




# Storylines with a Protagonist

Tim Hegemann   

Universität Würzburg, Germany

Alexander Wolff  

Universität Würzburg, Germany

---

## Abstract

*Storyline visualizations* show interactions between a given set of characters over time. Each *character* is represented by an x-monotone curve. A *meeting* is represented by a vertical bar that is crossed by the curves of exactly those characters that participate in the meeting. Therefore, character curves may have to cross each other. In the context of publication networks, we consider storylines where the characters are authors and the meetings are joint publications. We are especially interested in visualizing a group of colleagues centered around an author, the *protagonist*, who participates in all selected publications. For such instances, we propose a drawing style where the protagonist’s curve is drawn at a prominent position and never crossed by any other author’s curve.

We consider two variants of storylines with a protagonist. In the *one-sided* variant, the protagonist is required to be drawn at the top position. In this restricted setting, we can efficiently compute a drawing with the minimum number of *pairwise crossings*, whereas we show that it is NP-hard to minimize the number of *block crossings* (i.e., pairs of blocks of parallel curves that intersect each other). In the *two-sided* variant, the task is to split the set of co-authors of the protagonist into two groups, and to place the curves of one group above and the curves of the other group below the protagonist’s curve such that the total number of (block) crossings is minimized.

As our main result, we present an algorithm for bundling a sequence of pairwise crossings into a sequence of few block crossings (in the absence of meetings). It exploits a connection to a rectangle dissection problem. In the presence of meetings, it yields results that are very close to a lower bound. Based on this bundling algorithm and our exact algorithm for the one-sided variant, we present a new heuristic for computing two-sided storylines with few block crossings.

We perform an extensive experimental study using publication data of 81 protagonists from GD 2023 and their most frequent collaborators over the last ten years. Our study shows that, for two-sided storylines with a protagonist, our new heuristic uses fewer block crossings (and fewer pairwise crossings) than two heuristics for block crossing minimization in general storylines.

**2012 ACM Subject Classification** Theory of computation → Graph algorithms analysis; Human-centered computing → Graph drawings

**Keywords and phrases** Storyline visualization, storyline with a protagonist, crossing minimization, block crossings

**Digital Object Identifier** 10.4230/LIPIcs.GD.2024.26

**Supplementary Material** *Software (Source Code)*: <https://github.com/hegetim/publines> [14]  
archived at `swh:1:dir:8b6a0dd881ed0b7a4b0e5d8dbfa38f82827aa641`

**Funding** *Tim Hegemann*: Supported by BMBF grant 01IS22012C.

**Acknowledgements** We thank Felix Klesen for many discussions and Tim Herrmann for implementing the first version of GreedyBlocks. We also thank the reviewers of this paper for many helpful comments.

## 1 Introduction

Storyline visualizations are a visual tool to convey information about interactions between a group of entities – usually people – over time. Arguably, storyline visualizations have a long history that was started by Minard’s startling visualization of Napoleon’s Russian campaign [24], but certainly they have become quite popular since Munroe [26] used them to cleverly visualize several cinema classics. Our use case is different; see Figure 1.



© Tim Hegemann and Alexander Wolff;

licensed under Creative Commons License CC-BY 4.0

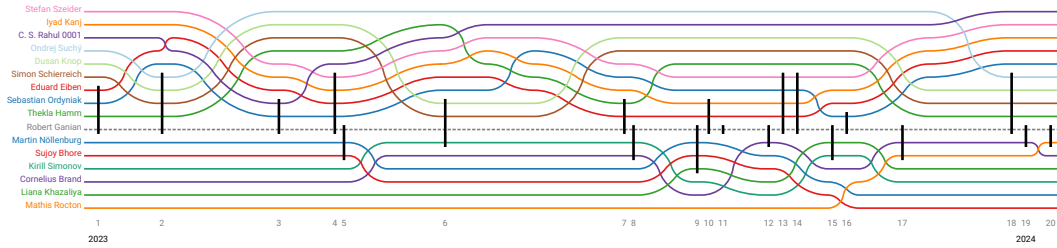
32nd International Symposium on Graph Drawing and Network Visualization (GD 2024).

Editors: Stefan Felsner and Karsten Klein; Article No. 26; pp. 26:1–26:22

Leibniz International Proceedings in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** Storyline visualization of Robert Galian and the 15 coauthors with whom he published most in the last year. In our drawing style, the protagonist (dashed line) has a prominent position – their curve is not crossed by any other curve. Make your own at <https://publines.github.io>!

More formally, a storyline  $\mathcal{S}$  is a pair  $(C, M)$ , where  $C = \{1, \dots, k\}$  is a set of *characters* and  $M = [m_1, \dots, m_n]$  is a sequence of *meetings*. If  $1 \leq i < j \leq n$ , we say that meeting  $m_i$  takes place before meeting  $m_j$  or that  $m_j$  is later than  $m_i$ . Every meeting is a subset of  $C$  of size at least 1. Note that we do not encode at what time a meeting happens exactly. In a *storyline visualization*, each character is represented by a continuous x-monotone curve. Let  $i \in \{1, \dots, n\}$ . At meeting  $m_i \subseteq C$ , the characters that participate in the meeting have to form an interval in the vertical order of the character curves. The meeting  $m_i$  is represented by a vertical line segment  $s_i$  at x-coordinate  $x_i$  such that (i) exactly the character curves in  $m_i$  cross  $s_i$  and (ii)  $x_i < x_j$  if  $i < j$ . A storyline visualization thus maps every point  $t \in [x_1, x_n]$  in time to a vertical order  $\pi_t$  of the characters.

In order to measure the quality of storyline visualizations (and to eventually design algorithms that produce readable drawings), various esthetic criteria have been suggested. Most works have focused on reducing the number of crossings of the character curves (simple pairwise crossings [12] or more complex types of crossings [31, 32]), others have also tried to minimize the number of wiggles [11] (that is, the number of turns) and/or the amount of vertical white-space [29, 23].

In this paper, we investigate a new variant of storylines that is motivated by the visualization of coauthor networks centered around a given main author or *protagonist*, that is, a character that is part of every meeting in a storyline. In order to stress the role of the protagonist, we disallow other curves to cross the protagonist’s curve which can thus be drawn as a straight horizontal line. We consider two variants concerning the placement of the protagonist: in the one-sided variant, the line of the protagonist is simply placed topmost, whereas in the two-sided variant, the line of the protagonist splits the other characters into two groups; those above and those below the line (see Figure 2). In both variants, we focus on minimizing crossings (but, in our experiments, we also keep track of the number of wiggles).

As we will show, minimizing pairwise crossings is easy in the one-sided variant. Therefore, we try to group these simple crossings into larger units, so-called *block crossings*. In a block crossing, two groups of curves are exchanged in the vertical ordering while no two curves within a group change their order. Such a grouping underlines the structure of the intersection pattern and leads to less visual clutter. Block crossings have also been investigated thoroughly beyond storyline visualization; in edge bundling [8], in metro maps [10], and due to their connection to the genus of a graph [3, 4, 5].

**Our contribution.** We introduce and formally define four variants of the new problem storyline crossing minimization with a protagonist; see Section 2. We show that one-sided storyline crossing minimization with a protagonist (1-SCM-P) can be solved efficiently; see Section 3. For the two-sided version of the problem (2-SCM-P), the characters have to

be split into two groups, which are then drawn above and below the horizontal line that represents the protagonist – using the algorithm for the one-sided case. We reduce the splitting problem to a max-cut problem and solve it heuristically; see Section 4. We then show that both variants of the storyline block crossing minimization problem (1-SBCM-P and 2-SBCM-P) are NP-hard; see Section 5.

In order to group pairwise crossings into blocks, we exploit a connection to partitioning the cells of crossing complexes into rectangular groups of cells that was observed by Fink et al. [8] in a more general topological setting. Note that in a purely geometric setting, the partitioning problem (namely of a simple orthogonal polygon into the minimum number of rectangles) can be solved efficiently, even in the presence of point holes [28]. In order to benefit from this, our bundling algorithm modifies the crossing complex heuristically such that it can be partitioned using algorithms for the geometric setting.

In our extensive experimental study we use publication data of 81 protagonists from GD 2023 and their most frequent collaborators over the last ten years; see Section 8. Somewhat unsurprisingly, a comparison shows that, for storylines with a protagonist, our specialized heuristic for 2-SBCM-P (combined with the bundling heuristic) uses fewer block crossings (and fewer pairwise crossings) than two heuristics for storyline block crossing minimization (SBCM) without a protagonist. On the other hand, these heuristics are free to allow the protagonist to cross other characters. From this point of view, it is indeed surprising that our 2-SBCM-P heuristic outperforms the two heuristics for SBCM.

We also evaluate the performance of our bundling heuristic against lower bounds that we obtain by partitioning the crossing complex without modifying it first. In general, this yields an optimal partitioning that cannot be realized geometrically, and hence, a lower bound. It turns out that the numbers of block crossings that our bundling heuristic produces are usually very close to this lower bound.

**More related work.** The *egocentric storylines* of Muelder, Crnovrsanin, Sallaberry, and Ma [25] probably come closest to our idea of storylines with a protagonist. However, their aim is to visualize large dynamic networks by selecting and drawing parts that are interesting for the user. As in our protagonist-setting, Muelder et al. allow the user to select a specific node  $p$  of the network, which is then displayed as a horizontal strip; in their case at the bottom of the layout. Then they select a subset of the nodes that are active in the current time step, namely nodes that are of relevance to  $p$  according to graph distance (combined with a time-based weight). Their input does not specify meetings; instead, more relevant nodes are placed closer to  $p$ . Nodes are shown as x-monotone strips between their first and last appearance; in time steps in which a node is not selected its strip becomes very thin and is placed behind the strips of selected nodes. Strips are colored according to a time-dependent clustering. Crossings are treated only implicitly, namely by reusing the vertical ordering of the nodes from the previous time step and inserting newly selected nodes according to their relevance.

Kim, Card, and Heer [17] visualized genealogical data using a storyline-like type of visualization. People are represented by x-monotone curves that start when they are born and end when they die (or when the diagram ends). The only type of meeting is marriage; that is, all meetings are of size 2. If a child is born, the start point of its curve is connected to its parents' curves by a dashed vertical line. Ancestors are placed by in-order, descendants by pre-order traversal of the family tree. Interestingly, when visualizing Elizabeth Taylor and her seven husbands [17, Fig. 9], the authors drew her as a protagonist and experiment with a 1- and a 2-sided layout. Again, crossings were not minimized; instead, after divorce, spouse curves simply return to the y-coordinate where they were before marriage.

Very recently, Kuo, Liu, and Ma [20] presented *SpreadLine*, a storyline-like visualization framework for egocentric networks that shares some properties with our two-sided approach. Instead of splitting the character set such that crossings are minimized, their split is based on an attribute of the characters given with the input. Hence, if the attribute of a character changes over time with respect to the protagonist’s attribute, the character’s curve will cross the protagonist’s curve. For generating the layout, Kuo et al. use the *StoryFlow* framework [23]. For crossing minimization, they use the well-known barycenter heuristic for one-sided crossing minimization in a sweeping fashion (i.e., starting on one side, and sweeping back and forth until the number of crossings no longer decreases).

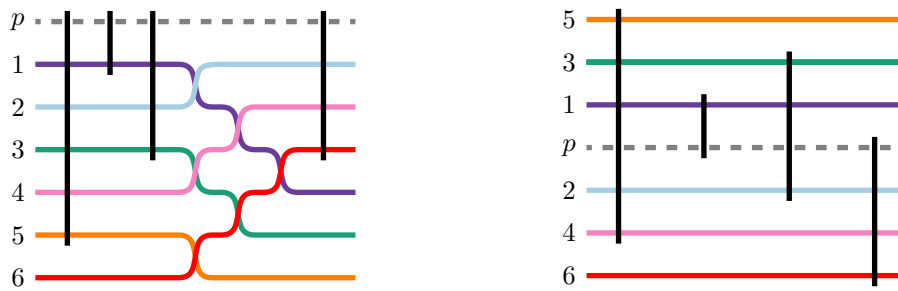
We now briefly review the existing (and not so closely related) approaches for storyline visualization in terms of computational techniques. Tanahashi and Ma [29] used a genetic algorithm to draw storylines with few crossings, few wiggles, and little white-space. Their algorithm is rather slow but produces esthetically pleasing results. Gronemann, Jünger, Liers, and Mambelli [12] used integer linear programming (ILP) to solve storyline crossing minimization (SCM) exactly. Fröschl and Nöllenburg [11] also used ILP, but in order to minimize the weighted number of wiggles (where each non-horizontal piece of a character’s curve is weighted by its height). Kostitsyna, Nöllenburg, Polishchuk, Schulz, and Strash [18] presented a fixed-parameter (FPT) algorithm for SCM. Van Dijk, Fink, Fischer, Markfelder, Ravsky, Suri, and Wolff [31] ([30]) gave the first FPT algorithm for SBCM and improved upon the running time of the FPT algorithm for SCM of Kostitsyna et al. Later, Van Dijk, Lipp, Markfelder, and Wolff [32] did an experimental study showing that, for SBCM, SAT-based algorithms are faster than ILP-based algorithms. Di Giacomo, Didimo, Liotta, Montecchiani, and Tappini [7] drew storylines where each character is represented by a plane tree rather than just by a curve, so a character can participate in several meetings simultaneously. They did two case studies, visualizing collaboration between scientists and work groups over time.

Another problem related to storyline visualization is metro-map layout, where metro lines are routed along the edges of an underlying graph whose vertices correspond to the metro stations. Note that other than the character curves in storyline visualization, the metro lines cannot go around a metro station if the station lies on the prescribed path that the metro line needs to follow. Fink and Pupyrev [9] showed that, given a fixed layout of the underlying graph, metro-line crossing minimization is NP-hard.

## 2 Preliminaries and Formal Problem Statement

We say that a meeting  $m$  *fits* a permutation  $\pi$  of  $C$  (or a permutation  $\pi$  *supports* a meeting  $m$ ) if the characters in  $m$  form an interval in  $\pi$ . In order to enable support for all meetings in  $M$ , the order of characters may have to change at several points in time. Whenever this order changes, the character curves cross. Since crossings make it harder for an observer to follow a character curve, we aim to minimize the number of crossings.

We describe a *crossing* that swaps two characters by their position in  $\pi$ . If a crossing swaps the character at  $a$  with that at  $a + 1$ , it maps the permutation  $\langle 1, \dots, a, a + 1, \dots, k \rangle$  to the permutation  $\langle 1, \dots, a - 1, a + 1, a, a + 2, \dots, k \rangle$ . When two disjoint blocks of curves cross all at once while staying parallel inside their respective blocks, we call this a *block crossing*. Let  $(a, b, c)$  with  $a \leq b < c$  be a block crossing that swaps the consecutive blocks  $\langle a, \dots, b \rangle$  and  $\langle b + 1, \dots, c \rangle$ . The permutation  $\langle 1, \dots, a, \dots, b, \dots, c, \dots, k \rangle$  is therefore mapped to the permutation  $\langle 1, \dots, a - 1, b + 1, \dots, c, a, \dots, b, c + 1, \dots, k \rangle$ . Note that the block crossing  $(a, a, a + 1)$  is equivalent to the pairwise crossing that swaps exactly the characters at  $a$  and  $a + 1$ . Let  $\pi_{\text{id}} = \langle 1, 2, \dots, k \rangle$  be the identity permutation for a set of  $k$  elements. For  $i \in \{1, \dots, k\}$ , we write  $\pi(i)$  for the position of character  $i$  in  $\pi$ . Analogously, for  $j \in \{1, \dots, k\}$ , we write  $\pi^{-1}(j)$  for the character at position  $j$  in  $\pi$ .



(a) One-sided storyline drawing with  $p$  at the top. (b) Two-sided drawing of the same storyline.

■ **Figure 2** In one- and two-sided storylines, the protagonist is drawn as a straight line and does not participate in any crossings.

In our visualization style, the protagonist maintains a fixed position and therefore, it must not participate in any (block) crossing that would move it away from that position. In a further restricted variant, we require the protagonist to be placed either at the top or bottom of the stack of character lines.

► **Definition 1** (*Two-Sided Storyline Crossing Minimization with a Protagonist (2-SCM-P)*). Given a storyline instance  $(C, M)$  with a protagonist  $p \in C$  and  $M = [m_1, m_2, \dots, m_n]$ , find a start permutation  $\pi_0$  and a sequence  $\mathcal{X} = [X_1, X_2, \dots, X_n]$  of (possibly empty) sequences of crossings such that (i)  $p$  is not involved in any crossing, (ii) for  $1 \leq i \leq n$ ,  $\pi_i = X_i(\pi_{i-1})$  supports  $m_i$ , and (iii) the total number of crossings is minimized.

The variant of the problem where we additionally require that  $\pi_0(1) = p$  is called *One-Sided Storyline Crossing Minimization with a Protagonist (1-SCM-P)*. The variants of the problem where we count block crossings instead of pairwise crossings are called *(One-/Two-Sided) Storyline Block Crossing Minimization with a Protagonist (1-/2-SBCM-P)*.

A *realization* of  $(C, M)$  is a pair  $(\pi_0, \mathcal{X})$ , where  $\pi_0$  is a start permutation and  $\mathcal{X}$  is a sequence of sequences of (block) crossings such that every meeting is supported. Figure 2 shows drawings of a one- and a two-sided storyline with a protagonist.

### 3 Minimizing Pairwise Crossings in 1-SCM-P

1-SCM-P has fewer degrees of freedom than the more general SCM problem. Indeed, we will show that 1-SCM-P can be solved in polynomial time. We start with a simple observation.

► **Observation 2.** Given two characters  $c$  and  $d$ , if a meeting  $m$  contains  $c$  but not  $d$  and a later meeting  $m'$  contains  $d$  but not  $c$ , then  $c$  and  $d$  must cross between  $m$  and  $m'$ .

A crossing that fulfills the condition stated in the above observation is *unavoidable*. For each character  $c \in C$ , we define its *attendance vector*  $v_c \in \{0, 1\}^{|M|}$  where  $v_c(i) = 1$  if and only if  $c \in m_i$ . For any pair  $(c, d)$  of characters with  $c < d$ , we count the number  $U_{cd}$  of unavoidable crossings between  $c$  and  $d$  by removing entries from the attendance vectors until only those remain that inflict unavoidable crossings. First, remove all entries  $i$  where  $v_c(i) = v_d(i)$ . Then, remove all entries  $j$  where  $(v_c(j), v_d(j)) = (v_c(j+1), v_d(j+1))$ . Now  $U_{cd}$  is the number of remaining entries minus one, and the number of unavoidable crossings for the whole storyline is  $\sum_{1 \leq c < d \leq k} U_{cd}$ .

► **Lemma 3.** Every instance of 1-SCM-P can be realized with unavoidable crossings only.

**Proof.** The statement is trivial for storylines that can be realized without crossings. For a storyline  $(C, M)$  with protagonist  $p$  that requires crossings, we show the statement by contradiction. Let  $(\pi_0, \mathcal{X})$  be a 1-SCM-P realization of  $(C, M)$  with the minimum number of crossings. Let  $c$  and  $d$  be two characters whose curves cross. Let  $c$  be the upper curve before the crossing, and let  $t \in [0, n]$  be the point in time when they cross. Clearly, in order to cross,  $c$  and  $d$  must lie on the same side of  $p$ , say below. Now assume that this crossing is not unavoidable by our definition. Hence, there is a meeting  $m_i$  with  $i < t$  that contains  $c$  but not  $d$  and there is no meeting  $m_j$  with  $j > i$  that contains  $d$  but not  $c$ . (The case that the later meeting contains only  $d$  but not  $c$ , and there is no earlier meeting that contains  $c$  but not  $d$  is symmetric to the case that we study and can be handled analogously.)

We can safely remove the crossing of  $c$  and  $d$  at time  $t$  and swap the positions of  $c$  and  $d$  in all permutations after  $t$ . Any meeting  $m_j$  with  $j > i$  that either contains both  $c$  and  $d$  or none of the two is not affected by this change. A meeting  $m_j$  that contains only  $c$  would require another crossing between  $c$  and  $d$ , which we could also remove safely. Hence, we obtain a realization with fewer crossings than  $(\pi_0, \mathcal{X})$ , contradicting our choice of  $(\pi_0, \mathcal{X})$ . ◀

► **Theorem 4.** *There is an algorithm that solves 1-SCM-P in  $\mathcal{O}(k^2n)$  time, where  $k$  is the number of characters and  $n$  is the number of meetings.*

**Proof.** From Lemma 3 we know that the unavoidable crossings are sufficient to realize a storyline with a protagonist. Therefore, we present Algorithm 1 and prove that it produces exactly the unavoidable crossings.

We set the start permutation  $\pi_0$  such that the characters are in descending lexicographic order with respect to their attendance vectors. Note that the first meeting  $m_1$  fits  $\pi_0$ . Let  $i \in \{2, 3, \dots, k\}$  and assume that we have already computed a storyline up to and including meeting  $m_{i-1}$ . Let  $\pi_{i-1}$  be the permutation right after  $m_{i-1}$ . Given  $\pi_{i-1}$ , in order to support  $m_i$ , we have to move all characters in  $S_1 = m_i \setminus m_{i-1}$  towards  $p$  (who attends all meetings) and all characters in  $S_2 = m_{i-1} \setminus m_i$  away from  $p$ . Since we must not cross  $p$ , every character in  $S_1$  must cross every character in  $S_2$ . By Observation 2, these crossings are unavoidable. For  $S_1$  and  $S_2$  to form contiguous blocks, we may have to introduce additional crossings. Assume that we need such a crossing between characters  $c$  and  $d$  to move  $d$  closer to  $p$ . Let  $m_k$  be the latest meeting before  $m_{i-1}$  that contains  $c$  but not  $d$ . Such a meeting must exist, otherwise  $c$  and  $d$  would be swapped in  $\pi_0$ . Hence, due to Observation 2,  $m_k$  and  $m_i$  induce an unavoidable crossing between  $c$  and  $d$ . Consequently, our algorithm produces only unavoidable crossings.

Any crossing introduced by one of the inner while-loops of Algorithm 1 moves character  $c$  upwards (that is, decreases  $\pi^{-1}(c)$  in that loop). Therefore, the foreach-loops check at most  $k^2$  pairs of characters, and Algorithm 1 runs in  $\mathcal{O}(k^2n)$  time. ◀

## 4 Minimizing Pairwise Crossings in 2-SCM-P

In 1-SCM-P, we restrict the storyline such that the protagonist  $p$  always is at the topmost position. This may introduce a lot of crossings compared to storyline visualizations without this constraint. The variant 2-SCM-P drops this constraint but still requires that  $p$  does not participate in any crossing (i.e., can be drawn straight); see Figure 2b.

We map an instance  $(C, M)$  of 2-SCM-P to two instances of 1-SCM-P by splitting  $C \setminus \{p\}$  into two sets  $C_1$  and  $C_2$ . For  $j \in \{1, 2\}$ , we set  $C_j = C_j \cup \{p\}$  and let  $M_j$  be the restriction of  $M$  to characters in  $C_j$ . In order to find a split, we define the *crossing graph* of  $(C, M)$  to be the complete graph with vertex set  $C \setminus \{p\}$ , where, for characters  $c$  and  $d$  with  $c < d$ , edge

■ **Algorithm 1** One-sided storyline with minimum number of pairwise crossings.

---

**Input:** Instance  $(C, M)$  of one-sided SCM, attendance vectors  $v_1, v_2, \dots, v_k$   
**Output:** Realization  $(\pi_0, \mathcal{X} = [X_1, \dots, X_{|M|}])$  of  $(C, M)$

- 1  $\pi_0 \leftarrow$  characters in descending lexicographic order by attendance vectors
- 2  $\pi \leftarrow \pi_0$
- 3  $X_1 \leftarrow \emptyset$
- 4 **for**  $i \leftarrow 2$  **to**  $|M|$  **do**
- 5      $X_i \leftarrow \emptyset$
- 6     **if**  $m_i$  fits  $\pi$  **then continue**
- 7      $S_0 \leftarrow m_{i-1} \cap m_i$
- 8      $S_1 \leftarrow m_{i-1} \setminus m_i$
- 9      $S_2 \leftarrow m_i \setminus m_{i-1}$
- 10     $S_3 \leftarrow C \setminus (m_{i-1} \cup m_i)$
- 11    **foreach**  $c \in S_0$  ordered by  $\pi^{-1}$  **do**
- 12        **while**  $d \leftarrow \pi(\pi^{-1}(c) - 1) \in S_1$  **do**
- 13            switch  $c$  and  $d$  in  $\pi$
- 14            append  $(\pi^{-1}(c), \pi^{-1}(c), \pi^{-1}(d))$  to  $X_i$                                 // a pairwise crossing
- 15    **foreach**  $c \in S_2$  ordered by  $\pi^{-1}$  **do**
- 16        **while**  $d \leftarrow \pi(\pi^{-1}(c) - 1) \in S_3$  **do**
- 17            switch  $c$  and  $d$  in  $\pi$
- 18            append  $(\pi^{-1}(c), \pi^{-1}(c), \pi^{-1}(d))$  to  $X_i$                                 // a pairwise crossing
- 19    append  $(|S_0| + 1, |m_i|, |m_i| + |S_2|)$  to  $X_i$                                         // a block crossing
- 20    apply  $X_i$  to  $\pi$
- 21 **return**  $(\pi_0, \mathcal{X})$

---

$\{c, d\}$  has weight  $U_{cd}$ . Consider the problem (WEIGHTED) MIN-UNCUT which asks for a 2-coloring of the vertices of a graph such that the number (total weight) of the monochromatic edges is minimized. If two characters  $c$  and  $d$  are on different sides of  $p$ , then they do not cross. Otherwise they cause exactly  $U_{cd}$  unavoidable crossings, independently of the presence of other characters. By Theorem 4, we can solve the two 1-SCM-P instances resulting from the split optimally. Hence a split that minimizes the number of crossings corresponds to a solution of WEIGHTED MIN-UNCUT in the crossing graph.

Unfortunately, MIN-UNCUT is MaxSNP-hard [27]; it admits an  $O(\sqrt{\log n})$ -approximation [1]. Note that MIN-UNCUT is the complement of MAX-CUT, which asks for a 2-coloring of the vertices of a graph such that the number of the bichromatic edges is maximized. In particular, the set of optimal solutions is the same for both problems.

We can efficiently detect instances of 2-SCM-P that can be drawn without crossings.

► **Theorem 5.** *Given a 2-SCM-P instance with  $k$  characters and  $n$  meetings, we can test in  $\mathcal{O}(nk^2)$  time whether it admits a solution without crossings.*

**Proof.** Construct the crossing graph, test whether it is bipartite, and if yes, draw the two resulting one-sided instances using Theorem 4. ◀

Also if the crossing graph is planar, WEIGHTED MAX-CUT and hence 2-SCM-P can be solved efficiently [22]. Similarly, exact FPT algorithms for WEIGHTED MAX-CUT [6] carry over to 2-SCM-P. For our application where we require fast response for larges instances, we use a heuristic for WEIGHTED MAX-CUT [16] that is easy to implement and, although asymptotically  $O(|C|^3)$ , sufficiently fast.

## 5 Hardness of 1-SBCM-P and 2-SBCM-P

Van Dijk et al. [31] showed that SBCM is NP-complete by reducing from SORTING BY TRANSPOSITIONS (SBT). In SBT, given a permutation  $\pi$  and an integer  $t$ , the task is to decide whether  $\pi$  can be transformed into the identity permutation by applying a sequence of at most  $t$  transpositions (which we call block crossings). SBT is NP-hard [2]. Whereas in SBT the start permutation is given, for SBCM we need a gadget in order to ensure that there is a point in time where the characters are ordered as in  $\pi$ . We now construct such a gadget.

► **Lemma 6.** *Given a set  $C$  of  $k$  characters and a permutation  $\pi$  of  $C$ , there exists a sequence of meetings  $M$  of size  $k - 1$  such that the one-sided storyline  $(C, M)$  with protagonist  $\pi(1)$  can be drawn crossing-free and  $\pi$  is the only permutation that supports all meetings in  $M$ .*

**Proof.** For any given order of characters the sequence of meetings  $[m_1, \dots, m_{k-1}]$  with  $m_i = \{1, \dots, i + 1\}$  is supported by the identity permutation  $\pi_{\text{id}}$ . Any permutation  $\pi' \neq \pi$  contains at least two characters  $c$  and  $d$  with  $c < d$  whose positions are swapped compared to  $\pi$ . By construction,  $\pi'$  does not support  $m_{c-1}$ . ◀

► **Theorem 7.** *The problems 1-SBCM-P and 2-SBCM-P are NP-complete.*

**Proof.** It is easy to see that the decision variants of both 1-SBCM-P and 2-SBCM-P lie in NP. The number of (block) crossings necessary to support any meeting is bounded by  $\binom{k}{2}$ . So we can simply check a solution from left to right, in time polynomial in  $k$  and  $n$ .

Next we show that 1-SBCM-P is NP-hard. Using Lemma 6 we can build a one-sided storyline with a protagonist  $p$  such that  $C \setminus \{p\}$  is in  $\pi$ -order just after  $m_{k-1}$  and in  $\pi_{\text{id}}$ -order just after  $m_k$ ; encoding the permutations for SBT (see the blue box in Figure 3).

Solving 1-SBCM-P for this instance gives us a start permutation  $\pi_0 = \pi$  and a sequence  $\mathcal{X} = [X_1, \dots, X_{k-1}, X_k, X_{k+1}, \dots, X_{2k-2}]$  of (possibly empty) sequences of block crossings. Let  $\pi_j$  denote the order of characters right after  $m_j$ . Meetings  $m_1, \dots, m_{k-1}$  allow us to maintain  $\pi$  from  $\pi_0$  to  $\pi_{k-1}$ . Because of that, we can prepend (in order) all block crossings from  $X_1, \dots, X_{k-1}$  to  $X_k$ . The same is true for meetings  $m_k, \dots, m_{2k-2}$  and the identity permutation. So block crossings from  $X_{k+1}, \dots, X_{2k-2}$  can be appended (in order) to  $X_k$ . Now,  $X_k$  contains a minimum set of block crossings (a.k.a. transpositions) necessary in order to transform  $\pi$  to the identity.

For the NP-hardness of 2-SBCM-P, we reuse our proof for the one-sided variant. We use a simple gadget consisting of one additional author  $q$  and  $2k^2$  meetings to fix an assignment of the authors to the sides that has all authors that are part of our one-sided instance in the same half of the drawing (see the red box in Figure 3). We add meetings  $M' = [m'_1, \dots, m'_{2k^2}]$  where meetings with even index contain exactly  $p$  and  $q$  while those with odd index contain  $p$  and all characters  $1, \dots, k$ . The resulting storyline  $(C, M')$  can be drawn crossing-free if and only if  $C \cup \{q\}$  is split  $(C \setminus \{q\}, \{q, p\})$ . Any other split introduces at least  $2k^2 - 2$  block crossings (one for each meeting in  $M'$  except for the first and the last one). On the other hand, transforming  $\pi$  into identity takes at most  $\binom{k}{2}$  crossings. Therefore,  $(C \setminus \{q\}, \{q, p\})$  is the only optimal split, and the two-sided instance  $(C \cup \{q\}, M' \circ M)$ , where  $\circ$  denotes concatenation, encodes the one-sided instance  $(C, M)$ . ◀



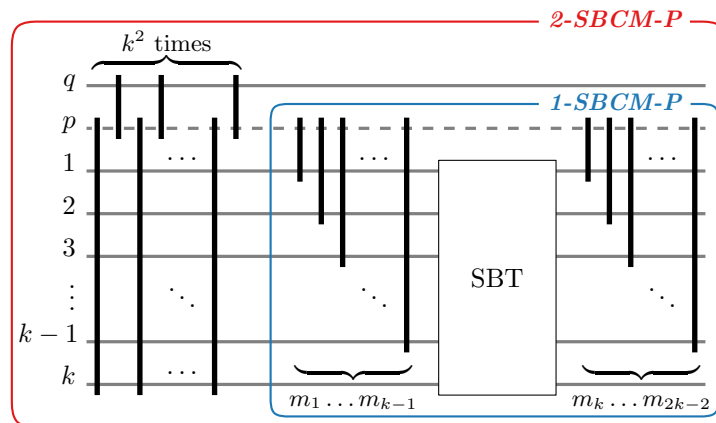


Figure 3 The NP-hardness construction for 2-SBCM-P reuses the one for 1-SBCM-P (blue box).

## 6 Bundling Pairwise Crossings in Storylines

While we can solve at least 1-SCM-P efficiently, we have seen that 1-SBCM-P is already NP-hard (Theorem 7). Still, we prefer block crossings from a cognitive point of view; they structure the set of pairwise crossings. Therefore, we now discuss the problem of covering a given set of pairwise crossings by the smallest number of block crossings. Fink et al. [8] introduced this problem for general graph embeddings and showed its NP-hardness.

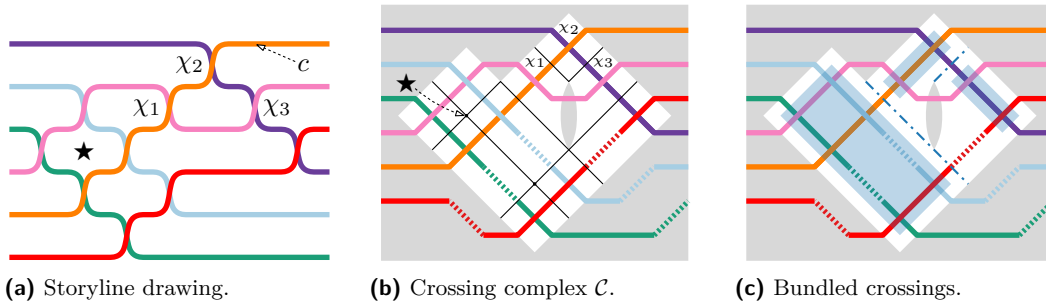
### 6.1 Bundling in the absence of meetings

Fortunately, in the special case of storylines, we can solve the following local version of the problem efficiently as shown below in Theorem 11. The version corresponds to the problem that needs to be solved between two consecutive meetings, given the sequence of pairwise crossings.

► **Definition 8** (Local Bundling for Storylines). *Given a permutation  $\pi$  and a sequence  $X$  of pairwise crossings, find the shortest sequence  $X'$  of block crossings such that, when applying  $X'$  to  $\pi$ , exactly the same lines cross as when applying  $X$  to  $\pi$ .*

Fink, Hershberger, Suri, and Verbeek [8] observed a connection between the bundled crossings problem and the minimum dissection problem for orthogonal polygons with holes. Soltan and Gorpinevich [28] showed that an orthogonal polygon with arbitrary holes can be dissected into the minimum number of rectangles in polynomial time. Their algorithm can be adapted to solve *Local Bundling for Storylines* as we will show now.

We say that two crossings  $\chi_1$  and  $\chi_2$  *touch* (and can hence be combined into a block crossing) if (i) they share exactly one character  $c$  and (ii) the part of the curve that represents  $c$  between the corresponding crossing points is  $y$ -monotone and is not crossed by any other curve. In Figure 4a, crossings  $\chi_1$  and  $\chi_2$  touch, but  $\chi_1$  and  $\chi_3$  do *not* touch and, unlike [8], we *cannot* combine them into a block crossing. We call the graph  $\mathcal{G}$  that has a vertex for every pairwise crossing in  $X$  and an edge for every pair of touching crossings the *touching graph* of  $X$ . Following Fink et al. [8], we define the *crossing complex*  $\mathcal{C}$  for a sequence  $X$  of pairwise crossings in a storyline as a special type of complex that consists of quadrilateral cells with sides and corners, where two respectively four cells touch:



■ **Figure 4** The crossing complex  $\mathcal{C}$  of a storyline drawing (a) contains a quadrilateral cell for each crossing. In (b) the cells are bounded by either the gray exterior (which includes holes) of  $\mathcal{C}$  or the thin black lines. Each cell shares a side (black line segments) with at most four other cells. A bundling corresponds to a dissection of  $\mathcal{C}$  into (blue shaded) rectangular groups of cells (c).

1. The complex  $\mathcal{C}$  contains a quadrilateral cell for each crossing in  $X$ . Each side of a cell corresponds to one “half” of a character curve; the one before and the one after the crossing.
2. If two crossings  $\chi = \{b, c\}$  and  $\psi = \{c, d\}$  in  $X$  touch each other, then their cells share the sides that correspond to the unique character in  $\chi \cap \psi$ . (Note that, in Figure 4b, the cells of  $\chi_1$  and  $\chi_2$  share a side, whereas the cells of  $\chi_1$  and  $\chi_3$  do not share a side.)
3. If the storyline drawing contains a quadrilateral face and the adjacent crossings form a cycle in  $\mathcal{G}$ , then the unique corner shared by the corresponding cells is part of  $\mathcal{C}$  (e.g., the starred face in Figure 4a becomes the starred corner in Figure 4b).

The crossing complex of the storyline drawing in Figure 4a is shown in Figure 4b. The exterior of  $\mathcal{C}$  is shaded in gray. Corners and sides that lie in the interior of  $\mathcal{C}$  are referred to as *internal*. In Figure 4b internal corners are marked by small black disks and internal sides by solid black line segments.

► **Lemma 9.** *The crossing complex  $\mathcal{C}$  of a storyline drawing can be laid out such that all internal sides are either falling or rising (i.e., have a slope of  $45^\circ$  or  $-45^\circ$ ).*

**Proof.** By item 2 of the definition above, an internal side can be drawn perpendicular to the curve of the common character of the two touching crossings. Since  $\mathcal{G}$  is a partial grid graph, it can be drawn orthogonally with respect to the embedding implied by the storyline drawing (see Figure 4b). Every crossing touches at most four other crossings, and the four arms of a crossing naturally yield a grid embedding. Therefore, internal sides can be drawn on a grid rotated by  $45^\circ$ . ◀

We assume that  $\mathcal{C}$  has been laid out as described in Lemma 9. In order to dissect  $\mathcal{C}$  into the minimum number of rectangular groups of cells, we cut  $\mathcal{C}$  along chords. A *chord* is a sequence of colinear internal sides that starts and ends in a *boundary corner* (i.e., a corner that is not internal) and has only internal corners in between. It is easy to see that cuts are necessary at every concave corner on the boundary  $\partial\mathcal{C}$  of  $\mathcal{C}$ . Following Fink et al. [8], for each boundary corner  $z$ , we define its measure of “concaveness”  $\kappa(z) = \lfloor (\alpha(z) - 1)/2 \rfloor$ , where  $\alpha(z)$  is the number of cells incident to  $z$ . This measure indicates how many cuts are necessary to make a corner convex. We will show that two cuts always suffice.

► **Lemma 10.** *Let  $z$  be a corner of  $\mathcal{C}$ . Then at most five cells are incident to  $z$ , and  $\kappa(z) \leq 2$ .*

**Proof.** Corners in  $\mathcal{C}$  (and on the boundary of  $\mathcal{C}$ ) map to faces in the original storyline drawing. Internal corners in  $\mathcal{C}$ , by construction, result from a quadrilateral face in the storyline drawing; therefore, they have only four adjacent cells. All other corners lie on points on  $\partial\mathcal{C}$  where internal sides have a common endpoint. This is the case when the crossings around a face in the storyline drawing form a path in  $\mathcal{G}$  (each touching pair of crossings implies an internal side). Since, at each touching pair of crossings, a face has a corner with an angle of  $90^\circ$  and the sum of angles of a face is  $360^\circ$ , such a path cannot be longer than five crossings.  $\blacktriangleleft$

All corners with four adjacent cells are internal by definition. By Lemma 10, we know that concave boundary corners have three or five adjacent cells and thus, their measure is 1 or 2. Hence, every cut along a chord starting in a concave corner  $z$  decreases  $\kappa(z)$  by 1.

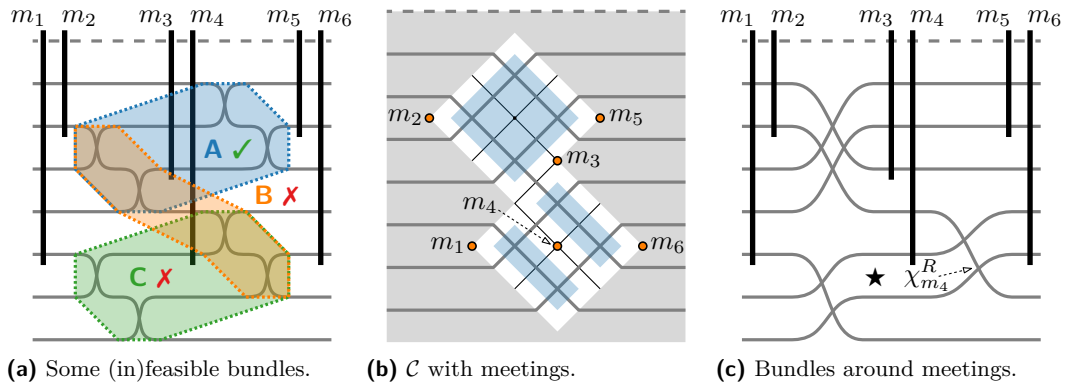
We identify two different types of chords; *effective chords*, which connect two concave corners and therefore decrease  $\kappa(\mathcal{C}) = \sum_{z \in \partial\mathcal{C}} \kappa(z)$  by 2, and *simple chords*, which connect a concave and a convex corner. In order to minimize the number of cuts (and therefore rectangles), we must maximize the number of cuts along effective chords. Note that not all combinations of effective chords are feasible. Two effective chords are *in conflict* if they share a common internal corner. After cutting along one of them, the other one is no longer effective by our definition. Instead, it was cut into two simple chords. Recall that concave corners have measure 1 or 2. Let  $z$  be a corner. If  $\kappa(z) = 1$ , then every pair of effective chords incident to  $z$  is in conflict, because a single cut is sufficient to make  $z$  convex. If  $\kappa(z) = 2$ , then only those pairs of effective chords incident to  $z$  are in conflict if the two chords together do not reduce  $\kappa(z)$  to 0. This is the case if, after cutting twice,  $z$  still has three consecutive cells.

In order to identify a maximum set of effective chords, we find a maximum independent set in their *conflict graph*. Note that due to the orthogonal nature of our chords, the conflict graph is bipartite and therefore a maximum independent set can be found in polynomial time (see König's theorem [21]). After applying cuts along a maximum set of effective chords, we handle the remaining concave corners by choosing an arbitrary simple chord starting at the corner and cutting along it. After that all connected components of  $\mathcal{C}$  are rectangular (i.e., their boundary corners are  $90^\circ$  or  $180^\circ$ ). For the storyline depicted in Figure 4a, a dissection into rectangles is shown in Figure 4c.

It remains to translate the order of rectangles implied by the embedding of  $\mathcal{C}$  back into a feasible order for the bundles. Given two bundles  $A$  and  $B$ ,  $A$  must happen before  $B$  if, for any two crossings  $\chi_A$  in  $A$  and  $\chi_B$  in  $B$ , (i) the cell of  $\chi_A$  has a top-right or bottom-right edge to the cell of  $\chi_B$  in  $\mathcal{G}$  (according to its grid embedding) or (ii)  $\chi_A$  and  $\chi_B$  are on the same *level* (i.e., if  $\pi_A$  is the permutation right before  $\chi_A$  and  $\pi_B$  is the one right before  $\chi_B$ , the character curves involved in  $\chi_A$  have the same position in  $\pi_A$  as those of  $\chi_B$  in  $\pi_B$ ) and  $\chi_A$  comes before  $\chi_B$  in  $X$ . We then topologically sort the directed graph with a vertex for every bundle and arcs corresponding to the ordering constraints. This gives us a feasible order for the bundles.

► **Theorem 11.** *Given a storyline with a sequence  $X$  of pairwise crossings, Local Bundling for Storylines can be solved in  $\mathcal{O}(|X|^2)$  time.*

**Proof.** The complex  $\mathcal{C}$  can be constructed in time linear in  $|X|$  since the storyline yields the topology and, for each crossing in  $X$ , we introduce at most four sides and corners. We then find the effective chords in time linear in  $|X|$  because at most four chords start, end, or go through any corner, and no two chords overlap. Since effective chords are in conflict only if they start at, end at, or go through the same corner, the number of nodes and edges in the conflict graph is linear in  $|X|$ . Hence, we can find a dissection of  $\mathcal{C}$  (corresponding to



■ **Figure 5** Not all bundles are feasible when it comes to meetings (a). Meetings correspond to point holes in the crossing complex  $\mathcal{C}$  (b). A possible bundling is shown in (c).

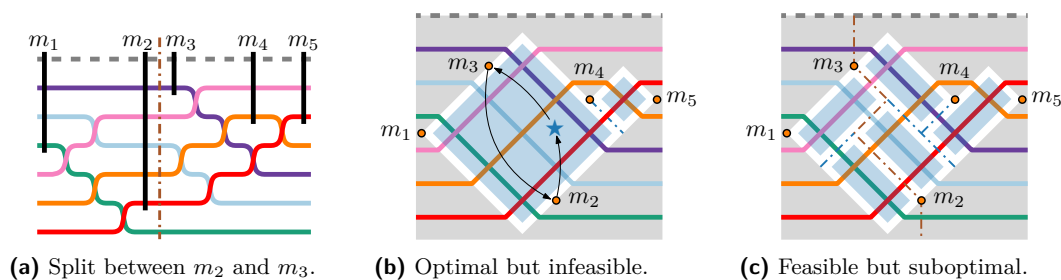
a maximum independent set in the conflict graph) in  $\mathcal{O}(|X|^2)$  time using the algorithm of Soltan and Gorpinevich [28]. Applying the set of effective chords and adding the remaining simple chords is again bounded by the number of sides because every side is cut at most once. The number of bundles is bounded by  $|X|$ . We can find all ordering constraints and topologically sort the conflict graph in  $\mathcal{O}(|X|^2)$  time. ◀

## 6.2 Bundling in the presence of meetings

So far we only bundled crossings between two neighboring meetings. It is obviously beneficial and often possible to bundle crossings across meetings. We allow a meeting only to happen before or after a block crossing (as shown in Figure 5c) but not inside. Hence, we may neither trap a meeting inside a bundle (see bundle  $C$  in Figure 5a) nor change the order of meetings. For example, in Figure 5a, meeting  $m_4$  must happen before the bundled crossing  $B$ , whereas meeting  $m_3$  must happen after  $B$ . Because we cannot change the order of meetings, such bundles are prohibited.

For a meeting  $m$ , let  $\chi_m^R$  be the first crossing after  $m$  where one of the characters whose curves cross is part of  $m$ , whereas the other is not. We say that  $\chi_m^R$  *touches  $m$  from the right* (see Figure 5c). Note that  $\chi_m^R$  is adjacent to one of the faces where  $m$  ends (see the black star in Figure 5c). In order to prevent bundles (such as  $C$  in Figure 5a) to trap meetings inside, we prohibit faces where meetings end from being part of a bundle. For the crossing complex  $\mathcal{C}$ , this means that we remove the corresponding corner from  $\mathcal{C}$ . The dissection algorithm can be easily enhanced to support point holes [28]. Point holes get a measure of two and effective chords starting or ending at a point hole are in conflict if and only if they have different slopes (one falling one rising).

The last step of the bundling procedure, *ordering bundles*, must adhere to two further kinds of constraints, namely meeting–meeting and meeting–bundle constraints. A meeting must be placed to the right of (or after) another meeting as indicated by the sequence  $M$  of meetings. A bundle must be placed to the right of a meeting  $m$  if any crossing in the bundle touches  $m$  from the right. In Figure 6b, the constraints between meetings  $m_2$  and  $m_3$  and the bundle marked with the blue star are depicted as black arrows. An arrow starting at  $a$  and pointing at  $b$  indicates that  $a$  must be placed to the right of  $b$ . Unfortunately, sometimes these constraints form cycles and therefore the bundling cannot be realized geometrically (for example the bundle with the blue star in Figure 6b or bundle  $B$  in Figure 5a).



■ **Figure 6** With the optimal dissection this bundling has a dependency cycle between meetings  $m_2$ ,  $m_3$ , and the bundle marked with the blue star (b). When we split the storyline between the conflicting meetings (a), the dissection becomes feasible but suboptimal (c) as three bundles would be sufficient.

We propose the following heuristic: whenever we encounter such a cycle, we split the storyline after the first meeting in the cycle (i.e., the meeting with the smallest index). See Figure 6a for an example. Note that every such cycle must involve at least one meeting. It remains open whether or not we can find a cut through  $\mathcal{C}$  that still produces an optimal result regarding the number of bundles. Cutting straight after a meeting and through the full height of the storyline can worsen the solution as shown in Figure 6c. We can insert the point holes and find the additional constraints within the same asymptotic runtime as above. We effectively run our algorithm once per split so theoretically our heuristic has a runtime of  $O(|X|^3)$ . However, in practice splits are rare and bundling is almost imperceptibly quick (see Section 8).

Summarizing, we can find an optimal bundling (i.e., with the minimum number of bundles) as long as no dependency cycles occur when ordering the resulting block crossings. Otherwise, we can use the number of bundles in the infeasible solution as a lower bound for the optimal number of bundles. Our experiments (see below) show that even our simple heuristic that splits the storyline whenever it encounters a conflict yields optimal results for many instances.

## 7 A Greedy Heuristic for SBCM

In this section we present a greedy algorithm to draw a storyline with few block crossings in  $\mathcal{O}(k^2sn)$  time, where  $|C| = k$  is the number of characters,  $|M| = n$  is the number of meetings, and  $\sum_i |X_i| = s$  is the number of block crossings. Because SBCM is NP-hard [31], we cannot hope for an optimal solution. Our algorithm repeatedly adds block crossings until all meetings fit. Our heuristic is based on previous work by Herrmann [15], but we use the following more involved scoring function. Our scoring function considers possible block crossings and chooses weights differently; moreover, we do not limit the number of meetings that are factored into the scoring. We use the attendance vector  $v_c$  of a character  $c$  (see Section 3) in order to find *t-conflict-free pairs*, that is, pairs that can stay in the same block for the next  $t$  meetings. A pair of characters  $(c, d)$  at a meeting  $m_i$  is *t-conflict-free* for a number  $t$  if  $v_c(j) = v_d(j)$  for  $i \leq j \leq i + t$ .

We process the meetings in order. For  $i \in \{2, 3, \dots, n\}$ , let  $\pi$  be the permutation right after  $m_{i-1}$ . Let  $\langle \pi(a), \dots, \pi(b-1) \rangle$  and  $\langle \pi(c), \dots, \pi(d) \rangle$  be two maximal blocks of characters in  $\pi$  that all attend  $m_i$  and are separated by another block of characters  $\langle \pi(b), \dots, \pi(c-1) \rangle$  that all do not attend  $m_i$ . Note that if no such configuration exists,  $m_i$  fits  $\pi$  and we continue with  $m_{i+1}$ . We now have several options to join the two blocks with a single block crossing. We can either merge the first block into the second or the second into the first and we can

---

**Algorithm 2** SELECTBLOCKCROSSING.
 

---

**Input:** Permutation  $\pi$ , meeting  $m_i$ , attendance vectors  $v_1, \dots, v_k$   
**Output:** Block crossing

```

1 find  $a < b < c \leq d$  such that
2    $\{\pi(j) \mid j \in \{a, \dots, b-1, c, \dots, d\}\} \subseteq m_i$  and
3    $\{\pi(j) \mid j \in \{1, \dots, a-1, b, \dots, c-1, d+1\}\} \cap m_i = \emptyset$ 
4  $best \leftarrow \emptyset$ 
5  $\sigma_{best} \leftarrow -\infty$ 
6 foreach  $\chi \in \{(z, c-1, d) \mid z \in \{a, \dots, b\}\} \cup \{(a, b-1, z) \mid z \in \{c-1, \dots, d\}\}$  do
7    $\pi' \leftarrow \pi$  with  $\chi$  applied
8    $\sigma_\chi \leftarrow 0$ 
9   foreach  $(e, f) \in \{(\pi'(j), \pi'(j+1)) \mid a-1 \leq j \leq d\}$  do
10     $i' \leftarrow i$ 
11    while  $v_e[i'] == v_f[i']$  do  $i' \leftarrow i' + 1$ 
12     $\sigma_\chi \leftarrow \sigma_\chi + i' - i$ 
13  if  $\sigma_{best} < \sigma_\chi$  or  $(\sigma_{best} == \sigma_\chi$  and  $size(best) > size(\chi)$ ) then
14     $best \leftarrow \chi$ 
15     $\sigma_{best} \leftarrow \sigma_\chi$ 
16 return  $best$ 

```

---

choose any position inside one of the blocks where we insert the other block. We use the procedure described in Algorithm 2 in order to rank the possible block crossings and select the best based on adjacent conflict-free pairs and the size of the block crossing (i.e., number of characters involved). Let  $\pi'$  be the permutation after applying the block crossing that we found in the previous step. If  $\pi'$  does not support  $m_i$ , we repeat the described procedure with  $\pi'$  instead of  $\pi$ .

Our algorithm heavily depends on a good start permutation. We propose the following strategy to find one. Let  $M'$  be the sequence of meetings obtained by reversing  $M$ , and let  $\tilde{\pi}$  be a random permutation. We apply our greedy algorithm to  $(C, M')$  with  $\tilde{\pi}$  as a start permutation and record the permutation  $\pi'_{\text{end}}$  after the last meeting. Note that the last meeting in  $M'$  is the first meeting in  $M$ . We use  $\pi'_{\text{end}}$  as the start permutation for the actual run.

For each block crossing in the final drawing, we call SELECTBLOCKCROSSING (see Algorithm 2) once. The parameters  $a, b, c$ , and  $d$  can be determined in  $\mathcal{O}(k)$  time. At most  $k$  block crossings are considered and each requires up to  $k-1$  pairs of characters to be scored. The scoring function can be computed in  $\mathcal{O}(n)$  time. Therefore, the total runtime is  $\mathcal{O}(k^2sn)$ , where  $s$  is the number of block crossings. Note that, in most cases, a lot fewer than  $n$  comparisons are required to score a pair of characters. If the worst-case runtime bound is a concern, the maximum number of comparisons can be limited by a constant, which reduces the runtime to  $\mathcal{O}(k^2s)$  (and may slightly increase the number of block crossings).

## 8 Experiments

Our use case for storylines with a protagonist is to visualize how the peer group of a scientific author changes over time. We are interested in comparing heuristics for visualizing storylines with a protagonist to heuristics for visualizing storylines without a protagonist. We want to measure whether the latter benefit from the additional degree of freedom that allows them to choose realizations where the protagonist's curve is crossed by the curves of other characters. We also want to evaluate the performance of the bundling heuristic that we presented in Section 6.2.

**Benchmark set.** For our experiments, we identified the 81 authors of short and long papers in the GD 2023 proceedings that have at least 20 coauthors on publications over the last 10 years that are listed in the dblp computer science bibliography at <https://dblp.org>, a very reliable source of publication data (for computer science). For each among the 81 authors, we created storylines with their 5, 10, 15, and 20 most frequent coauthors. This yielded our benchmark set with 324 instances.

**Metrics.** Metrics for assessing the quality of a visualization look for measurable features that influence legibility and aesthetics. In the context of storylines, Tanahashi and Ma [29] discussed formative design criteria and suggested to measure line wiggles, line crossings, and white-space. Because our drawing style does not produce white-space, we measure *wiggles* and *crossings*. Additionally, we count the number of *block crossings* (or bundled crossings). We define the number of wiggles as the number of times a character curve changes position. This is equivalent to the sum of the sizes of all block crossings where the size of a block crossing is the number of lines involved. Note that the wiggles metric, as we defined it, is equivalent to what is known as passages in the context of metro maps [13].

**Algorithms.** We implemented the following four algorithms for drawing storylines with a protagonist with few (block) crossings. We call the algorithm that solves 1-SCM-P exactly *1-Sider* (see Section 3). Recall that our heuristic for the two-sided variant first splits the set of characters using a heuristic for MAX-CUT and then solves the resulting 1-SCM-P instances exactly (see Section 4). We call this heuristic for 2-SCM-P *2-Sider*. We call our greedy heuristic for SBCM *GreedyBlocks* (see Section 7).

As a baseline for our experiments we used a simple heuristic that we call *Median*. For each meeting, it selects the median of all attending characters (by their position in the current permutation) and join the remaining participants with the minimum number of crossings. For finding a decent start permutation, we use the same trick as with *GreedyBlocks* and draw the reversed storyline with a random start permutation.

We combine *1-Sider*, *2-Sider*, and *Median* with our algorithm for bundling pairwise crossings into block crossings (see Section 6). Note that our implementation uses Kuhn’s classic algorithm [19] for computing a maximum independent set in a bipartite graph; its cubic runtime is worse than what we stated in Theorem 11, but the algorithm is easy to implement and fast enough in our setting.

We implemented all algorithms as TypeScript web applications. We performed the tests under Fedora Linux 40 and node.js v20 on an AMD Ryzen 7 7840HS with 64 GB of RAM.

**Results.** We applied the four algorithms to each instance in our benchmark set and compared the results in terms of crossings, block crossings, and wiggles. In Table 1,  $\mu$  denotes the average of that metric over all 81 realizations of that algorithm and number of characters. For every algorithm we also counted how often they produced the best result (with that metric) out of all algorithms tested; see the columns labeled  $\beta$ .

The results for the largest storylines with 21 authors are shown in detail in Figure 7. All measurements are relative to *GreedyBlocks*, so an algorithm with a measure below 1 performs better than *GreedyBlock* for that specific data set. For each algorithm, a horizontal line marks the median of its results. The medians and the detailed results in general support the overall trend that Table 1 shows.

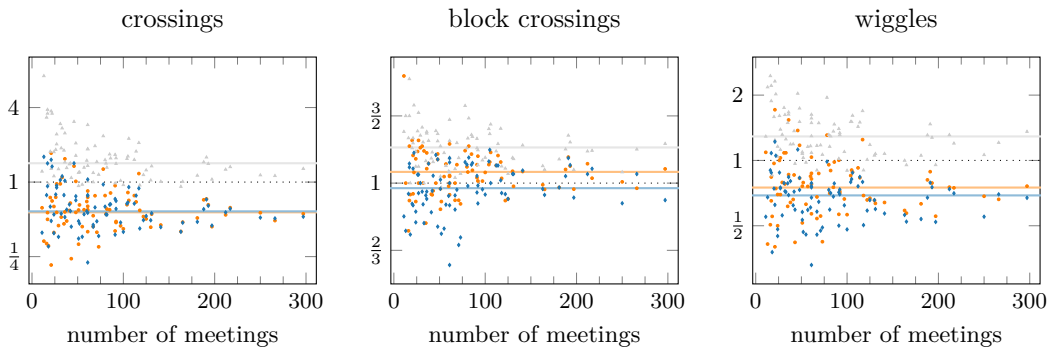
## 26:16 Storylines with a Protagonist

■ **Table 1** Experimental results on a dataset of 81 protagonists. The mean  $\mu$  is calculated for each algorithm and each number of authors  $k$ .  $\beta$  measures the percentage of cases where an algorithm achieved the best result. Sums over 100% are possible due to ties.

	$k$	1-Sider		2-Sider		Median		GreedyBlocks	
		$\mu$	$\beta$	$\mu$	$\beta$	$\mu$	$\beta$	$\mu$	$\beta$
crossings	6	85.1	0	24.3	<b>88</b>	32.8	21	47.0	11
	11	283.0	0	103.3	<b>70</b>	113.3	32	197.3	2
	16	536.9	0	211.8	<b>59</b>	222.6	33	418.0	10
	21	823.6	0	342.6	<b>53</b>	346.2	42	623.5	5
block crossings	6	33.8	0	18.7	<b>68</b>	21.1	32	23.0	26
	11	59.2	0	41.8	<b>59</b>	45.2	22	45.5	30
	16	75.9	0	58.9	<b>63</b>	63.4	22	62.8	27
	21	87.9	0	72.3	<b>54</b>	76.2	19	72.8	38
wiggles	6	112.6	0	42.9	<b>81</b>	53.5	26	67.3	11
	11	286.5	0	136.7	<b>75</b>	152.6	25	208.8	4
	16	469.0	0	239.3	<b>68</b>	266.6	26	388.3	6
	21	649.1	0	344.1	<b>77</b>	386.6	21	535.8	2

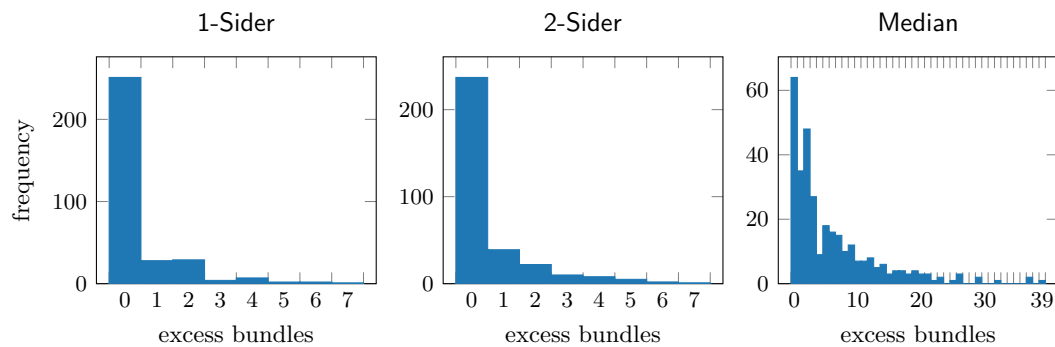
We expected the algorithms that implement our protagonist-focused style to be at a disadvantage compared to algorithms without that restriction. With about 30% more block crossings, 80% more wiggles, and more than twice as many crossings, for the 1-Sider algorithm this was clearly the case. The 2-sider algorithm in contrast was competitive in every metric. Both the 2-sider and Median algorithms match or even outperform the GreedyBlocks heuristic when it comes to block crossings, showing the effectiveness of our bundling algorithm.

As discussed in Section 6, the bundling algorithm is not always optimal but we can determine a lower bound on the optimal number of bundles for any input. We evaluate the quality of our heuristic that splits the storyline whenever it encounters a conflict by comparing its results with the lower bound. The results can be found in Figure 8. For 1-Sider, 77% of our test set is bundled optimally; for 2-Sider, 73%. The results for Median are rather inconclusive. While most instances are close to the optimum, the overall gap between the lower bound and the actual number of block crossings is wider. Apparently, the protagonist benefits bundling.

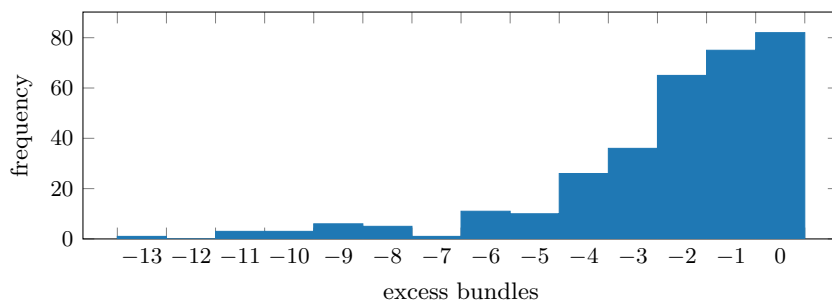


■ **Figure 7** Results of 1-Sider  $\blacktriangle$ , 2-Sider  $\blacklozenge$ , and Median  $\bullet$  relative to GreedyBlocks when applied to the largest instances (with 21 characters) in our benchmark set. Horizontal lines mark the medians.





■ **Figure 8** For most of the 324 storylines in our benchmark set, the bundling heuristic operated at (or very close to) the lower bound.



■ **Figure 9** For most of the 324 storylines, rebundling reduced the number of block crossings.

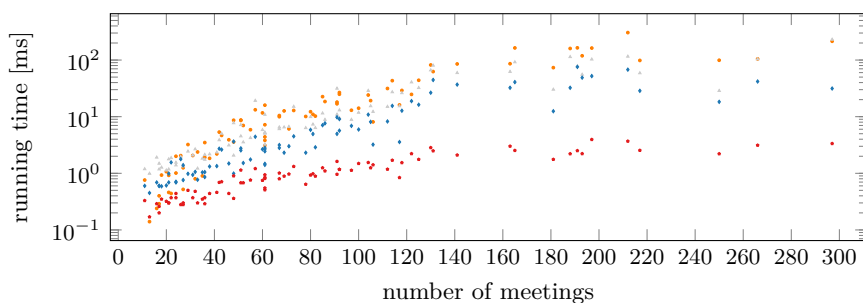
The bundling heuristic can also be applied to the results of `GreedyBlocks` as a post-processing step. So we run `GreedyBlocks`, resolve all block crossings into pairwise ones (we replace any block crossing with a block of  $a$  lines intersecting a block of  $b$  lines by  $a \times b$  pairwise crossings), and then use the bundling heuristic to “rebundle” them again into block crossings. On the same dataset as used before (324 storylines derived from 81 protagonists) none of the instances in our dataset had more block crossings than without rebundling, despite the heuristic nature of our algorithm. See Figure 9 for detailed results. Rebundling decreased the number of block crossings in 75% of cases. As a post-processing step it showed positive results across the board.

**Running times.** For the dataset of large storylines (20 coauthors) we measured running times of 100 repetitions per input and algorithm. See Figure 10 for the results. We can clearly see that bundling takes a toll but overall, for our interactive use case, the running times are always tolerable and most of the time imperceptible.

## 9 Conclusion

Storylines with a protagonist arise naturally when visualizing how the peer group of a scientific author changes over time. Minimizing the number of (block) crossings helps to make such visualizations more readable. We have presented an efficient algorithm for minimizing *pairwise* crossings in a restricted case (1-SCM-P), and we have shown that it is NP-hard to minimize the number of *block* crossings (1-SBCM-P) even in the simpler protagonist setting. Our experimental evaluation has shown that our heuristic for bundling pairwise crossings

## 26:18 Storylines with a Protagonist

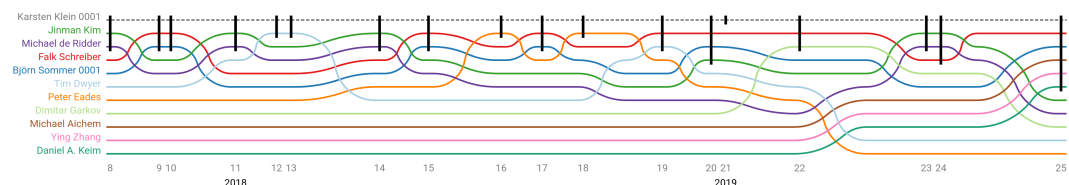


■ **Figure 10** Running times of 1-Sider  $\blacktriangle$ , 2-Sider  $\blacklozenge$ , Median  $\bullet$ , and GreedyBlocks  $\blacklozenge$  for the largest instances (with 21 characters) in our benchmark set. Each point is the average of 100 repetitions.

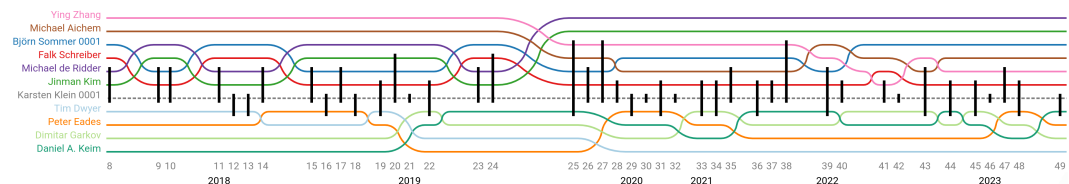
into block crossings performed close to optimal on our benchmark set. Our heuristics for 2-SBCM(-P) are fast enough for interactive applications. The fact that 2-Sider outperformed GreedyBlocks and Median underlines that more cleverness is needed to exploit the additional freedom that 2-SBCM offers compared to 2-SBCM-P. On the other hand, having a designated protagonist can be beneficial in use cases other than the visualization of publication histories; Kuo et al. [20] used a somewhat less strict notion of a protagonist to visualize (i) interactions among actors in social media and (ii) disease propagation centered around a primary outbreak.

Still, some questions remain open. Is 2-SCM-P NP-hard? Can bundling in the presence of meetings (see Section 6.2) be solved efficiently? Can we efficiently minimize the weighted number of wiggles [11] in the one-sided setting?

Figures 11–14 show some storyline visualizations produced by our algorithms. The storylines depicted are not necessarily part of our benchmark set. They were specifically chosen (and some of them cropped) in order to highlight noteworthy properties of the algorithms. Clicking on the star in a caption opens an interactive storyline visualization of the same setting using *current* publication data on dblp.

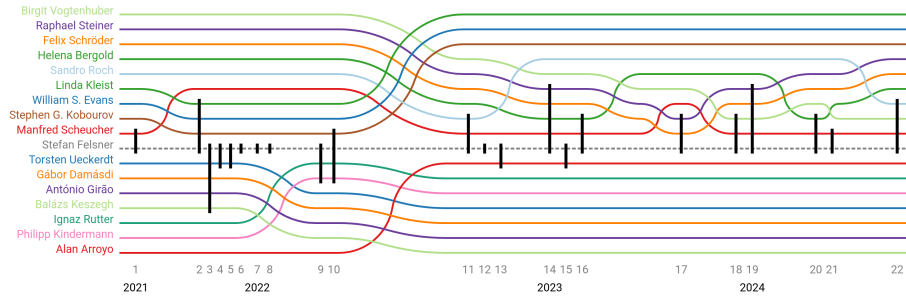


(a) Drawn by 1-Sider with bundling. ★

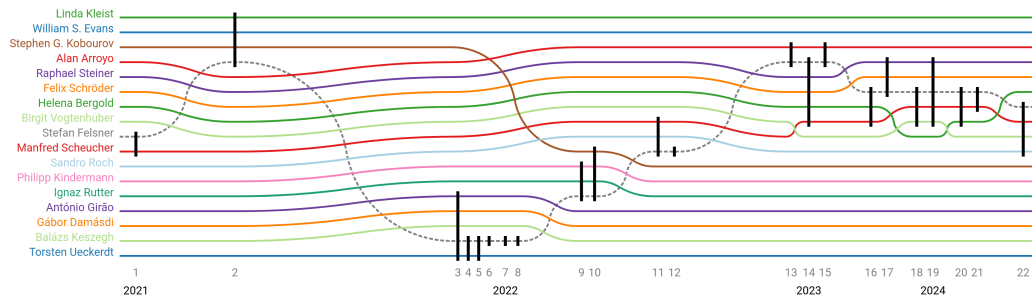


(b) Drawn by 2-Sider with bundling. ★

■ **Figure 11** Clippings of two storylines visualizing Karsten Klein and his 10 most frequent coauthors in the past 7 years. Note that 2-Sider yields a much more compact drawing (more meetings in the same scene space).

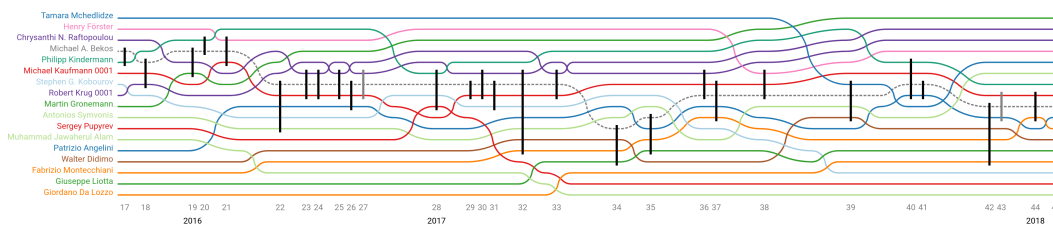


(a) Drawn by 2-Sider with bundling (55 pairwise crossings, 11 block crossings, and 52 wiggles). ★



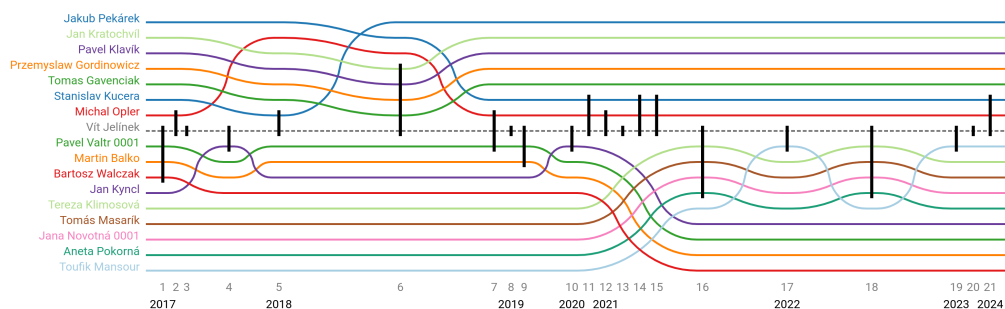
(b) Drawn by GreedyBlocks (44 pairwise crossings, 11 block crossings, and 55 wiggles). ★

■ **Figure 12** Two complete storylines visualizing Stefan Felsner and his 16 most frequent coauthors in the past 3 years. Note that GreedyBlocks produces larger block crossings than 2-Sider.

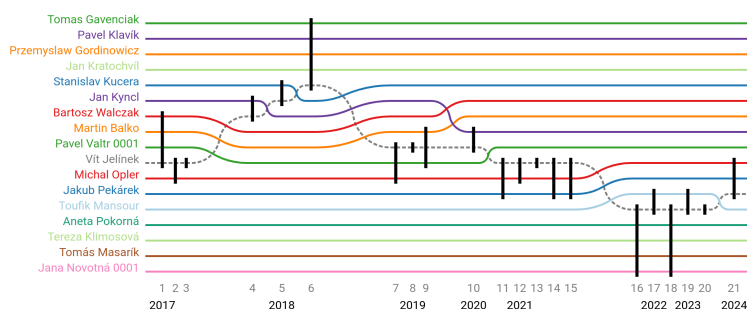


■ **Figure 13** A clipping of a storyline visualizing Michael Bekos and his 16 most frequent coauthors in the past 10 years. This realization uses the Median algorithm with bundling. Note that, at meetings 32 and 33, our bundling heuristic missed an obvious opportunity to merge two  $2 \times 1$  block crossings into a  $2 \times 2$  block crossing. ★

## 26:20 Storylines with a Protagonist



(a) Drawn by 2-Sider with bundling (58 pairwise crossings, 10 block crossings, and 53 wiggles). ★



(b) Drawn by GreedyBlocks (16 pairwise crossings, 8 block crossings, and 24 wiggles). ★

■ **Figure 14** Two complete storylines visualizing Vít Jelínek and his 16 most frequent coauthors in the past 7 years. Note that when more than two disjoint groups collaborate, allowing the protagonist's curve to cross other curves can reduce the number of (block) crossings. This happened, however, not very often in the data set we analyzed.

## References

- 1 Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev.  $O(\sqrt{\log n})$  approximation algorithms for Min UnCut, Min 2CNF deletion, and directed cut problems. In *ACM Symp. Theory Comput. (STOC)*, pages 573–581, 2005. doi:10.1145/1060590.1060675.
- 2 Laurent Bulteau, Guillaume Fertin, and Irena Rusu. Sorting by transpositions is difficult. *SIAM J. Discrete Math.*, 26(3):1148–1180, 2012. doi:10.1137/110851390.
- 3 Steven Chaplick, Myroslav Kryven, Giuseppe Liotta, Andre Löffler, and Alexander Wolff. Beyond outerplanarity. In Fabrizio Frati and Kwan-Liu Ma, editors, *Graph Drawing & Network Vis. (GD)*, volume 10692 of *LNCS*, pages 546–559. Springer, 2018. doi:10.1007/978-3-319-73915-1\_42.
- 4 Steven Chaplick, Thomas C. van Dijk, Myroslav Kryven, Ji-won Park, Alexander Ravsky, and Alexander Wolff. Bundled crossings revisited. In Daniel Archambault and Csaba D. Tóth, editors, *Graph Drawing & Network Vis. (GD)*, volume 11904 of *LNCS*, pages 63–77. Springer, 2019. doi:10.1007/978-3-030-35802-0\_5.
- 5 Steven Chaplick, Thomas C. van Dijk, Myroslav Kryven, Ji-won Park, Alexander Ravsky, and Alexander Wolff. Bundled crossings revisited. *J. Graph Alg. Appl.*, 24(4):621–655, 2020. doi:10.7155/JGAA.00535.
- 6 Markus Chimani, Christine Dahn, Martina Juhnke-Kubitzke, Nils Kriege, Petra Mutzel, and Alexander Nover. Maximum cut parameterized by crossing number. *J. Graph Alg. Appl.*, 24(3):155–170, 2020. doi:10.7155/jgaa.00523.
- 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, *Graph Drawing & Network Vis. (GD)*, volume 12590 of *LNCS*, pages 324–332. Springer, 2020. doi:10.1007/978-3-030-68766-3\_25.

- 8 Martin Fink, John Hershberger, Subhash Suri, and Kevin Verbeek. Bundled crossings in embedded graphs. In Evangelos Kranakis, Gonzalo Navarro, and Edgar Chávez, editors, *Latin Amer. Symp. Theoret. Inform. (LATIN)*, volume 9644 of *LNCS*, pages 454–468. Springer, 2016. doi:10.1007/978-3-662-49529-2\_34.
- 9 Martin Fink and Sergey Pupyrev. Metro-line crossing minimization: Hardness, approximations, and tractable cases. In Stephen K. Wismath and Alexander Wolff, editors, *Graph Drawing (GD)*, volume 8242 of *LNCS*, pages 328–339. Springer, 2013. doi:10.1007/978-3-319-03841-4\_29.
- 10 Martin Fink, Sergey Pupyrev, and Alexander Wolff. Ordering metro lines by block crossings. *J. Graph Alg. Appl.*, 19(1):111–153, 2015. doi:10.7155/JGAA.00351.
- 11 Theresa Fröschl and Martin Nöllenburg. Minimizing wiggles in storyline visualizations. In Fabrizio Frati and Kwan-Liu Ma, editors, *Graph Drawing & Network Vis. (GD)*, volume 10692 of *LNCS*, pages 585–587. Springer, 2018. URL: <https://www.ac.tuwien.ac.at/files/pub/fn-mwsv-18.pdf>.
- 12 Martin Gronemann, Michael Jünger, Frauke Liers, and Francesco Mambelli. Crossing minimization in storyline visualization. In Yifan Hu and Martin Nöllenburg, editors, *Graph Drawing & Network Vis. (GD)*, volume 8901 of *LNCS*, pages 367–381. Springer, 2016. doi:10.1007/978-3-319-50106-2\_29.
- 13 Herman Haverkort. Metaphorical metro maps: Design challenges. Schematic Mapping Workshop, 2022. URL: <https://www.ruhr-uni-bochum.de/schematicmapping/papers/smw-haverkort.pdf>.
- 14 Tim Hegemann and Alexander Wolff. Publines. Software, swbId: swb:1:dir:8b6a0dd881ed0b7a4b0e5d8dbfa38f82827aa641 (visited on 2024-10-14). URL: <https://github.com/hegetim/publines>.
- 15 Tim Herrmann. Storyline-Visualisierungen für wissenschaftliche Kollaborationsgraphen. Master’s thesis, University of Würzburg, 2022. URL: <https://www1.pub.informatik.uni-wuerzburg.de/pub/theses/2022-herrmann-masterarbeit.pdf>.
- 16 Sera Kahruman, Elif Kolotoglu, Sergiy Butenko, and Illya V. Hicks. On greedy construction heuristics for the max-cut problem. *Int. J. Comput. Sci. Eng.*, 3(3):211–218, 2007. doi:10.1504/IJCSE.2007.017827.
- 17 Nam Wook Kim, Stuart K. Card, and Jeffrey Heer. Tracing genealogical data with timenets. In Giuseppe Santucci, editor, *Advanced Visual Interfaces (AVI)*, pages 241–248. ACM Press, 2010. doi:10.1145/1842993.1843035.
- 18 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, *Graph Drawing & Network Vis. (GD)*, volume 9411 of *LNCS*, pages 192–198. Springer, 2015. doi:10.1007/978-3-319-27261-0\_16.
- 19 Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Res. Logist. Quart.*, 2(1–2):83–97, 1955. doi:10.1002/nav.3800020109.
- 20 Yun-Hsin Kuo, Dongyu Liu, and Kwan-Liu Ma. SpreadLine: Visualizing egocentric dynamic influence. *IEEE Trans. Visual. Comput. Graphics*, 2024. To appear. arXiv:2408.08992.
- 21 Dénes König. Gráfok és mátrixok. *Matematikai és Fizikai Lapok*, 38:116–119, 1931.
- 22 Frauke Liers and Gregor Pardella. Partitioning planar graphs: A fast combinatorial approach for max-cut. *Comput. Optim. Appl.*, 51(1):323–344, 2012. doi:10.1007/S10589-010-9335-5.
- 23 Shixia Liu, Yingcai Wu, Enxun Wei, Mengchen Liu, and Yang Liu. StoryFlow: Tracking the evolution of stories. *IEEE Trans. Visual. Comput. Graphics*, 19(12):2436–2445, 2013. doi:10.1109/TVCG.2013.196.
- 24 Charles Joseph Minard. The Russian campaign 1812–1813. Diagram available at <https://commons.wikimedia.org/wiki/File:Minard.png>, 1869. Accessed 2017/04/03.
- 25 Chris W. Muehler, Tarik Crnovrsanin, Arnaud Sallaberry, and Kwan-Liu Ma. Egocentric storylines for visual analysis of large dynamic graphs. In *Proc. IEEE Int. Conf. Big Data*, pages 56–62, 2013. doi:10.1109/BigData.2013.6691715.

## 26:22 Storylines with a Protagonist

- 26 Randall Munroe. Movie narrative charts. Diagram available at <https://xkcd.com/657/>, 2009. Accessed 2017/04/03.
- 27 Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991. doi:10.1016/0022-0000(91)90023-X.
- 28 Valeriu Soltan and Alexei Gorpinevich. Minimum dissection of a rectilinear polygon with arbitrary holes into rectangles. *Discrete Comput. Geom.*, 9:57–79, 1993. doi:10.1007/BF02189307.
- 29 Yuzuru Tanahashi and Kwan-Liu Ma. Design considerations for optimizing storyline visualizations. *IEEE Trans. Visual. Comput. Graphics*, 18(12):2679–2688, 2012. doi:10.1109/TVCG.2012.212.
- 30 Thomas C. van Dijk, Martin Fink, Norbert Fischer, Fabian Lipp, Peter Markfelder, Alexander Ravsky, Subhash Suri, and Alexander Wolff. Block crossings in storyline visualizations. In Yifan Hu and Martin Nöllenburg, editors, *Graph Drawing & Network Vis. (GD)*, volume 9801 of *LNCS*, pages 382–398. Springer, 2016. doi:10.1007/978-3-319-50106-2\_30.
- 31 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 Alg. Appl.*, 21(5):873–913, 2017. doi:10.7155/jgaa.00443.
- 32 Thomas C. van Dijk, Fabian Lipp, Peter Markfelder, and Alexander Wolff. Computing storylines with few block crossings. In Fabrizio Frati and Kwan-Liu Ma, editors, *Graph Drawing & Network Vis. (GD)*, volume 10692 of *LNCS*, pages 365–378. Springer, 2018. doi:10.1007/978-3-319-73915-1\_29.