DOI: 10.1111/cgf.70499

EUROVIS 2026
A. Abdul-Rahman, M. Angelini, and B. Preim
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 3
STAR – State of The Art Report

The Story(*line*) So Far: A Survey on Storyline Visualization

S. Di Bartolomeo [1], A. Dobler [1], V. Filipov [1], M. Nöllenburg [1], H. Ehlers [1,2]

[1]TU Wien, Vienna, Austria [2] Works GmbH, Tübingen, Germany

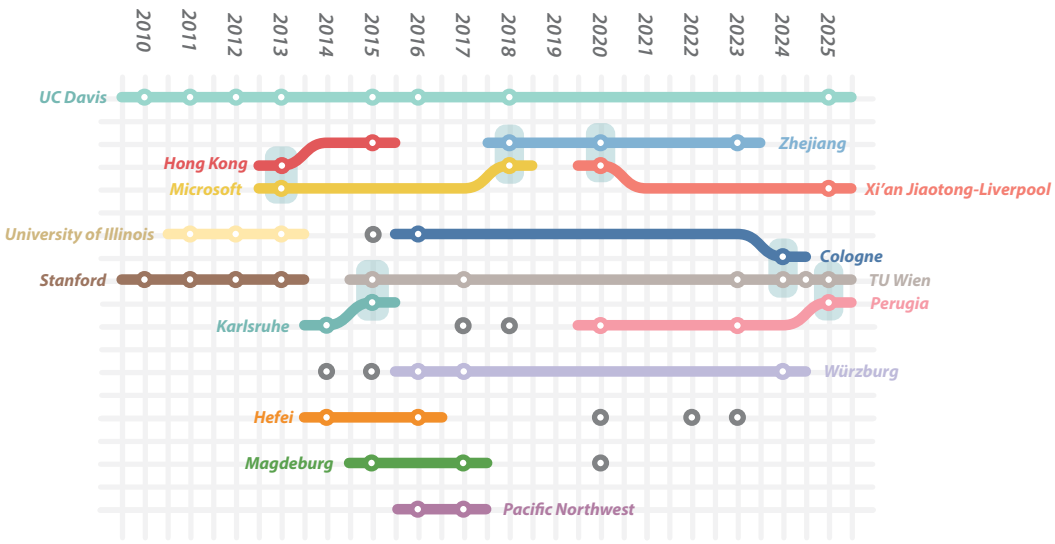

Figure 1: *A storyline of publications included in this survey over time. Colored dots represent individual papers ("events"). Lines correspond to institutions that appear in more than one publication, while papers from other institutions are shown as isolated dots. Collaborations between institutions are indicated by blue rectangles. The visualization highlights a vibrant research field, with recurring collaborations and a steady stream of publications across years. Note that, in this case in particular, the graph is laid out manually.*

Abstract

Storyline visualizations model narratives as temporal networks, using x-monotone lines to represent entities and their interactions over time. This technique offers an intuitive way to reveal structural patterns over time, such as character co-occurrence and narrative flow. Storylines represent a visualization approach with growing interest from the visualization community and applications in diverse contexts. Researchers have developed various layout algorithms and formalized a set of optimization objectives with the goal of automating their generation and balancing their readability, graph aesthetics, and efficiency. These methods vary in their algorithmic formulations and implementations as well as the visual elements they support, such as labels, grouping, or continuity preservation across time. This state-of-the-art report maps the current landscape of storyline visualization approaches, with a specific focus on the visual structures, optimization objectives, and the characteristic of the layout algorithms that generate these.

1. Introduction

Storylines are a visualization technique used to depict evolving relationships and interactions over time. At its core, storylines model a narrative as a temporal network, where entities or characters are represented by separate x-monotone lines arranged along the *y*-axis progressing along time, the *x*-axis (see Figure 1). Each line traces the presence or involvement of a character across moments in the narrative, and when characters interact, such as appearing together in a scene, their lines move close to one another or even

converge. In these storylines, spatial proximity (i.e., line convergence) visually encodes temporal, social, and structural relationships. The concept gained widespread attention through a popular XKCD comic titled "Movie Narrative Charts" [Mun09], which illustrated the movements and interactions of characters from major movies. This showcased how simple and intuitive visuals could reveal the structure and rhythm of a story, i.e., what characters appear when, with whom, and for how long.

Following this debut, researchers began developing automatic **layout algorithms** to reproduce and expand on the manually crafted aesthetic of XKCD's version, aiming to balance readability, (graph) aesthetic clarity, and efficiency, while accommodating a wide range of narrative structures and facilitate for additional (visual) features, such as labels, grouping, or continuity preservation across time. The general challenge lies in arranging the lines to show groupings and transitions clearly and without clutter. Specific goals include minimizing line crossings (which obscure relationships), reducing wiggles, or bends (which distract from continuity), and limiting white space (to preserve compactness) [TM12].

Due to the method of representing time in storyline visualizations (i.e., a sequence of discrete time steps or layers), many approaches draw on techniques from layered graph drawing. In particular, adaptations of the Sugiyama framework [STT81] and similar heuristics are used to compute vertical orderings of characters over time by minimizing crossings. Methods such as barycentric ordering [GKNV93] are common, often with additional constraints to ensure that interacting characters are placed adjacently [GJLM16].

Existing storyline methods vary not only in the algorithmic techniques they employ, ranging from heuristics, to exact, to hybrid approaches, but also in the visual and structural elements they support. For example, some methods explicitly account for labels, extended group interactions, or character continuity across multiple time steps, all of which impose additional constraints on the layout algorithm's optimization objectives.

The landscape of layout methods and features is growing quickly. To help make sense of this evolving space, this paper offers a **state-of-the-art report of storyline visualizations**, centered on their optimization objectives, visual structures, and algorithmic foundations. Our goal is to support future researchers in identifying unexplored directions and to assist those looking to implement storyline visualizations on specific datasets in finding algorithms that suit their particular needs. To do so, we have structured this state-of-the-art report as follows. First, for the sake of reproducibility, we discuss the methodology utilized to collect our corpus of storyline-visualization-relevant literature. Second, we discuss the taxonomic classification of this corpus of literature as well as the results of said classification. Subsequently, we take a closer look at the different types of layout algorithms that are utilized to minimize/maximize various optimization objectives and how they are commonly evaluated across several domains. With the results of the previous sections in hand, we discuss identified outstanding challenges and opportunities. Finally, we summarize the results of the paper and outline some limitations of this paper.

An interactive website containing the classified papers used in this survey is at https://velitchko.github.io/storyline-survey.

1.1. What is a Storyline?

Throughout our survey of the literature, we found ourselves frequently scrutinizing various visualization approaches, e.g., flow charts or area charts, and wondering "Can we consider these as storylines?"

Ogawa and Ma [OM10], to the best of our knowledge, are the earliest authors describing a storyline visualization, provide these following constraints for the visualization of storylines:

> From studying the XKCD chart, we derived general rules for an aesthetic layout of lines:
>
> 1. Clustered developers must be placed in contiguous, adjacent lines.
> 2. Clusters should be spaced apart from each other.
> 3. Existing tubes should change y-position very little, if at all.
> 4. Tube crossings are inevitable, but avoid them if possible.

While the language used by Ogawa and Ma reflects the infancy of the niche ("tubes" are character lines, "clusters" are scenes in which characters appear together), these criteria will go on to define the basis of subsequent storyline visualization efforts. In their fundamental paper defining criteria for optimizing storyline visualization, Tanahashi et al. refer directly to the earlier Ogawa and Ma paper in order to expand and refine the definition of storylines as follows [TM12, p. 2]:

- "Storyline visualization is a technique that portrays the temporal dynamics of social interactions by projecting the timeline of the interaction onto an axis. [...]"
- "Storyline visualization consists of a series of lines, going from left to right along the time-axis, that converge and diverge in the course of their paths. Each line in this visualization represents a unique social entity (character) in the data."
- "The starting and ending points of each line represent the lifespan of the corresponding character."
- "Lines representing interacting characters are bundled together during the time period of their interaction, causing them to converge as they start interacting and to diverge as they stop interacting."

They go on to define their own criteria for storylines:

- Lines representing interacting characters must be adjacent.
- Otherwise, lines must not be adjacent.
- A line must not deviate unless it converges or diverges with another line.

With the proliferation of the field, these definitions were relaxed to allow for a broader exploration of the design spaces. For instance, ubiquitous characters, characters that participate in multiple scenes at the same time, could now be allowed as an additional feature [GDL*20].

In order to limit the scope of this survey to papers that can be formally defined as storyline visualizations, we present the following exclusion criteria:

E1 Every entity in a storyline visualization should be represented by one line, which can be followed through the story.

E2 The focus of the visualization should be on the interactions between these lines.
E3 The visualization should extend along one axis that represents time.
E4 The vertical ordering of characters needs to be determined by a layout algorithm informed by their meetings, and not by additional attributes or metadata associated with the nodes.

As such, we **exclude**:

- **Criterion E1:** Layered graph visualizations in which the *layers do not represent time*.
  While such layouts share structural similarities with storylines, their layers encode abstract graph structure rather than temporal progression and fall outside the scope of this survey.
- **Criterion E2:** Visualizations in which there is not a one-to-one correspondence between visual lines and individual characters.
  *This includes temporal treemaps [KW19, DN24], in which entities are aggregated and focus on composition rather than on interactions between characters. For the same reason, we exclude Sankey diagrams [RTJ*11], aggregated area charts [SWL*14, XWW*13], and related approaches in which multiple entities are visually merged such that it is not possible to follow an individual character over time [OMB*07, RB10, CLWW14].*
- **Criterion E3:** Approaches in which the layout is primarily driven by metadata or attributes rather than by interactions between characters.
  An example of this category is the work by Di Bartolomeo et al. [DBZSD21], where storyline-like layouts are used but spatial organization is not derived from character interactions.
- **Criterion E4:** Layouts in which characters that are absent at a given time step completely disappear from the visualization and are not visually tracked across time.
  *An example of this category is in Yeh et al. [YMA*25]* in which characters are not presented in a visually continuous manner.

1.2. A Brief Glossary of Storyline-related Terms

Storyline visualizations have been referred to using various descriptions. The visual and structural elements within storylines are also described using terminology that comes from graph drawing and visualization as well as from the narrative aspects of the visualization. The visualization itself has been referred to under multiple names: originally labeled as a "narrative chart" [Mun09], however in the seminal paper of Ogawa et al. [OM10], it is formally defined as a "storyline".

The following presents a summary of terms used throughout our collection of literature, listing their synonyms and explaining what is meant by them through the paper. The terms in **bold** are the ones we try to use in the rest of the paper as standardized terms.

A "**Character**": an individual participant in the narrative, visually represented by a single traceable element; also referred to as "Entity" [QC16], "Actor" [ZCY25], or using a name based on context (e.g., "Developer" [OM10]). The continuous visual trace representing a character's presence across time can be called a "**Character Line**" [LWW*13] (or just "Line"), but also a "Tube" [OM10] or a "Trajectory" [YMA*25]. In layered graph terminology, character lines would be comprised of multiple "Edges" or be referred to as "Paths".

A temporal interval during which multiple characters participate together can be referred to as "**Interaction**" [TBS14], "Event" [Are15], "Scene" [ZCY25], "Meeting" [HW24], "**Group**" [GDL*20] or "Cluster" [OM10]. A temporal interval during which a character is active in the narrative but does not participate in a multi-character interaction is just an "**Event**" [Are15] (perhaps even specified as "Solo Event"), or, in graph terminology, a "Node".

A discrete unit of time at which character presence and interactions take place are defined as a "**Time Step**" [PBH18]. Time steps can contain one or more events or groups. In papers specifically tackling dynamic and/or layered graph drawing, a time step can also be called a "Layer" [GJLM16], but also a "Slice" [DJJ*24], or in other cases a "Snapshot" [BBDW17].

At every given time step, a storyline visualization needs to have defined a total vertical order of characters. This can be defined as "**Vertical Order**" [HW24], "Permutation" [GJLM16] or "Ranking" [PLM*25].

1.3. Related Surveys

Drawing on Filipov et al.'s recent survey of surveys on network visualization [FAM23], we highlight a few relevant perspectives in current surveys that, in our estimation, relate strongly to the visualization of storylines: the visualization of dynamic graph drawing, layered graph drawing, and clustered graph visualization.

As storylines can be viewed as a form of **dynamic network**, it is useful to consider typical approaches to dynamic network visualization. Several surveys have explored the visualization of such evolving networks [MMB05, KKC14]. In particular, Beck et al. [BBDW17] classify dynamic visualizations into categories such as *animation*, *timeline*, and *generic*. Storyline visualizations fall into the *juxtaposed* timeline category, where sequential snapshots of a network are placed side by side, aligned vertically and ordered left to right along the x-axis, as is common in storyline layouts.

On a similar note, storyline visualizations are also closely related to **layered graph drawing**. However, unlike general layered graphs, where layers encode an abstract structure, storylines assume that the horizontal axis represents time and that each entity is visually tracked as a continuous trace across this axis. Interactions are not encoded as explicit edges, but emerge through spatial proximity and grouping of these traces. This emphasis on continuity, persistence, and interaction-centric layouts introduces constraints and optimization goals that are specific to storylines and are not commonly addressed in typical layered graph drawing or broader dynamic network visualization. Nonetheless, the algorithmic foundations used in storyline layouts are indeed provided by layered graph drawing. Here, a large body of work exists on the topics, e.g., McGee et al.'s [MGM*19] survey on multilayer network visualization. Techniques such as Sugiyama-style frameworks [STT81], barycentric ordering, and median heuristics

are frequently adapted in storyline systems to compute vertical orderings across time steps. However, these approaches are typically developed for graphs where nodes are discrete and layer-specific, rather than for representations that require persistent visual identity across layers. As a result, many challenges central to storylines, such as minimizing unnecessary vertical movement or preserving continuity during periods of inactivity, fall outside the purview of typical layered graph drawing research.

Another related field is the visualization of **groups or clusters** within a network. Both Vehlow et al. [VBW17] as well as Ehlers et al. [EMWR24] present taxonomies that focus on the explicit representation of group structures in network visualization, especially useful in storyline contexts where visual coherence among recurring character groups or themes is desirable.

Finally, storylines can also be seen as **a form of multivariate data visualization**, since time acts as an additional attribute displayed in tandem with nodes and their relationships. Archambault et al. [AAK*13] formalize the concept of multivariate temporal networks, which lie at the intersection of dynamic and multivariate visualization; a topic subsequently surveyed thoroughly by Nobre et al. [NMSL19]. Their work is particularly relevant here for its emphasis on the integration of time alongside other dimensions, an aspect at the heart of storyline design.

While these surveys address broader categories of network visualization, storylines represent a focused technique that brings together multiple aspects, temporal/layered structures, dynamic behavior, and multivariate attributes. As this style gains increasing attention in the visualization community, we believe the time is right to take stock of existing work and offer a dedicated survey of storyline visualization methods.

2. Methodology

In this section, we aim to provide the reader with a succinct overview of how papers were collected, how the collected set of papers was (manually) refined, and finally categorized.

Search Query In order to collect the papers for this survey, we ran a search query over all the biggest digital libraries (listed in the table below) for scientific publications. We adapted the following search query for the specific methods to search on each one of the libraries, looking for the following terms in the paper title or its metadata:

> "Storyline Visualization" OR "Storyline Drawing" OR "Storyline Chart" OR "Storytelling Chart" OR "Timeline Visualization" OR "Movie Narrative Chart" OR "Narrative Chart" OR "Temporal event sequence visualization" OR "Visual Narrative"

We also included the entirety of the Graph Drawing (and Network Visualization) conference proceedings in the process, i.e., from 1994 until 2025, as we deemed their contents highly relevant to this survey. Note that, in order to ensure as many relevant papers were collected as possible, we supplemented this (API-driven) querying with an additional final snowballing step. Table 1 illustrates the number of papers returned from the search for every queried venue as well as the manual snowballing.

Engine	Query results	Included
ACM	486	3
Sage	64	1
IEEE	376	15
DigLib	116	0
ScienceDirect	235	5
SpringerLink	130	5
Graph Drawing	1275	10
Snowballing		14
Total:		53

Table 1: *Sources of papers included in the survey*

Refinement and Tagging After this aforementioned collection step, and after establishing our exclusion criteria (documented as part of Section 1.1), our team went through an initial, manual filtering of the papers to exclude those that did not fit the scope of this state-of-the-art report (explained in Section 1.1). After establishing the categorization dimensions (Section 3), our team, in a second manual pass through the collected literature, refined and tagged all remaining papers. The results of this process are illustrated in the second column of Table 1.

Scope This survey focuses specifically on storyline visualization, i.e., the depiction of dynamic relational data, such as characters and their interactions over time. While broader fields like storytelling and narrative visualization are relevant, we limit our scope to techniques that explicitly represent dynamic networks as storylines. Papers discussing general storytelling or narrative visualization are excluded unless they include a storyline-based representation of dynamic relationships.
This narrow focus allows us to analyze the algorithmic methods, optimization objectives, and the design choices unique to storyline visualization techniques. The challenge in storyline visualizations is in maintaining a readable and interpretable layout of the evolving network of characters, which has a fundamentally different objectives to narrative visualization.

3. Classification

All collected papers were classified along a set of dimensions that reflect both how storyline visualizations are constructed and what aspects of the visualization they emphasize. The goal of this classification is twofold. First, to provide a structured overview of the design and algorithmic choices that reoccur across the literature. And second, to highlight where different approaches make fundamentally different assumptions.

Our classification focuses primarily on the **layout algorithms** used to generate storyline visualizations. In particular, we examine which **optimization objectives** are considered, how these objectives are combined or staged, and what types of **algorithmic techniques** are employed to compute character orderings and positions over time. This emphasis reflects the fact that most storyline research is driven by layout concerns, and that many papers distinguish themselves mainly through the optimization problems they formulate or the heuristics they propose to solve them.

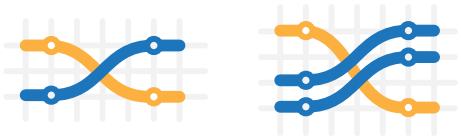

Figure 2: *Counting crossings: while the figure on the left is unambiguous (it has one crossing), the figure on the right can be interpreted in two different ways, depending on if we are counting pair crossings (in which case, it would have 2 crossings) or block crossings (in which case, it would only have one crossing).*

For completeness, we also record a number of **visual and data-related aspects** that influence how storylines are applied in practice. These include supported visual features, interaction encodings, and application domains. However, these aspects are not treated as primary classification axes. Instead, they serve to contextualize the algorithmic choices and to illustrate how different methods are tailored to specific data characteristics or use cases.

The following sections discuss each classification dimension in turn. Rather than assigning each paper to a single category, we allow papers to appear in multiple categories where appropriate, as many approaches combine several objectives or techniques.

3.1. Optimization Objectives

Most storyline visualization approaches are guided by one or more optimization objectives that capture intuitive notions of readability and visual clarity. These objectives typically formalize properties of the character lines and their relative ordering over time, such as reducing visual clutter caused by crossings, limiting unnecessary vertical movement, or controlling the overall compactness of the layout. The following subsections detail optimization objectives found in the papers included in this survey.

3.1.1. Crossings

Optimizing the order of characters in a storyline to minimize crossings is one of the central objectives of a storyline layout algorithm. A poor selection of the vertical order is one of the primary sources of clutter and reduced readability associated with increased cognitive load. When character lines or groups thereof exchange positions over time, the consequent crossings disrupt the visualization's continuity and flow of narratives, which is argued to impact a viewer's capability to follow individual characters and their development. Crossings in storyline visualizations can broadly fall into two categories: (i) *pair* and (ii) *block* crossings (see Figure 2). Related work also addresses skewness and overlap reduction, which considers geometric factors that are perceptually close to crossings since they focus on improving curve separability and reducing visual clutter.

Pair Crossings (also known as *pairwise crossings*) are by far the most common type of crossing in storyline visualizations (with 31 occurrences in the queried literature; see Figure 2-left and Table 2). Pair crossings treat each inversion or intersection between two individual character lines as a separate crossing. Pair crossings present the most common quality metric that is used to evaluate the readability of storyline layouts and frequently guides algorithm design. Optimizing for pair crossings is a central objective of layout algorithms that reduces the *total* number of these inversions or intersections between individual lines across the layout. This is a similar problem to the classical crossing minimization often discussed in layered graph drawing literature [STT81]. Pair crossings

Crossings (34)	
Pair Crossings (31)	[OM10, TM12, LWW*13, LZLC14, TBS14, Are15, KNP*15, THM15, SAHW15, GJLM16, QC16, SEA*16, LCZ17, AP17, LLLW17, Frö18, PBH18, TRL*19, LHZ*21, TLW*21, SLW*22, DNS*23, HW24, DJJ*24, WSZ*24, DCS*24, ZCY25, LYK25, KLM25, YWR*25, PLM*25]
Block Crossings (4)	[vDFF*16, vDLMW17, GDL*20, HW24]
Wiggles & Bends (21)	
Wiggle Count (17)	[TM12, LWW*13, Are15, THM15, QC16, LCZ17, AP17, LLLW17, Frö18, TRL*19, LHZ*21, TLW*21, SLW*22, DCS*24, KLM25, YWR*25, DHNW25]
Wiggle Height (6)	[LZLC14, QC16, LCZ17, Frö18, DHNW25, PLM*25]
Quadratic Wiggle Height (8)	[LWW*13, TBS14, HBK15, TRL*19, LHZ*21, TLW*21, SLW*22, YWR*25]
Compactness (19)	
Limit Whitespace (16)	[TM12, LWW*13, LZLC14, Are15, THM15, QC16, LCZ17, AP17, TRL*19, GDL*20, LHZ*21, TLW*21, SLW*22, DCS*24, KLM25, YWR*25]
Number of Layers (1)	[DNS*23]
Limit Width (1)	[DHNW25]
Separation (1)	[LYK25]
Emerging Objectives (4)	
Actor-Centric (2)	[HBK15, GDL*20]
Stability (1)	[ZCY25]
Fairness (1)	[PLM*25]

Table 2: *Optimization objectives and their counts. The tables illustrates, by tag, all the citations for the papers corresponding to such tags. Papers that have multiple tags (e.g., more than one concurrent optimization objective) appear in multiple rows of the table.*

provide precise and low-level measures of a storyline's visual complexity and readability, it favors improvements that are local rather than global [HW24]. Skewness reduction relates to pair crossing by controlling how the curves intersect (i.e., at what angle), favoring clearer intersection angles (similar to angular resolution in graph drawing). The objective is to avoid steep diagonal crossings and reduce the impact of these [PLM*25]. Considering crossings between groups of characters (rather than individuals) and the larger-scale structures of the storylines motivates research on optimizing for *block crossings*.

Block Crossings constitute a smaller body and more recent development of related work (with 4 papers in total that fit our inclusion criteria; see Figure 2-right and Table 2). The motivation behind optimizing for block crossings is that not all crossings are equal in terms of the perceived complexity of the resulting storyline layout. More specifically, two sets of lines that cross each other at a specific point and are bundled together are easier to comprehend compared to the same number of pair crossings that are distributed irregularly throughout the storyline [vDLMW17]. This consideration aligns with Gestalt principles of proximity and continuity [Wer38], according to which readers group approximately close and parallel lines together and interpret their behavior as a collective. Overlap reduction is also uniquely considered in 3D [YZC*25] and targets minimizing the visual occlusion of curves by offsetting them along the z-axis while maintaining the 2D ordering and temporal alignment. In contrast to pair crossings, in block crossings, we consider optimizing the ordering of entire groups of characters, which results in layouts that highlight collective behavior and transitions. Perceptually, this provides storyline layouts that have fewer large-scale structural changes and disruptions and more emphasis on collective (group structure) dynamics.

3.1.2. Wiggles (& *Bends*)

The (geometric) smoothness of a character's curve is another important aspect and fundamental optimization objective, often referred to as wiggle (*count* or *height*) minimization [DHNW25]: A character in a storyline changing its vertical position introduces bends or *wiggles*. Similar to crossings, few wiggles are assumed to correspond to increased readability and a more aesthetically pleasing storyline layout [AP17]. Optimizing wiggles in a storyline can consider minimizing their: (i) *count* or (ii) *height* (see Figure 3). There are also various rendering methods to displaying wiggles, illustrated in Figure 4.

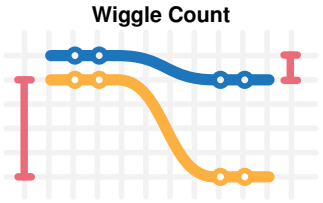

Figure 3: *Counting wiggles. Two main approaches to count wiggles: in wiggle count, we count each directional change as 1 – thus the figure above would have just 2. If, instead, we care about wiggle height, we consider the extent of the vertical movement of the line – thus, the figure above would have a total sum of 5.*

Wiggle Count captures the number of direction changes along an individual character's line throughout the storyline, i.e., the number of inflection points across a character's line(s). This problem is popular in storyline visualization literature (with 17 occurrences throughout the queried literature; see Figure 3 and Table 2). Minimizing the total number of wiggles is an important objective, resulting in overall smoother evolving curves, easily traceable paths and narratives, and reducing jitter that causes visual clutter. In storyline layout algorithms, this step is commonly performed after crossing minimization and is typically addressed using combinatorial formulations, heuristics, or independent set problems [LWW*13, THM15, AP17]. Similar to pair crossings, the wiggle count provides a low-level measure of local improvements to individual curves, rather than a global smoothing over the whole storyline layout. This presents an opportunity for optimization objectives that account for the *wiggles' height* instead of just their total count.

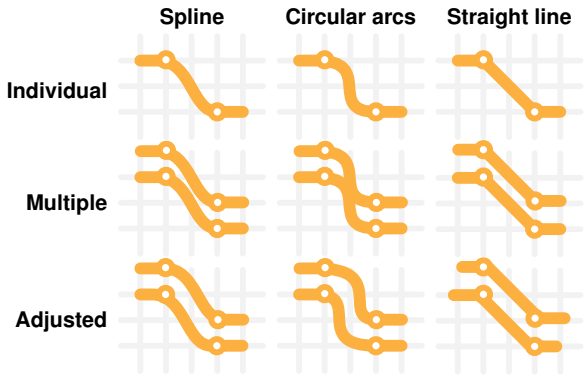

Figure 4: *Different representations of line wiggles include splines, circular arcs, and straight segments. In all cases, and particularly for circular arcs, anchor points must be adjusted to maintain consistent spacing between lines. This becomes critical when segments turn vertical, as unadjusted lines may overlap, as shown in the unadjusted circular arc example above. An discussion on spline adjustment can be found in [DHNW25].*

Wiggle Height is another metric to quantify the smoothness of the character curves, measures how far they move vertically between the events in a storyline visualization, instead of the curves direction changes (i.e., *wiggle count*). This particular criterion is discussed in five related papers, see Figure 3 and Table 2. Measuring the amount of vertical movement along each curve in literature is typically computed either linearly [Frö18], i.e., by summing up the absolute vertical distances between events, or quadratically [DHNW25], i.e., squaring the movements before summing [LWW*13]. The main distinction between the two approaches is how the penalty function is interpreted, i.e., how strongly large movements are penalized. In quadratic formulations, abrupt vertical jumps and displacements are more strongly penalized than in linear formulations. This metric represents a global description of geometric smoothness and complements wiggle count, which is focused on more local changes.

3.1.3. Compactness

Compactness describes the overall amount of space used by a storyline visualization. In contrast to crossing and wiggle minimization, which focus on the optimization of continuity, smoothness, and reduction of visual clutter with the goal of improving a storyline visualization's readability, compactness aims to avoid overly sparse layouts that contain excessive white space (see Figure 5). Such excessive white space has perceptual effects such as impacting the eye travel distance between characters and events, weakening a readers ability to connect events, and understand/explore the narratives.

Limiting Whitespace reduces the amount of unused vertical and horizontal space between character lines and their interactions (blocks/events) [TM12]. Storyline layout algorithms typically approach this objective after the crossings and wiggles have been minimized. The remaining whitespace is then tightened, shrinking any gaps between lines and events, while avoiding occlusions that might occur. It is quite a common objective of most storyline layout algorithms (16 occurrences in queried literature; see Table 2). This objective seeks to produce denser but still readable layouts that make narrative structure visible without excessive and unnecessary white space.

Spacing is an objective that refers to the distances between individual character curves or character groups (2 occurrences in literature; see Figure 5 and Table 2). *Separation* is a constraint that enforces minimum vertical distances between character curves or groups to avoid overlaps or visual clutter in denser parts of the storyline. This objective aims to ensure that characters and interactions remain distinguishable, particularly when interaction encoding relies on spatial separation rather than explicit containers or marks [LYK25] (see the spacing example in Figure 5). *Proximity*, on the other hand, pulls characters that are related or that frequently co-occur closer together by penalizing large distances between selected pairs or groups of character curves [WSZ*24]. Both objectives contribute to visual clarity by grouping meaningful elements, but their effect is most relevant when grouping is implied through spatial proximity, rather than through explicit group encodings such as enclosures or containers (see Figure 10).

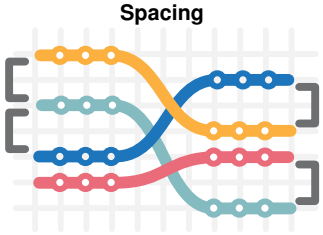

Figure 5: *In the figure above, characters that do not interact in the same time step are separated by enforced vertical spacing between their lines. This spacing visually distinguishes interactions, where characters appear together, from periods in which they do not. The requirement for spacing depends on the chosen interaction encoding (see Figure 10).*

Layout Bounds optimize specific constraints related to the drawing dimensions of the storyline visualizations. It presents an under-investigated optimization objective as only 2 papers on the topic could be identified (see Table 2). *Limiting Width* is formulated as a minimization problem to determine the minimum distance between time steps needed to maintain a smooth, x-monotone character curve that should balance between smoothness and width usage [DHNW25]. *Number of layers* presents another objective that is minimized through graph coloring techniques to reduce the total number of vertical layers needed when multiple interactions occur within the same time [DNS*23].

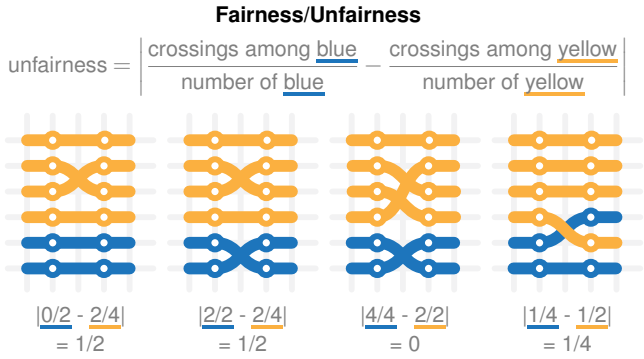

Figure 6: *Unfairness in different cases: blue and yellow represent two different groups of characters, with blue being a minority and yellow being a majority. In a fair layout unsolvable crossings would be distributed evenly between the two categories.*

3.1.4. Emerging Objectives

Recently, new kinds of semantic (character and story-driven) objectives have been introduced that address specialized narrative structures, equitable representation of characters, and preserve (interaction) stability.

Recent objectives for specialized character roles, i.e., optimizing the main character's (the *protagonist* or *ego* [EPF*24]) prominence throughout the storyline visualization, achieve this by representing the character centrally and ensuring that there are no crossings or wiggles, as illustrated in Figure 7. Hegemann and Wolff [HW24] propose a storyline visualization in which either a single individual is given particular importance or a group of characters is considered especially relevant. Accordingly, their layout method assigns greater weight to crossings and wiggles that involve the protagonists of the visualization. A similar idea is proposed in a more relaxed form (visually only, and not explicitly addressed by the layout algorithm) by Kuo et al. [KLM25].

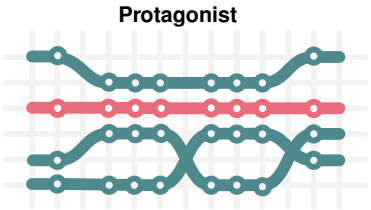

Figure 7: *The pink line represents a protagonist in the visualization: the optimization criteria are done so that the main character is given more importance, central to the visualization and deprived (as much as possible) of crossings, in addition to having no wiggles. [HW24]*

When characters simultaneously participate in multiple interactions at the same time, a very common scenario, a different representation compared to established straight character lines is required. Character trees have been introduced as structured representations that aim to address the challenge of characters participating in multiple interactions (*ubiquitous characters*, see Figure 8) [GDL*20]. This approach emphasizes a character's participation as branching behavior and introduces new objectives for optimization, such as actor planarity, branch continuity, and branch degree minimization.

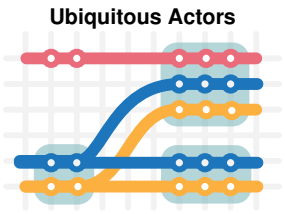

Figure 8: *Ubiquitous actors: yellow and blue both belong to two different groups simultaneously. In [GDL*20], this is represented as the character being a tree, with branching paths that represent their participation in multiple simultaneous groups.*

Fairness objectives have been recently introduced [PLM*25] that formalize the equitable distribution of visual complexity across groups of characters (see Figure 6). Different modes of optimization are introduced: (i) *fairness mode* ensures balanced distribution regardless of the group's size, whereas (ii) *focus mode*, on the other hand, prioritizes specific character groups.

Finally, stability objectives aim to preserve a reader's mental map during interactive exploration. This is achieved by minimizing the storyline's layout changes when filters or selections are changed [ZCY25].

3.2. Visual Structures & Entities

Apart from optimizing the layout for the various objectives discussed in Section 3.1, storyline visualization approaches differ substantially in terms of the visual and structural elements used to convey the underlying data or narrative. An overview of the structural elements is available in Table 3. These elements determine how characters, interactions, relationships, and other facets, e.g., attributes, locations, other timelines, are encoded, placed, and visualized. Optimization approaches aim to improve the readability, whereas the visual structures and features focus on *what* information is being displayed, *where* is it placed in the visualization, and *how* is it encoded. In storytelling and narrative visualization these aspects act as important considerations to make in order to support the interpretation of the data [SH10].

Labels (28)	
Character Labels (26)	[OM10, KCH10, TM12, LZLC14, TBS14, Are15, THM15, SAHW15, HBK15, LLLW17, OK17, PC18, SBB*18, PBH18, TRL*19, AXP*22, TLW*21, SLW*22, HAB23, CN23, WSZ*24, ZCY25, LYK25, KLM25, YWR*25, PLM*25]
Event Labels (5)	[TM12, SAHW15, LHZ*21, WWD23, CN23]
Annotations (14)	
General Annotations (4)	[SEA*16, OK17, SBB*18, WWD23]
User Annotations (3)	[TRL*19, TLW*21, SLW*22]
Audio Annotations (1)	[CLMY13]
Locations (8)	[TM12, LWW*13, QC16, LCZ17, OK17, HAB23, WWD23, ZCY25]
Character-Specific Encodings (6)	
Character Groups (4)	[ZWQ*15, PC18, LYK25, PLM*25]
Character Trees (1)	[GDL*20]
Genealogy (1)	[KCH10]
Interaction Encodings (5)	
Multi-Character Interactions (1)	[WSZ*24]
Multi-variate Encodings (4)	[ZWQ*15, PC18, YZC*25, KLM25]
Extensions (4)	
Non-Interacting Characters (2)	[OM10, TBS14]
Multiple Timelines (1)	[PBH18]
Uncertainty (1)	[KCH10]

Table 3: *Visual Structures & Entities. The table illustrates, by tag, all the citations for the papers corresponding to such tags. Papers that have multiple tags (e.g., more than one concurrent optimization objective) appear in multiple rows of the table.*

3.2.1. Labels & Annotations

Manual or automatically generated labels and annotations can provide explicit, additional context and information. These are one of the most common visual elements that are present in storyline visualizations. Surprisingly, however, only 28 occurrences across the queried literature were counted (see Table 3). More specifically, these annotations revolved around characters (26 occurrences), annotations (4 occurrences), and events (5 occurrences). Note that these occurrences were counted if the papers in question featured some form of labels in their visualizations; not that the paper explicitly featured an optimization of label placement. Generally, these elements support identifying specific narrative entities, i.e., characters, interpreting significant events and interactions, as well as communicating any contextual or explanatory information that cannot be encoded by the visualization alone.

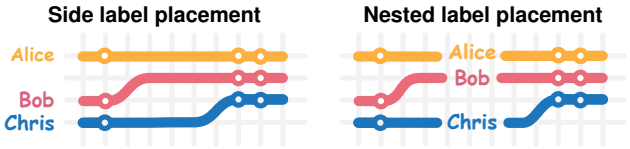

Figure 9: *Different label placements for characters: labels can be placed outside the visualization or embedded directly within the lines. Embedded labels can save space and help maintain character identity, but require sufficient straight segments in the layout to accommodate the text.*

Figure 10: *Different ways to represent events where multiple characters participate. The can be represented through an elongated mark (left picture), through any kind of container (center picture), or just identified through the separation between lines (right picture). In the latter case, a minimum space between lines needs to always be enforced, thus the aspect needs to be taken into account in the layout algorithm (see Figure 5).*

Characters Labels Character labels primarily serve the goal of explicitly annotating the main (or most significant) entity/entities within a storyline to guide the viewer through the narrative. The majority of the approaches places labels adjacent to (or directly atop of) their respective lines, either by embedding them inline when the spacing allows it, placing them to the side of the visualization, or offsetting and angling them to reduce clutter (see Figure 9) [KCH10, LZLC14, LLLW17, SBB*18, AXP*22, TLW*21, SLW*22, WSZ*24, LYK25, YWR*25, PLM*25]. Several approaches selectively reduce labeling density, i.e., omitting labels completely, by computing and ascribing a significance score to each character. This is commonly achieved by identifying long-living or salient characters within storylines [KLM25] or by allowing viewers to interactively/manually reposition labels in order to avoid overlap and visual clutter [ZCY25].

Event Labels Events are additionally labeled or annotated to make the interactions throughout the development of a narrative explicit. These labels are commonly found along the temporal x-axis. However, some approaches use bounding boxes, background segments, text, or containers that surround converging lines, conveying duration and participation explicitly [TM12, SAHW15, OK17]. Other approaches utilize the visual properties of line convergence and divergence to draw attention (and use these as) implicit event encodings, i.e, character interactions; commonly supplemented with textual annotations, interactive tooltips, or linked views showing metadata and details associated with the events [LHZ*21, WWD23, CN23]. These approaches attempt to balance the readability of the entire storyline with access to detailed information about the narrative and context.

3.2.2. Annotations

Annotations extend beyond basic labeling of the storyline's core elements and entities by attributing or emphasizing explanations and contextualizing narrative elements. They represent the second most common visual structure within storyline visualizations (with 14 occurrences across literature; see Table 3). Such elements are often employed as a method of externalizing important details/facts and reducing the need for viewers to retain this information in working memory.

Audio & User Annotations A number of approaches explore various modalities, such as, audio annotations [CLMY13], which was shown to i) improve recall of narrative elements when viewers revisited the storyline visualization, and ii) allow users to remember key moments through audio alone. Other approaches provided methods for viewers to manually add, annotate, or modify labels and other visual elements supporting the presentation [TRL*19, TLW*21, SLW*22]

Locations Location encodings aim to provide spatial context, e.g., movie scenes and locations, for events and character interactions. Commonly employed approaches include shaded background regions, colored contours, and bounding boxes that visually group entities located at the same place together [TM12, LWW*13, OK17]. Recent approaches investigate integrating imagery, (mini-) maps, or other linked views depicting geographical information alongside the storyline visualization [HAB23, WWD23]. Trees and tables are further used to provide an overview and support exploration of location hierarchies and metadata [QC16, LCZ17, ZCY25].

3.2.3. Character-Specific Encodings

Character-specific encodings enrich storyline visualizations by representing roles, group memberships, or further structural relationships beyond the main narrative. We have identified various structural elements and encodings, extending past labels and annotations, that we have placed in this category (6 publications in our literature collection; see Table 3).

Character Groups Character groups and clusters represent collective behavior or group interactions. Groups are commonly represented by spatial proximity or shared bounding boxes sharing the same visual attributes, e.g., color, or line/glyph style (see Figure 10 and Figure 11). This similarity enables viewers to quickly identify the groups and the joint participation of individuals within them [TLW*21, PC18, LYK25]. Color and line styling are also used to encode categorical distinctions such as team memberships [LYK25, ZWQ*15] or narrative focus [PLM*25]. Clustering provides further higher-level summaries of character organization, often aggregating characters into clusters and visualizing these as bundles or embedded glyphs [LLLW17, PC18].

Character Trees & Genealogy To tackle the problem of visualizing concurrent character participation in multiple interactions, [GMD*23] propose a design that generalizes the storyline visualization to tree structures: character trees. Each character starts

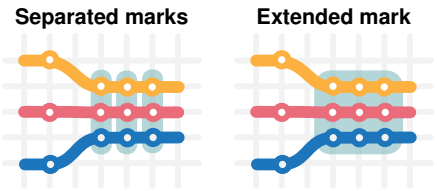

Figure 11: *Groups spanning multiple time steps. Interactions can be shown either as separate marks at each time step (left) or as a single mark spanning multiple time steps (right). The latter requires the layout algorithm to maintain consistent vertical positions for all involved characters to preserve visual continuity.*

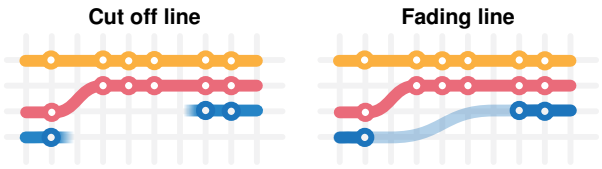

Figure 12: *When characters are absent at a given time step, their lines can either be interrupted entirely (left) or shown as a faint continuation indicating temporary absence (right). In the latter case, these faded segments must still be considered by the layout algorithm to avoid unnecessary crossings and visual clutter.*

with a line that branches when that character participates in multiple groups or interactions simultaneously, while preserving the temporal and visual continuity. In genealogy, storylines have been employed to encode familial relationships and lifelines (where the horizontal extent represents the life span) [KCH10]. Branching and linking are also employed here to encode marriages and lineage, where as other visual attributes, i.e., symbols and color gradients, are used to encode demographic information and uncertainty.

3.2.4. Interaction Encodings

Interaction encodings describe how joint activities or (multiple) character participation in events are visually represented. These type of encodings typically express multiple character relationships (see Figure 10) or multivariate encodings using glyphs (see Figure 14). This category represents special cases and circumstances where interactions between characters need to be explicitly encoded as the focal point of the visualization with 5 publications in this category (see Table 3).

Multi-Character Interactions These interactions are commonly depicted using extended marks, explicit containers, or through enforced line separation (see Figure 10). When these interactions span multiple time steps, they are duplicated and shown as repeated marks or continuous regions with stable ordering [TM12, TLW*21, SBB*18] to preserve visual continuity (see Figure 11). Wang et al. [WSZ*24] have complemented storyline visualizations with a matrix-based glyph that encodes character interactions. This relationship matrix encodes the character interactions as compact grids and reduces crossings that could occur.

Multi-variate & Multi-relational Encodings Storylines are frequently augmented with additional data facets, such as multiple attributes and multiple relationships between characters. These facets are often visualized in supporting visualizations, i.e., using coordinated linked views or small multiples, such as network diagrams and other standard charts [PC18, KLM25, ZWQ*15]. Yao et al. [YZC*25] propose a 3D storyline visualization, where the spatial and temporal aspects are encoded using a space-time-cube integrating geographic movement with narrative progression.

3.2.5. Extensions

Other extensions to storyline visualizations do not neatly fall into the other existing categories discussed above (4 occurrences; see Table 3). We therefore discuss these here in their own section, describing metaphors that address the narrative completeness, uncertainty, or alternative narrative structures.

Non-Interacting Characters considers characters that are inactive or peripheral to the central narrative being conveyed. In existing approaches, these characters are still retained to preserve narrative context, but are often visually de-emphasized. Common strategies include pushing character lines toward the periphery of the layout, e.g., the top or bottom of the storyline (see Figure 7), simplifying their visual appearance, or allocating them reduced vertical space to compress their impact on the main layout [OM10, TBS14]. This allows the visualization to maintain narrative completeness without overwhelming the depiction of the primary characters and interactions [PC18]. Related designs consider characters that temporarily vanish from the narrative. Some systems remove a character's line entirely during periods of absence, allowing the layout to reclaim space and reduce overall height. Other approaches avoid abrupt disappearance and instead preserve visual continuity. Tapaswi et al. [TBS14], for example, used dashed lines to indicate periods in which a character is inactive. This choice has direct implications for layout optimization: preserving continuity requires the algorithm to maintain vertical slots for absent characters, whereas removing inactive characters enables more aggressive space compaction (see Figure 12).

Multiple Timelines often show alternative or hypothetical narratives rather than a singular timeline. These approaches are often centered around movie scenarios and support illustrating or comparing the narrative paths and outcomes [PBH18].

Uncertainty encodings aim to faithfully represent the missing or uncertain data attributes, i.e., time points or intervals. Kim et al. [KCH10], for example, utilize gradients and fading colors to communicate uncertainty related to temporal spans and support cautious interpretation of the data.

3.3. Temporal Model

Most storyline visualizations adopt a discrete (*timesliced*) temporal model, where the horizontal axis is divided into a sequence of time steps and each interaction is assigned to exactly one of these steps. In this setting, time advances in uniform increments and interactions are treated as instantaneous events, even when they concep-

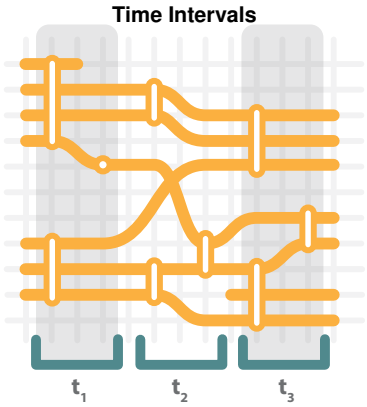

Figure 13: *This figure illustrates an example from time-interval storylines [DNS*23]. In every time interval, illustrated by segments at the bottom of the visualization, events occur simultaneously.*

tually span longer periods. This model simplifies layout and optimization, as it allows character orderings and positions to be computed independently for each time step.

A notable deviation from this assumption is introduced by Dobler et al. [DNS*23], where the horizontal axis is partitioned into time intervals rather than single instants (see Figure 13). Each interval occupies a certain amount of horizontal space and may contain multiple events that occur simultaneously, e.g., in the same year. While these events share the same temporal position, the allocated width allows them to be visually separated and arranged within the interval. With the exception of this variation, the temporal models used across the surveyed literature remain consistent and rely on discrete time to structure the layout process as a layered graph.

3.3.1. Multivariate Data Encoding

Storyline visualizations can be interpreted as a form of multivariate temporal network visualization, as they inherently encode multiple data dimensions such as time, entity identity, interaction membership, and character lifespan. Despite this, most approaches consider only interactions and time, and do not integrate additional attributes into the visualization.

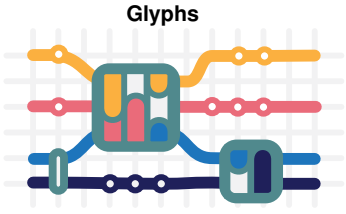

Figure 14: *In the figure above, adapted from [WSZ*24], multivariate interaction data is encoded using glyphs. In this example, the glyph represents script-level information by showing which characters interact with one another through vertical line elements.*

Only a small number of papers explicitly integrate additional multivariate data beyond this standard set. A notable example hereof is the work by Wang et al. [WSZ*24], which overlays *glyphs* on interaction events to encode extra semantic information. Here, glyphs represent subject–predicate–object relationships extracted from scripts, allowing the visualization to convey not only who interacts with whom, but also the nature of these interactions (see Figure 14). Importantly, this additional data is encoded visually but does not directly influence the layout optimization. Another example of such multivariate visualization appears in the fairness-driven storyline layouts by Piselli et al. [PLM*25], where characters are assigned to *categorical groups*, e.g., factions, that are not derived from interaction structure alone. These group memberships introduce an additional attribute that is taken into account by the optimization process, with the goal of distributing crossings and wiggles evenly across groups.

> ***Takeaways:*** The optimization objectives described above are not independent and, in many cases, cannot be satisfied simultaneously. For instance, enforcing adjacency of interacting characters may introduce crossings, while minimizing crossings can separate groups or increase vertical movement. As a result, most approaches do not aim to optimize all criteria at once. Instead, they prioritize subsets of objectives or decompose the problem into stages, as seen in pipeline-based methods. Storyline layout is fundamentally a trade-off-driven process, where different applications may require different compromises between readability, compactness, and structural fidelity.

4. Layout Algorithms

Here, we discuss different types of layout algorithms used to compute storyline visualizations. From our collection of papers, we can formulate a split between two major types: heuristic-based layout algorithms, and exact ones. The algorithm chosen for the visualization is deeply tied with the chosen metrics the authors aim to optimize (see Section 3.1). Further, the layout process is often divided into several steps (see Section 4.1.1), and the different steps are targeted at different quality metrics. We describe algorithmic approaches for the different steps separately.

Formal Definitions and Notation

$[k]$	the set $\{1,2,\dots,k\}$
$\binom{S}{k}$	the set of k-element subset of S
$\mathcal{C} = \{c_1, c_2, \dots, c_n\}$	the set of characters
$\mathcal{I} = \{I_1, I_2, \dots, I_m\}$	the set of interactions
$T = \{1,2,\dots,\ell\}$	the set of time steps
$A(t)$	the set of characters alive at time step t
$c \in I$	character c is part of interaction I
$\mathcal{I}(t) \subseteq \mathcal{I}$	the set of interactions at time step t
π_t	the ordering of characters at time step t
$c \prec_{\pi_t} c'$	c is before c' in π_t
$y_{t,c}$	the y-coordinate of character c at time step t

Table 4: *Definitions of important concepts used through the rest of this section and their notation.*

4.1. Heuristic Approaches

In this section, we present heuristic approaches in the literature and discuss the typical stages of optimizing storyline visualizations. We further provide details about the specific heuristics used, i.e., layered sweep, wiggle minimization, gradient descent, force-directed, and multi-dimensional scaling methods (see Table 5).

Heuristic Approaches	
Pipeline Approach (16)	[DCS*24, LZLC14, LLLW17, YWR*25, Are15, LCZ17, LHZ*21, SLW*22, TRL*19, LWW*13, KLM25, TLW*21, QC16, THM15, TM12, AP17]
Layered Sweep (18)	[DCS*24, LZLC14, LLLW17, YWR*25, OM10, LCZ17, LHZ*21, SLW*22, ZCY25, TRL*19, LWW*13, KLM25, TLW*21, QC16, THM15, TM12, AP17, PBH18]
Greedy (8)	[LWW*13, THM15, TM12, KLM25, DCS*24, OM10, vDFF*16, HW24]
Local Search (4)	[PLM*25, GDL*20, THM15, AP17]
Gradient Descent (1)	[TBS14]
Genetic Algorithm (1)	[TM12]
Force-Based (2)	[SAHW15, SEA*16]
MDS (2)	[ZCY25, LLLW17]
Other (4)	[TLW*21, GDL*20, HW24, LYK25]
Specific to Wiggle Count	
Longest Common Subsequence (12)	[DCS*24, LLLW17, YWR*25, LCZ17, LHZ*21, SLW*22, TRL*19, LWW*13, KLM25, TLW*21, QC16, THM15]
Maximum Independent Set (2)	[Are15, AP17]

Table 5: *Heuristic approaches*

4.1.1. Pipeline Approach

Computing a storyline layout is a complex problem with many optimization criteria that depend on different aspects of the visualization. As dealing with all criteria at once is challenging, the layout process is often divided into several steps, each of which computes a different part of the layout and optimizes a subset of the quality criteria.

This usually comprises the following three steps (see Figure 15).

1. Compute a *topological layout* that consists of a vertical order of characters π_t, for each time step t. Coordinates are not set yet. This step usually serves to minimize crossings, and may inherently reduce the number of wiggles.
2. **(optional)** In an optional second step, some layout algorithms fix other parts of the layout. In most cases where this happens, this step reduces the wiggle count by forcing an equality of the y-coordinates of a character between consecutive time steps.
3. The last step computes the full layout while respecting those parts of the layout computed in the previous steps. That is, in this step real-valued coordinates are computed for each character and for each time step, while respecting the character orders from step 1 and the potential restrictions from step 2. This step usually minimizes whitespace and some definition of wiggle.

What follows is a description of such pipeline approaches in chronological order. Tanahashi and Ma [TM12] incorporate a pipeline approach into their genetic algorithm. They do so by applying steps 1 and 3 of the pipeline to a genome of their algorithm to evaluate its fitness; for more details, see Section 4.1.3. This approach is ultimately extended by the authors themselves [THM15], i.e., greedily computing a single genome and extending the pipeline to include a wiggle count minimization on the longest common subsequence (Section 4.1.6).

Storyflow [LWW*13] is the most standard example of a pipeline approach, which applies steps 1-3 as described above: Crossings are minimized in step 1 with a layered sweep (Section 4.1.2), wiggle count is minimized in step 2 with an approach based on the longest common subsequence problem (Section 4.1.6), and step 3 reduces squared wiggle height and whitespace with linear programming (Section 4.2.1). Several papers adopt this approach. Liu et al. [LLLW17] preface their pipeline with a cluster assignment step based on multi-dimensional scaling (Section 4.1.10). Tang et al. [TRL*19] extended the layout process with constraints imposed by a user and each step of the Storyflow pipeline is adapted accordingly. This approach has since been adapted multiple times [TLW*21, SLW*22]. Lu et al. [LHZ*21], Ye et al. [YWR*25], and Qiang et al. [QC16] all apply the Storyflow algorithm directly without modifications. Deng et al. [DCS*24] only apply the first two steps of the Storyflow pipeline but add a new greedy whitespace reduction step as a third step (Section 4.1.4). Lastly, Kuo et al. [KLM25] adopt parts of the Storyflow pipeline, including slightly constrained steps 1 and 2, but add a new greedy compaction in step 3.

An approach applying only steps 1 and 3 can be found in the work of Lu et al. [LZLC14].

Arendt [Are15] uses the Graphviz library to perform their first pipeline step. In a second step, wiggle count is minimized by reducing the problem to that of a maximum independent set (Section 4.1.7). The last step is again done by linear programming. Arendt and Pirrung [AP17] apply the same steps, but local search to further reduce crossings is incorporated into step 2.

Finally, Dobler et al. [DNS*23] make use of a quite different pipeline approach. They do not assign interactions to fixed time steps but instead give them a partial order. The pipeline, thus, looks as follows: First, interactions are assigned to partially ordered layers (by a coloring approach) in order to minimize horizontal space; The second step orders layers based on a traveling

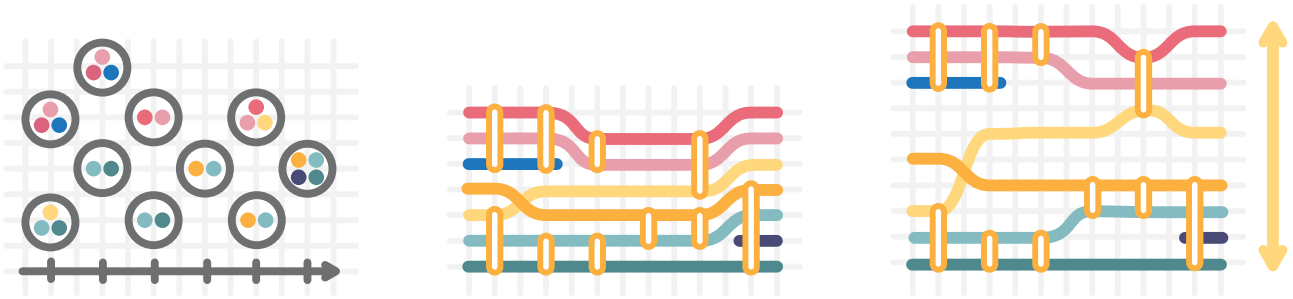

Figure 15: *Pipeline approach. The input (left) is a set of interactions over time. In the first step (middle), the vertical orderings π_t, $t \in T$, of characters are computed for each time step, without assigning final spacing. In the final step (right), geometric refinement minimizes wiggles and whitespace while enforcing separation constraints, with improved separation enabling further reduction of wiggles.*

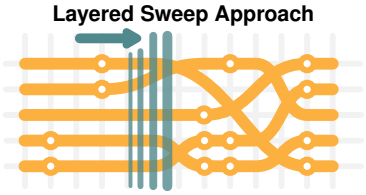

Figure 16: *Layered sweep approach. The algorithm iteratively processes layers to reduce crossings, sweeping through the visualization either in a single direction or in alternating passes. This technique is commonly used for crossing minimization in layered graph drawing [STT81].*

salesperson (TSP) formulation, placing similar layers next to each other; The last step uses the integer linear program of Gronemann et al. [GJLM16] to minimize crossings (see Section 4.2.3).

> **Use a pipeline approach when** you have several goals that clash, and you're okay solving them step by step instead of all at once. This separates the problem into steps: first fix the structure (like crossings), then refine it (like wiggles/spacing). The downside is that early decisions limit what can be improved later.

4.1.2. Layered Sweep

By far the most papers identified in this literature review use a method from graph drawing to compute the order of characters at each time step in order to minimize the number of pair-wise crossings (see Figure 16). Usually, this is incorporated in the context of a pipeline approach, where the computed order serves as the input for the next step of the pipeline. This method stems from layered graph drawing [STT81], and then computes character orders in an iterative way, starting with the first time step, and computing orders for each additional time step t based on the order of the previous time step $t-1$. This approach can then be carried out in reverse order again, i.e., computing orders for t based on $t+1$, iteratively from the last time step to the first.

Usually, the initial order of the first time step is set randomly or by some criterion. The ordering of each consecutive time step tries to keep relative orders of pairs of characters as similar as possible to the previous time step. The most popular approach for this is the following: Given an ordering π_t of characters at time step t, the aim is to compute the order π_{t+1}. Each character at $t+1$ receives a weight which corresponds to its position in π_t. An interaction receives a list of weights corresponding to the positions of its characters in π_t. Then, the list of characters and interactions are sorted by their weight and by the average/median weights, respectively. Characters inside an interaction can then again be sorted by this weight.

This kind of layered sweep approach is adopted in a plethora of papers. The first notable example hereof was authored by Ogawa and Ma [OM10]. They do not only compute the order of characters this way, but also the coordinates of characters at each time step. Starting with a random order of characters at the first time step (equally spaced along the vertical axis), they then, for each next time step, calculate the desired y-coordinate for each interaction as the average of y-coordinates of its characters in the previous time step. The desired y-coordinate of characters not present in an interaction is carried over from the last timestep. Characters and interactions are then assigned y-coordinates that are as close as possible to the desired y-coordinate greedily, retaining their relative order.

Tanahashi and Ma [TM12] apply this method as part of their genetic algorithm. Here, a genome assigns interactions and characters to vertical slots, and a pipeline method, including a Sugiyama-style sweep, evaluates a genome (refer to Section 4.1.3 for more details). Each slot can only accommodate a single interaction for each time step. Tanahashi, Hsueh, and Ma [THM15] apply a similar algorithm. However, instead of a genetic algorithm, a greedy algorithm computes slot assignments. Both papers do not state how the initial order of the first time step is computed. As slots only contain a single interaction at each time step, and slots already have a fixed ordering, both approaches simply carry over relative orders of characters from the previous time step within a slot. Characters that appear are placed at either the top or bottom-most position within their slot.

The Storyflow pipeline [LWW*13] uses the standard barycenter method. Further, the computed orders must comply with the fixed vertical order of locations, which is computed at the first time step of the pipeline. This is achieved by applying the barycenter heuristic for each location separately for each time step. The al-

gorithm uses a random initial order which conforms to the location and interaction constraints. The Storyflow method for crossing minimization has since also been adapted into several visualizations [TRL*19, LHZ*21, TLW*21, SLW*22, DCS*24, KLM25, YWR*25, QC16, LCZ17]. Notably, Tang et al. [TRL*19] also consider such a variant, where a subset of relative (user-defined) orders of character pairs is fixed. Here, they use an extension of the barycenter method from graph drawing tailored towards this setting [For04]. Similarly, Lu et al. [LZLC14] adopt an equivalent approach based on the barycenter method, also incorporating a fixed order of locations. The initial order is not discussed.

Arendt [Are15] models the storyline as a directed acyclic graph where arcs are directed from a time step to the next, and where interactions and characters outside an interaction are modeled as a single vertex. Vertices can have multiple incoming or outgoing arcs if they correspond to an interaction, as the characters from an interaction do not necessarily belong to the same interaction in the previous or next time step. They then use Graphviz to compute the orderings of characters. Usually, the applied algorithm corresponds to an adaptation of the Sugiyama-framework [STT81], which applies the barycenter or median method.

Arendt and Pirrung [AP17] are the first ones to use the median instead of the barycenter for crossing minimization, applied in a Sugiyama-style sweep. The initial order is not discussed. Padia et al. [PBH18] use the barycenter method for computing character orders, and then use d3.js to draw the storyline. Finally, Zhang, Chen, and Yong [ZCY25] apply a variant of the barycenter method as a post-processing step to sort characters within locations.

> **Use layered sweep when** your main goal is to reduce crossings quickly. It is fast and scales well, which is why it is widely used. However, it only looks at local changes, so the result is not guaranteed to be globally good.

4.1.3. Genetic Algorithm

Tanahashi and Ma [TM12] propose a genetic algorithm. The algorithm works with slots, which are disjoint horizontal strips in the visualization having a vertical fixed order. A specific genome assigns each so-called *interaction session* one of the available slots, while there cannot be two interaction sessions in a single slot for a time step. An interaction session is essentially, either an interaction having a duration with start and end time step, or a single character over a maximal time span where it is not part of any interaction. The paper also optimizes the length of a genome by sometimes merging interaction sessions if it is always optimal to place them into the same slot. Thus, the number of slots is set as the maximum number of interaction session over all time steps. The approach is extended to incorporate a variable order of locations, which is also encoded into the genome.

The fitness of a genome is then defined by first computing a layout of the storyline by applying an algorithmic pipeline to the slot assignment, and then computing a weighted sum of wiggle count and crossings. The genetic algorithm classically maintains a population of genomes, and in each iteration applies crossover, mutation, evaluation, and selection phases.

The algorithmic pipeline to compute the layout of a genome first applies a Sugiyama-style sweep (see Section 4.1.2 for a discussion), and a greedy approach to reduce whitespace by shifting sets of interaction sessions downwards as long as possible (see Section 4.1.4), and by not altering the wiggle count.

> **Use genetic algorithms when** the problem is messy and you want to try many different solutions to see what works. They explore a wide range of possibilities and can combine multiple objectives. This flexibility comes at the cost of higher computational effort and less predictable outcomes.

4.1.4. Greedy

Many layout algorithm incorporate some greedy aspect. That is, when an iterative layout algorithm is faced with a choice, it always chooses the locally best one, even when this might not be globally optimal. The applications of the greedy paradigm are vastly different between the layout algorithms, so we describe them separately, based on which aspect of the visualization they are optimizing.

Greedy Crossing Minimization Van Dijk et al. [vDFF*16] propose a heuristic for minimizing block crossings in storylines, where each interaction consists of exactly two characters. The algorithm starts with an arbitrary initial order of the characters and the list M of interactions sorted by time step. Then, in each step, a block crossing is computed, such that it *supports* the maximum number of yet unsupported interactions from the beginning of M. That is, for each supported interaction, the two corresponding characters appear consecutively after the block crossing is applied. Because each interaction has only two characters, this process of computing *greedy block crossings* can be done in time $\mathcal{O}(|\mathcal{C}| \cdot |\mathcal{I}|)$.

Hegemann and Wolff [HW24] apply a very similar approach for their storylines with a protagonist. The difference is that interactions can contain arbitrarily many characters. Again, they start with a random start permutation and apply the block crossing, which optimizes a local score. This is done until all interactions are supported – in the same sense as in [vDFF*16].

Greedy Coordinate Computation The approach of Tanahashi and Ma [TM12] incorporates a greedy whitespace reduction step, that shifts groups of interaction sessions (see Section 4.1.3) down as long as possible.

Deng et al. [DCS*24] determine character coordinates by first assigning initial positions at the first time step. Then, for each subsequent time step, the method greedily attempts to keep character curves straight; when this is not possible, certain characters are shifted vertically to allow the placement of the currently considered character. In Spreadline [KLM25], a similar approach is used; however, the description is not complete.

Other Greedy Approaches Tanahashi, Hsueh, and Ma [THM15] extend the approach from [TM12]: While in [TM12] a genetic algorithm is used to assign interaction sessions to slots (see Section 4.1.3), in [THM15] this is done with a greedy approach; that is, interaction sessions are assigned one by one to the best possible slots with regard to crossings. This approach is further extended by introducing new slots in between two other slots during this greedy

assignment, if this new slot would be better for the interaction session.

Storyflow [LWW*13] uses a greedy algorithm to determine a vertical order of locations in their variant, where each interaction happens at a location. The locations are processed iteratively in order of the number of interactions they contain, starting with the location with the most interactions; then, in each step, the location is inserted at a position that produces the least amount of crossings.

> **Use greedy methods when** you want a fast solution and don't mind that it may not be the best overall. They make simple choices step by step, which makes them efficient. But once a decision is made, it is usually not revisited, which can lead to poor global layouts.

4.1.5. Local Search

Local search is a heuristic technique to improve a solution – possibly computed by another heuristic – by iteratively performing small changes to the solution as long as this improves the solution quality.

Tanahashi, Hsueh, and Ma [THM15] employ a local search for adjusting their layout. As described in Section 4.1.4 and Section 4.1.1, they first compute a slot assignment of interaction sessions and run a pipeline that computes the layout based on these slot assignments. After this initial layout with the initial slot assignment is computed, the authors try to locally improve this slot assignment, run their pipeline again, and see if this improves the quality metrics. In particular, they first compute a weighting of subsets of interaction sessions of a slot – resulting in subsets of interactions sessions for every slot; the weight corresponds to a sum of the wiggle count and crossings of line segments starting and ending in these interaction sessions. The subsets are then processed by decreasing weight. In each step, it is tested if the currently processed subset can be moved to a new slot such that this improves the quality metrics after running the whole pipeline again. If this is the case, the new solution is accepted and the process is iterated. The authors also employ multi-threading to speed up this process.

Arrendt and Pirrung [AP17] apply a local search for reducing crossings. That is, after initial orderings of characters are computed by their sweep approach (Section 4.1.2), they swap vertically adjacent characters of a layer as long as this decreases crossings.

Di Giacomo et al. [GDL*20] consider storylines where a character is represented by a tree instead of a single curve. After the initial layout is computed, the authors employ a heuristic which exchanges tree edges as long as this results in fewer crossings.

Piselli et al. [PLM*25] propose a post-processing wiggle reduction. Their mixed-integer-linear program first computes a layout where characters are equally spaced vertically. If this is not a requirement, a local search moves character groups up or down as long as this improves the wiggle height.

> **Use local search when** you already have a decent layout and want to improve it little by little. It works by making small changes and keeping the ones that help. This can refine a layout nicely, but it can also get stuck in a solution that is only locally good.

4.1.6. Longest Common Subsequence Wiggle Count Minimization

Some approaches use a weighted longest common subsequence approach for minimizing wiggle count. The *longest common subsequence* problem is given two strings $A = a_1 a_2 \dots a_p$ and $b_1 b_2 \dots b_q$ and a weighting function $w\colon A \times B \to \mathbb{R}$ and the question is to find a common subsequence $a_{i_1} a_{i_2} \dots a_{i_r}$ of A and a subsequence $b_{j_1} b_{j_2} \dots b_{j_r}$ of B maximizing $\sum_{k=1}^{r} w(a_{i_k}, b_{j_k})$. Note that this is not the original definition of the weighted common subsequence, but a variant used in the storyline papers.

This problem has a simple $\mathcal{O}((n+m)^2)$ polynomial time algorithm. The idea is to use positive weights for characters that can be drawn without wiggle. This approach works once the order of characters in each time step is known.

The Storyflow algorithm [LWW*13] was the first to use this approach: given a fixed order of characters of each time step, the idea is to minimize wiggle counts between each pair of two consecutive time steps locally. We will see later why this works globally. First, the algorithm splits the orderings π_t and π_{t+1} of two consecutive time steps into two orderings of sessions: each session is either a single character outside an interaction, or all characters from an interaction (see Section 4.1.3 and Section 4.1.4 where sessions span over multiple time steps). Given the two sequences $A = a_1 a_2 \dots a_p$ and $B = b_1 b_2 \dots b_q$ of interaction sessions at time steps t and $t+1$ respectively, the idea is to define a weight between each pair of interaction sessions a_i and b_j that reflects how many character curves can be straightened if these two interaction sessions are aligned. The Storyflow algorithms chooses the weight

$$w(a_i, b_j) = \text{straight}(a_i, b_j) + \alpha \cdot (1 - \left|\frac{i}{p} - \frac{j}{q}\right|),$$

where $\alpha = 0.1$ and $\text{straight}(a_i, b_j)$ is the maximum number of characters curves that can be straightened when aligning a_i and b_j. The second term should also decrease wiggle height. It is not described how $\text{straight}(a_i, b_j)$ is computed, but it is easy to see how this can be done, as the characters from a_i and b_j must have respective equidistant vertical spacing. As only sessions are aligned, by increasing vertical spacing, this local wiggle count minimization can also be achieved locally. Note that this approach does not achieve optimal wiggle count minimization, as it does not account for the possibility that multiple sessions of one time step could be aligned with one session of the next time step, and the other way around.

Several papers adopt this approach [THM15, LLLW17, TRL*19, LHZ*21, TLW*21, SLW*22, DCS*24, KLM25, YWR*25, QC16, LCZ17], either as part of their own algorithm or because they apply the Storyflow algorithm for their visualization.

> **Use methods based on longest common subsequence when** you want to keep characters in a similar order over time to reduce wiggles.

4.1.7. Maximum Independent Set Wiggle Count Minimization

Two papers use a reduction to maximum weighted independent set to minimize wiggle count. The first is by Arendt [Are15], however there is no description on how this procedure works exactly.

Similarly, Arendt and Pirrung [AP17] implement such an approach. The reduction works as follows: As described in Section 4.1.2, first crossings are minimized by modeling interactions and characters without an interaction as vertices in a directed acyclic graph, which they call G_{flow}. From this, an ordering of the vertices of each time step is computed. The output of this step serves as input for the wiggle count minimization. Arendt and Pirrung then compute a new graph which they call G_{align}. In G_{align}, each vertex corresponds to a directed arc in G_{flow}, and taking such a vertex into the independent set means that the corresponding interaction/character is aligned. The weight of such a vertex is the number of character curves that could be straightened between the corresponding two time steps. It is not described how this can be determined, but we suspect that this can be computed with dynamic programming in polynomial time. Edges $\{u,v\}$ in G_{align} model two arcs that cannot be aligned simultaneously. That is, consider the arcs (a,b) and (a',b') in G_{flow} that correspond to u and v respectively. If the relative order of a and a' is not the same as the relative order of b and b', then it is impossible to align both arcs and the edge $\{u,v\}$ is added to G_{align}. To compute the maximum weighted independent set, the authors use a known greedy heuristic [STY03] as the problem is NP-hard. Note again, that as in the longest common subsequence reductions (Section 4.1.6), solving the weighted maximum independent set problem exactly does not give the optimal number of wiggles. This is again because the possibility of having straight character curves between one interaction of a time step and multiple interactions of the previous/next timestep, which are not modeled.

> **Use methods based on maximum independent set** when you want to reduce wiggles by keeping as many straight segments as possible without creating conflicts. The idea is to choose a set of "no-wiggle" decisions that do not interfere with each other. This usually gives cleaner results than simple step-by-step fixes, but it still depends on the ordering chosen earlier.

4.1.8. Objective Abstraction & Gradient Descent

A non-standard approach is used by Tapaswi et al. [TBS14]: The layout aims to optimize proximity of character pairs in common interactions, separation of characters pairs that are not in a common interaction, wiggle, minimum separation of characters, and crossings. This is all done in a gradient descent approach that computes coordinates $y_{t,c}$ of all characters $c \in \mathcal{C}$ at each time step $t \in T$ and optimizes for a sum of these objective criteria by abstracting each criterion into a differentiable function.

Perhaps the two most interesting abstractions are the minimum separation and the crossings. For a minimum separation of value μ_s, the following smoothly differentiable modification of $\frac{1}{x}$ is defined first:

$$\mathcal{Z}(x,\mu) = \begin{cases} \frac{1}{x} \cdot (\sqrt{1+(x-\mu)^2} - 1) & 0 < x < \mu \\ 0 & x \geq \mu. \end{cases}$$

Then, to ensure minimum separation, the following function is minimized:

$$\frac{1}{|T|\left|\binom{\mathcal{C}}{2}\right|} \sum_{\{c,c'\} \in \binom{\mathcal{C}}{2}, t \in T} \mathcal{Z}((y_{t,c} - y_{t,c'})^2, \mu_s).$$

To abstract crossings, the authors apply the pseudo-Huber loss function [Hub92], that is, the following function:

$$\mathcal{H}(x,\mu) = \begin{cases} \sqrt{1+(x-\mu)^2} - 1 & x < \mu \\ 0 & x \geq \mu. \end{cases}$$

To minimize crossings, the following function is minimized:

$$\frac{1}{|T|\left|\binom{\mathcal{C}}{2}\right|} \sum_{\{c,c'\} \in \binom{\mathcal{C}}{2}, t \in T} \mathcal{H}((y_{t,c} - y_{t,c'})(y_{t+1,c} - y_{t+1,c'}), \mu_c),$$

where they set $\mu_c = 0$.

> **Use gradient descent when** your problem is about smooth shapes and positions, not strict ordering rules. It adjusts positions continuously, which makes it good for reducing wiggles or improving geometry. It is less suited for discrete constraints like keeping characters adjacent.

4.1.9. Force-Based Approaches

Force-directed approaches are most prevalent in drawings of general graphs. Usually, forces for pairs of nodes are defined and depend on some deviation between the ideal distance and actual distance between the pair of nodes. These forces are then being minimized by some iterated approach, similar to gradient descent. The most famous approach for this is the Fruchtermann-Reingold algorithm [FR91].

In general graphs, forces are usually two-dimensional, as nodes can be placed anywhere in the plane. In storylines, nodes – which correspond to positions of characters at each time step – usually have a fixed x-coordinate. Thus, in the case of storylines, forces are 1-dimensional, i.e., along the y-axis.

The first approach like this is by Silvia et al. [SAHW15]. They define the following types of forces:

node-node forces Nodes within a time step repulse and attract each other, using an inverse-squared force with equilibrium distance; this is as in the Fruchtermann-Reingold model, but vertically. That is, they define some desired distance d_0. Whenever the distance is larger than d_0, the two nodes attract each other with a force that is proportional with the reciprocal of their distance. If it is smaller, they repulse each other. The equilibrium distance is not described.

node-edge forces Nodes connected by an edge, i.e., two nodes that correspond to the same character of consecutive time steps, attract each other vertically.

node-interaction forces nodes within the same interaction attract each other vertically.

interaction-interaction forces Interactions within the same time step repulse each other vertically, using an inverse square force on the centroid of each interaction.

The full mathematical description of these forces is missing. Clearly, node-edge forces are designed to minimize wiggle and crossings, and node-interaction and interaction-interaction forces are designed to meet the design criterion of storylines that characters in an interaction are grouped together. It is not described how the forces are minimized.

A work by Silvia et al. [SEA*16] is an extended version of the

approach described just above. The same forces are used, and it is further described how they are minimized. We provide this algorithm in Algorithm 1, with a different notation than in the original paper. The procedure is applied 30 times, followed by applying strong local symmetry and alignment forces for one more iteration, which is not further described in the paper.

Algorithm 1: Force minimization step in [SEA*16].

```
foreach t ∈ T do
    foreach pair of nodes u, v in t do
        if vertical distance(u, v) < d_0 then
            apply node-node repulse force on u, v;
        else if u and v are in an interaction together then
            apply node-interaction attractive forces on u, v;
        if a non-interaction member node u overlaps with
          an interaction I then
            apply interaction-interaction forces on u;
    foreach pair of nodes u, v in time steps t, t + 1 do
        if u and v corr. to the same character then
            apply node-edge attractive forces on u, v;
```

> **Use force-based methods when** you want a flexible layout that balances several soft goals. They model the layout with forces, which makes them easy to adapt and extend. The trade-off is that the results can be unstable and hard to predict.

4.1.10. Multi-Dimensional Scaling (MDS)

Multi-dimensional scaling is an approach that takes a similarity matrix between entities and computes a mapping of the entities to some metric space such that similar entities are close together in the mapping.

Such an approach has been used for the computation of a storyline layout by Liu et al. [LLLW17]: Each character represents an entity; the similarity of two characters is the number of time steps they appear together in an interaction. The authors then use a one-dimensional scaling for the computed similarities [Bor11]. Based on the mapping to the single dimension, clusters of characters are formed greedily. In the visualization, clusters are horizontal strips, which have a fixed vertical ordering. Each character, whenever it is not participating in a meeting, is drawn within the corresponding cluster. Next, interactions are assigned to clusters based on how many of the contained characters are within each cluster; that is, the cluster with the most participating characters is chosen. To compute the ordering of clusters and the final layout, the Storyflow algorithm is used [LWW*13].

Zhang, Chen, and Yong [ZCY25] employ a similar approach for their location-focused storyline: Each character and location receives a significance score. Essentially, significance scores correspond to how often the character appears in an interaction, and how often significant actors interact within the location, respectively. The locations are ordered vertically using one-dimensional scaling: Similarity values between pairs of locations are essentially defined as the number of times a character moves from one location to the other, weighted by their significance. A global character ordering is computed to sort the characters within each interaction. Then a layered sweep is performed to adjust character orders and minimize crossings; see Section 4.1.2. The exact definitions of significance scores and the global character orders are quite technical, so we refer to the original paper [ZCY25].

> **Use MDS when** you have large-scale dense data and you want to go from a coarse to a fine layout.

4.1.11. Other

Tang et al. [TLW*21] propose a user-AI collaboration tool based on iStoryline [TRL*19]. The base layout is computed with the iStoryline approach; then, the user modifies the layout; based on this interaction, the AI proposes different new layouts in an attempt to predict the users' behavior. The user can choose one of these new layouts, and the process starts anew.

For their storyline model, where interactions have a partial order, Dobler et al. [DNS*23] propose a pipeline approach to compute a total order of interactions. This is done by a packing and ordering step. The packing step tries to put as many interactions as possible in a single layer by modeling the problem as a graph coloring problem. The ordering step then tries to put similar groups of interactions (as computed by the packing step) next to each other by finding a constrained TSP-tour in a specified graph, where weights between nodes model the dissimilarity of two groups of interactions. The rationale is that similar groups of interaction enable the possibility of few crossings in the crossing minimization step that follows.

In the protagonist setting of Hegemann and Wolff [HW24] (recall Figure 7), a heuristic for crossing minimization is proposed that is based on Max-Cut: namely, for each non-protagonist character they need to decide whether it runs above or below the protagonist. If a pair of characters runs on the same side, this introduces some fixed amount of crossings. A weighted edge between the pair of characters models this amount. A maximum cut in the resulting graph gives a partition of the characters into two sides that minimizes these crossings. Thus, Hegemann and Wolff use a heuristic for the maximum cut problem to find such a partition. Further, they propose yet another heuristic: for a crossing-minimized solution, they use a geometric packing algorithm to group crossings into a minimum number of block crossings.

> ***Summary & Design Implications:*** Heuristic approaches are widely used in storyline visualization because they scale well to larger datasets and many interactions. Some work *globally*, shaping the whole layout at once, while others work *locally*, improving it step by step. Global methods, such as genetic algorithms, force-based layouts, or MDS, try to balance several goals at the same time and often produce a coherent overall structure. Local methods, such as greedy strategies, local search, or LCS-based techniques, focus on small improvements and offer more control, but are more affected by earlier choices. In practice, not all goals can be satisfied at once, so the choice of method depends on which aspects of the visualization matter most.

4.2. Exact Approaches

In contrast to heuristic approaches, exact methods formulate parts of the storyline layout problem as well-defined optimization or constraint satisfaction (SAT) problems and aim to compute provably optimal or feasible solutions with respect to specific objectives. These approaches rely on formal models that capture geometric constraints or temporal dependencies explicitly. Due to their computational complexity, exact methods may be applied to restricted subproblems or individual stages of the layout process, rather than to the full storyline pipeline. In this section, we address different methods to compute exact solutions used in the storyline literature (see Table 6).

Exact Approaches	
Linear Programming [12]	[DHNW25, HBK15, LZLC14, YWR*25, Are15, LHZ*21, SLW*22, TRL*19, LWW*13, TLW*21, THM15, AP17]
Dynamic Programming [1]	[DHNW25]
ILP [7]	[DHNW25, PLM*25, DNS*23, GJLM16, Frö18, DJJ*24, WSZ*24]
SAT [3]	[Frö18, vDLMW17, GDL*20]
Other [3]	[KNP*15, vDFF*16, HW24]

Table 6: *Exact approaches*

4.2.1. Linear Programming

Linear programming is a technique that allows us to compute values of real-valued variables under some linear constraints, optimizing a linear objective. This is especially useful, when computing the coordinates of character curves at each time step. Many approaches apply linear programming as one of the last steps of their pipeline to compute vertical coordinates when the order of characters has already been determined by the previous steps of the pipeline. The advantage of linear programming is that readily available linear programming solvers can find an optimal solution in polynomial time. Some approaches extend the linear objective function to a quadratic one and realize that the program is still polynomial-time solvable. To describe the different approaches, we use real-valued variables $y_{t,c}$ for each $t \in T$ and $c \in \mathcal{C}$. These correspond to the y-coordinate of character c at time step t. In some works, the variable $y_{t,c}$ only exists if c is active at t.

The Storyflow approach [LWW*13] uses linear programming for computing character coordinates. The inputs are character orders π_t, $t \in T$, for the characters that are active at time step t, and a set of pairs $S \subset [\ell-1] \times \mathcal{C}$. Each pair $(t,c) \in S$ enforces the constraint that the character curve of c must run straight between t and $t+1$, as computed by a previous wiggle count minimization step. For an interaction I, let $C^2(I) \subseteq I \times I$ be the set of character pairs (c,c') such that c is directly before c' in the permutation π_t that contains I, and let $C^2_{\text{out}}(t) \subseteq A(t) \times A(t)$ be the set of character pairs (c,c') such that $c \neq c'$, and c and c' do not appear in an interaction together at time step t. The linear program is then given as follows.

$$\text{minimize} \sum_{t=1}^{\ell-1} \sum_{c \in A(t) \cap A(t+1)} (y_{t,c} - y_{t+1,c})^2 + \beta \sum_{t=1}^{\ell} \sum_{c \in A(t)} y_{t,c}^2 \tag{1}$$

$$y_{t,c} = y_{t+1,c}, \qquad (t,c) \in S \tag{2}$$

$$y_{t,c} < y_{t,c'}, \qquad t \in T, c, c' \in A(t), c \prec_{\pi_t} c' \tag{3}$$

$$y_{t,c'} - y_{t,c} = d_{\text{in}}, \qquad t \in T, I \in \mathcal{I}(t), (c,c') \in C^2(I) \tag{4}$$

$$|y_{t,c'} - y_{t,c}| \geq d_{\text{out}}, \qquad t \in T, (c,c') \in C^2_{\text{out}}(t) \tag{5}$$

The objective (1) minimizes a weighted sum of squared wiggle height and a definition of whitespace. The value β is a constant set to $\beta = 1$ in the paper. Constraint (2) ensures that the constraints imposed by the wiggle count minimization are followed. Constraint (3) ensures that the coordinates adhere to the given character orderings. Constraints (4) and (5) ensure that characters in an interaction together are spaced equally with spacing d_{in} and characters that are not in an interaction together are at least d_{out} apart. Here $d_{\text{in}} < d_{\text{out}}$ are two constants. The authors claim that this is a quadratic optimization problem with linear constraints and can thus be optimized in polynomial time; for this, they use the Mosek package. This approach has been adopted in [LHZ*21, YWR*25].

In [LZLC14], two objectives corresponding to the total height of the layout and the total wiggle height are given, using the same $y_{t,c}$-variables. It can be assumed, that linear programming was used to minimize them, however a description is absent. Similarly, Arendt [Are15] writes they use linear programming for minimizing whitespace; there is no further description about how this is done.

The approach of Tanahashi, Hsueh, and Ma [THM15] extends the Storyflow linear programming approach. As they deal with streaming data, their linear program tries to keep the existing layout as is and only optimize for the new data, grouping parts of the visualization together and treating them as one in the linear program. They then define a linear program to only minimize whitespace, and not wiggle height. As the description is technical and optimized for streaming data, we refer to their paper for the full details.

Held, Braune, and Kruse [HBK15] deal with real-valued time-series data representing distance values between entities. They try to represent these distance values as a storyline along the y-axis. That is, the distance value between two characters c and c' at time step t is $d_t(c,c')$. To do this, they minimize the following objective:

$$\begin{aligned} \text{minimize} \quad & \sum_{t \in T} \sum_{\{c,c'\} \in \binom{A(t)}{2}} (|y_{t,c} - y_{t,c'}| - d_t(c,c'))^2 + \\ & \sum_{t \in [\ell-1]} \sum_{c \in A(t) \cap A(t+1)} (y_{t,c} - y_{t+1,c})^2 \end{aligned} \tag{6}$$

This objective minimizes the squared deviation from the real distance-values plus the squared wiggle height. The authors do not describe how this objective is minimized.

Arendt and Pirrung [AP17] claim to use the Networkx Simplex algorithm to compute coordinates in the last step of their pipeline. However, the linear program which is solved is not described.

Tang et al. [TRL*19] extend the Storyflow approach to allow more extensive user interactions. Some of these alter the step of

the pipeline which computes coordinates. These interactions are as follows:

Bending. Here, one interaction I can be moved below another interaction I'. This introduces some inequality constraints between y-variables to the linear program.

Scaling. Another interaction allows scaling a range of time steps or a local part of the storyline. It is not described how this is done, but we assume that this either introduces some new constraints into the linear program or increases d_{in} and d_{out} locally.

This approach is then also used in [TLW*21, SLW*22].

Dobler et al. [DHNW25] give two linear programs, one for minimizing squared wiggle height and one for minimizing wiggle height. The corresponding programs are very similar to (1)-(5); they also use the two constants d_{in} and d_{out}, but they allow the constants to be equal. The program for minimizing squared wiggle height essentially consists of the constraints (3)–(5) and the following objective function.

$$\text{minimize} \quad \sum_{t=1}^{\ell-1} \sum_{c \in A(t) \cap A(t+1)} (y_{t,c} - y_{t+1,c})^2 \tag{7}$$

The authors show that the objective matrix is positive semidefinite and the program is thus solvable in polynomial time.

To minimize wiggle height, the authors again use constraints (3)–(5). They introduce the real variables $w_{t,c}$ for all $t \in [\ell-1]$ and all $c \in A(t) \cap A(t+1)$ which correspond to the wiggle height of character c between time steps t and $t+1$. This is achieved by the following two constraints:

$$w_{t,c} \geq y_{t,c} - y_{t+1,c}, \qquad t \in [\ell-1], c \in A(t) \cap A(t+1) \tag{8}$$
$$w_{t,c} \geq y_{t+1,c} - y_{t,c}, \qquad t \in [\ell-1], c \in A(t) \cap A(t+1) \tag{9}$$
$$\tag{10}$$

The objective is then given as follows:

$$\text{minimize} \quad \sum_{t=1}^{\ell-1} \sum_{c \in A(t) \cap A(t+1)} w_{t,c} \tag{11}$$

The authors show that if d_{in} and d_{out} are both natural numbers, then all extreme points of the given polytope are integer-valued. This means that there is always an optimal solution having all y-variables corresponding to integer values and that a linear programming solver will give such a solution.

In [DHNW25], another type of linear program is described which minimizes the horizontal distance between two time steps. This is necessary due to the specific way in which the character curves are drawn in this paper. Namely, a character curves that wiggles between two time steps is drawn as two circular arcs instead of a Bezier curve with two intermediate control points. This has the effect that the radial distance between two vertically consecutive characters can remain the same in the space between the two time steps. However, the width between time steps needs to be adjusted, which is done with this linear program. For the full details, we refer to the paper.

> **Use linear programming when** the character order is fixed and the goal is to compute positions to improve spacing, compactness or wiggles.

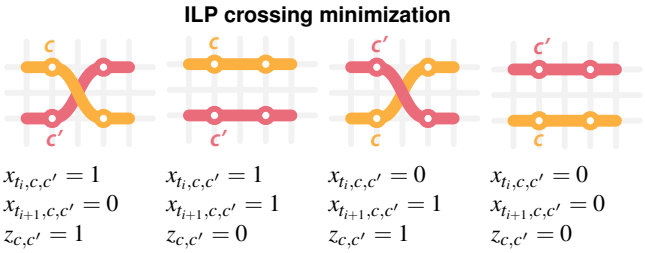

Figure 17: *The figure illustrates one of the ideas at the base of ILP crossing minimization [GJLM16]. There is a crossing iff the two endpoints of two lines are in inverted relative positions at t_i and t_{i+1}. This is captured by variables $x_{t_i,c,c'}$ that indicate the relative positions of the nodes at every timestep in the visualization. The formulas to compute this are in Equation (16) and Equation (17).*

4.2.2. Dynamic Programming

Dobler et al. [DHNW25] present a dynamic program to optimally minimize wiggle count for storylines consisting of exactly two time steps, assuming a fixed order of characters at each time step, and a fixed vertical distance d_{in} between consecutive characters of an interaction and a minimal distance d_{out} between characters not in an interaction together. The idea is to model two characters that can be simultaneously straightened as an arc in a directed acyclic graph. A dynamic program finds the longest path in this graph, which corresponds to the maximum set of characters that can be laid out without wiggles.

4.2.3. Integer Linear Programming (ILP)

Integer linear programming is an approach to solve complex, usually NP-hard, problems. The idea is to define the problem as a set of linear constraints over integer variables. An assignment of the integer variables that simultaneously minimizes (or maximizes) some linear objective function corresponds to an optimal solution of the original problem. As very powerful ILP-solvers exist nowadays, this approach is feasible for small to medium-sized instances which are common in visualization.

In storylines, ILP-approaches are mostly used for crossing minimization, which is an NP-hard problem (see Section 4.3.1). The first work to formalize the storyline crossing minimization problem into an ILP is by Gronemann et al. [GJLM16]. They model the orderings π_t for $t \in T$ as binary variables, by having for each $t \in T$ and distinct $c, c' \in A(t)$ a binary variable $x_{t,c,c'}$ which is exactly one when c is above c' at time step t, or more formally, $c \prec_{\pi_t} c'$. The most basic formulation, using a quadratic objective, is as follows:

$$\text{minimize} \quad \sum_{t \in [\ell-1], c,c' \in A(t) \cap A(t+1), c \neq c'} x_{t,c,c'} x_{t,c',c} \tag{12}$$

$$0 \leq x_{t,c,c'} + x_{t,c',c''} - x_{t,c,c''} \leq 1, \quad t \in T, \{c,c',c''\} \in \binom{A(t)}{3} \tag{13}$$

$$x_{t,c,c'} = 1 - x_{t,c',c}, \quad t \in T, \{c,c'\} \in \binom{A(t)}{2} \tag{14}$$

While the objective (12) minimizes crossings and (13)-(14) ensure transitivity and symmetry of the orders π_t, this model does not yet ensure the continuity of characters within an interaction. For this, Gronemann et al. propose so-called *tree-constraints*. These are defined for each interaction I, each two characters $\{c,c'\} \in \binom{I}{2}$ and each third character $c'' \in A(t) \setminus I$, where t is the time step such that $I \in \mathcal{I}(t)$. The tree constraint for this triple of characters is then given in (15).

$$x_{t,c,c''} = x_{t,c',c''} \tag{15}$$

This constraint ensures that the relative order of the pair (c,c'') and the pair (c',c'') is the same at time step t, i.e., c'' cannot be between c and c' in the order π_t.

Gronemann et al. [GJLM16] then realize that they can get rid of half of the x-variables due to symmetries between variables of the form $x_{t,c,c'}$ and $x_{t,c',c}$, and they apply standard projections to the constraints afterward. The symmetry constraints (14) are removed. Lastly, they realize that the problem corresponds to a maximum-cut problem with additional transitivity constraints, which can be represented as an integer linear program with an exponential number of so-called *odd cycle constraints*. They implement this program, and apply a branch-and-cut approach to lazily add the odd-cycle-constraints. For the full details of the maximum-cut transformation and implementation details, we refer to the original paper. The largest instances which they can solve have around 60 characters and 120 time steps, and the optimization runs for around half an hour.

Fröschl and Nöllenburg [FN18,Frö18] use a different ILP model to simultaneously minimize wiggle height, and optionally also wiggle count and crossings, using a weighted summation of these objectives. However, their model restricts character coordinates to a constant range of integer values. That is, the y-coordinate of each character at each point in time can only be chosen from a set $Y = \{1,2,\dots,y_{\max}\}$, and the size of their models depends on $y_{\max}$. For the full details of these models, we refer to the corresponding master thesis [Frö18].

Dobler et al. [DNS*23] consider the setting where interactions are only partially ordered, instead of having a total order. For the minimization of crossings in this setting, they propose two approaches. The first approach computes a total order of interactions by applying a pipeline consisting of a graph coloring step and a TSP step, and then applies the ILP of Gronemann et al. [GJLM16]. The second approach incorporates all these steps into a single ILP. The idea is to use binary assignment variables $y_{t,I}$ to decide whether interaction I is placed at time step t and map the tree constraints of Gronemann et al. to these variables, that is, tree constraints for interaction I are active at time step t if and only if $y_{t,I} = I$. For the formal definition, we refer to the paper.

In a follow-up work, Dobler et al. [DJJ*24] revise the models for crossing minimization by Gronemann et al. [GJLM16]. Firstly, they start with the quadratic objective (12). They linearize the objective by introducing binary variables $z_{t,c,c'}$ for each $t \in [\ell-1]$, $\{c,c'\} \in \binom{A(t)\cap A(t+1)}{2}$ with $z_{t,c,c'} = 1$ if and only if c and c' cross between time steps t and $t+1$. They include the following new constraints:

$$z_{t,c,c'} \geq x_{t,c,c'} - x_{t,c',c}, \quad t \in [\ell-1], c, c' \in \binom{A(t)\cap A(t+1)}{2} \tag{16}$$

$$z_{t,c,c'} \geq x_{t,c',c} - x_{t,c,c'}, \quad t \in [\ell-1], c, c' \in \binom{A(t)\cap A(t+1)}{2} \tag{17}$$

Together with the following new objective, this has the desired effect (see Figure 17):

$$\text{minimize} \sum_{t\in[\ell-1],\{c,c'\}\in\binom{A(t)\cap A(t+1)}{2}} z_{t,c,c'} \tag{18}$$

Dobler et al. [DJJ*24] then prove a set of structural properties of crossing-optimal storylines, which allow them to introduce equality-constraints that reduce the search space for ILP solvers. Based on these properties they also introduce a new ILP formulation, which essentially gets rid of many transitivity constraints (13), and has a quadratic number of constraints per time step instead of a cubic amount. Due to the technicalities of the structural insights, we refer to the original paper for the new model. Together with the structural insights, Dobler et al. present new heuristics which guide the solution finding process of the ILP-solver. This allows them to improve upon the approach by Gronemann et al. [GJLM16], with a reduction of runtime by a factor of 2–3.

Wang et al. [WSZ*24] propose a visualization for entity pairs over time. That is, in their visualization, characters correspond to entities. Each interaction is a set of entity pairs, which is explicitly shown with a matrix that has a column for each entity pair and a row for each entity; the two cells corresponding to the entities are filled in that column. The entity rows are at the same y-coordinate as the corresponding character curve. They propose a multi-objective ILP to minimize line crossings and the sum of all vertical distances between entity pairs. To realize the crossing minimization, they apply the quadratic model by Gronemann et al. [GJLM16] described above ((12)-(14)), however without the tree constraints. To minimize the vertical distances, they use a model where characters can have a fixed vertical position from 1 to $n = |\mathcal{C}|$ for each point in time. They use assignment variables to encode whether an entity is at a specific position at a specific point in time, and link these assignment variables to the ordering variables in the quadratic model. To ensure interaction continuity, they use a set of constraints which is arguably more complicated and harder to solve than the tree constraints of Gronemann et al. [GJLM16].

Piselli et al. [PLM*25] propose a multi-objective mixed integer linear program for fair storylines based on previous ILP work for layered graphs [DBRGD22]. That is, in their model the characters are partitioned into two groups, and metrics such as crossings should be either fairly distributed or primarily optimized for only one of the groups. Simultaneously, a subset of the objectives crossings, wiggle height, and skewness can be optimized. To minimize crossings, they use the linearized model described above ((13)-(15), (16)-(18)). Their approach to minimizing wiggle height is similar to the one by Wang et al. [WSZ*24]. They assume that each pair of vertically consecutive characters at a time step has the same

distance. Thus, they also use binary assignment variables to track whether a character is at a specific vertical position at a time step, and use these variables to calculate wiggle height. However, this approach is also extended with a baseline integer variable b_t for each time step t. This variable encodes the height of the bottom-most character, and thus allows that the baseline vertical offset is different for each time step. To minimize skewness, they essentially use binary variables for each character, which encodes whether the character is crossed, and then minimize the sum of these variables. For fair minimization of crossing, wiggle height, skewness, and wiggle count, they essentially minimize the weighted absolute difference between the two quality metrics, when they are computed only within characters of the same group.

The most recent paper which uses an ILP approach, proposes one for wiggle count minimization [DHNW25]. As in the linear programs for (quadratic) wiggle height minimization, they assume a fixed order of characters at each time step, and a fixed vertical distance d_{in} between consecutive characters of an interaction and a minimal distance d_{out} between characters not in an interaction together. They add constraints similar to (3)–(5), and additionally introduce binary variables $z_{t,c}$ for each $t \in [\ell-1]$ and $c \in A(t) \cap A(t+1)$ which are one if and only if character c wiggles between time steps t and $t+1$. The objective is then given as:

$$\text{minimize} \quad \sum_{t\in[\ell-1]} \sum_{c\in A(t)\cap A(t+1)} z_{t,c} \tag{19}$$

To ensure, the correct values of the z-variables, the following constraints are added:

$$y_{t,c} + Y \cdot z_{t,c} \geq y_{t+1,c}, \quad t \in [\ell-1], c \in A(t) \cap A(t+1) \tag{20}$$

$$y_{t,c} - Y \cdot z_{t,c} \leq y_{t+1,c}, \quad t \in [\ell-1], c \in A(t) \cap A(t+1) \tag{21}$$

The constant Y is a large upper bound set to $\max\{d_{\text{in}}, d_{\text{out}}\} \cdot \sum_{t\in T} |A(t)|$. Essentially, if $z_{t,c} = 1$, then both of the constraints are trivially satisfied. Otherwise, $y_{t,c} = y_{t+1,c}$ must hold for the constraints to be satisfied.

> **Use ILP when** the goal is to compute an optimal ordering of characters — including under multiple constraints. It can encode crossings, adjacency, and other layout rules simultaneously. This also makes it very expensive, so it is mainly used for small instances or offline computation.

4.2.4. SAT

Similar to ILP, SAT is usually used to solve NP-hard problems exactly by modeling them as a set of Boolean formulas. In the case of SAT, the problem is stated in logical formulas, and a solution in the form of a satisfying variable assignment constitutes an optimal solution. Modern Max-SAT solvers are even able to optimize an objective value in the form of a weighted sum of binary variables. SAT has been used mainly in two papers: Fröschl and Nöllenburg [FN18, Frö18] have used SAT to minimize crossings and wiggles in a multi-objective setting. Essentially, they have modeled the same problem with SAT as they did with ILP (see Section 4.2.3).

Van Dijk et al. [vDLMW17] used SAT to minimize block crossings. The basis of the model is similar to crossing minimization models in ILP. They use a logic 0-1 variable $x^p_{c,c'}$ to describe the relative order of characters c and c' at some layer p – they have multiple layers per time step as a pair of consecutive layers can only support a single block crossings. The remaining construction is however rather complex, and we refer to the paper for the full details. This approach has also been used in [GDL*20].

> **Use SAT-based methods when** the ordering problem can be expressed as logical constraints between characters. Like ILP, they are computationally heavy and best suited for smaller problems.

4.2.5. Other

Van Dijk et al. [vDFF*16] show that deciding whether a storyline visualization with zero crossings and block crossings exists can be done in polynomial time. This is done by modelling the interactions as a binary matrix and testing whether the matrix has the consecutive ones property [BL76].

Hegemann and Wolff [HW24] consider the setting where a specified protagonist character is part of all interactions, each character persists throughout all time steps, and all non-protagonist characters must run below the protagonist. In this setting, they show that there exists an exact polynomial-time algorithm to minimize crossings. The idea is to identify a set of unavoidable crossings and to find a solution where only these crossings exist.

> ***Summary & Design Implications:*** Exact methods compute optimal solutions for specific layout objectives, which makes them useful in two main cases: as a reference to evaluate heuristic approaches, and for applications where strong guarantees on the layout are important. This level of precision comes at a much higher computational cost than heuristic methods, and is the main reason why exact approaches are less commonly used in practice. Linear and quadratic programming are the most commonly used techniques and can be solved efficiently when the character order is fixed. This allows design requirements to be expressed directly as constraints, such as enforcing group separation or fixing the position of a protagonist. In contrast, methods such as ILP or SAT address the harder problem of finding a good ordering, but are significantly more expensive and do not scale well. For this reason, they are typically applied to smaller instances or used in offline settings. Complexity results (Section 4.3.1) also suggest that certain structural properties of the data, such as persistent interactions or the presence of a main character, can make exact solutions easier to compute, and are worth considering before choosing an algorithm.

4.3. Computational Complexity Results

In this section, we aim to summarize findings related to the computational complexity of storyline visualization, i.e., the NP-completeness of layouting sub-problems, various developed approximation algorithms, and fixed-parameter tractable solutions to NP-Hard problems.

4.3.1. NP-complete Subproblems

Many of the layout generation problems in storylines were shown to be NP-complete. Kostitsyna et al. [KNP*15], for example, have shown that minimizing crossings in storylines is an NP-complete

problem. Similarly, Van Dijk et al. [vDFF*16] show that block crossing minimization is NP-complete.

In the protagonist-storyline setting of Hegemann and Wolff [HW24], the authors show that crossing as well as block-crossing minimization is NP-complete. More specifically, crossing minimization is NP-complete when non-protagonist characters can be placed above and below the protagonist; block crossing minimization is already NP-complete when they can only be on one side.

Dobler et al. [DHNW25] show that minimizing wiggle count is NP-complete in the setting where the permutations π_t for each time step are already fixed, and where there is a fixed vertical distance d_{in} between consecutive interaction-characters, and a minimum vertical distance d_{out} between character pairs not in the same interaction.

4.3.2. Approximations

Approximation algorithms are usually polynomial-time algorithms that give a performance guarantee, that is, they compute a solution with objective value at most $c \cdot \text{opt}$, where opt is the objective value of the optimal solution. Although technically not an approximation algorithm, Kostitsyna et al. [KNP*15] show how to compute a storyline visualization with at most $\mathcal{O}(n \log n)$ crossings when each interaction has size two, characters persist throughout all time steps, and the interactions can be modeled as a tree (for details refer to [KNP*15]). In the case where interactions cannot be repeated, and characters persist throughout all time steps, the authors of [vDFF*16] present an approximation algorithm for block crossing minimization that computes solutions with at most $3d^2(d^2-1)$ times the optimum amount of block crossings in polynomial time, where d is the maximum number of characters in an interaction.

4.3.3. Fixed-Parameter Tractability

Fixed-parameter tractability is a tool from algorithmic complexity theory to find a way to tackle NP-hard problems. Usually, one starts with an NP-hard combinatorial optimization problem and identifies some underlying parameter of the problem input or the objective function itself. This parameter is usually called k. Then, the problem is *fixed-parameter tractable* with respect to k if there exists an algorithm that solves the problem in time $\mathcal{O}(f(k) \cdot n^c)$, where f is a computable function, n is the input size of the problem, and c is a constant. Usually this is done for parameters where the hope is that k is small, and thus the algorithm runs efficiently.

Two papers show that some optimization problem in storylines is fixed-parameter tractable: Kostitsyna et al. [KNP*15] consider the problem of minimizing crossings in a storyline by computing a ordering of characters at each time step. They show that there is an algorithm that runs in time $\mathcal{O}(|\mathcal{C}|!^2|\mathcal{C}|\log(|\mathcal{C}|) + |\mathcal{C}|!^2|T|)$ and uses exponential space in $|\mathcal{C}|$. That is, the algorithm runs in fixed-parameter tractable time when parameterized by the number $|\mathcal{C}|$ of characters.

Van Dijk et al. [vDFF*16] consider the problem of minimizing block crossings. They propose two fixed-parameter tractable algorithms. The first one runs in time $\mathcal{O}(|\mathcal{C}|! \cdot (\frac{|\mathcal{C}|^3-|\mathcal{C}|}{6})^\beta \cdot (\beta+\mu))$ and uses polynomial space, where β is the optimal number of block crossings and $\mu = \sum_{I \in \mathcal{I}} |I|$. The second runs in time $\mathcal{O}(|\mathcal{C}|! \cdot |\mathcal{C}|^3 \cdot |T|)$ and uses exponential space in $|\mathcal{C}|$. Hence, the first algorithm runs in fixed-parameter tractable time when parameterized by $|\mathcal{C}|+\beta$ and the second when parameterized by $|\mathcal{C}|$ but uses exponential space.

4.4. Evaluation

Benchmarks	
Quantitative Metrics	[DHNW25, PLM*25, DCS*24, DNS*23, vDFF*16, GJLM16, Frö18, vDLMW17, HW24, DJJ*24, ZCY25, LWW*13, THM15]
Runtime only	[TLW*21, PBH18]
Human involvement	
User Study	[DCS*24, OK17, LZLC14, SEA*16, YWR*25, LCZ17, LHZ*21, YZC*25, AXP*22, KLM25, HAB23, CLMY13, QC16, SBB*18, AP17, KCH10, WSZ*24, WWD23]
Expert Interview	[CN23, LYK25, TRL*19, LWW*13, TLW*21, ZWQ*15, AP17]
Task scenario inspection	[PC18]
Case studies	[DCS*24, HBK15, YWR*25, OM10, GDL*20, LCZ17, SLW*22, ZCY25, LWW*13, KLM25, QC16, TM12, KCH10, PLM*25, PBH18]

Table 7: *Evaluation methods*

When it comes to evaluating the quality of a storyline layout approach, there are multiple methods to do it.

Benchmarks Many papers rely on computational measures directly tied to layout quality. These metrics are objective, reproducible, and allow comparison of the values obtained in these benchmarks over a broad range of instances of varying sizes and features, and this is in general a very common way to report quality of any graph layout algorithm, not just storylines [DBCS*24]. Often, these are just the optimization objectives described in Section 3.1.

Good examples of this are found for instance in Gronemann et al. [GJLM16] and van Dijk et al. [vDFF*16], who evaluate their ILP and block-crossing heuristics by reporting crossing counts and comparing them to heuristic baselines, while Tanahashi et al. [THM15] explicitly measure wiggle count and wiggle height when reporting performance of their pipeline and streaming approaches.

In addition to optimization objectives, **runtime** is particularly emphasized in works proposing integer linear programs or dynamic programming, as the cost in time and resources of solving a problem can quickly scale to large numbers. Gronemann et al. [GJLM16] and Dobler et al. [DNS*23, DJJ*24], for instance, report solvable instance sizes and runtimes of ILP formulations. In a

few cases runtime is the only metric reported: Tang et al. [TLW*21] and Padia et al. [PBH18] explicitly evaluate only runtime.

Human Involvement Papers focused primarily on perceptual clarity or interaction design rely more on **user studies** for reporting the quality of their results. For instance, user studies are used to compare the effectiveness of particular encodings. Deng et al. [DCS*24] use user studies to evaluate comprehension when storyline layouts are overplotted against alternative temporal network views. To evaluate interaction, Osmakcic et al. [OK17] test the effect of features such as bundling or zooming.

In some cases, it might be of interest to focus on **tasks**. Common tasks include identifying co-occurrences, ordering events, following a specific character, or detecting structural motifs. Chou et al. [CLMY13] test whether users can track topic evolution in textual corpora, and a similar task is also evaluated by Araya et al. [AXP*22] (albeit in 3D). Evaluating tasks related to the perception of continuity and grouping is also relevant. For instance, Hulstein et al. [HAB23] evaluate geographic storylines and continuity cues by a user study. Kim, Card, and Heer [KCH10] evaluate how genealogical storylines affect users' understanding of lineage structures.

In some cases, **expert interviews** are used. Costa and Nunes [CN23] collect feedback from digital humanities scholars about storyline-based depictions of historical data. Tang et al. [TRL*19] involve domain experts to assess whether iStoryline's interactive editing matches real analysts' workflows. Liu et al. [LWW*13] incorporate expert interviews when refining the StoryFlow pipeline to align with narrative analysis needs.

A final category of evaluations are **case studies**. Often, the authors of papers use case studies to show compelling examples of applicability of the methods they present in their papers. While this is a good method to show the results of a particular method at a glance, if the case studies are not accompanied by additional results it might not be as informative for the reader, lacking in broadness of scope to show applicability over cases different than the ones presented in the paper. This method to present results depends heavily on reader interpretation and provides little empirical evidence, thus should be handled carefully and, if possible, accompanied with another type of analysis. Examples are: Ogawa and Ma [OM10] and Tanahashi and Ma [TM12] present narrative examples (e.g., software developers, conference interactions) and reason about readability informally. Zhang et al. [ZCY25] and Kuo et al. [KLM25] show successive refinements of storyline layouts and discuss artifact reduction qualitatively.

Software	[OM10]
Genealogy	[KCH10]
Movies, books and literature	[PLM*25, DNS*23, ACYB14, OK17, YWR*25, vDFF*16, GJLM16, Frö18, vDLMW17, DJJ*24, LCZ17, GMD*23, ZCY25, TRL*19, LWW*13, TLW*21, QC16, THM15, TM12, WSZ*24]
Social - other	[DHNW25, KLM25, HAB23, CLMY13, AWA20]
Project by person	[LZLC14, LHZ*21]
Video scene detection	[TBS14]
Movement data	[DCS*24, HBK15, YZC*25, ZCY25, HAB23, ZWQ*15, AP17]
(Sport) movement data	[LLLW17, SLW*22, LYK25, WSZ*24]
Critical editions	[SAHW15, SEA*16]
Literature	[PC18]
Social meetings	[SBB*18]
Hierarchical task network	[PBH18]
Coauthorship network	[DNS*23, GDL*20, HW24, DJJ*24, KLM25]
News articles	[CN23, AXP*22]
Disease outbreak	[KLM25]
Rolling stock schedule	[DHNW25]

Table 8: *Domains used in the papers we collected as example applications or motivation for developing their method.*

4.5. Domain

In the papers we collected, the vast majority definitely uses, as sample datasets or motivations for the work, the **narratives** from either **books, movies, or literature** (see, for instance, [TM12, LWW*13], or refer to Table 8). Indeed, not only is this the original motivation of the XKCD movie narrative charts [Mun09], but it is also implicit in the name of *story*lines, making this a canonical application of this kind of visualization. These datasets are usually discrete, relatively small in scale, and sparse in attributes beyond character identity and scene membership. Because the underlying data is often curated rather than observed, many papers implicitly assume clean interaction boundaries and unambiguous temporal orderings. This partly explains why crossing minimization and wiggle reduction dominate optimization objectives in this domain: the narrative structure is treated as fixed, and visual clarity becomes the primary concern.

Storyline visualizations have also been applied to **movement** and trajectory data, where entities correspond to moving objects such as people, animals, or vehicles, and interactions reflect co-location or

shared presence at the same place and time. The storyline metaphor shifts from narrative or lineage to spatial movement over time: lines represent individual trajectories, convergence indicates encounters, meetings, or shared locations [DCS*24, ZCY25, HAB23, ZWQ*15, AP17]. In **sport** analytics, this approach is used to trace the movement of players over the course of a match, with time aligned to game phases or events rather than absolute clock time. Characters correspond to individual players, and interactions capture moments of proximity, team formation, or coordinated movement [LLLW17, SLW*22, LYK25, WSZ*24].

A smaller application domain for storyline visualization concerns the **evolution of texts across translations**, editions, and revisions. In these datasets, entities typically correspond to textual versions such as editions of a book or translations into different languages, while interactions encode derivation, reuse, or shared textual segments across versions [SAHW15, SEA*16]. Time is modeled discretely, reflecting publication dates or major revision milestones rather than continuous change. Unlike narrative or collaboration datasets, interactions in this domain are not social but editorial: lines converge to indicate shared sources, common ancestry, or reuse of passages, and diverge as versions evolve independently. Storyline visualizations are used to reveal branching structures, parallel developments, and points of convergence where multiple editions draw from the same source. Closely related to this domain are also **genealogical** datasets, where storyline visualizations are used to represent family lineages, marriages, and lifespans over time [KCH10]. Here, entities correspond to individuals, and interactions encode family relationships. As with translations and editions, the emphasis is on descent and inheritance, with lines converging to indicate unions and diverging to represent offspring. In [KCH10], time includes uncertainty, reflecting uncertain or approximate birth and death dates.

Another application domain is **collaborative work** (such as software engineering), where characters correspond to developers and interactions are derived from shared activity, such as committing to the same file or repository within a given time window [OM10]. The work from Ogawa and Ma [OM10] is one of the earliest works using storylines as a visual metaphor, and uses a dataset extracted from version control systems and feature dense interaction patterns, large numbers of entities, and long temporal extents. Compared to narrative media, the emphasis shifts toward scalability and aggregation: it adopts a form of bundling bundling to address visual clutter.

Storyline visualizations of **coauthorship** data are commonly used to reveal the evolution of research collaborations, recurring partnerships, and the formation or dissolution of research groups over time [DNS*23, GDL*20, HW24, DJJ*24, KLM25]. These visualizations support exploratory analysis of collaboration structure and temporal trends. A coauthorship dataset is also used in the teaser of this paper (Figure 1), where institutions are treated as characters and joint publications as interactions.

5. Discussion

While the community has developed a large collection of knowledge on storyline visualizations, there is still a lot of research to do. A couple of challenges remain to be faced:

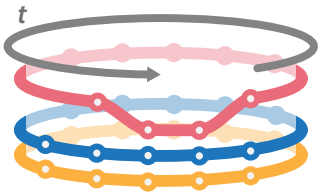

Figure 18: *How would a storyline look in a narrative where time is circular? What if there are time jumps? The picture represents one of the attempts at displaying circular time proposed during the Graph Drawing Contest 2025 [DBKMW25], where the visualization was displayed on a circular screen.*

5.1. Co-optimization

Co-optimization refers to the simultaneous optimization of multiple aesthetic or structural criteria rather than treating them in a strict sequence or isolation. Storyline visualization research traditionally decomposes the layout pipeline into serial stages, e.g., crossing minimization, then wiggle reduction, then whitespace reduction (see Section 4.1.1), solving each subproblem in turn. Co-optimization, instead, searches for layouts that trade off these quality metrics jointly. A useful analogue from the broader graph drawing literature is found in multi-criteria layout frameworks. For example, the Stress-Plus-X [DADL*19] graph layout model simultaneously optimizes multiple readability metrics such as edge crossings, minimum crossing angle, and stress (a measure of how well geometric distances match graph-theoretic distances) in a single objective framework, rather than sequentially optimizing each criterion in isolation. Computing everything at once results in layouts that have overall higher readability compared to pipelines, often optimizing one metric at a time. However, this usually comes at the cost of higher computational complexity.

In storylines, *crossing minimization, wiggle reduction, and even whitespace control are not independent*: minimizing crossings alone can increase wiggles, and aggressively compressing whitespace can induce undesirable line tangling. Recent work on wiggle minimization in storylines highlights this interaction by formally characterizing the complexity of minimizing different wiggle objectives, e.g., count vs. vertical movement, exploring solutions that balance those objectives [DHNW25].

5.2. Temporal Oddities

One open challenge is how to represent stories that have time jumps or time travel (see Figure 20): Few have seemingly noticed that in the very famous visualization from XKCD [Mun09], multiple examples are presented, the last of which (*Primer*) is perhaps jokingly placed there and drawn in convoluted scribbles, as if it is something too complicated to represent. That visualization represents the story of *Primer*, an indie production about time machines, where the characters, by jumping back and forth through time, create multiple new timelines that intersect and interact with each other. To this day, and to the best of our knowledge, no one has attempted at replicating the last visualization in XKCD's overly famous visu-

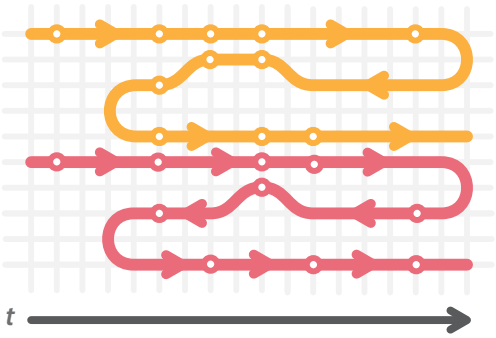

Figure 19: *What if characters could go back and forth in time, as it happens in the movie Tenet? And even meet themselves while they are going through time in reverse, and normally?*

alization, which is interesting given the popularity of all the other visualizations in the same image.

Inspired by this, the most recent Creative Contest organized by the Symposium on Graph Drawing & Network Visualization [DBKMW25] presented a challenge to represent a story with temporal jumps (the German TV series *Dark*, which prominently features time jumps of exactly 33 years towards the past or the future). A large circular 360 degree screen was made available to the participants of the contest, that could be used as a continuous surface to represent a visualization. This produced interesting submissions that wrap the storyline around the whole screen, making it possible for the story to "wrap" every 33 years, highlighting higher-order patterns as a result of the the circular aspect of the story, and time jumps were typically drawn as vertical lines between the circles (see Figure 18).

There are also further variations and possible models of how time and narratives are considered in science fiction books and movies. We refer to these as temporal oddities and describe them in the following:

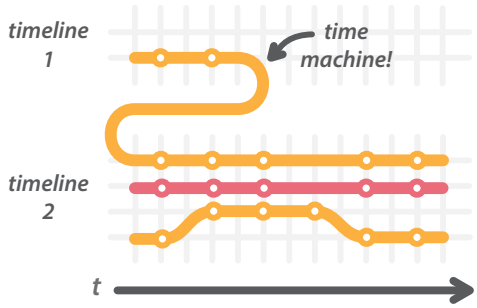

Figure 20: *Yellow has a time machine! When they use it in timeline 1, they not only travel back in time, but can also meet themselves from a few timesteps ago. This creates an alternate reality (timeline 2) where yellow has met themselves from the future and makes different decisions based on their meeting. How would we display multiple timelines and time travel on a storyline visualization?*

For instance, we can take into account a model like the one used in *Tenet*, in which time (and characters) are flowing forward and backward in time simultaneously (see Figure 19). Alternatively, consider the possibility of multiple parallel realities that might interact with each other, making it necessary to have an additional dimension, perhaps. Other forms of temporal oddities are expressed in science fiction narratives. Taking inspiration from other Christopher Nolan movies, we can investigate how the passage of time is conveyed. Within these narratives the passage of time for the characters involved differs either due to the general principle of relativity or how the passage of time is conceptualized in dream states. We illustrate this concept in Figure 22. Specifically, *Interstellar* has a wrapping storyline, but also time passing differently for different groups of characters due the way that gravity warps time (i.e., general relativity). *Inception*, on the other hand, has time passing differently for different characters, in increasingly smaller pockets due to the fact that each dream level runs on a slower timescale than the one above it. The deeper the characters go into nested dreams, the more time they experience relative to the real world.

Temporal oddities also arise in less extreme forms, such as events with unclear boundaries, interactions whose durations overlap without a clear start or end, or characters whose presence is inferred rather than observed. In these cases, discretization of time is introduced and the resulting time steps or events introduce additional uncertainty. While these modeling decisions are necessary for layout algorithms, they are rarely communicated visually, and the resulting storyline may appear more precise than the data or narrative warrants. From this perspective, temporal oddities can be seen as manifestations of uncertainty along the temporal dimension. Just as storyline visualizations typically do not encode uncertainty in interaction membership or layout choice, they also struggle to represent uncertainty in temporal ordering. Addressing these cases explicitly would require moving beyond a single linear timeline and considering alternative temporal structures, parallel timelines, or visual encodings that allow ambiguity to remain visible rather than resolved.

5.3. Uncertainty

Communicating uncertainty in storyline visualization is also a pressing challenge. For example, in Akira Kurosawa's *Rashomon*,

Figure 21: *The story of yellow is told by two different unreliable narrators: according to one, yellow was with pink in the center of the narrative, while according to the other, yellow was with teal. The two recollections are only consistent at the beginning and at the end of the story. How would we display the uncertainty of the narrative?*

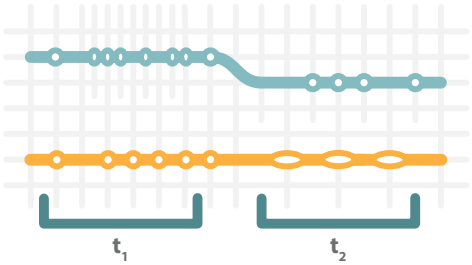

Figure 22: *This storyline has characters experiencing time at different times, as it happens in the movie Interstellar. Teal experiences time in t_1 at double speed compared to yellow. In t_2, instead, yellow is experiencing time at half speed compared to teal.*

a single narrative is recounted multiple times from different perspectives, each internally consistent but mutually incompatible. From a visualization perspective, such narratives challenge the notion of a single authoritative timeline. "How would one go about visualizing such incompatible storylines?" Inspired by compound graph visualizations, one could visualize each such perspective as a separate juxtaposed or interchangeable small multiples [EMWR24, VBW17]. Alternatively, as highlighted in Figure 21, one could visualize all perspectives in a single storyline and make use of a combination of branching paths and opacity to visually communicate the disagreements between the various perspectives.

Beyond incompatible perspectives, however, uncertainty poses a challenge in several different ways. While storyline visualizations are well-suited to depicting parallel actions and simultaneous interactions, they generally presuppose that the temporal order, locations of events, and the participants in events are fixed and known. However, this in real-life contexts this is not always the case. "How would a storyline visualization represent a series of events in which the exact start and endpoints of a particular event are unknown?" Inspired by metro map visualizations, one could perhaps elongate and extend an event across time for a particular (set of) character(s). Alternatively, how would one visualize the uncertain participation of characters, e.g., a character whose participation in an event is merely implied? We see opportunities for (classical) uncertainty visual channels to be employed, such as opacity, fuzziness, enclosure, or (the recently proposed) node wiggliness [EPDB*25]. Finally, in the context of spatial storylines, i.e., storylines whose events have and share some notion of location, how would one tackle uncertainty in location? If closed curves around events were employed to visualize an event's location, perhaps (as discussed by Vehlow et al. [VBW17] in their survey on compound graph visualization) color, opacity, and overlap of these surfaces could be utilized to communicate such uncertainties.

Uncertainty in storyline visualization, and graph drawing more generally, remains an open challenge, further complicated by the many types of uncertainty that can manifest, e.g., node and edge attributes [EPDB*25, SSSE16], edge presence or weights [SNG*16], or other structures (such as group structures or time) [VRW13] and the layout itself [YC17]. This indicates visualizing uncertainty in storylines, as well as in the field of graph drawing more generally, is still an open direction and far from settled.

6. Conclusion

In this survey, we reviewed existing work on storyline visualization with a focus on layout algorithms and the assumptions behind them. Our goal was to clarify what is commonly done in this area, how different approaches relate to each other, and where they start to diverge.

Looking across the literature, most storyline methods follow a similar structure. Time is usually modeled as a sequence of discrete steps, layouts are driven by character interactions, and optimization focuses on a small set of well-known objectives such as reducing crossings, limiting vertical movement, and keeping the layout compact. Many papers build on the same algorithmic ideas and refine them incrementally, often combining several objectives in a pipeline.

Scalability remains a relevant concern for the visualization community, specifically, the *visual* and *cognitive and perceptual* aspects of it as introduced by Richer et al. [RPA*24]. Algorithmically, exact methods are limited to smaller datasets, however, heuristic approaches attempt to resolve this gap. Visually, as clutter grows due to the number of characters and interactions between them, only a handful of approaches address this explicitly, often through the use of glyphs or groupings and aggregation methods. Most storyline visualization techniques make the assumption that fewer crossings and wiggles may imply better comprehension and has perceptual and cognitive benefits, however this has not been empirically evaluated. This also remains an open challenge in literature.

At the same time, some aspects appear much less frequently. Only a few approaches move beyond basic interaction data to include additional attributes, alternative temporal models, or higher-level constraints such as fairness or uncertainty. When these aspects do appear, they are often treated as special cases rather than as part of the core layout process.

A **limitation** of this work is that we focus exclusively on the algorithmic and design choices underlying storyline visualizations. Consequently, our analysis considers visualization decisions primarily from an algorithmic perspective, rather than from the viewpoint of end users. Narrative visualization research addresses a much wider array of data types compared to storyline visualization and addresses user-centric challenges, such as plot comprehension, engagement, and storytelling. In contrast, out survey addresses the specific challenge of representing dynamic relational data using storyline visualization techniques. Future work could bring these two distant perspectives closer and investigate how layout decisions in storyline visualizations influence narrative comprehension or complement this perspective by examining how specific design choices relate to user tasks and by evaluateing these choices through empirical user studies. Building on the topic of empirical evaluations, we could systematically compare a wide range of storyline visualization algorithms across a well-chosen set of case studies to investigate the perceptual and cognitive implications that common optimization objectives have, such as, crossing/wiggle

minimization, temporal coherence, or compactness and aspect ratio. Another promising future research direction involve interaction techniques in storyline visualizations. While interactive approaches were not exclusively omitted from out survey, they were rarely described in the queried literature and our search terms did not target "interaction" specifically. Most papers focus on the algorithmic or optimization challenges, leaving space for the design and evaluation of interaction techniques to support analysis in storyline visualizations. In particular, there is no established taxonomy of interaction techniques aimed for storyline visualizations. Understanding which methods would be effective for which use cases, as well as, examining their impact to the visualization, i.e., how they introduce computational and/or layout constraints are open challenges.

We hope this survey helps readers orient themselves within the space of storyline visualization, identify suitable algorithms for different data and design requirements, and understand the trade-offs behind existing approaches.

Companion Website

We offer an interactive website with filtering f unctions t o navigate through the papers used to write this survey. The website can be found at: https://velitchko.github.io/storyline-survey/

Acknowledgements

Sara Di Bartolomeo was supported by the Austrian Science Fund (FWF) project [10.55776/ESP513]. Velitchko Filipov was supported by the Austrian Science Fund (FWF) projects SANE [10.55776/I6635] and ArtVis [10.55776/P35767]. Alexander Dobler and Martin Nöllenburg were supported by the Vienna Science and Technology Fund (WWTF) under grant [10.47379/ICT19035].

The authors acknowledge TU Wien Bibliothek for financial support through its Open Access Funding Programme. For open access purposes, the authors have applied a CC BY public copyright license to any author accepted manuscript version arising from this submission. Open Access funding provided by Technische Universitat Wien.

References

[AAK*13] ARCHAMBAULT D., ABELLO J., KENNEDY J., KOBOUROV S. G., MA K., MIKSCH S., MUELDER C., TELEA A. C.: Temporal multivariate networks. In *Multivariate Network Visualization - Dagstuhl Seminar #13201, Dagstuhl Castle, Germany, May 12-17, 2013, Revised Discussions* (2013), Kerren A., Purchase H. C., Ward M. O., (Eds.), vol. 8380 of *Lecture Notes in Computer Science*, Springer, pp. 151–174. doi:10.1007/978-3-319-06793-3_8. 4

[ACYB14] AKYIGIT E. E., CENGIZ T., YILDIRIM O. B., BALCISOY S.: Visual exploratory tool for storyline generation. In *9th IEEE Conference on Visual Analytics Science and Technology, IEEE VAST 2014, Paris, France, October 25-31, 2014* (2014), Chen M., Ebert D. S., North C., (Eds.), IEEE Computer Society, pp. 215–216. doi:10.1109/VAST.2014.7042497. 23

[AP17] ARENDT D., PIRRUNG M.: The "y" of it matters, even for storyline visualization. In *12th IEEE Conference on Visual Analytics Science and Technology, IEEE VAST 2017, Phoenix, AZ, USA, October 3-6, 2017* (2017), Fisher B. D., Liu S., Schreck T., (Eds.), IEEE Computer Society, pp. 81–91. doi:10.1109/VAST.2017.8585487. 5, 6, 12, 14, 15, 16, 18, 22, 23, 24

[Are15] ARENDT D. L.: SVEN: an alternative storyline framework for dynamic graph visualization. In *Graph Drawing and Network Visualization - 23rd International Symposium, GD 2015, Los Angeles, CA, USA, September 24-26, 2015, Revised Selected Papers* (2015), Giacomo E. D., Lubiw A., (Eds.), vol. 9411 of *Lecture Notes in Computer Science*, Springer, pp. 554–555. doi:10.1007/978-3-319-27261-0_48. 3, 5, 8, 12, 14, 15, 18

[AWA20] ABIDIN Z., WIDYANTORO D. H., AKBAR S.: A survey on visualization techniques to narrate interpersonal interactions between sportsmen. In *2020 International Conference on Smart Technology and Applications (ICoSTA)* (2020), pp. 1–6. doi:10.1109/ICoSTA48221.2020.1570614056. 23

[AXP*22] ARAYA V. P., XUE T., PIETRIGA E., AMSALEG L., BEZERIANOS A.: Hyperstorylines: Interactively untangling dynamic hypergraphs. *Inf. Vis. 21*, 1 (2022), 38–62. doi:10.1177/14738716211045007. 8, 9, 22, 23

[BBDW17] BECK F., BURCH M., DIEHL S., WEISKOPF D.: A taxonomy and survey of dynamic graph visualization. *Comput. Graph. Forum 36*, 1 (2017), 133–159. doi:10.1111/CGF.12791. 3

[BL76] BOOTH K. S., LUEKER G. S.: Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *J. Comput. Syst. Sci. 13*, 3 (1976), 335–379. doi:10.1016/S0022-0000(76)80045-1. 21

[Bor11] BORG I.: Multidimensional scaling. In *International Encyclopedia of Statistical Science*, Lovric M., (Ed.). Springer, 2011, pp. 875–878. doi:10.1007/978-3-642-04898-2_385. 17

[CLMY13] CHOU J., LIAO I., MA K., YANG C.: A study on enhancing timeline-like visualization with verbal text. In *2013 International Conference on Cyberworlds, Yokohama, Japan, October 21-23, 2013* (2013), Mao X., Hong L., (Eds.), IEEE Computer Society, pp. 206–213. doi:10.1109/CW.2013.36. 8, 9, 22, 23

[CLWW14] CUI W., LIU S., WU Z., WEI H.: How hierarchical topics evolve in large text corpora. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 2281–2290. doi:10.1109/TVCG.2014.2346433. 3

[CN23] COSTA M., NUNES S.: Newslines: Narrative visualization of news stories. In *Proceedings of Text2Story - Sixth Workshop on Narrative Extraction From Texts held in conjunction with the 45th European Conference on Information Retrieval (ECIR 2023), Dublin, Ireland, April 2, 2023* (2023), Campos R., Jorge A. M., Jatowt A., Bhatia S., Litvak M., (Eds.), vol. 3370 of *CEUR Workshop Proceedings*, CEUR-WS.org, pp. 37–46. URL: https://ceur-ws.org/Vol-3370/paper4.pdf. 8, 9, 22, 23

[DADL*19] DEVKOTA S., AHMED R., DE LUCA F., ISAACS K. E., KOBOUROV S.: Stress-plus-x (spx) graph layout. In *Graph Drawing and Network Visualization* (Cham, 2019), Archambault D., Tóth C. D., (Eds.), Springer International Publishing, pp. 291–304. doi:10.1007/978-3-030-35802-0_23. 24

[DBCS*24] DI BARTOLOMEO S., CRNOVRSANIN T., SAFFO D., PUERTA E., WILSON C., DUNNE C.: Evaluating graph layout algorithms: A systematic review of methods and best practices. *Computer Graphics Forum 43*, 6 (2024), e15073. doi:https://doi.org/10.1111/cgf.15073. 22

[DBKMW25] DI BARTOLOMEO S., KLUTE F., MONDAL D., WULMS J.: Graph Drawing Contest Report. In *33rd International Symposium on Graph Drawing and Network Visualization (GD 2025)* (Dagstuhl, Germany, 2025), Dujmović V., Montecchiani F., (Eds.), vol. 357 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, pp. 41:1–41:11. doi:10.4230/LIPIcs.GD.2025.41. 24, 25

[DBRGD22] DI BARTOLOMEO S., RIEDEWALD M., GATTERBAUER W., DUNNE C.: Stratisfimal layout: A modular optimization model

for laying out layered node-link network visualizations. *IEEE Transactions on Visualization and Computer Graphics 28*, 1 (2022), 324–334. doi:10.1109/TVCG.2021.3114756. 20

[DBZSD21] Di Bartolomeo S., Zhang Y., Sheng F., Dunne C.: Sequence braiding: Visual overviews of temporal event sequences and attributes. *IEEE Trans. Vis. Comput. Graph. 27*, 2 (2021), 1353–1363. doi:10.1109/TVCG.2020.3030442. 3

[DCS*24] Deng Z., Chen S., Schreck T., Deng D., Tang T., Xu M., Weng D., Wu Y.: Visualizing Large-Scale Spatial Time Series with GeoChron. *IEEE Transactions on Visualization and Computer Graphics 30*, 1 (Jan. 2024), 1194–1204. doi:10.1109/TVCG.2023.3327162. 5, 12, 14, 15, 22, 23, 24

[DHNW25] Dobler A., Hegemann T., Nöllenburg M., Wolff A.: Optimizing wiggle in storylines. In *Graph Drawing and Network Visualization (GD'25)* (2025), Dujmović V., Montecchiani F., (Eds.), vol. 357 of *LIPIcs*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, pp. 39:1–39:17. doi:10.4230/LIPIcs.GD.2025.39. 5, 6, 7, 18, 19, 21, 22, 23, 24

[DJJ*24] Dobler A., Jünger M., Jünger P. J., Meffert J., Mutzel P., Nöllenburg M.: Revisiting ILP models for exact crossing minimization in storyline drawings. In *32nd International Symposium on Graph Drawing and Network Visualization, GD 2024, September 18-20, 2024, Vienna, Austria* (2024), Felsner S., Klein K., (Eds.), vol. 320 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 31:1–31:19. doi:10.4230/LIPICS.GD.2024.31. 3, 5, 18, 20, 22, 23, 24

[DN24] Dobler A., Nöllenburg M.: Improving Temporal Treemaps by Minimizing Crossings. *Computer Graphics Forum* (2024). doi:10.1111/cgf.15087. 3

[DNS*23] Dobler A., Nöllenburg M., Stojanovic D., Villedieu A., Wulms J.: Crossing minimization in time interval storylines. *CoRR abs/2302.14213* (2023). doi:10.48550/ARXIV.2302.14213. 5, 7, 11, 12, 17, 18, 20, 22, 23, 24

[EMWR24] Ehlers H., Marin D., Wu H.-Y., Raidou R. G.: Visualizing group structure in compound graphs: The current state, lessons learned, and outstanding opportunities. In *VISIGRAPP (1): GRAPP, HUCAPP, IVAPP* (2024), pp. 697–708. doi:10.5220/0012431200003660. 4, 26

[EPDB*25] Ehlers H., Pahr D., Di Bartolomeo S., Filipov V., Wu H.-Y., Raidou R. G.: Wiggle! wiggle! wiggle! visualizing uncertainty in node attributes in straight-line node-link diagrams using animated wiggliness. *Computers & Graphics* (2025), 104290. doi:10.1016/j.cag.2025.104290. 26

[EPF*24] Ehlers H., Pahr D., Filipov V., Wu H.-Y., Raidou R. G.: Me! me! me! me! a study and comparison of ego network representations. *Computers & Graphics 125* (2024), 104123. doi:10.1016/j.cag.2024.104123. 7

[FAM23] Filipov V. A., Arleo A., Miksch S.: Are we there yet? A roadmap of network visualization from surveys to task taxonomies. *Comput. Graph. Forum 42*, 6 (2023). URL: https://doi.org/10.1111/cgf.14794, doi:10.1111/CGF.14794. 3

[FN18] Fröschl T., Nöllenburg M.: Minimzing wiggles in storyline visualizations. In *Graph Drawing and Network Visualization (GD'17)* (2018), Frati F., Ma K.-L., (Eds.), vol. 10692 of *LNCS*, Springer, pp. 585–587. URL: https://www.ac.tuwien.ac.at/files/pub/fn-mwsv-18.pdf. 20, 21

[For04] Forster M.: A fast and simple heuristic for constrained two-level crossing reduction. In *Graph Drawing, 12th International Symposium, GD 2004, New York, NY, USA, September 29 - October 2, 2004, Revised Selected Papers* (2004), Pach J., (Ed.), vol. 3383 of *Lecture Notes in Computer Science*, Springer, pp. 206–216. doi:10.1007/978-3-540-31843-9_22. 14

[FR91] Fruchterman T. M., Reingold E. M.: Graph drawing by force-directed placement. *Software: Practice and experience 21*, 11 (1991), 1129–1164. doi:10.1002/spe.4380211102. 16

[Frö18] Fröschl T.: *Minimizing wiggles in storyline visualizations*. Thesis, Technische Universität Wien, 2018. Accepted: 2020-06-29T15:57:57Z. doi:10.34726/hss.2018.53581. 5, 6, 18, 20, 21, 22, 23

[GDL*20] Giacomo E. D., Didimo W., Liotta G., Montecchiani F., Tappini A.: Storyline visualizations with ubiquitous actors. In *Graph Drawing and Network Visualization - 28th International Symposium, GD 2020, Vancouver, BC, Canada, September 16-18, 2020, Revised Selected Papers* (2020), Auber D., Valtr P., (Eds.), vol. 12590 of *Lecture Notes in Computer Science*, Springer, pp. 324–332. doi:10.1007/978-3-030-68766-3_25. 2, 3, 5, 8, 12, 15, 18, 21, 22, 23, 24

[GJLM16] Gronemann M., Jünger M., Liers F., Mambelli F.: Crossing minimization in storyline visualization. In *Graph Drawing and Network Visualization - 24th International Symposium, GD 2016, Athens, Greece, September 19-21, 2016, Revised Selected Papers* (2016), Hu Y., Nöllenburg M., (Eds.), vol. 9801 of *Lecture Notes in Computer Science*, Springer, pp. 367–381. doi:10.1007/978-3-319-50106-2_29. 2, 3, 5, 13, 18, 19, 20, 22, 23

[GKNV93] Gansner E., Koutsofios E., North S., Vo K.-P.: A technique for drawing directed graphs. *IEEE Transactions on Software Engineering 19*, 3 (1993), 214–230. doi:10.1109/32.221135. 2

[GMD*23] Giacomo E. D., Martino B. D., Didimo W., Esposito A., Liotta G., Montecchiani F.: Design of a process and a container-based cloud architecture for the automatic generation of storyline visualizations. In *Advanced Information Networking and Applications - Proceedings of the 37th International Conference on Advanced Information Networking and Applications (AINA-2023), Juiz de Fora, Brazil, 29-31 March 2023, Volume 3* (2023), Barolli L., (Ed.), vol. 655 of *Lecture Notes in Networks and Systems*, Springer, pp. 90–99. doi:10.1007/978-3-031-28694-0_9. 9, 23

[HAB23] Hulstein G., Araya V. P., Bezerianos A.: Geostorylines: Integrating maps into storyline visualizations. *IEEE Trans. Vis. Comput. Graph. 29*, 1 (2023), 994–1004. doi:10.1109/TVCG.2022.3209480. 8, 9, 22, 23, 24

[HBK15] Held P., Braune C., Kruse R.: Exploring dinofun park happenings. In *10th IEEE Conference on Visual Analytics Science and Technology, IEEE VAST 2015, Chicago, IL, USA, October 25-30, 2015* (2015), Chen M., Andrienko G. L., (Eds.), IEEE Computer Society, pp. 157–158. doi:10.1109/VAST.2015.7347658. 5, 8, 18, 22, 23

[Hub92] Huber P. J.: Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*. Springer, 1992, pp. 492–518. doi:10.1007/978-1-4612-4380-9_35. 16

[HW24] Hegemann T., Wolff A.: Storylines with a protagonist. In *32nd International Symposium on Graph Drawing and Network Visualization, GD 2024, September 18-20, 2024, Vienna, Austria* (2024), Felsner S., Klein K., (Eds.), vol. 320 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 26:1–26:22. doi:10.4230/LIPICS.GD.2024.26. 3, 5, 6, 7, 12, 14, 17, 18, 21, 22, 23, 24

[KCH10] Kim N. W., Card S. K., Heer J.: Tracing genealogical data with timenets. In *Proceedings of the International Conference on Advanced Visual Interfaces, AVI 2010, Roma, Italy, May 26-28, 2010* (2010), Santucci G., (Ed.), ACM Press, pp. 241–248. doi:10.1145/1842993.1843035. 8, 9, 10, 22, 23, 24

[KKC14] Kerracher N., Kennedy J., Chalmers K.: The Design Space of Temporal Graph Visualisation . In *EuroVis - Short Papers* (2014), Elmqvist N., Hlawitschka M., Kennedy J., (Eds.), The Eurographics Association. doi:/10.2312/eurovisshort.20141149. 3

[KLM25] Kuo Y.-H., Liu D., Ma K.-L.: SpreadLine: Visualizing Egocentric Dynamic Influence. *IEEE Transactions on Visualization and Computer Graphics 31*, 1 (Jan. 2025), 1050–1060. doi:10.1109/TVCG.2024.3456373. 5, 7, 8, 9, 10, 12, 14, 15, 22, 23, 24

[KNP*15] Kostitsyna I., Nöllenburg M., Polishchuk V.,

SCHULZ A., STRASH D.: On minimizing crossings in storyline visualizations. In *Graph Drawing and Network Visualization - 23rd International Symposium, GD 2015, Los Angeles, CA, USA, September 24-26, 2015, Revised Selected Papers* (2015), Giacomo E. D., Lubiw A., (Eds.), vol. 9411 of *Lecture Notes in Computer Science*, Springer, pp. 192–198. doi:10.1007/978-3-319-27261-0_16. 5, 18, 21, 22

[KW19] KÖPP W., WEINKAUF T.: Temporal treemaps: Static visualization of evolving trees. *IEEE Transactions on Visualization and Computer Graphics 25*, 1 (2019), 534–543. doi:10.1109/TVCG.2018.2865265. 3

[LCZ17] LU Q., CHAI B., ZHANG H.: Storytelling by the storycake visualization. *Vis. Comput. 33*, 10 (2017), 1241–1252. doi:10.1007/S00371-017-1409-2. 5, 8, 9, 12, 14, 15, 22, 23

[LHZ*21] LU Q., HUANG J., ZHANG Q., ET AL.: Evaluation on visualization methods of dynamic collaborative relationships for project management. *The Visual Computer 37*, 1 (2021), 161–174. doi:10.1007/s00371-019-01789-1. 5, 8, 9, 12, 14, 15, 18, 22, 23

[LLLW17] LIU Y., LIN H., LIANG Y., WANG C.: An application of optimization method for storyline based on cluster analysis. In *Proceedings of the 10th International Symposium on Visual Information Communication and Interaction, VINCI 2017, Bangkok, Thailand, August 14-16, 2017* (2017), Biuk-Aghai R. P., Li J., Takahashi S., (Eds.), ACM, pp. 24–28. doi:10.1145/3105971.3105986. 5, 8, 9, 12, 15, 17, 23, 24

[LWW*13] LIU S., WU Y., WEI E., LIU M., LIU Y.: Storyflow: Tracking the evolution of stories. *IEEE Trans. Vis. Comput. Graph. 19*, 12 (2013), 2436–2445. doi:10.1109/TVCG.2013.196. 3, 5, 6, 8, 9, 12, 13, 15, 17, 18, 22, 23

[LYK25] LIN J., YANG C.-K., KAO C.-H.: Visualizing nba information via storylines. *Computers & Graphics 127* (2025), 104169. doi:10.1016/j.cag.2025.104169. 5, 7, 8, 9, 12, 22, 23, 24

[LZLC14] LU Q., ZHU X., LIU L., CAO S.: An effective demonstration for group collaboration based on storyline visualization technology. In *Proceedings of the IEEE 18th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2014, Hsinchu, Taiwan, May 21-23, 2014* (2014), Hou J., Trappey A. J. C., Wu C., Chang K., Liao C., Shen W., Barthès J. A., Luo J., (Eds.), IEEE, pp. 47–52. doi:10.1109/CSCWD.2014.6846815. 5, 8, 9, 12, 14, 18, 22, 23

[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. doi:https://doi.org/10.1111/cgf.13610. 3

[MMB05] MOODY J., MCFARLAND D., BENDER-DEMOLL S.: Dynamic network visualization. *American Journal of Sociology 110*, 4 (2005), 1206–1241. doi:10.1086/421509. 3

[Mun09] MUNROE R.: Movie narrative charts, 2009. https://xkcd.com/657/, Accessed: 2025-02-20. URL: https://xkcd.com/657/. 2, 3, 23, 24

[NMSL19] NOBRE C., MEYER M. D., STREIT M., LEX A.: The state of the art in visualizing multivariate networks. *Comput. Graph. Forum 38*, 3 (2019), 807–832. doi:10.1111/CGF.13728. 4

[OK17] OSMAKCIC K., KOCIJAN K.: Story of a 'storyline visualization' in high school readings. In *40th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2017, Opatija, Croatia, May 22-26, 2017* (2017), Biljanovic P., Koricic M., Skala K., Grbac T. G., Cicin-Sain M., Sruk V., Ribaric S., Gros S., Vrdoljak B., Mauher M., Tijan E., Hormot F., (Eds.), IEEE, pp. 806–811. doi:10.23919/MIPRO.2017.7973532. 8, 9, 22, 23

[OM10] OGAWA M., MA K.: Software evolution storylines. In *Proceedings of the ACM 2010 Symposium on Software Visualization, Salt Lake City, UT, USA, October 25-26, 2010* (2010), Telea A. C., Görg C., Reiss S. P., (Eds.), ACM, pp. 35–42. doi:10.1145/1879211.1879219. 2, 3, 5, 8, 10, 12, 13, 22, 23, 24

[OMB*07] OGAWA M., MA K.-L., BIRD C., DEVANBU P., GOURLEY A.: Visualizing social interaction in open source software projects. In *2007 6th International Asia-Pacific Symposium on Visualization* (2007), pp. 25–32. doi:10.1109/APVIS.2007.329305. 3

[PBH18] PADIA K., BANDARA K. H., HEALEY C. G.: Yarn: Generating storyline visualizations using HTN planning. In *Proceedings of the 44th Graphics Interface Conference, Toronto, Canada, May 8-11, 2018* (2018), Batty C., Reilly D., (Eds.), ACM, pp. 26–33. 3, 5, 8, 10, 12, 14, 22, 23

[PC18] PING Q., CHEN C.: Litstoryteller+: an interactive system for multi-level scientific paper visual storytelling with a supportive text mining toolbox. *Scientometrics 116*, 3 (2018), 1887–1944. doi:10.1007/S11192-018-2803-X. 8, 9, 10, 22, 23

[PLM*25] PISELLI T., LIOTTA G., MONTECCHIANI F., NÖLLENBURG M., DI BARTOLOMEO S.: F^2 stories: A modular framework for multi-objective optimization of storylines with a focus on fairness. *IEEE VIS* (2025). doi:0.1109/TVCG.2025.3634228. 3, 5, 6, 8, 9, 11, 12, 15, 18, 20, 22, 23

[QC16] QIANG L., CHAI B.: Storycake: A hierarchical plot visualization method for storytelling in polar coordinates. In *2016 International Conference on Cyberworlds, CW 2016, Chongqing, China, September 28-30, 2016* (2016), Sourin A., (Ed.), IEEE Computer Society, pp. 211–218. doi:10.1109/CW.2016.43. 3, 5, 8, 9, 12, 14, 15, 22, 23

[RB10] ROSVALL M., BERGSTROM C. T.: Mapping change in large networks. *PLoS ONE 5*, 1 (2010), e8694. doi:10.1371/journal.pone.0008694. 3

[RPA*24] RICHER G., PISTER A., ABDELAAL M., FEKETE J.-D., SEDLMAIR M., WEISKOPF D.: Scalability in visualization. *IEEE Transactions on Visualization and Computer Graphics 30*, 7 (2024), 3314–3330. doi:10.1109/TVCG.2022.3231230. 26

[RTJ*11] REDA K., TANTIPATHANANANDH C., JOHNSON A. E., LEIGH J., BERGER-WOLF T. Y.: Visualizing the evolution of community structures in dynamic social networks. *Comput. Graph. Forum 30*, 3 (2011), 1061–1070. doi:10.1111/J.1467-8659.2011.01955.X. 3

[SAHW15] SILVIA S., ABBAS J., HUSKEY S., WEAVER C.: Storyline Visualization with Force Directed Layout. URL: https://www.cs.ou.edu/~weaver/academic/publications/silvia-2015a/materials/silvia-2015a.pdf. 5, 8, 9, 12, 16, 23, 24

[SBB*18] SHI Y., BRYAN C., BHAMIDIPATI S., ZHAO Y., ZHANG Y., MA K.: Meetingvis: Visual narratives to assist in recalling meeting context and content. *IEEE Trans. Vis. Comput. Graph. 24*, 6 (2018), 1918–1929. doi:10.1109/TVCG.2018.2816203. 8, 9, 10, 22, 23

[SEA*16] SILVIA S., ETEMADPOUR R., ABBAS J., HUSKEY S., WEAVER C.: Visualizing Variation in Classical Text with Force Directed Storylines. *Workshop on Visualization for the Digital Humanities* (2016). 5, 8, 12, 16, 17, 22, 23, 24

[SH10] SEGEL E., HEER J.: Narrative visualization: Telling stories with data. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1139–1148. doi:10.1109/TVCG.2010.179. 8

[SLW*22] SHI Z., LI M., WANG M., SHEN J., CHEN W., LUO X.: Npipvis: A visualization system involving NBA visual analysis and integrated learning model prediction. *Virtual Real. Intell. Hardw. 4*, 5 (2022), 444–458. doi:10.1016/J.VRIH.2022.08.008. 5, 8, 9, 12, 14, 15, 18, 19, 22, 23, 24

[SNG*16] SCHULZ C., NOCAJ A., GOERTLER J., DEUSSEN O., BRANDES U., WEISKOPF D.: Probabilistic graph layout for uncertain network visualization. *IEEE transactions on visualization and computer graphics 23*, 1 (2016), 531–540. doi:10.1109/TVCG.2016.2598919. 26

[SSSE16] SCHWANK J., SCHÖFFEL S., STÄRZ J., EBERT A.: Visualizing uncertainty of edge attributes in node-link diagrams. In *2016 20th International conference information visualisation (IV)* (2016), IEEE, pp. 45–50. doi:10.1109/IV.2016.19. 26

[STT81] SUGIYAMA K., TAGAWA S., TODA M.: Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics 11*, 2 (1981), 109–125. doi:10.1109/TSMC.1981.4308636. 2, 3, 5, 13, 14

[STY03] SAKAI S., TOGASAKI M., YAMAZAKI K.: A note on greedy algorithms for the maximum weighted independent set problem. *Discret. Appl. Math. 126*, 2-3 (2003), 313–322. doi:10.1016/S0166-218X(02)00205-6. 16

[SWL*14] SUN G., WU Y., LIU S., PENG T.-Q., ZHU J. J. H., LIANG R.: Evoriver: Visual analysis of topic coopetition on social media. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 1753–1762. doi:10.1109/TVCG.2014.2346919. 3

[TBS14] TAPASWI M., BÄUML M., STIEFELHAGEN R.: Storygraphs: Visualizing character interactions as a timeline. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014* (2014), IEEE Computer Society, pp. 827–834. doi:10.1109/CVPR.2014.111. 3, 5, 8, 10, 12, 16, 23

[THM15] TANAHASHI Y., HSUEH C., MA K.: An efficient framework for generating storyline visualizations from streaming data. *IEEE Trans. Vis. Comput. Graph. 21*, 6 (2015), 730–742. doi:10.1109/TVCG.2015.2392771. 5, 6, 8, 12, 13, 14, 15, 18, 22, 23

[TLW*21] TANG T., LI R., WU X., LIU S., KNITTEL J., KOCH S., YU L., REN P., ERTL T., WU Y.: Plotthread: Creating expressive storyline visualizations using reinforcement learning. *IEEE Trans. Vis. Comput. Graph. 27*, 2 (2021), 294–303. doi:10.1109/TVCG.2020.3030467. 5, 8, 9, 10, 12, 14, 15, 17, 18, 19, 22, 23

[TM12] TANAHASHI Y., MA K.: Design considerations for optimizing storyline visualizations. *IEEE Trans. Vis. Comput. Graph. 18*, 12 (2012), 2679–2688. doi:10.1109/TVCG.2012.212. 2, 5, 7, 8, 9, 10, 12, 13, 14, 22, 23

[TRL*19] TANG T., RUBAB S., LAI J., CUI W., YU L., WU Y.: istoryline: Effective convergence to hand-drawn storylines. *IEEE Trans. Vis. Comput. Graph. 25*, 1 (2019), 769–778. doi:10.1109/TVCG.2018.2864899. 5, 8, 9, 12, 14, 15, 17, 18, 22, 23

[VBW17] VEHLOW C., BECK F., WEISKOPF D.: Visualizing group structures in graphs: A survey. *Comput. Graph. Forum 36*, 6 (2017), 201–225. doi:10.1111/CGF.12872. 4, 26

[vDFF*16] VAN DIJK T. C., FINK M., FISCHER N., LIPP F., MARKFELDER P., RAVSKY A., SURI S., WOLFF A.: Block crossings in storyline visualizations. In *Graph Drawing and Network Visualization - 24th International Symposium, GD 2016, Athens, Greece, September 19-21, 2016, Revised Selected Papers* (2016), Hu Y., Nöllenburg M., (Eds.), vol. 9801 of *Lecture Notes in Computer Science*, Springer, pp. 382–398. doi:10.1007/978-3-319-50106-2_30. 5, 12, 14, 18, 21, 22, 23

[vDLMW17] VAN DIJK T. C., LIPP F., MARKFELDER P., WOLFF A.: Computing storyline visualizations with few block crossings. In *Graph Drawing and Network Visualization - 25th International Symposium, GD 2017, Boston, MA, USA, September 25-27, 2017, Revised Selected Papers* (2017), Frati F., Ma K., (Eds.), vol. 10692 of *Lecture Notes in Computer Science*, Springer, pp. 365–378. doi:10.1007/978-3-319-73915-1_29. 5, 6, 18, 21, 22, 23

[VRW13] VEHLOW C., REINHARDT T., WEISKOPF D.: Visualizing fuzzy overlapping communities in networks. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2486–2495. doi:10.1109/TVCG.2013.232. 26

[Wer38] WERTHEIMER M.: Laws of organization in perceptual forms. In *A source book of Gestalt psychology*, Ellis W. D., (Ed.). Kegan Paul, Trench, Trubner & Company, 1938, pp. 71–88. doi:10.1037/11496-005. 6

[WSZ*24] WANG Y., SUN G., ZHU Z., LI T., CHEN L., LIANG R.: E2storyline: Visualizing the relationship with triplet entities and event discovery. *ACM Trans. Intell. Syst. Technol. 15*, 1 (Jan. 2024). URL: https://doi.org/10.1145/3633519, doi:10.1145/3633519. 5, 7, 8, 9, 10, 11, 18, 20, 22, 23, 24

[WWD23] WALLNER G., WANG L., DORMANN C.: Visualizing the spatio-temporal evolution of gameplay using storyline visualization: A study with league of legends. *Proc. ACM Hum.-Comput. Interact. 7*, CHI PLAY (Oct. 2023). doi:10.1145/3611058. 8, 9, 22

[XWW*13] XU P., WU Y., WEI E., PENG T.-Q., LIU S., ZHU J. J., QU H.: Visual Analysis of Topic Competition on Social Media. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (Dec. 2013), 2012–2021. doi:10.1109/TVCG.2013.221. 3

[YC17] YAN K., CUI W.: Visualizing the uncertainty induced by graph layout algorithms. In *2017 IEEE Pacific Visualization Symposium (PacificVis)* (2017), IEEE, pp. 200–209. doi:10.1109/PACIFICVIS.2017.8031595. 26

[YMA*25] YEH C., MENON T., ARYA R. S., HE H., WEIGEL M., VIÉGAS F., WATTENBERG M.: Story ribbons: Reimagining storyline visualizations with large language models, 2025. URL: https://arxiv.org/abs/2508.06772, arXiv:2508.06772. 3

[YWR*25] YE L., WANG L., RUAN S., WANG H., MENG Y., WANG Y., CHEN W., ZHOU Z.: Storyexplorer: A visualization framework for storyline generation of textual narratives. *IEEE Trans. Hum. Mach. Syst. 55*, 5 (2025), 886–895. doi:10.1109/THMS.2025.3592357. 5, 8, 9, 12, 14, 15, 18, 22, 23

[YZC*25] YAO H., ZHAO L., CHEN B., LI K., LIANG H., YU L.: 3dstoryline: immersive visual storytelling. *J. Vis. 28*, 3 (2025), 681–697. doi:10.1007/S12650-025-01058-5. 6, 8, 10, 22, 23

[ZCY25] ZHANG M., CHEN L., YONG J.: Stratiline: A visualization system based on stratified storyline. *Comput. Graph. 127* (2025), 104166. doi:10.1016/J.CAG.2025.104166. 3, 5, 8, 9, 12, 14, 17, 22, 23, 24

[ZWQ*15] ZHENG Y., WU W., QU H., MA C., NI L. M.: Visual analysis of bi-directional movement behavior. In *2015 IEEE International Conference on Big Data (IEEE BigData 2015), Santa Clara, CA, USA, October 29 - November 1, 2015* (2015), IEEE Computer Society, pp. 581–590. doi:10.1109/BIGDATA.2015.7363802. 8, 9, 10, 22, 23, 24