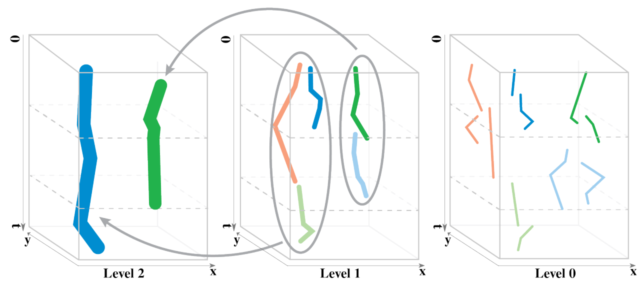
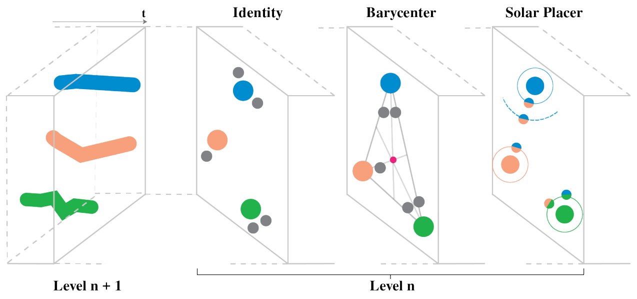
A Multilevel Approach for Event-Based Dynamic Graph Drawing

A. Arleo[1], S. Miksch[1], and D. Archambault[2]

[1]TU Wien, Institute of Visual Computing and Human-Centered Technology, Austria
[2]Swansea University, United Kingdom



(a) *Example of the coarsening process. When two or more nodes are merged together, their presence intervals are merged as well.*



(b) *Placement strategies for the three* `MultiDynNoS` *variants.*

Figure 1: *Multilevel strategies have two important stages: coarsening and placement. In this event-based multilevel approach, we coarsen and place trajectories. An example of the coarsening (a) and placement (b) stages used by the approach.*

Abstract

The timeslice is the predominant method for drawing and visualizing dynamic graphs. However, when nodes and edges have real coordinates along the time axis, it becomes difficult to organize them into discrete timeslices, without a loss of temporal information due to projection. Event-based dynamic graph drawing rejects the notion of a timeslice and allows each node and edge to have its own real-valued time coordinate. Nodes are represented as trajectories of adaptive complexity that are drawn directly in the three-dimensional space-time cube (2D + t). Existing work has demonstrated clear advantages for this approach, but these advantages come at a running time cost. In response to this scalability issue, we present **`MultiDynNoS`**, *the first multilevel approach for event-based dynamic graph drawing. We consider three operators for coarsening and placement, inspired by Walshaw, GRIP, and FM^3, which we couple with an event-based graph drawing algorithm. We evaluate our approach on a selection of real graphs, showing that it outperforms timeslice-based and existing event-based techniques.*

CCS Concepts

• ***Human-centered computing*** $\rightarrow$ ***Graph drawings;*** *Visualization;*

1. Introduction

Usually, a dynamic graph is defined as a succession of individual static graphs [BBDW17], each one representing the state of the graph at a specific time instant (also known as a *timeslice*). This definition has two advantages: it works well for clearly defined time intervals (e.g., yearly, monthly, etc.) and allows for existing static layout algorithms to be used directly for drawing. However, when nodes and edges have independent time coordinates, projecting onto the nearest timeslice results in a quantization error, potentially reducing drawing quality (see this video). *Event-based* networks (also known as *temporal networks* [HS12]) do not suffer from this issue as real-valued time coordinates are specified for each node and edge. Unlike timeslice-based approaches, event-based drawing algorithms exploit the full temporal resolution of the data by optimizing node *trajectories* in the *space-time cube* (2D + *t*), outperforming timeslice-based techniques in terms of drawing quality [SAK17, SAK20] albeit with significant costs in terms of running time and computational resources. These higher running times have limited the use of event-based graph drawing on net-

DOI: 10.2312/evs.20211063

works with a small number of events, despite the quality improvements over timeslice-based techniques.

This paper presents `MultiDynNoS`: the first multilevel event-based graph drawing algorithm, capable of bringing the time to draw event-based networks comparable to timeslice-based approaches. Similar to standard multilevel techniques for static graphs, `MultiDynNoS` follows a *coarsening-refinement* strategy. We adapt the coarsening and placement strategies of Walshaw [Wal03], GRIP [GK00], and FM^3 [HJ04], designed for static graphs, to operate on node trajectories for drawing temporal graphs in the space-time cube. Our experiments show that `MultiDynNoS` drawing quality, in terms of stress, is comparable to existing event-based techniques [SAK17, SAK20] but with significant running time improvements, making them more competitive with timeslice-based approaches [BM11].

2. Related Work

The visualization of dynamic graphs has been studied extensively [BBDW17] with animated techniques [APP10, AP16, FQ11, BPF14] and timeline approaches [APP10, SA06, BVB*11, LHS*15, AB20] receiving considerable attention. We focus on the closest related work to our work in this section.

Multilevel Graph Drawing. In the 2000s, multilevel graph drawing algorithms [Wal03, AMA07, GK00, HJ04, BGKM10] were devised to scale to larger static graphs. These algorithms construct a hierarchy of coarse graphs and exploit this hierarchy to accelerate the drawing. Multilevel graph drawing approaches have been adapted to an online dynamic setting [CCM17, Vel07, Cra16]. Multi-*layer* networks, where several node and edge layers have different meaning [MGM*19], have been used for visualization.

Temporal Networks and Event-Based Visualization. Temporal and event-based networks [HS12, LVM18] have been studied extensively for automatic graph analysis. For most of the past two decades, visualization of temporal networks has focused on drawing a series of projected timeslices in a way that encourages a stable drawing [BBDW17] – the position of nodes and edges should change as little as possible when a change is made to the graph [CP96] so that nodes and edges can be easily identified [AP12, AP16]. Algorithms have been explored to optimize the simultaneous drawing of timeslices in offline [DG02, DGK01, EHK*03, BM11] and online [MELS95, GDBG12, FT08] scenarios. Event-based visualization techniques [DSP*17, MLMdO*13, MLL*13] visualise event sequences with real time coordinates for each data point. Event-based dynamic graph drawing algorithms directly draw these event-based/temporal graphs in the space-time cube [SAK17, SAK20]. Other techniques, such as HOTVis [PS21], exploit the temporal ordering of the edges (the *causal paths*) to influence the layout. However, they focus on 2D visualizations and do not optimize the drawing across the space-time cube.

Contribution. The literature indicates a growing interest in event-based visualizations of networks for visual analytics applications. Event-based dynamic graph drawings can potentially yield improved drawing quality over timeslice-based approaches, motivating our research on more scalable techniques for embedding temporal networks in the space-time cube.

3. `MultiDynNoS` Pipeline

Consider a temporal network $D = (V, E)$ where each node and edge possesses a number of *attributes* which are functions of time. The *appearance* of a node $v \in V$ is defined as $A_v : V \times T \to [true, false]$ (edge appearance is defined similarly) which maps to node/edge insertion and deletion in the event-based graphs. A_v defines a series of intervals in T (time) in which the node/edge is present. The *position* of a node in the plane over time is defined as $P_v : V \times T \to R^2$. When defined in this way, the appearance and position of the nodes are represented as a series of trajectories through time embedded in the space-time cube (e.g., Fig. 1a): lines that define node movement in the two dimensional plane as time passes downwards in the cube. We also define a *flattened* graph as the weighted static counterpart of a temporal graph where node and edge weights represent the cumulative duration of the time intervals in which their appearance attribute function yields true.

Layout Process. First, a *coarsening* operator is applied on D to generate a *coarse hierarchy* of the graph. Starting from the coarsest graph, each level gets *refined*: its drawing is computed and its coordinates are used to *place* (i.e. assign the initial coordinates) the vertices on the level below. This initial placement provides quicker convergence in the next refinement cycle. Refinement ends when the layout for the input graph is computed.

Coarsening. Coarsening yields a hierarchy of coarse event-based graphs $D_H = \{D_w, D_1, ..., D_k\}$, with "depth" k, to be used in the refinement stage. D_w is the input graph with an added attribute constant function representing the node and edge weights from the flattened graph D_f. The finest level is D_1 and the coarsest D_k. For each level $D_n = (V_n, E_n)$, we order the vertices of V_n by their weight and put them on a stack. We pop the stack and get the heaviest vertex v_n: its copy v_{n+1} is then assigned to V_{n+1}. At this point, we select a subset of the neighbors of v_n, depending on the coarsening strategy, summing their weights and merging their appearance intervals with v_{n+1}. We refer to v_n as the "representative" in V_{n+1} of the vertices merged with it in V_n. We refer to the set of representatives of level n as $\overline{V}_n$. Once complete, v_n and the vertices merged with it are removed from the stack. This process is repeated until the stack is empty. Coarsening stops at the coarsest hierarchy level D_k when the node count falls below a threshold or it is $\geq 95\%$ the size of level D_{k-1}. The latter condition is introduced to avoid a deep hierarchy with very similar level sizes, which would slow down drawing significantly. We implemented three different coarsening strategies. First, we implemented the *Maximal Matching*, found in the multilevel approach by Walshaw [Wal03], where pairs of vertices connected by an edge belonging to the graph maximal matching are merged together in each level. Second, we implemented the *Maximal Independent Set* coarsening, used by *GRIP* [GK00]. Once a vertex is selected to be part of the new level, it is merged together with all of its neighbors. Finally, we implemented the *Solar Merger* algorithm, used by *FM*^{3} [HJ04]. Each selected vertex is merged together with its neighbors up to distance 2, creating a "Solar System Partitioning" of the graph. Once the vertex set for the new level is created, we generate E_{n+1}: for each edge $e_n = (v_n, w_n)$, we create an edge $e_{n+1} = (v_{n+1}, w_{n+1})$ such that v_n and w_n were merged in v_{n+1} and w_{n+1} respectively. If that edge already exists, its presence is merged with the one of e_n.

Coarsest Level Placement. Initial placement assigns the initial coordinates of the trajectories in D_k as follows: we flatten D_k to obtain $D'_k$and draw it using a static layout algorithm. Node trajectories are centered in these newly computed coordinates and extruded vertically downwards across time. Subsequent refinement steps with an event-based drawing algorithm [SAK20] "bend" these trajectories to change the position of the nodes across time. A good initial placement is expected to yield smoother trajectories with few bends, which resolves in nodes with smoother movement.

Refinement. During each refinement iteration, DynNoSlice [SAK20] is run on D_n. One of the key points of the multilevel strategy is that more quality-oriented layout parameters can be used on coarse graphs, since they are smaller in size and therefore quicker to draw. As the size of the graph to layout increases, speed can be emphasized. In our approach, we tune two parameters: the maximum node mobility and the number of layout algorithm iterations. Coarser levels will benefit from more flexible trajectories, while finer levels are more conservative with reduced iterations and movement. The parameters decrease linearly by 7% at each level. This value was obtained empirically when the considering quality/running time trade off. Time trajectory post-processing of DynNoSlice [SAK17, SAK20] runs once every two layout iterations in the coarser levels and the interval grows by 2 with each new level. Once the layout for D_n is computed (and $D_n \neq D_w$), the final coordinates are used to *place* the node trajectories on level D_{n-1}. First, each representative $v_{n-1} \in \overline{V}_{n-1}$ is placed at the coordinates of the corresponding vertex in V_n. We compute the initial coordinates of the remaining vertices based on the new coordinates of their representative . We implemented three placement operators (Fig. 1b) inspired by Walshaw [Wal03], *GRIP* [GK00], and *FM*^{3} [HJ04]. The first strategy is the *identity placer*: the nodes are placed in the same position as their representative. The second strategy places the trajectories close to the *barycenter* of the coordinates of the representative's neighbors at level $n+1$. The final position of the node is skewed towards its own representative by a fixed rate. The third strategy is similar to barycenter but changes the attraction of the representative cluster. Specifically, given any two neighboring nodes $v_{n+1}, w_{n+1} \in V_{n+1}$, the solar system partitioning guarantees that representatives at level n, v_n and w_n, are at most distance 5 from each other. Since v_n and w_n neighbors up to distance 2 are merged together in the *FM*^{3} coarsening, with this information it is possible to reconstruct the relative position of any of the merged trajectories in the paths between v_n and w_n, and place them accordingly. When the path position is not known it uses the barycenter placement strategy. For all approaches, randomness is added to the final coordinates to avoid possible accidental coordinate overlaps.

4. Experimental Evaluation

We conduct an evaluation where we repeat the experiment performed in DynNoSlice [SAK20] to compare MultiDynNoS to state-of-the-art dynamic graph layout algorithms on known metrics. Our *research question* can be formulated as follows: "Is MultiDynNoS faster than DynNoSlice, while providing layouts with comparable drawing quality?". We implemented MultiDynNoS and tested it (results in Table 1).

Metrics and Strategies. We evaluate the layouts using quality and readability metrics. We include: (i) the **time**, drawing time in seconds; (ii) **Movement**, the average distance travelled by a node during graph evolution [BM11, SAK20]; (iii) **Crowding**: the number of times nodes pass close to each other in the animation of the dynamic graph [SAK20]; (iv) **Depth**: coarsening depth (multilevel strategies only); (v) ***StressOn*** and (vi) ***StressOff***, which are the layout stress computed on a per-timesliced basis or between timeslices, respectively, with optimal scaling [SAK20] applied.
We test three MultiDynNoS variants: MultiDynNoS wi_id is the Walshaw variant of MultiDynNoS with maximal matching of trajectories and identity placement; MultiDynNoS is_gr is the GRIP variant of MultiDynNoS with maximal independent set coarsening of trajectories and barycenter placement; MultiDynNoS sm_sp is the FM^3 variant of MultiDynNoS with the FM^3 coarsening and placement strategy. Each variant is tested alternating the drawing algorithm for the coarsest level placement between sfdp [Hu05] and fdp [FR91]. The variants of MultiDynNoS are tested against Visone [BW04], a state-of-the-art timeslice-based dynamic graph drawing algorithm, and DynNoSlice [SAK17, SAK20]. sfdp flat flattens the entire event-based data and draws it once as a static graph using sfdp [Hu05], and is our baseline.

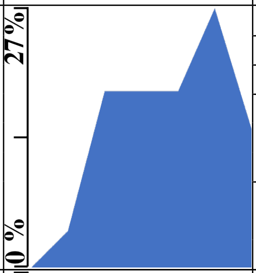
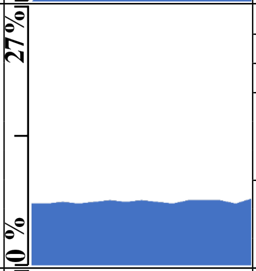
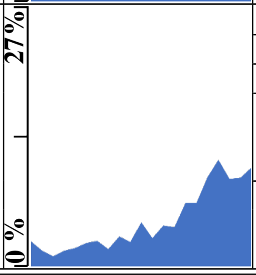
Results. Table 1 shows the results of our experiments. In terms of running time, on all the experiment instances MultiDynNoS is competitive with Visone and can be an order of magnitude faster than DynNoSlice. This represents a leap forward than previous studies [SAK17,SAK20] (whose results have been replicated here), where Visone always had the best performance when compared to DynNoSlice on this same set of graphs. In terms of drawing quality, MultiDynNoS approaches have competitive or lower levels of stress and crowding than DynNoSlice, thus confirming our research hypothesis, with smaller amounts of movement due to the initial placement. In timesliced graphs, Visone had unsurprisingly the least stress, with the notable exception of **InfoVis**, where MultiDynNoS and DynNoSlice perform better in terms of both types of stress and crowding. As previously discussed [SAK17], **InfoVis** is very similar to an event-based data, since there are drastic changes between timeslices as author sets rarely remain stable across consecutive years. On the event-based data, MultiDynNoS and DynNoSlice outperform or match Visone in terms of stress, movement, and crowding. Visone cannot optimize for stress between the timeslices imposed on this naturally expressed event-based data. The video in the supplementary material demonstrates these improvements. The *sfdp flat*, our baseline, is not able to perform very well in terms of stress on these smaller datasets. However, it is a multilevel algorithm and its strengths are in terms of scalability.

5. Conclusion and Future Work

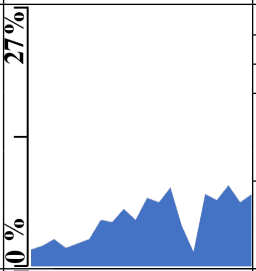
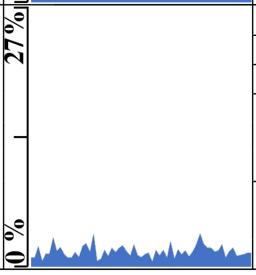
In this paper, we present MultiDynNoS: a multilevel approach for event-based dynamic graph drawing. Our experiment shows an improvement up to an order of magnitude in terms of running time compared to DynNoSlice while retaining its advantages. Future work includes performing a new evaluation on larger datasets that were previously inaccessible to event-based layout techniques.

Table 1: *Results of the experiment. $|V|$ and $|E|$ columns report the number of nodes and edges in the flattened graph. $|E_v|$ reports the number of events in thousands. The* ***Trend*** *column visualizes the number of events per timeslice on a scale from 0 to 27% of the total events of the graph. The number of timeslices is reported by the name of the graph in brackets. The* ***Type*** *column reports the tested algorithm. The* `MultiDynNoS` *variant used is presented as the combination of the initial placement layout (*`fdp` *or* `sfdp`*) and the coarsening/placement technique used.* ***T*** *column reports the algorithm running time in seconds.* ***Sc.(aling)*** *column reports the scaling value. Columns* ***On*** *and* ***Off*** *show the StressOn and StressOff values. Columns M and C represent Movement and Crowding respectively; D reports the depth of the coarsened hierarchy.* `MultiDynNoS` *is implemented in Java 14 and the experiments are run on an i7-8750H CPU with 16GB of RAM.*

Timesliced Graphs

	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert E_v\rvert$	**Trend**	**Type**		***T*** **(s)**	**Sc.**	***On***	***Off***	*M*	*C*	*D*
`VanDebunt (7)`	39	32	0.1*k*		`Visone`		**0.12**	1	1.14	1.46	3.79	0	-
					`DynNoSlice`		5.04	0.62	1.23	1.21	3.92	0	-
					`sfdp flat`		0.14	1.61	2.77	2.81	-	0	-
					`fdp`	`wi_id`	0.48	0.68	1.55	1.62	1.03	0	5
						`is_gr`	0.47	0.75	**1.03**	**1.06**	0.99	0	3
						`sm_sp`	0.46	0.75	1.05	1.08	1.00	0	3
					`sfdp`	`wi_id`	0.56	0.68	1.37	1.39	0.98	0	6
						`is_gr`	0.58	0.75	1.09	1.12	0.97	0	3
						`sm_sp`	0.58	0.68	1.42	1.48	**0.92**	0	3
`Newcomb (15)`	17	93	0.6*k*		`Visone`		0.10	1	**14.04**	**14.76**	16.36	8	-
					`DynNoSlice`		7.58	0.68	16.60	16.57	13.44	1	-
					`sfdp flat`		**0.15**	1.33	26.54	26.52	-	**0**	-
					`fdp`	`wi_id`	0.32	0.82	28.40	28.48	2.87	2	6
						`is_gr`	0.31	0.82	21.01	20.86	2.95	4	3
						`sm_sp`	0.32	0.82	22.55	22.39	2.87	1	3
					`sfdp`	`wi_id`	0.42	0.82	27.05	26.94	2.89	2	6
						`is_gr`	0.38	0.82	20.89	20.70	**2.82**	1	3
						`sm_sp`	0.39	0.82	21.79	21.71	2.85	2	3
`InfoVis (21)`	1,136	2,506	2.8*k*		`Visone`		77.43	0.46	51.66	52.97	2.14	36	-
					`DynNoSlice`		224.93	0.56	30.14	30.19	2.03	2	-
					`sfdp flat`		**0.55**	1.33	105.29	102.87	-	1,253	-
					`fdp`	`wi_id`	143.95	0.51	47.26	47.49	0.78	16	7
						`is_gr`	87.79	0.56	**28.08**	**27.79**	1.50	4	4
						`sm_sp`	138.95	0.56	28.88	28.65	1.51	4	3
					`sfdp`	`wi_id`	110.00	0.46	51.03	50.97	**0.70**	36	7
						`is_gr`	83.00	0.62	28.69	28.59	1.62	2	4
						`sm_sp`	85.00	0.56	27.21	27.02	1.48	**1**	3

Event-Based Graphs

	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert E_v\rvert$	**Trend**	**Type**		***T*** **(s)**	**Sc.**	***On***	***Off***	*M*	*C*	*D*
`Rugby (20)`	12	66	3.1*k*		`Visone`		**0.07**	0.68	3.08	2.70	25.46	6	-
					`DynNoSlice`		2.84	0.51	1.86	**1.78**	6.64	0	-
					`sfdp flat`		0.18	0.90	2.07	2.02	-	0	-
					`fdp`	`wi_id`	0.75	0.56	2.18	2.01	1.74	1	5
						`is_gr`	1.84	0.56	**1.76**	1.84	1.25	0	2
						`sm_sp`	0.52	0.51	2.10	1.94	1.28	0	2
					`sfdp`	`wi_id`	0.88	0.51	2.19	1.97	1.51	1	5
						`is_gr`	1.04	0.513	1.99	1.87	**1.11**	0	2
						`sm_sp`	0.77	0.56	2.03	1.95	1.41	0	2
`Dialogs (61)`	118	501	4.0*k*		`Visone`		3.39	0.17	0.62	0.87	5.44	682	-
					`DynNoSlice`		49.53	0.28	0.75	0.90	1.35	0	-
					`sfdp flat`		**0.21**	1	0.65	0.69	-	6	-
					`fdp`	`wi_id`	1.53	0.42	**0.53**	0.60	0	711	14
						`is_gr`	5.05	0.35	0.66	0.96	0.76	1	4
						`sm_sp`	5.49	0.35	0.65	0.91	0.73	0	3
					`sfdp`	`wi_id`	1.63	0.42	0.55	**0.58**	0	441	13
						`is_gr`	5.07	0.35	0.64	0.92	0.71	0	4
						`sm_sp`	5.96	0.31	0.74	0.88	**0.64**	0	3

References

[AB20] AGARWAL S., BECK F.: Set streams: Visual exploration of dynamic overlapping sets. *Computer Graphics Forum 39*, 3 (2020), 383–391. 2

[AMA07] ARCHAMBAULT D., MUNZNER T., AUBER D.: TopoLayout: Multilevel graph layout by topological features. *IEEE Transactions on Visualization and Computer Graphics 13*, 2 (2007), 305–317. 2

[AP12] ARCHAMBAULT D., PURCHASE H. C.: Mental map preservation helps user orientation in dynamic graphs. In *International Symposium on Graph Drawing* (2012), Springer, pp. 475–486. 2

[AP16] ARCHAMBAULT D., PURCHASE H. C.: Can animation support the visualization of dynamic graphs? *Information Sciences 330* (2016), 495–509. 2

[APP10] ARCHAMBAULT D., PURCHASE H., PINAUD B.: Animation, small multiples, and the effect of mental map preservation in dynamic graphs. *IEEE transactions on visualization and computer graphics 17*, 4 (2010), 539–552. 2

[BBDW17] BECK F., BURCH M., DIEHL S., WEISKOPF D.: A taxonomy and survey of dynamic graph visualization. *Computer Graphics Forum 36*, 1 (2017), 133–159. 1, 2

[BGKM10] BARTEL G., GUTWENGER C., KLEIN K., MUTZEL P.: An experimental evaluation of multilevel layout methods. In *International Symposium on Graph Drawing* (2010), Springer, pp. 80–91. 2

[BM11] BRANDES U., MADER M.: A quantitative comparison of stress-minimization approaches for offline dynamic graph drawing. In *International Symposium on Graph Drawing* (2011), Springer, pp. 99–110. 2, 3

[BPF14] BACH B., PIETRIGA E., FEKETE J. D.: GraphDiaries: Animated transitions and temporal navigation for dynamic networks. *IEEE Transactions on Visualization and Computer Graphics 20*, 5 (2014), 740–754. 2

[BVB*11] BURCH M., VEHLOW C., BECK F., DIEHL S., WEISKOPF D.: Parallel edge splatting for scalable dynamic graph visualization. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2344–2353. 2

[BW04] BRANDES U., WAGNER D.: *Analysis and visualization of social networks*. Springer, 2004, pp. 321–340. 3

[CCM17] CRNOVRSANIN T., CHU J., MA K.-L.: An incremental layout method for visualizing online dynamic graphs. *Journal of Graph Algorithms and Applications 21*, 1 (2017), 55–80. 2

[CP96] COLEMAN M. K., PARKER D. S.: Aesthetics-based graph layout for human consumption. *Software: Practice and Experience 26*, 12 (1996), 1415–1438. 2

[Cra16] CRAWFORD C. J.: *Dynamic multilevel graph layout and visualisation*. PhD thesis, University of Greenwich, 2016. 2

[DG02] DIEHL S., GÖRG C.: Graphs, they are changing. In *International Symposium on Graph Drawing* (2002), Springer, pp. 23–31. 2

[DGK01] DIEHL S., GÖRG C., KERREN A.: Preserving the mental map using foresighted layout. In *Data Visualization 2001* (2001), Springer, pp. 175–184. 2

[DSP*17] DU F., SHNEIDERMAN B., PLAISANT C., MALIK S., PERER A.: Coping with volume and variety in temporal event sequences: Strategies for sharpening analytic focus. *IEEE Transactions on Visualization and Computer Graphics 23*, 6 (2017), 1636–1649. 2

[EHK*03] ERTEN C., HARDING P. J., KOBOUROV S. G., WAMPLER K., YEE G.: Graphael: Graph animations with evolving layouts. In *International Symposium on Graph Drawing* (2003), Springer, pp. 98–110. 2

[FQ11] FARRUGIA M., QUIGLEY A.: Effective temporal graph layout: A comparative study of animation versus static display methods. *Journal of Information Visualization 10*, 1 (2011), 47–64. 2

[FR91] FRUCHTERMAN T. M., REINGOLD E. M.: Graph drawing by force-directed placement. *Software: Practice and experience 21*, 11 (1991), 1129–1164. 3

[FT08] FRISHMAN Y., TAL A.: Online dynamic graph drawing. *IEEE Transactions on Visualization and Computer Graphics 14*, 4 (2008), 727–740. 2

[GDBG12] GOROCHOWSKI T. E., DI BERNARDO M., GRIERSON C. S.: Using aging to visually uncover evolutionary processes on networks. *IEEE Transactions on Visualization and Computer Graphics 18*, 8 (2012), 1343–1352. 2

[GK00] GAJER P., KOBOUROV S. G.: Grip: Graph drawing with intelligent placement. In *International Symposium on Graph Drawing* (2000), Springer, pp. 222–228. 2, 3

[HJ04] HACHUL S., JÜNGER M.: Drawing large graphs with a potential-field-based multilevel algorithm. In *International Symposium on Graph Drawing* (2004), Springer, pp. 285–295. 2, 3

[HS12] HOLME P., SARAMÄKI J.: Temporal networks. *Physics Reports 519*, 3 (2012), 97 – 125. 1, 2

[Hu05] HU Y.: Efficient, high-quality force-directed graph drawing. *Mathematica Journal 10*, 1 (2005), 37–71. 3

[LHS*15] LIU Q., HU Y., SHI L., MU X., ZHANG Y., TANG J.: Egonet-cloud: Event-based egocentric dynamic network visualization. In *2015 IEEE Conference on Visual Analytics Science and Technology (VAST)* (2015), IEEE, pp. 65–72. 2

[LVM18] LATAPY M., VIARD T., MAGNIEN C.: Stream graphs and link streams for the modeling of interactions over time. *Social Network Analysis and Mining 8*, 1 (2018), 61. 2

[MELS95] MISUE K., EADES P., LAI W., SUGIYAMA K.: Layout adjustment and the mental map. *Journal of Visual Languages & Computing 6*, 2 (1995), 183–210. 2

[MGM*19] MCGEE F., GHONIEM M., MELANÇON G., OTJACQUES B., PINAUD B.: The state of the art in multilayer network visualization. *Computer Graphics Forum 38*, 6 (2019), 125–149. 2

[MLL*13] MONROE M., LAN R., LEE H., PLAISANT C., SHNEIDERMAN B.: Temporal event sequence simplification. *IEEE transactions on visualization and computer graphics 19*, 12 (2013), 2227–2236. 2

[MLMdO*13] MONROE M., LAN R., MORALES DEL OLMO J., SHNEIDERMAN B., PLAISANT C., MILLSTEIN J.: The challenges of specifying intervals and absences in temporal queries: A graphical language approach. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2013), pp. 2349–2358. 2

[PS21] PERRI V., SCHOLTES I.: HOTVis: Higher-order time-aware visualisation of dynamic graphs. In *International Symposium on Graph Drawing and Network Visualization* (2021). 2

[SA06] SHNEIDERMAN B., ARIS A.: Network visualization by semantic substrates. *IEEE transactions on visualization and computer graphics 12*, 5 (2006), 733–740. 2

[SAK17] SIMONETTO P., ARCHAMBAULT D., KOBOUROV S.: Drawing dynamic graphs without timeslices. In *International Symposium on Graph Drawing and Network Visualization* (2017), Springer, pp. 394–409. 1, 2, 3

[SAK20] SIMONETTO P., ARCHAMBAULT D., KOBOUROV S.: Event-based dynamic graph visualisation. *IEEE Transactions on Visualization and Computer Graphics 26*, 7 (2020), 2373–2386. 1, 2, 3

[Vel07] VELDHUIZEN T. L.: Dynamic multilevel graph visualization. *arXiv preprint arXiv:0712.1549 [cs.GR]* (2007). 2

[Wal03] WALSHAW C.: A multilevel algorithm for force-directed graph-drawing. *Journal of Graph Algorithms and Applications 7*, 3 (2003), 253–285. 2, 3