Event Network Visualization

Mehmet Gençer

SELF LOOPS ARE IGNORED DURING IMPORT

1 Model

An event network is represented with a multiplex graph, $N = (V, E)$, consisting of a vertex set representing actors, $V = \{v_1, v_2, \cdots, v_n\}$, and a time ordered set of dyadic events, $E = \{e_1, e_2, \cdots, e_m\}$. Each event, e_k, is represented with the two vertices between which the event takes place, its time, and its weight: $e_k = (v_i, v_j, t, w)$. This event definition ignores the direction information about the event.

For convenience we define some functions to extract event information as: $t(e)$ (time of event), $w(e)$ (weight of event), $v(e) = \{v_i, v_j\}$ (vertices with which the event is incident).

For an event network we define function $T_s(N)$ as time of the first event (i.e. $t(e_1)$), $T_e(N)$ as time of the last event (i.e. $t(e_m)$). Thus the time span of the event network is found as $T(N) = T_e(N) - T_s(N) = t(e_m) - t(e_1)$. Furthermore, age of network at a given time is denotd by function $a(N, t) = t - T_s(N)$.

1.1 Event and network decay

Events are observed at discrete time instances. Consider, for example, an e-mail group communication. The events are instantaneous but actors have lasting memory of events. However, impact of an event deteriorates as time passes. For this reason one can compute strength of relation between two actors as one which decays in time (Brandes et al., 2009). Such a relation is revived when new events take place between actors. Assuming a decay function, $d(e, t) : E, R \to R$, one can compute strength of relation between two actors at a given time as:

$$r(v_i, v_j, t) = \sum_{e_k \in E, t(e_k) <= t, v(e) \equiv \{v_i, v_j\}} d(e_k, t)$$

Note that the relation strength defined above ignores the direction of events, as it is intended to produce a measure of how close the vertices must be placed ,in visualization.

A decay function such as exponential decay would be appropriate for use in this computation:

$$d(e,t) = \begin{cases} 0 & \text{if } t < t(e) \\ w(e)e^{-D(t-t(e))} & \text{if } t \geq t(e) \end{cases}$$

Here, D is a decay constant which must be chosen according to time resolution, and possibly considering time span of the event network. Since $d(.)$ is a "memory-less" function, the above formulation can be easily computed in practice, using an algorithm which updates relation strength as events arrive.

In practice one would also need to have effect of events to disappear after some decay. This is not only compatible with actors' experiences, but also promises a visual quality improvement: inactive actors will be detached from the network and thus can be removed from visualization with an appropriate algorithm. Thus a decay function which satisfies this requirements can be described as follows:

$$d(e,t) = \begin{cases} 0 & \text{if } t < t(e) \\ w(e)e^{-D(t-t(e))} & \text{if } t \geq t(e) \text{ and } w(e)e^{-D(t-t(e))} > D' \\ 0 & \text{if } t \geq t(e) \text{ and } w(e)e^{-D(t-t(e))} \leq D' \end{cases}$$

where D' is a treshold value.

Also, for a vertex, v_i, we define its age at a given time moment, t, on the basis of the set of its past events, $E(v_i,t) = \{e : e \in E, v_i \in v(e), t(e) \leq t\}$, as:

$$a(v_i,t) = t - \min_{e \in E(v_i)} t(e)$$

1.2 Spring model

We adopt the force calculation scheme of Fruchterman and Reingold (1991). However, animation of an event network through time needs to visually indicate how relations deteriorate or get stronger. In order to use relation strength, we use the attractive and repulsive forces between two vertices as follows, where p_i and p_j indicates positions of vertices and $s_{ij} = ||p_i - p_j||$ represents their separation:

$$f_a(t,v_i,v_j) = \frac{s_{ij}^2 r(v_i,v_j,t)}{k}$$

$$f_r(t,v_i,v_j) = -\frac{k^2}{s_{ij}}$$

Where the constant k must be chosen for quality of visualization. This formulation assumes an ideal distance between related vertices as $\frac{1}{k/r(v_i,v_j,t)}$, unlike Fruchterman and Reingold. Therefore the attractive and repulsive forces cancel each other at the ideal distance of $\frac{k}{\sqrt[3]{s_{ij}}}$. TOO UNINDICATIVE??

The choice of k in Fruchterman and Reingold as $k = C\sqrt{\frac{\text{area}}{n}}$ is not appropriate here since event network starts empty. Number of vertices may be unknown (i.e. when visualizing in real time). Instead we use the number of vertices, $V(t)$, at any given time, thus allowing the visualization to scale as network size changes:

$$V(t) = \{v : |E(v,t)| > 0\}$$
$$k(t) = C\sqrt{\frac{\text{area}}{|V(t)|+1}}$$

For a three dimensional layout, the above calculation must use cube-root instead of square-root.

1.3 Algorithm

Animation of an event network is different from graph visualization problems where algorithms run against a stable graph structure. In our problem current state of the network must be visualized through incremental changes to its form in the previous time moment, while network events arrive.

More "central nodes" should move less during adjustment rounds for aesthetically pleasing output!!! For this reason we define a heat factor for each vertex at a given time as:

$$h(v_i,t) = H' \frac{a(N,t)}{a(N,t) + Ha(v_i,t)} \frac{1}{|E(v_i,t)|+1}$$

where H and H' are constants. In other words heat of a vertex decreases (1) as it ages (normalized to network age), and (2) as number of alive events incident to it increases. The above definition of heat limits its value as $0 < h(.) \leq 1$. The constant, H, adjusts the importance of age in determining the heat factor, where H' is used to match overall allowance for movement to framebox size (set to $\sqrt[box-dim]{box - area - or - volume}$).

1.4 Further imporvements

Initial placement at peer diagonal.

Event networks does not seem to lend itself easily to multi-level visualization, which would help to get rid of "clumping" at the center. To remedy this problem we propose a force calculation which makes zero-force edge length proportional to degrees of pair of nodes to which the event is adjacent. Following adjustment for repulsive power "only" between connected nodes:

$$f_r(t,v_i,v_j) = -\frac{k^2}{s_{ij}}(|E(v_i,t)||E(v_j,t)|), \text{if } v_i \text{ and } v_j \text{ are connected, i.e. } r(v_i,v_j,t) > 0.$$

seemed to work well, which makes zero-force length equal to $(|E(v_i,t)||E(v_j,t)|)^{1/3}$.

References

Brandes, U., J. Lerner, and T. A. Snijders (2009). Networks evolving step by step: Statistical analysis of dyadic event data. *2009 Advances in Social Network Analysis and Mining*, 200–205.

Fruchterman, T. M. J. and E. M. Reingold (1991). Graph drawing by force-directed placement. *Software: Practice and Experience 21*(11), 1129–1164.