Crossing Minimization in Time Interval Storylines*

Alexander Dobler, Martin Nöllenburg, Daniel Stojanovic, Anaïs Villedieu, and Jules Wulms

Algorithms and Complexity Group, TU Wien
{adobler|noellenburg|avilledieu|jwulms}@ac.tuwien.ac.at,
e11907566@student.tuwien.ac.at

Abstract

Storyline visualizations are a popular way of visualizing characters and their interactions over time: Characters are drawn as x-monotone curves and interactions are visualized through close proximity of the corresponding character curves in a vertical strip. Existing methods to generate storylines assume a total ordering of the interactions, although real-world data often do not contain such a total order. Instead, multiple interactions are often grouped into coarser time intervals such as years. We exploit this grouping property by introducing a new model called storylines with time intervals and present two methods to minimize the number of crossings and horizontal space usage. We then evaluate these algorithms on a small benchmark set to show their effectiveness.

Related Version arXiv:2302.14213

1 Introduction

Storyline visualizations are a popular way of visualizing characters and their interactions through time. They were popularized by Munroe's xkcd comic [13] (see Fig. 1 for a storyline describing a movie as a series of scenes through time, in which the characters participate). A character is drawn using an x-monotone curve, and the vertical ordering of the character curves varies from left to right. A scene is represented by closely gathering the curves of characters involved in said scene at the relevant spot on the x-axis, which represents time. Storylines attracted significant interest in visualization research, especially the question of designing automated methods to create storylines adhering to certain quality criteria [12, 14, 15].

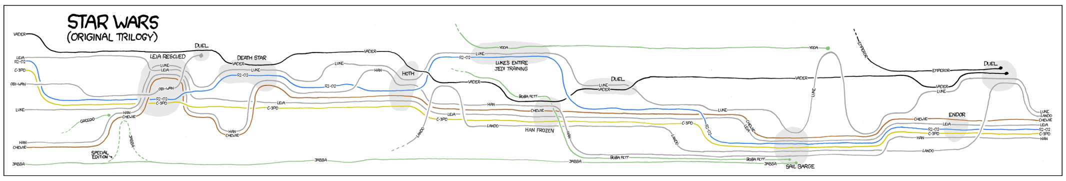


Figure 1 The xkcd comic showing a storyline of the Star Wars movie.

While different design optimization goals can be specified, most theoretical research has been focused on crossing minimization [8, 11] and variants like block crossing minimization [16, 17]. This problem is NP-hard [11, 16] and is commonly solved using ILP and SAT formulations [8, 17]; it has many similarities with the metro line crossing minimization problem [1–3, 5]. Recently a new model for storylines was proposed by Di Giacomo et al. [7] that allows for one character to be part of multiple interactions at the same point in time, by modeling each character as a tree rather than a curve. Using this model, it is possible to

* Alexander Dobler was supported by the Vienna Science and Technology Fund (WWTF) [10.47379/ICT19035]. Anaïs Villedieu was supported by the Austrian Science Fund (FWF) under grant P31119. We thank Martin Gronemann for the initial discussion.

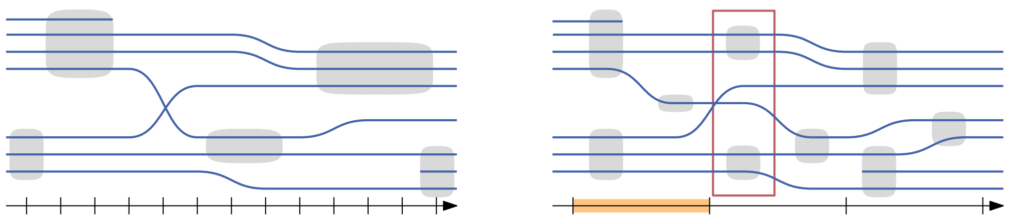


Figure 2 (a) A classic storyline with blue character lines. Interactions are shown in gray, they happen on specific timestamps and have a duration. **(b)** A time interval storyline. The horizontal orange segment shows a slice, every interaction on this segment has the same timestamp. A layer is highlighted in red, containing two interactions with the same timestamp but not sharing a character.

represent data sets which have a more loosely defined ordering of interactions. Furthermore, authorship networks have been a popular application for storylines visualizations [7, 10]. In this paper we introduce *time interval* storylines, an alternative approach to visualize data sets with less precise temporal attributes. In the time interval model, a set of discrete, totally ordered timestamps is given, which serve to label disjoint time intervals (e.g., the timestamp 2021 represents all interactions occurring between January and December of the year 2021). Each interval is represented in a storyline as a horizontal section in which all interactions with the same timestamp occur. The horizontal ordering within this section, however, does not correspond to a temporal ordering anymore (see Fig. 2). For example, an authorship network often sorts publications by year. In a traditional storyline model, the complete temporal ordering of the interactions must be provided. Previous models like the one by van Dijk et al. [17] can place multiple disjoint interactions in the same vertical layer, but the assignment of interactions to the totally ordered set of layers must be given as input. Unlike the traditional model, we have no pre-specified assignment of interactions to layers, but interactions with the same timestamp can be assigned to any layer within the time interval of this timestamp.

Problem setting. We are given a triple $\mathcal{S} = (\mathcal{C}, \mathcal{I}, T)$, of characters $\mathcal{C} = \{c_1, \ldots, c_n\}$, interactions $\mathcal{I} = \{I_1, \ldots, I_m\}$, and totally ordered timestamps $T = \{t_1, \ldots, t_p\}$ as input. Each interaction $(C_j, t) = I_j \in \mathcal{I}$ consists of a set $C_j \subseteq \mathcal{C}$ of characters involved in the interaction and a timestamp $t \in T$ at which the interaction I_j occurred, respectively denoted by $\texttt{char}(I_j) = C_j$ and $\texttt{time}(I_j) = t$. A subset of interactions can form a *layer* ℓ, when for every pair of interactions I, I' in ℓ, $\texttt{time}(I) = \texttt{time}(I')$. A time interval storyline is composed of a sequence of layers to which interactions are assigned. Intuitively, a layer represents a column in the storyline visualization, in which interactions are represented as vertical stacks. Thus, to each layer we associate a vertical ordering of $\mathcal{C}$. Consider the set S containing all interactions with timestamp t, we call the union of layers containing S a *slice*.

Characters are represented with curves passing through each layer at most once. To represent an interaction $I = (C, t)$ in a layer ℓ, the ordering of the characters in ℓ must be such that the characters of C appear consecutively in that ordering. For a pair I, I' of interactions in the same layer, it must hold that $\texttt{char}(I) \cap \texttt{char}(I') = \emptyset$.

For a layer ℓ, we denote the set of interactions by $\texttt{inter}(\ell)$ and the timestamp of a layer by $\texttt{time}(\ell)$ (with slight abuse of notation). We focus on combinatorial storylines, as opposed to geometric storylines, meaning that our algorithm should output a (horizontal) ordering $o_L(\mathcal{S})$ of layers, and for each layer ℓ, a (vertical) ordering $o_c(\ell)$ of the characters, and all interactions must occur in some layer. For two interactions I, I' such that $\texttt{time}(I) < \texttt{time}(I')$, let ℓ and ℓ' be the layers of I and I', respectively. Then ℓ must be before ℓ' in $o_L(\mathcal{S})$. A character is *active* in a layer if it appears in the character ordering for that layer. A character must be active in a contiguous range of layers including the first and last interaction it is involved in. A character is active in a layer if it appears in the character ordering for that layer.

Contributions. In this paper we introduce the time interval storylines model, as well as two methods to compute layer and character orderings. In Section 2.1 we introduce an algorithmic pipeline based on ILP formulations and heuristics that computes time interval storylines. We further present an ILP formulation that outputs a crossing-minimal time interval storyline in Section 2.2. Lastly in Section 3, we experimentally evaluate our pipeline and ILP formulation. Due to space constraints, some details are omitted and can be found in the full version of the paper [4].

2 Computing combinatorial storylines

2.1 A pipeline heuristic

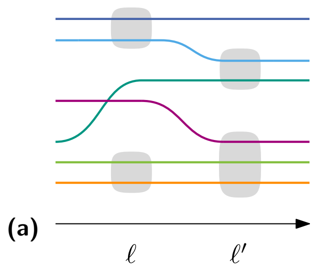
As the traditional storyline crossing minimization problem is a restricted version of the time interval formulation, our problem is immediately NP-hard [11]. Thus, we first aim to design an efficient heuristic to generate time interval storylines, which consists of the following stages.

(i) Initially, we assign each interaction to a layer,
(ii) then, we compute a horizontal ordering $o_L(\mathcal{S})$ of the layers obtained in step (i), and
(iii) finally, we compute a vertical ordering $o_c(\ell)$ of the characters for each layer $\ell \in o_L(\mathcal{S})$.

For step (i), the assignment is obtained using graph coloring. For each $t \in T$, we create a conflict graph $G_t = (\mathcal{I}_t, E)$ where $\mathcal{I}_t \subseteq \mathcal{I}$ and $I \in \mathcal{I}_t$ if and only if $\texttt{time}(I) = t$. Two interactions are connected by an edge if they share at least one character. Each color class then corresponds to a set of interactions which share no characters and can appear together in a layer. We solve this problem using a straightforward ILP formulation based on variables $x_{v,c} = 1$ if color c is assigned to vertex v and 0 otherwise. We can choose to limit the size of each color class by adding an upper bound on the number of interactions assigned to each color, which forces fewer interactions per layer. While this allows us to limit the height of each slice, it likely results in more layers.

To compute a horizontal ordering of the layers in step (ii), we use a traveling salesperson (TSP) model. Concretely, for the slice corresponding to the timestamp t, we create a complete weighted graph $G = (\mathcal{L}, E)$, where $\mathcal{L}$ corresponds to all the layers ℓ such that $\texttt{time}(\ell) = t$. For each edge e between a pair of layers ℓ and ℓ' in $\mathcal{L}$, we associate a weight w_e, estimating the number of crossings that may occur if the two layers are consecutive as follows.

Minimizing the crossings of the curves representing the characters is NP-complete [6, 11], thus we propose two heuristics to estimate the number of crossings. First, we propose to use set similarity measures to describe how similar the interactions in two layers ℓ and ℓ' are: If ℓ and ℓ' both have an interaction that contains the same set of characters, then no crossing should be induced by the curves corresponding to those characters, when these two layers are consecutive (see Fig. 3a). Second, we consider pattern matching methods that guess how many crossings could be induced by a certain ordering of the characters. There are certain patterns of interactions between two layers for which a crossing is unavoidable (see Fig. 3b). We count how many of these patterns occur between each pair of layers in G and set the weight of the corresponding edge to that crossing count. More details on these heuristics can be found in the full version [4].



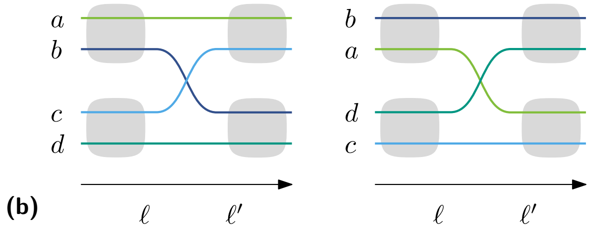


Figure 3 **(a)** The orange and light green characters are together in two interactions, which increases similarity between ℓ and ℓ', but the two blue characters are once together and once apart, which decreases similarity. **(b)** An example of an unavoidable crossing pattern.

To finish step (ii), we solve the path formulation of the TSP problem on G and find a horizontal ordering of the layers for each time slice. We have now obtained a traditional storyline, in which each interaction belongs to a specific layer, and all layers are totally ordered. Thus, we can solve step (iii) using the state-of-the-art crossing minimization ILP by Gronemann et al. [8].

We call the pipeline variants $\mathrm{P_s}$ and $\mathrm{P_p}$, when using the set similarity heuristic and the pattern matching heuristic in step (ii), respectively.

2.2 An ILP formulation

Crossing minimization in storylines is generally solved using ILP formulations [8,16]. We propose two formulations to handle slices, which build on the ideas of Gronemann et al. [8]. Both formulations will give us an assignment of interactions to layers, that are already totally ordered, and an ordering of characters per layer. For each timestamp $t \in T$, let $\mathcal{L}_t$ be a set of $|\{I \mid \mathtt{time}(I) = t\}|$ layers corresponding the number of interactions at t, and let $\mathcal{L} = \bigcup_{t \in T} \mathcal{L}_t$. In the first formulation we assume that a character c is active in all layers between the first timestamp and last timestamp, inclusively, where there exists an interaction I such that $c \in \mathtt{char}(I)$. In the second formulation we will introduce additional variables that model whether a character really needs to be active, since, in fact, character curves do not need to be active before their first interaction or after their last interaction. In contrast to the pipeline approach, the presented ILP formulations are able to find the crossing-minimal solution for the explored search space.

First formulation. Let $\mathcal{C}_\ell$ be the characters that appear in layer $\ell \in \mathcal{L}$, as discussed before. First we introduce for each $t \in T$ the binary variables $y_{\ell,I}$ for $\ell \in \mathcal{L}_t$ and $I \in \mathcal{I}$ where $\mathtt{time}(I) = t$. These should be one iff interaction I is assigned to layer ℓ. This is realized by constraints of type (1). If two different interactions I and I' share a character they cannot be in the same layer, realized by type (2) constraints.

$$\sum_{\ell \in \mathcal{L}_t} y_{\ell,I} = 1 \qquad t \in T, I \in \mathcal{I}, \mathtt{time}(I) = t \tag{1}$$

$$y_{\ell,I} + y_{\ell,I'} \leq 1 \qquad \mathtt{time}(I) = \mathtt{time}(I') = t, \mathtt{char}(I) \cap \mathtt{char}(I') \neq \emptyset, \ell \in \mathcal{L}_t \tag{2}$$

Next we introduce binary ordering variables x_{ℓ,c_i,c_j} for each layer $\ell \in \mathcal{L}$ and $c_i, c_j \in \mathcal{C}_\ell$ with $i < j$. Variable x_{ℓ,c_i,c_j} should be one iff c_i comes before c_j on layer ℓ. Standard transitivity constraints (3) (see e.g. [9]) ensure that the binary variables induce a total order.

$$0 \leq x_{\ell,c_h,c_i} + x_{\ell,c_i,c_j} - x_{\ell,c_h,c_j} \leq 1 \qquad c_i, c_j, c_h \in \mathcal{C}_\ell, i < j < h \tag{3}$$

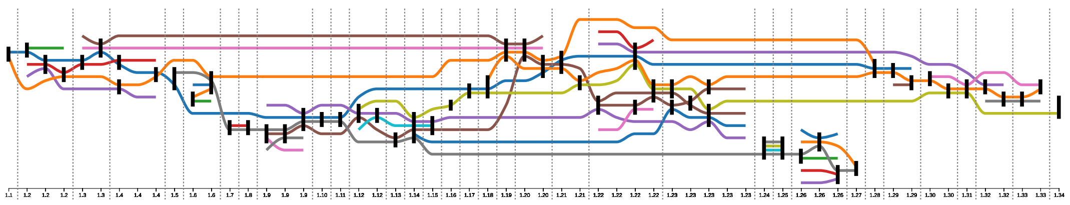


Figure 4 A storyline for a dataset corresponding to the first chapter of *Anna Karenina* created by ILP1. The x-axis is labeled by the scenes of the book, which are separated by dashed gray lines and correspond to the timestamps. Interactions are visualized with black vertical bars and correspond to the characters in the book interacting with each other shown as x-monotone curves. The storyline contains 58 layers, 34 timestamps, and 23 crossings.

The crux is now to model the assignment of some interaction I to some layer ℓ, linking the x- and y-variables together. This is done with so-called tree-constraints [8]: Let $\ell \in \mathcal{L}_t$, $I \in \mathcal{I}$ with $\texttt{time}(I) = t$ and $c_i, c_j, c_k \in \mathcal{C}_\ell$ such that $i < j$, $c_i, c_j \in \texttt{char}(I)$, and $c_k \notin \texttt{char}(I)$. If $i < j < k$ we add constraints (4) and (5), which ensure that c_k is either before or after both c_i and c_j. We elaborate on the analogous cases $k < i < j$ and $i < k < j$ in the full version [4].

$$x_{\ell,c_i,c_k} \leq x_{\ell,c_j,c_k} + 1 - y_{\ell,I} \tag{4}$$

$$x_{\ell,c_j,c_k} \leq x_{\ell,c_i,c_k} + 1 - y_{\ell,I} \tag{5}$$

Lastly, to optimize the number of crossings we have to provide an objective function. For this we introduce binary variables z_{ℓ,c_i,c_j} for all layers ℓ but the rightmost one and all $c_i, c_j \in \mathcal{C}_\ell \cap \mathcal{C}_{\ell'}$ where ℓ' is the adjacent layer of ℓ to the right. Variable z_{ℓ,c_i,c_j} should be one iff the character lines of c_i and c_j cross between layers ℓ and ℓ'. Linking variables z_{ℓ,c_i,c_j} is done by introducing the constraints corresponding to setting $z_{\ell,c_i,c_j} \geq x_{\ell,c_i,c_j} \oplus x_{\ell',c_i,c_j}$ (see the full version [4]) where $x \oplus y$ denotes the exclusive-or relation of two binary variables x and y. The objective is then to simply minimize $\sum z_{\ell,c_i,c_j}$. A solution to the ILP model is then transformed into a storyline realization of the input. We call this formulation ILP1. Figure 4 shows a storyline visualization that was computed with ILP1 and a simple post-processing method that assigns x- and y-coordinates (refer to the full version [4] for more information).

Extensions and Second Formulation In the above formulation we have one layer for each interaction, which does not utilize the potential of having multiple interactions in one layer. We can, however, minimize the number of layers beforehand, using the graph coloring problem as in Section 2.1. If we need q colors for timestamp t, we let $\mathcal{L}_t$ only consist of q layers. This can of course result in more crossings in the end. We call this adapted formulation ILP1ML.

Additionally, in the above model a character was contained in all layers of the first and last timestamp that contains an interaction, in which the character appears. By introducing further variables and adapting the constraints given in ILP1, this can be relaxed, so that a character's active range actually only spans from the first to the last interaction it appears in. This new formulation is given in the full version [4] and is referred to as ILP2. We can then introduce the fewest possible number of layers as above, resulting in formulation ILP2ML.

3 Evaluation

We evaluated our six algorithms on seven instances that were used in previous work on storylines. The number of layers that could be saved by the coloring pipeline-step, the number of crossings, and the runtimes are reported in the full paper [4] together with the complete description of instances, experimental setup, and evaluation. It also contains visualizations of storylines produced by our algorithms. Generally, the layer-minimization step of the pipeline, using graph coloring, reduces the number of layers for all but one instance, in one case even by 50%. The ILP-formulations perform better than the pipeline-approaches w.r.t. crossings, if they do not time out (3600 s). And even if they time out, the best feasible solution found by the solver is sometimes better than the solution provided by the pipeline-approaches. The ILP-approaches perform far worse w.r.t. runtime and three of the four instances timed out for all ILP-approaches. It has to be mentioned that, by construction, optimal solutions for ILP2 will always have the least crossings, and optimal solutions for ILP2ML will always have fewer crossings than all other algorithms minimizing layers. We think though, that our ILP-formulations can be further optimized for scalability.

References

1 Matthew Asquith, Joachim Gudmundsson, and Damian Merrick. An ILP for the metro-line crossing problem. In *Proc. 14th Symposium on Computing: Australasian Theory Symposium (CATS)*, volume 77 of *CRPIT*, pages 49–56, 2008.

2 Michael A. Bekos, Michael Kaufmann, Katerina Potika, and Antonios Symvonis. Line crossing minimization on metro maps. In Seok-Hee Hong, Takao Nishizeki, and Wu Quan, editors, *Proc. 15th International Symposium on Graph Drawing (GD)*, volume 4875 of *LNCS*, pages 231–242. Springer, 2007. doi:10.1007/978-3-540-77537-9_24.

3 Marc Benkert, Martin Nöllenburg, Takeaki Uno, and Alexander Wolff. Minimizing intra-edge crossings in wiring diagrams and public transportation maps. In M. Kaufmann and D. Wagner, editors, *Proc. 14th International Symposium on Graph Drawing (GD)*, volume 4372 of *LNCS*, pages 270–281. Springer-Verlag, 2007. doi:10.1007/978-3-540-70904-6_27.

4 Alexander Dobler, Martin Nöllenburg, Daniel Stojanovic, Anaïs Villedieu, and Jules Wulms. Crossing minimization in time interval storylines, 2023. doi:10.48550/ARXIV.2302.14213.

5 Martin Fink and Sergey Pupyrev. Metro-line crossing minimization: Hardness, approximations, and tractable cases. In Stephen Wismath and Alexander Wolff, editors, *Proc. 22nd International Symposium on Graph Drawing and Network Visualization (GD)*, volume 8242 of *LNCS*, pages 328–339. Springer-Verlag, 2013. doi:10.1007/978-3-319-03841-4_29.

6 M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM. J. Alg. Discr. Meth.*, 4(3):312–316, 1983. doi:10.1137/0604033.

7 Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, Fabrizio Montecchiani, and Alessandra Tappini. Storyline visualizations with ubiquitous actors. In David Auber and Pavel Valtr, editors, *Proc. 28th International Symposium on Graph Drawing and Network Visualization (GD)*, volume 12590 of *LNCS*, pages 324–332. Springer, 2020. doi:10.1007/978-3-030-68766-3_25.

8 Martin Gronemann, Michael Jünger, Frauke Liers, and Francesco Mambelli. Crossing minimization in storyline visualization. In Yifan Hu and Martin Nöllenburg, editors, *Proc. 24th International Symposium on Graph Drawing and Network Visualization (GD)*, volume 9801 of *LNCS*, pages 367–381. Springer, 2016. doi:10.1007/978-3-319-50106-2_29.

9 Martin Grötschel, Michael Jünger, and Gerhard Reinelt. Facets of the linear ordering polytope. *Math. Program.*, 33(1):43–60, 1985. doi:10.1007/BF01582010.

10 Tim Herrmann. Storyline-Visualisierungen für wissenschaftliche Kollaborationsgraphen. Master's thesis, University of Würzburg, 2022.

11 Irina Kostitsyna, Martin Nöllenburg, Valentin Polishchuk, André Schulz, and Darren Strash. On minimizing crossings in storyline visualizations. In Emilio Di Giacomo and Anna Lubiw, editors, *Proc. 23rd International Symposium on Graph Drawing and Network*

Visualization (GD), volume 9411 of *LNCS*, pages 192–198. Springer, 2015. `doi:10.1007/978-3-319-27261-0_16`.

12 Shixia Liu, Yingcai Wu, Enxun Wei, Mengchen Liu, and Yang Liu. Storyflow: Tracking the evolution of stories. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2436–2445, 2013. `doi:10.1109/TVCG.2013.196`.

13 Randall Munroe. Movie Narrative Charts, November 2009. URL: `https://xkcd.com/657/`.

14 Michael Ogawa and Kwan-Liu Ma. Software evolution storylines. In Alexandru C. Telea, Carsten Görg, and Steven P. Reiss, editors, *Proc. ACM 5th Symposium on Software Visualization (SOFTVIS)*, pages 35–42. ACM, 2010. `doi:10.1145/1879211.1879219`.

15 Yuzuru Tanahashi and Kwan-Liu Ma. Design considerations for optimizing storyline visualizations. *IEEE Trans. Vis. Comput. Graph.*, 18(12):2679–2688, 2012. `doi:10.1109/TVCG.2012.212`.

16 Thomas C. van Dijk, Martin Fink, Norbert Fischer, Fabian Lipp, Peter Markfelder, Alexander Ravsky, Subhash Suri, and Alexander Wolff. Block crossings in storyline visualizations. *J. Graph Algorithms Appl.*, 21(5):873–913, 2017. `doi:10.7155/jgaa.00443`.

17 Thomas C. van Dijk, Fabian Lipp, Peter Markfelder, and Alexander Wolff. Computing storyline visualizations with few block crossings. In Fabrizio Frati and Kwan-Liu Ma, editors, *Proc. 25th International Symposium on Graph Drawing and Network Visualization (GD)*, volume 10692 of *LNCS*, pages 365–378. Springer, 2017. `doi:10.1007/978-3-319-73915-1_29`.