEEN THE VECTOR CLOCKS FOR A GIVEN DATASET AND TIME-SLICE. THE ROWS REPRESENT THE FIVE DATASETS AND THE COLUMNS REPRESENT THE 25^{TH}, 50^{TH}, 75^{TH} AND 100^{TH} TIME-SLICES.

[7] S. North, "Incremental Layout in DynaDAG," in *Proceedings of the 3rd International Symposium on Graph Drawing (GD'95)*, F. Brandenburg, Ed. Springer, 1996, pp. 409–418.

[8] J. Branke, "Dynamic Graph Drawing," in *Drawing Graphs, Methods and Models*, M. Kaufmann and D. Wagner, Eds. Springer, 2001, pp. 228–246.

[9] S. Diehl and C. Görg, "Graphs, They Are Changing," in *Proceedings of the 10th International Symposium on Graph Drawing (GD'02)*, S. Kobourov and M. Goodrich, Eds. Springer, 2002, pp. 23–30.

[10] U. Brandes, D. Fleischer, and T. Puppe, "Dynamic Spectral Layout with an Application to Small Worlds," *Journal of Graph Algorithms and Applications*, vol. 11, no. 2, pp. 325–343, 2007.

[11] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," *Communications of the ACM*, vol. 21, no. 7, pp. 558–565, 1978.

[12] C. Fidge, "Timestamps in Message-Passing Systems that Preserve Partial Ordering," *Australian Computer Science Communications*, vol. 10, no. 1, pp. 56–66, 1988.

[13] F. Mattern, "Virtual Time and Global States of Distributed Systems," in *Proceedings of the International Workshop on Parallel and Distributed Algorithms*, M. Corsnard, Y. Robert, and P. Quinton, Eds. Elsevier, 1989, pp. 215—226.

[14] T. Cox and M. Cox, *Multidimensional Scaling*, 2nd ed. Chapman and Hall, 2000.

[15] U. Brandes and C. Pich, "Eigensolver Methods for Progressive Multidimensional Scaling of Large Data,"

in *Proceedings of the 14th International Symposium on Graph Drawing (GD'06)*, M. Kaufmann and D. Wagner, Eds. Springer, 2007, pp. 42–43.

[16] C. Grinstein, C. Plaisant, S. Laskowski, T. O' Connell, J. Scholtz, and M. Whiting, "VAST 2008 Challenge: Introducing Mini-Challenges," in *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST'08)*. IEEE Computer Society, 2008, pp. 195–196.

[17] K. Berman, "Vulnerability of Scheduled Networks and a Generalization of Menger's Theorem," *Networks*, vol. 28, no. 3, pp. 125–134, 1996.

[18] D. Kempe, J. Kleinberg, and A. Kumar, "Connectivity and Inference Problems for Temporal Networks," *Journal of Computer and System Sciences*, vol. 64, no. 2, pp. 820–842, 2002.

[19] E. Cheng, J. Grossman, and M. Lipman, "Time-Stamped Graphs and their Associated Influence Digraphs," *Discrete Applied Mathematics*, vol. 128, no. 2–3, pp. 317–335, 2003.

[20] A. Johansen, "Probing Human Response Times," *Physica A*, vol. 338, no. 1–2, pp. 286–291, 2004. [Online]. Available: arXiv:cond-mat/0305079

[21] P. Holme, "Network Reachability of Real-World Contact Sequences," *Physical Review E*, vol. 71, no. 046119, 2005. [Online]. Available: arXiv:cond-mat/0410313 [cond-mat.other]

[22] S. Bogaerts and D. Leake, "Facilitating CBR for Incompletely-Described Cases: Distance Metrics for Partial Problem Descriptions," in *Proceedings of the 7th European Conference on Case-Based Reasoning (ECCBR'04)*, P. Funk and P. González-Calero, Eds. Springer, 2004, pp. 62–76.

[23] J. Mingers, "An Empirical Comparison of Selection Measures for Decision-Tree Induction," *Machine Learning*, vol. 3, no. 4, pp. 319–342, 2004.

[24] Algorithmics Group, "MDSJ: Java Library for Multidimensional Scaling," University of Konstanz, 2009. [Online]. Available: http://www.inf.uni-konstanz.de/algo/software/mdsj

[25] R. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[26] X. Wang, J. Mayo, W. Gao, and J. Slusser, "An Efficient Implementation of Vector Clocks in Dynamic Systems," in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'06)*, H. Arabnia, Ed. CSREA Press, 2006, pp. 593–599.