StoryFlow: Tracking the Evolution of Stories

Shixia Liu, *Senior Member, IEEE*, Yingcai Wu, *Member, IEEE*, Enxun Wei, Mengchen Liu, and Yang Liu

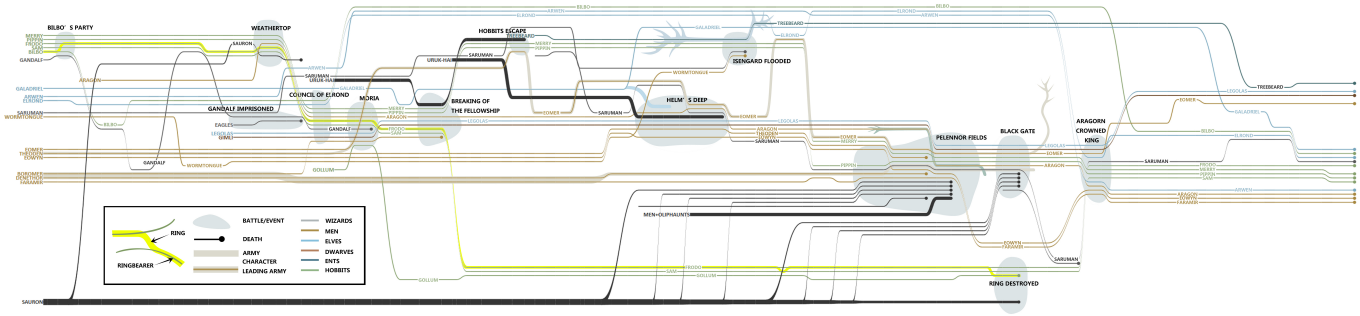

Fig. 1. A storyline visualization of the movie *The Lord of the Rings*. The yellow line indicates the path of the ring. The black layers are the armies. For example, the one at the bottom is *Sauron*'s army bundled by the level of detail technique.

Abstract— Storyline visualizations, which are useful in many applications, aim to illustrate the dynamic relationships between entities in a story. However, the growing complexity and scalability of stories pose great challenges for existing approaches. In this paper, we propose an efficient optimization approach to generating an aesthetically appealing storyline visualization, which effectively handles the hierarchical relationships between entities over time. The approach formulates the storyline layout as a novel hybrid optimization approach that combines discrete and continuous optimization. The discrete method generates an initial layout through the ordering and alignment of entities, and the continuous method optimizes the initial layout to produce the optimal one. The efficient approach makes real-time interactions (e.g., bundling and straightening) possible, thus enabling users to better understand and track how the story evolves. Experiments and case studies are conducted to demonstrate the effectiveness and usefulness of the optimization approach.

Index Terms—Storylines, story-telling visualization, user interactions, level-of-detail, optimization.

1 Introduction

A story refers to a recounting of a sequence of temporally and causally related events [3]. Each story has a beginning, a middle, and an end. It also involves one or more characters (or entities) who determine the way the story will develop, as well as several settings/locations where the story takes place and the entities interact with each other. We refer to the interactions among the entities in different settings/locations as entity relationships. Understanding how entity relationships evolve from the beginning to the end in a story is very important in many applications such as information exploration and understanding, interpersonal communication and storytelling, and media analysis [9, 16]. For example, given a set of tweets related to the story of "the US presidential election," a social scientist might be very interested in the key opinion leaders (entities) in each of the related topics and how their impact on these topics (relationships) changes over time.

To help users better understand and analyze a complex story, storyline visualizations have been developed to convey the temporal patterns of entity relationships. Such a visualization was first introduced by Munroe as a movie narrative chart [27]. It aims to illustrate the entity relationships in the narratives of various movie epics. As shown in Fig. 11(c), the characters are encoded by lines running from left to right. Consequently, their relationships in different locations are mapped to the distance between lines at each time frame. Lines are drawn adjacently when the corresponding characters are in the same location or setting. The movie charts were drawn by hand. Subsequent research has focused on automatically generating a storyline layout [29, 42].

However, existing visualization techniques either produce an aesthetically pleasing and legible storyline picture with much time [42] or an interactive visualization with too many wiggly lines and too much visual inconsistency [29]. Typically, the aesthetics and legibility are measured by three metrics: line crossings, line wiggles, and space efficiency [40]. Although these techniques have achieved a certain amount of success in conveying the temporal dynamics of entity relationships inside a story, they may not fully support real-world storytelling and analysis tasks. First, the performance of the state-of-the-art storyline layout fails to meet the requirements of real-time interactions. The method proposed by Tanahashi et al. [42] may take hours to generate a storyline with hundreds of entities and hundreds of time frames. Second, the available approaches aim solely to generate a storyline layout with flat relationships among entities, while in many real-world applications, settings/locations are naturally organized in a hierarchy [6]. Third, the existing visualizations are not designed to accommodate more than hundreds of entity lines. They cannot provide legible results when the number of entities is in the thousands or even hundreds.

We have designed a storyline visualization system, *StoryFlow*, to generate an aesthetically pleasing and legible storyline visualization. Compared with the state-of-the-art approach [42], it supports real-time user interactions, hierarchical relationships among entities, and the rendering of a large number of entity lines. The approach formulates the difficult problem of creating an effective storyline layout as an optimization problem, which is solved efficiently using a novel hybrid optimization strategy with a judicious combination of discrete and continuous optimization methods. The discrete method creates an initial layout through ordering and aligning line entities, while the continuous

- *S. Liu, Y. Wu, and Y. Liu are with Microsoft Research Asia. Y. Wu is the corresponding author. E-mail: {shliu, ycwu, yangliu}@microsoft.com.*
- *E. Wei is with Shanghai Jiao Tong University. E-mail: weienxun@gmail.com.*
- *M. Liu is with Tsinghua University. E-mail: v-meli@microsoft.com. E. Wei and M. Liu performed this work while at Microsoft Research Asia.*

Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 4 October 2013.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

method optimizes the layout based on quadratic convex optimization. The efficient algorithm enables a rich set of real-time user interactions, including adding, removing, dragging, straightening, and bundling line entities. In addition to being efficient, this approach provides a flexible mechanism for adding grouping constraints (e.g., same location or topic) at different time frames. Hierarchical relationships based on the grouping constraints are also considered in our layout method to visually convey the hierarchical interactions among the entities. Furthermore, we leverage the hierarchical structure among entities and level-of-detail (LOD) rendering technique to handle thousands of entities in the storyline layout.

Specifically, our work makes the following contributions.

- **An efficient hybrid optimization approach** for creating storyline layouts with the capability of progressive updates as well as a flexible mechanism for adding grouping constraints.
- **A hierarchy-aware storyline layout** to visually illustrate the hierarchical relationships among entities.
- **A method for interactively and progressively rendering** a large number of storylines with adaptive levels of detail.

2 Related Work

2.1 Temporal Event Visualization

Considerable research has been directed towards developing a time-based visualization to illustrate event sequences over time. Researchers have proposed many timeline visualizations such as *LifeLines* [31, 32] and its extension [44], *PlanningLines* [4], *TimeLine* [7], TIARA [23,24], RankExplorer [37], EventRiver [25], and *SIMILE* [2] to facilitate understanding event sequences. Fails et al. [14] presented *PatternFinder* to help users visually formulate powerful queries to find temporal event patterns in multivariate datasets. *LifeFlow* [46] and *Outflow* [45] merged multiple event sequences into a tree and a graph, respectively, to provide an abstraction of the data and enable more scalable timeline visualizations. Compared with *LifeFlow*, the graph-based *Outflow* enables more effective comparison of alternative paths of events. However, relationships between entities are not explicitly visualized in these approaches. In *LifeLines* [31], for example, a user can only link to related entities mentally when they are highlighted [5, 19]. More recently, Xu et al. [47] studied the problem of temporal topic competition over time.

Since the relationships between events are important for many analysis tasks, various techniques have been presented to explicitly visualize event relationships. *tmViewer* [22], one of the earliest visualizations to emphasize the importance of relationships between entities, explicitly connects related entities along the timeline using simple lines with arrows (directed edges). Burch et al. [8] used a horizontally-oriented tree layout to represent the hierarchical relationships of transaction sequences along the timeline. *SemaTime* [19] aims to display different types of entity relationships inside or across stacked timelines using distinct types of directed edges. André et al. [5] presented *Continuum* to facilitate visual exploration of relationships between events across periods. *Continuum* uses taxonomic hierarchies, which can be dynamically reordered, to navigate the event space. Recently, Stab et al. [38] developed *SemaTime* to visualize time-dependent relationships of entities using rectangles. Although using lines or rectangles can do a good job of illustrating the relationships between a small number of events, these methods often fail to show large numbers of relationships due to visual clutter caused by line crossings.

2.2 Storytelling

Recently, there has been increased interest in leveraging interactive visualizations to convey stories by journalists, politicians, social activists, and scientific researchers [15, 18, 26, 35, 36]. Here, we briefly review the literature on time-based storytelling techniques.

Inspired by Munroe's movie narrative charts [27], a new visual metaphor called storyline has emerged recently to visualize the relationships between entities over time. Compared to other timeline visualizations, a storyline visualization represents an entity as a line and encodes the relationships between the entities according to the distances between the associated lines over time. A variety of specialized techniques based on the storyline visualization have been proposed for different applications, such as tracing generational relationships in genealogical data [20], tracking community evolution in dynamic networks [34], and illustrating topic merging/splitting relationships and their evolution in a text stream [11,21]. Substantial successes have been achieved by the techniques in solving real-world problems. However, these techniques are only applicable for certain applications, as the visual representations used are significantly simplified in the storyline by imposing more application-specific constraints such as the specific ordering of the lines.

Recent research on storyline visualization has focused on developing a generic visualization tool. PlotWeaver [30], an online editing tool, allows users to interactively create a general storyline visualization, but interactive editing without sufficient computational support is often time-consuming. To solve this problem, Ogawa and Ma [29] derived a set of heuristic rules for a good storyline layout and developed an automatic algorithm based on the rules to create a storyline visualization for tracking software evolution. However, the heuristic rules are relatively simple and the algorithm cannot produce results that are as good as those created by professional artists. Tanahashi and Ma [42] suggested a more complete set of design principles and formulated the storyline layout problem as an optimization approach based on a genetic algorithm. The approach can successfully produce an aesthetically-appealing and legible storyline layout that is comparable to those generated by professional artists. However, it takes considerable time to create a storyline layout when the number of characters and time frames are in the hundreds. As a result, no user interaction is supported. Compared with Tanahashi's and Ma's work, our optimization-based method can produce a desired storyline visualization significantly faster. For a story with hundreds of entities and hundreds of time frames, our method can generate the layout in less than one second. With the efficient hybrid optimization approach, the system provides a set of rich interactions such as aggregating lines and straightening a line, thus allowing for real-time exploration of the dynamic relationships within a larger and complex dataset. In addition, to the best of our knowledge, our approach is the first to automatically generate a hierarchy-aware storyline layout, which considers the hierarchical relationships among entities in the layout.

3 StoryFlow Overview

To create a storyline layout with the capability of progressive updates as well as a flexible mechanism for adding grouping constraints, we have developed StoryFlow. Fig. 2 illustrates the system overview. It contains a layout pipeline and provides a set of rich interactions. For simplicity, we take the movie as an example to illustrate the basic idea of our method. Our method can be easily applied to other datasets such as a Twitter dataset as illustrated in Sec. 7.2.

The input is an XML file containing a session table and a location tree. The session table stores the dynamic relationships between entities. We call the interaction of multiple entities during two adjacent time frames a *session*, which can be regarded as an *event* in a story. Each element $a_{i,j}$ in the session table (also called a line segment or segment) has a session ID that the entity i belongs to at time j. The location tree describes the hierarchical structure of locations. Each tree node represents a location and includes all the session IDs that occur at the location. Thus, the location tree together with the table defines a set of dynamic hierarchies (relationship trees) characterizing the hierarchical relationships between entities at different time frames.

The layout pipeline consists of four steps: relationship tree generation, session/line ordering, session/line alignment, and layout compaction (Fig. 2(a)). In the first step, StoryFlow creates a set of dynamic relationship trees for different time frames. Then the relationship trees are used to order sessions and entity lines. Next, sessions/lines between successive time frames are aligned to maximize the number of straight lines in the layout. Finally, a quadratic optimization algorithm is performed to obtain a compact storyline layout.

The efficient layout algorithm enables a rich set of user interactions for real-time data exploration and analysis. The user interactions can be classified into four categories (Fig. 2(b)): (1) adding/deleting entities, sessions, or locations, which modifies the relationship trees; (2) dragging the entities and sessions to a new position, which is a complement

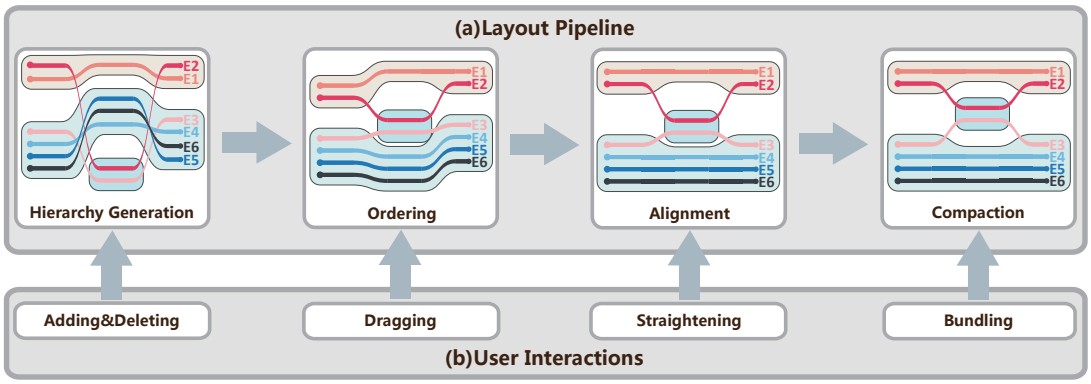

Fig. 2. System overview. StoryFlow contains a layout pipeline and provides a set of interactions. The layout pipeline starts by generating a set of relationship trees, followed by session/line ordering, session/line alignment, and layout compaction. Different user interactions including adding/deleting, dragging, straightening, and bundling are supported to facilitate data exploration.

to the ordering step; (3) straightening a line representing an entity, which runs the alignment step; (4) bundling a group of sessions or releasing them later, which is achieved by running the compaction step.

4 Considerations For StoryFlow Visualization

Given a set of entities and their relationships in different locations over time, StoryFlow aims to generate a legible storyline visualization as well as support real-time exploration and analysis. To this end, we formulate the StoryFlow layout into a computationally efficient optimization problem. In this section, we introduce our considerations for the optimization formulation.

4.1 Design Guidelines

The following design criteria are widely used when creating a storyline visualization [20, 29, 42]:

◇ A line going from left to right along the horizontal timeline represents the lifespan of an entity in a story.
◇ Multiple lines bundled together during a time period indicate that the entities are interacting with each other. Line convergence (or divergence) indicates that the interaction starts (or terminates).

Following the above criteria, early efforts [20, 29] have achieved a certain amount of success in creating an effective storyline layout. However, these layout methods do not model the contextual information such as the locations of the story, which are important for people to better understand the story. To emphasize the locations where important events take place, Tanahashi and Ma [42] draw a closed contour surrounding the event in the background. The enclosed areas are filled with different colors to distinguish different locations. This strategy can handle flat contextual information very well. However, in many real-world applications, the contextual information, such as the location hierarchy in a movie or the topic hierarchy in a document collection, is often hierarchically structured [6]. Without considering the contextual information hierarchy, the simple strategy may produce a misleading layout. Fig. 3(b) demonstrates the problem where Florida is incorrectly included in California. Moreover, the hierarchical structure is an intuitive and effective means to organize a huge amount of data. It allows us to employ a level-of-detail rendering mechanism readily to handle the scalability problem. Thus, the hierarchical contextual information needs to be modeled in the storyline layout. Accordingly, we define two more criteria to convey the spatial information properly.

◇ Contours filled by the same color represent the same location along the timeline. A line inside a background area during a time period indicates that the entity occurs in the location at that time.
◇ The inclusion relationships among the filled contours represent the location hierarchy.

4.2 Optimization Metrics and Constraints

In our layout optimization, we use three optimization metrics suggested by Tanahashi and Ma [42] to define the optimization goal.

- **Line crossings**: reduce line crossings that may lead to occlusion and visual clutter.
- **Line wiggles**: boost the straightness and continuousness of the lines and avoid visual discontinuities.
- **Wiggle distance**: in addition to reducing the wiggle number as in [42], the wiggle distance is also minimized to obtain a compact layout.
- **White space**: minimize the empty space that may cause an unbalanced layout and a waste of screen space.

Furthermore, to satisfy the design criteria introduced in Section 4.1, we define the following two hard constraints that a layout must satisfy:

- **Line adjacency**: if entities are interacting, their lines must be placed adjacently. Otherwise, they must be separate.
- **Hierarchical location**: if the locations are hierarchically organized, the storyline layout must convey this with the inclusion relationships among the related background contours.

4.3 Optimization Strategy

The major goal of our optimization is to minimize the following four metrics: line crossing number, wiggle number, wiggle distance, and white space. If we design a approach that simultaneously optimizes all the metrics given the constraints, it typically requires exhaustive exploration of a large search space. A natural solution is to use a genetic algorithm as in [42], to find a nearly optimal layout. However, a genetic algorithm has the drawbacks of poor local search and premature convergence [43].

To overcome these drawbacks and reduce the prohibitively large search space, we split the optimization into separate subproblems. According to the types of optimization metrics, we divide it into two parts: discrete optimization to minimize the number of line crossings and wiggles (a good initial layout), and continuous optimization to minimize the wiggle distance and white space (fine tune). Our goal is to optimize the important metrics first. Then the subsequent optimization of the less important metrics will not affect the important metrics. A previous study has shown that reducing the line crossing number is the most important metric, while minimizing the wiggle number and maximizing symmetry are less important [33]. The experts (one film professor and two sociologists in media and communication studies) who worked with us on storyline visualization also confirmed this fact. Based on this result, we further divide the optimization into three steps. The first two steps aim to minimize the numbers of line crossings and wiggles, while the last step focuses on optimizing the wiggle distance and white space.

◇ **Step 1: reduce line crossings.** The ordering of visual elements directly impacts the line crossing number. In addition, the ordering also needs to satisfy the hierarchy constraints. Thus this step aims to determine an optimal ordering of line segments, sessions, and locations, to meet the constraints and minimize the line crossing number.
◇ **Step 2: reduce the wiggle number.** A better alignment of sessions and entity lines over time helps to reduce the wiggle number. In this step, we aim to align as many sessions and lines as possible between adjacent time frames.
◇ **Step 3: reduce the wiggle distance and white space.** To generate a compact and aesthetically pleasing storyline visualization with

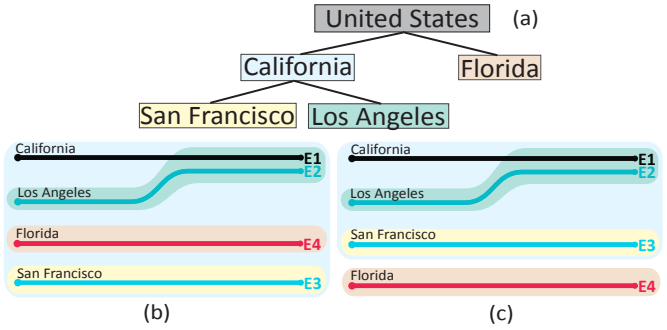

Fig. 3. An example storyline layout with location information: (a) The location tree; (b) A layout created by Tanahashi's and Ma's algorithm [42] with an incorrect location hierarchy (Florida is included in California); (c) A layout created by our method that ensures that the layout visualizes the location hierarchy correctly.

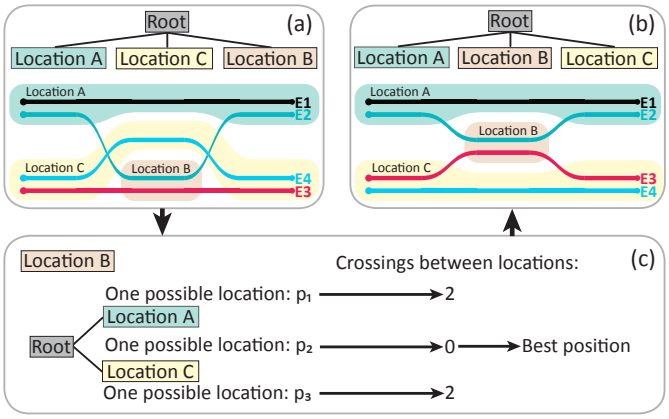

Fig. 4. An example of ordering with hierarchy constraints: (a) without ordering; (b) with ordering; (c) the greedy method to order locations.

informative content, we further optimize two continuous metrics, the wiggle distance and white space.

5 StoryFlow Layout

Fig. 2(a) illustrates the layout pipeline of StoryFlow. It starts by creating a dynamic relationship tree for hierarchically organizing the visual elements (e.g., lines), followed by three optimization algorithms for creating an effective and aesthetically appealing layout.

5.1 Construction of the Relationship Tree

At each time, the hierarchical relationships among entities and sessions should be consistent with those in the location tree. Fig. 3(a) shows a tree depicting the hierarchical structure of multiple locations. A layout result that unambiguously reveals the hierarchical relationship of the locations is shown in Fig. 3(c). To achieve this, we generate a sequence of relationship trees and use each of them to constrain the layout at each time frame. The relationship tree at each time is generated by leveraging the input location tree, as well as the locations, sessions, and entities occurring at that time.

5.2 Ordering

To minimize the line crossing number while satisfying the hierarchy constraints, we sort the entity lines vertically at each time frame. In particular, we assume: at each time frame, if node **A** is on the left of **B** at the same level of the relationship tree, then **A** will be placed above **B** in the layout. With this assumption, the line ordering is formulated to estimate the best ordering of nodes at each level of the relationship tree to reduce line crossings. However, it is computationally complex and time-consuming to produce such an ordering since it is an NP hard problem [13]. To tackle this issue, we divide the ordering problems

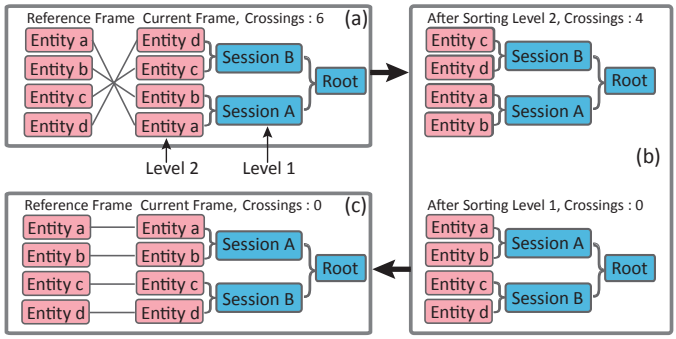

Fig. 5. Illustration of the ordering method: (a) an initial order with six crossings; (b) sorting the nodes level-by-level; (c) the final result without any crossings.

into two parts: (1) sorting the location nodes; (2) ordering the session and entity nodes within each location is computed.

We use a greedy algorithm to order the location nodes from bottom to top, recursively. The basic idea is that for the nodes under the same parent, we first place the node with the largest number of entities. The remaining nodes should be placed in a position that introduces a minimum crossing number with the already placed location nodes. If there is more than one position that introduces the same crossing number, we select the top one. As shown in Fig. 4, there are three locations under the root node. Nodes **A** and **C** are placed first, which provides three candidate positions p_1, p_2, and p_3 for **B**. The algorithm places **B** at p_2, which results in the minimum crossing number among the three locations.

Next, we order the sessions and entities under each location node. The problem is similar to minimizing edge crossings in a multi-level directed acyclic graph (DAG) with a fixed ordering of some of the non-leaf levels that correspond to the locations. Even if the graph has two levels with a fixed ordering at one level, the problem is NP-hard [13]. To solve the problem, we extend a well-established DAG sweeping algorithm [17, 39]. The algorithm starts by generating an initial ordering of the first time frame, which satisfies the hierarchy constraints. It then treats the ordering of the first frame as a reference to calculate the ordering of the second one. This step is repeated until the last frame is reached. Once the order of the last step is fixed, we then treat it as the reference and sweep back. We iteratively improve the ordering result by sweeping back and forth until the line crossing number reaches a stable number or the maximum iteration number is reached. In our implementation, the maximum iteration number is set at 20. In each step of the sweep, we use the barycenter method [39] to sort the tree nodes. The weight of an entity is set as its order at the last time frame. The weight of a session node is then computed by averaging the weights of the entities in it.

We then sort the tree nodes at each level according to their weights. We use a bottom-up sorting scheme to order the nodes in the relationship tree, except for the location nodes that were fixed in the previous step. Compared with existing algorithms [17, 39], our method explicitly satisfies the hierarchy constraints imposed by the relationship trees during the sweeping process. Fig. 5 shows an example of how the sorting works. An initial ordering of two frames with six crossings is shown in Fig. 5(a). First we sort the second level by swapping entities c, d and entities a, b because c is above d and a is above b in the reference frame. After this swapping, the crossing number is reduced to 4. Then we sort the first level which contains two sessions. The weight of session **B** is 3.5 and the weight of session **A** is 1.5. Thus, **B** and **A** are swapped. The result is shown in Fig. 5(c), in which the crossing number is zero.

As the number of locations is usually much smaller than that of sessions and entities, the computation time is mainly determined by sorting the sessions and entities. We can see that the algorithm is a tree based sort with a time complexity of $O(n_t n_e \log n_e)$ where n_e is the number of entities and n_t is the number of time frames.

5.3 Alignment

To minimize the wiggle number, we try to align as many line segments as possible (assign them the same y-coordinate). Mathematically, it can be expressed as:

$$E_{\texttt{align}} = \max \sum_{t=1}^{n_t-1} H(t), \tag{1}$$

where $H(t)$ is the number of straight line segments between t and $t+1$.

The alignment consists of two major steps: session alignment and line alignment within the matched session pair. Once the sessions are aligned, the line alignment in the matched sessions is achieved by simply moving the sessions to align the matched line segments. Thus the alignment result is mainly determined by the session alignment. In the alignment we can add enough white space between sessions so that the alignment of two frames will not influence that of others. Based on this observation, we can rewrite Eq. (1) as

$$E_{\texttt{align}} = \sum_{t=1}^{n_t-1} \max H(t). \tag{2}$$

Accordingly, the optimization problem is simplified to find a set of local optima between adjacent time frames.

Let's take the session as an example to illustrate the basic idea. Suppose we have two adjacent time frames L and R. From top to bottom, L has a sequence of sessions $S_L = \{l_1, \ldots, l_m\}$ and R has $S_R = \{r_1, \ldots, r_n\}$. $sim(l_i, r_j)$ denotes the similarity between two sessions l_i and r_j:

$$sim(l_i, r_j) = straight(l_i, r_j) + \alpha \cdot \left(1 - \left|\frac{i}{m} - \frac{j}{n}\right|\right). \tag{3}$$

The first term $straight(l_i, r_j)$ is the maximum number of straight segments we can get from l_i and r_j. The second term measures how similar their relative positions are in two frames. This term tends to align the sessions with similar relative positions and thus produces a more symmetric layout. Parameter α balances the symmetry of lines and the number of straight segments. In our examples, α is set at 0.1.

For any i and j, $match(i, j)$ denotes the maximum sum of the matched pairs between $l_1, \ldots, l_i$ and $r_1, \ldots, r_j$. Finding a local optimum $H(t)$ is equivalent to computing $match(m, n)$. The problem can be formulated as a weighted Longest-Common-Subsequence (LCS) problem [10]. The basic idea of LCS is to find the longest common subsequence in a set of sequences. In our case, two sequences are considered. Here, a subsequence is a sequence derived from another sequence, in which the elements appear in the same relative order as the original one, but are not necessarily contiguous.

Let $S'_L = \{l_{i_k}, \ldots, l_{i_p}\}$ be a subsequence of S_L and $S'_R = \{r_{j_k}, \ldots, r_{j_p}\}$ be an equal-length subsequence of S_R. The session alignment is formulated as:

$$match(i, j) = \max \sum_{k=1}^{p} sim(l_{i_k}, r_{j_k}) \tag{4}$$

In our case, we use a similarity function to measure the two elements instead of a boolean one. After solving the LCS, we get:

$$match(i, j) = \begin{cases} \max\{match(i-1, j-1) + sim(l_i, r_j), \\ match(i-1, j),\ match(i, j-1)\}, & \text{if } i > 0 \text{ and } j > 0; \\ 0, & \text{if } i = 0 \text{ or } j = 0. \end{cases}$$

This recursive function can be efficiently solved by a dynamic programming technique with time complexity $O(n_e^2)$. Thus, the computational complexity of the alignment algorithm is $O(n_e^2 n_t)$.

Fig. 6 demonstrates one result generated by our algorithm. The layout after ordering (Fig. 6(a)) is the input of our algorithm. We then dynamically construct a table for $match(i, j)$ (store the LCS sequence for each step of the calculation) and trace back the path leading to the optimal alignment (Fig. 6(b)). As shown in Fig. 6(c), our method increases the number of straight segments and achieves a more symmetric result.

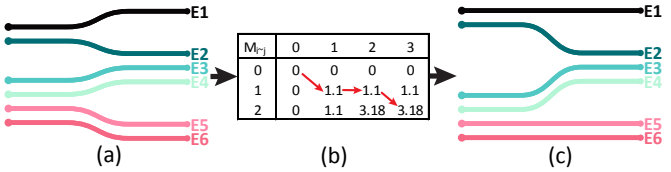

Fig. 6. Line alignment example: (a) input; (b) the dynamic table (the red line represents the best path, $M_{i\sim j} = match(i, j)$); (c) alignment result.

5.4 Compaction

To generate a compact and symmetric storyline layout, the wiggle distance and white space need to be minimized. Mathematically, we formulate it as a linearly constrained quadratic programming problem.

$$\texttt{min} \sum_{i=1}^{n_e} \sum_{j}^{n_t-1} (y_{i,j} - y_{i,j+1})^2 + \beta \sum_{i=1}^{n_e} \sum_{j=1}^{n_t} y_{i,j}^2. \tag{5}$$

Subject to

$$y_{i_1,j} < y_{i_2,j}, \quad \text{if } S_{i_1,j} \prec S_{i_2,j}; \tag{5a}$$
$$y_{i,j} = y_{i,j+1}, \quad \text{if } S_{i,j} \leftrightarrow S_{i,j+1}; \tag{5b}$$
$$y_{i,j} - y_{i+1,j} = d_{\texttt{in}}, \quad \text{if } SID(S_{i,j}) = SID(S_{i+1,j}); \tag{5c}$$
$$\left|y_{i,j} - y_{i+1,j}\right| \geq d_{\texttt{out}}, \quad \text{if } SID(S_{i,j}) \neq SID(S_{i+1,j}). \tag{5d}$$

Here $S_{i,j}$ is the line segment that represents entity i at time j, $y_{i,j}$ denotes the y-coordinate of $S_{i,j}$, and $SID(S_{i,j})$ is the session ID of $S_{i,j}$. The first term aims to penalize the wiggle distance, and the second term penalizes unnecessary white space. β is a parameter that balances these two terms. In our example, $\beta = 1$. The constraints are illustrated below:

(5a) Line order constraint. If the line order of $S_{i_1,j}$ is less than the line order of $S_{i_2,j}$ (we use $S_{i_1,j} \prec S_{i_2,j}$ to denote this order relationship), then segment $S_{i_1,j}$ is placed above $S_{i_2,j}$.

(5b) Line alignment constraint. We use the symbol "$\leftrightarrow$" to indicate that two segments are aligned.

(5c) Adjacency constraint (in the same session). The distance of two adjacent line segments in the same session is defined as $d_{\texttt{in}}$. Typically, $d_{\texttt{in}} = 3 \times \texttt{LineWidth}$. `LineWidth` is the width of the line that represent an entity.

(5d) Adjacency constraint (not in the same session). The minimum distance between two adjacent segments that are not in the same session is $d_{\texttt{out}}$. Here $d_{\texttt{out}} > d_{\texttt{in}}$. In most of our examples, $d_{\texttt{out}}$ is set as $9 \times \texttt{LineWidth}$. Note that the sign of $y_{i,j} - y_{i+1,j}$ is known due to (5a), so the absolute value operator can be removed.

This is a quadratic convex optimization with linear constraints whose global optimum can be calculated in polynomial time [28]. In StoryFlow, we use the Mosek package [1] which implements the state-of-art interior point method to solve this quadratic programming problem.

6 Interactive Exploration

To better aid users in understanding complex story evolution and performing deep analysis, StoryFlow provides a set of user interactions.

Bundling. In many applications, it is quite common to have hundreds and even thousands of entities. Due to limited screen real estate, only

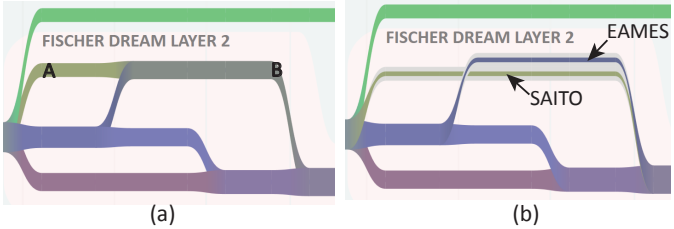

Fig. 7. Bundling operation (The layout is part of the *Inception* layout that describes what happens in the second layer of Fischer's Dream): (a) bundling result; (b) layer **AB** is expanded. It can be clearly seen that Eames and Saito first leave the team and rejoin the team later.

a subset of entities can be displayed. Without an overview, users may find the displayed information inadequate for their tasks. To solve this problem, we leverage the level-of-detail (LOD) rendering technique to bundle the lines in the same session at each time. More precisely, we set the distance between segments (d_{in}) in a session at 0 and run the optimization algorithm again. In the aggregated layout, the layer color of each session is a blend of all the entity colors in it, and the layer width is proportional to the number of entities.

If the user is interested in one session layer and wants to examine the details, s/he can double-click on it. The layer will be gradually expanded and all the segments in it will be displayed. For example, in Fig. 7, after layer **AB** is expanded, it can be clearly seen that Eames and Saito first leave the team and rejoin the team later.

Adding/Deleting. Since different users may have different information needs and preferences, StoryFlow allows users to delete unimportant entities and add entities (with necessary session information) that are useful for their exploration and analysis. After the entities are deleted/added, the relationship trees and storyline layout are changed accordingly. One example is shown in Fig. 8.

Fig. 8. Deleting operation: (a) before deleting; (b) after deleting

Interactive Ordering. In StoryFlow, an ordering method is proposed to automatically order locations, sessions, and lines. However, the optimization method is less than perfect and users may have different needs. We provide a backup schedule to allow users to interactively order the visual elements. One interactive ordering example is shown in Fig. 9.

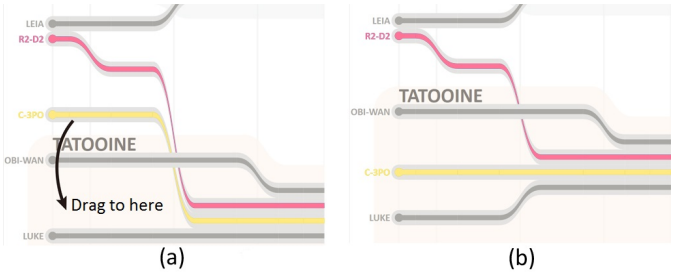

Fig. 9. Interactive ordering: (a) before ordering; (b) after ordering.

Line straightening. In many applications, users are interested in a specific entity. One simple way to help users study this entity is to highlight the related entity line. However, if this line has several wiggles, it is still difficult for users to follow it over time. To solve this problem, StoryFlow provides an interaction that directly straightens a line throughout its life span. Fig. 10 is an example of the straightening interaction. We can see that the yellow line representing "Ma Bangde" has a lot of wiggles in the original layout. After straightening the yellow line, we can easily track his story, understanding how he leaves one group of people and joins another.

7 Evaluation

Based on the proposed layout method and interaction techniques, we have developed StoryFlow. In StoryFlow, we leverage the line geometry adjustment method proposed by Tanahashi and Ma [42] to relax line segments and deemphasize unimportant segments. In this section, we aim to evaluate the efficiency, effectiveness and usefulness of StoryFlow. First, we conducted a comparison experiment to demonstrate the performance of our algorithm. Then we compared the layout results derived from our method with those generated by the state-of-the-art method. Next, to further illustrate the capability and usefulness of the

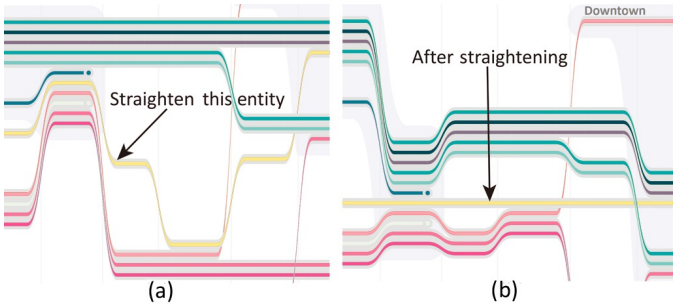

Fig. 10. Straightening: (a) before straightening; (b) after straightening.

StoryFlow system, we applied it to a dataset collected from Twitter. All the experiments were conducted on a desktop PC with an Intel i7-2600 CPU (3.4 GHz) and 8 GB memory. The movie datasets used in our experiments are available at [41].

7.1 Quantitative Analysis

To evaluate the performance of our layout method, we compared it with the state-of-the-art method proposed by Tanahashi and Ma [42]. For brevity, we denote their method as the TM method. We conducted the experiment based on three movie datasets and a network dataset: *Star Wars*, *Inception*, *The Matrix*, and the MID network dataset (from 1817 to 1947). In the experiments, the initial element ordering of the two methods is the same, which is given by the input XML files. We compared our method with the TM method in terms of time, the crossing number, and the wiggle number. The results are shown in Table 1.

In our experiments, we used the open source code provided by Tanahashi [41] to obtain their results. Since different replications of a genetic algorithm may produce different solutions, the results produced by the TM method were the average values of ten trials. The output of our method was stable, so we only ran our program once for each data.

Tanahashi's and Ma's paper [42] provided the statistical data of *Star Wars*, so we used the form of A(B) to represent the related data items in Table 1. Here A is the average value of the ten trials and B is the data from their paper. We failed to generate the storyline layout of the MID data, so we used the results provided by Tanahashi. The experimental results provided by the authors were run on a MacBook Pro with Intel i7-620M CPU (2.66 GHz) and 4 GB memory.

Table 1. Quantitative comparison with the TM method.

	Data		Time (seconds)		#Crossings		#Wiggles	
	#Entity	#Frame	Ours	TM	Ours	TM	Ours	TM
StarWars	14	50	0.16	129.79	48	93(51)	82	133(91)
Inception	8	71	0.16	149.67	23	99	88	162
Matrix	14	42	0.16	172.47	14	43	54	94
MID	79	523	0.60	10^5	1267	1871	831	874

Table 2. Evaluation of StoryFlow on random input orders.

	#Crossings			#Wiggles		
	average	best	worst	average	best	worst
StarWars	53	47	61	83	81	86
Inception	27	23	31	86	84	88
Matrix	14	14	15	54	53	56
MID	1455	1321	1607	876	838	912

We first compared the time required by the two methods to generate the four storyline layouts. As shown in Table 1, our method was over hundreds of times faster than the TM method. The reason is that the TM method is based on a genetic algorithm. One drawback of genetic algorithms is their expensive computational cost. In particular, computing the layout for each genome takes time complexity $O(CI + STI)$. Here, C is the number of lines, I is the number of sessions, S is the number of slots, and T is the number of time frames. Thus the whole process will cost $n_p n_r O(CI + STI)$ where n_p is the number of genomes in each generation and n_r is the number of iterations. On the other hand, the time complexity of StoryFlow is $O(n_e^2 T + T^3)$. The first term comes from the alignment step and the second term comes from the compaction step. Parameter S is set to 580 in the open source

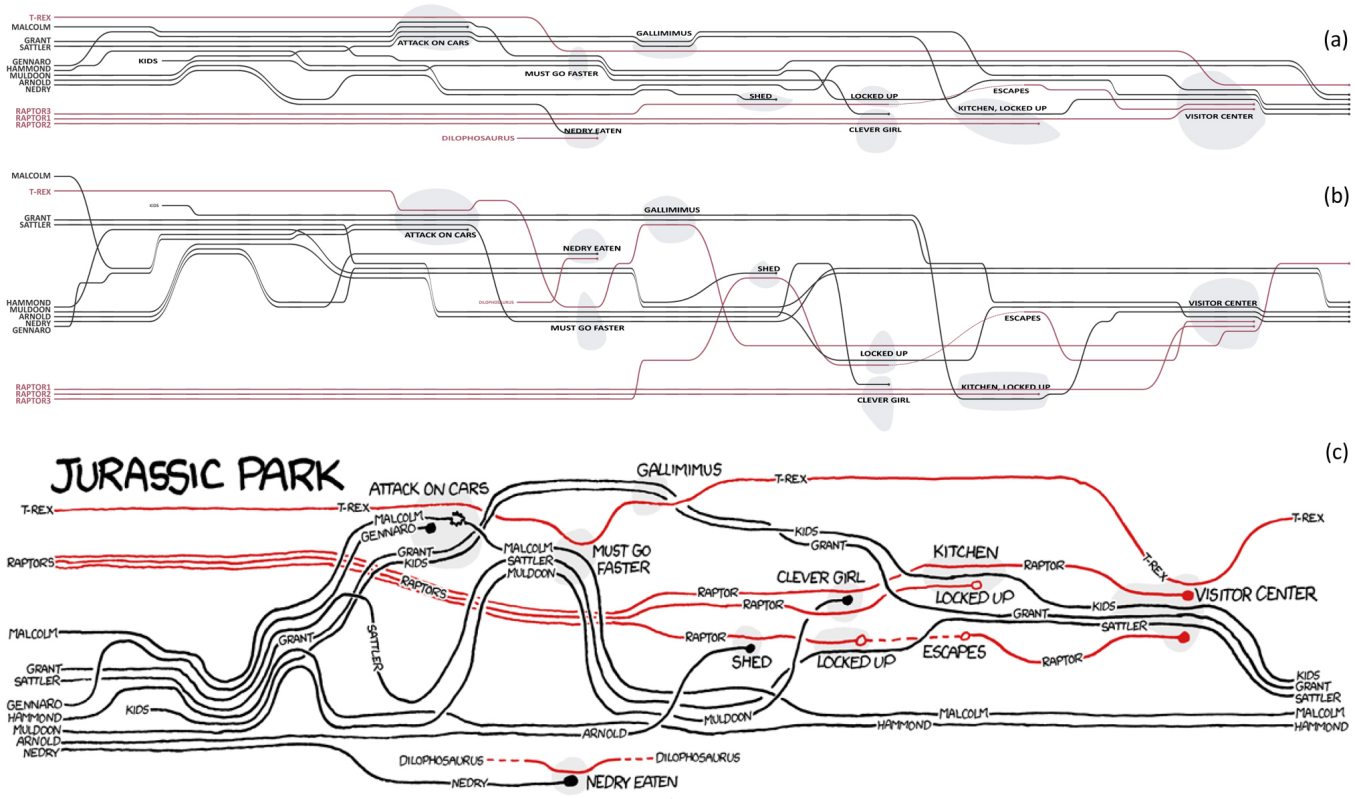

Fig. 11. Comparison of the movie *Jurassic Park*: (a) layout by StoryFlow; (b) layout by the TM method; (c) hand-drawn illustration from XKCD.

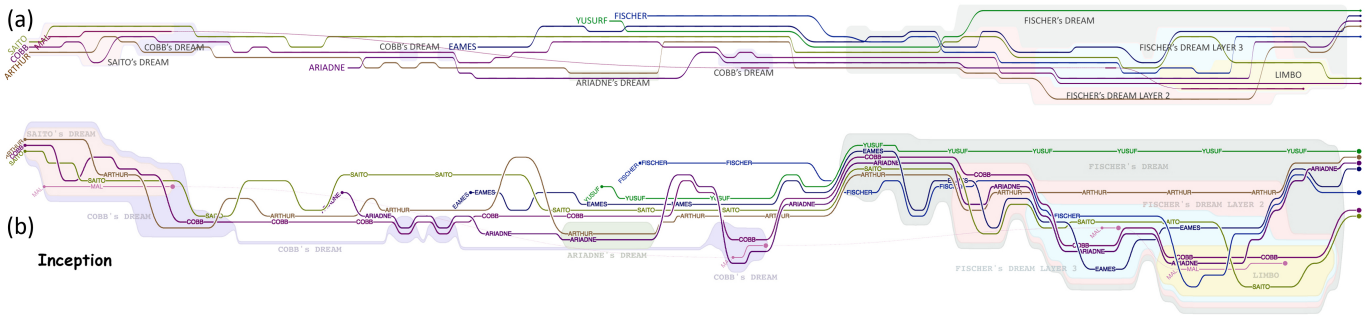

Fig. 12. Comparison of the movie *Inception* on encoding hierarchical locations: (a) layout by StoryFlow, (b) layout by the TM method.

code of the TM method, which is comparable to T in most movies and dramas. So the time cost of the TM method can be roughly estimated by $n_p n_r O(T^3)$. n_p and n_r are set to 200 and 550, respectively. Compared with StoryFlow, n_p and n_r are the major reasons for the expensive computational cost in the TM method.

Next, we compared the numbers of line crossings and wiggles. As shown in Table 1, our method achieves better performance in reducing line crossings and wiggles. This is because the genetic algorithm used by the TM method has drawbacks such as premature convergence and local optima [43]. Our method provides a good initial layout from the ordering and alignment steps, so dynamic-programming-based optimization can find a better local optima.

Initialization. To examine the influence of different input (with different ordering) on the layout results generated by StoryFlow, we conducted the following experiment. In each trial, we randomly ordered the entities at the first time frame and then generated the storyline layout using our optimization approach. Ten trials were conducted for each dataset. We evaluated the layout quality in terms of line crossings and wiggles. Under each metric, we compared the average value, best value, and worst value. The results are shown in Table 2. We can see that the layout quality is quite stable with different inputs, which implies the effectiveness of our optimization approach.

7.2 Case Studies

In this section, we first use a movie example to illustrate the usefulness and capability of StoryFlow. Then we compare our layout results with existing techniques, including the state-of-the-art method proposed by Tanahashi and Ma [42] and the hand-drawn illustration from XKCD [27]. Finally, a case study on Twitter data was conducted to further illustrate the usefulness and extensibility of StoryFlow.

Fig. 1 illustrates the story evolution in the movie *The Lord of the Rings*. The yellow line is the path of the ring, Frodo is the ring bearer who most often carried the ring. He and Sam are together most of the time. In this visualization, to reduce visual clutter, the armies encoded by the black lines are bundled together. With the LOD-based bundling technique, the marching routes of the armies are clearly conveyed. This implies that our layout can handle large data using the LOD technique.

Fig. 11 shows the storyline visualizations of the movie *Jurassic Park*, which were generated by our method, the TM method, and the hand-drawn illustration from XKCD [27], respectively. Compared with the TM method (b), our layout (a) is more symmetric and compact with fewer line crossings, wiggles, and white space. Furthermore, its overall shape is much closer to the hand-drawn illustration (c). The total number of line crossings and wiggles in the original illustration

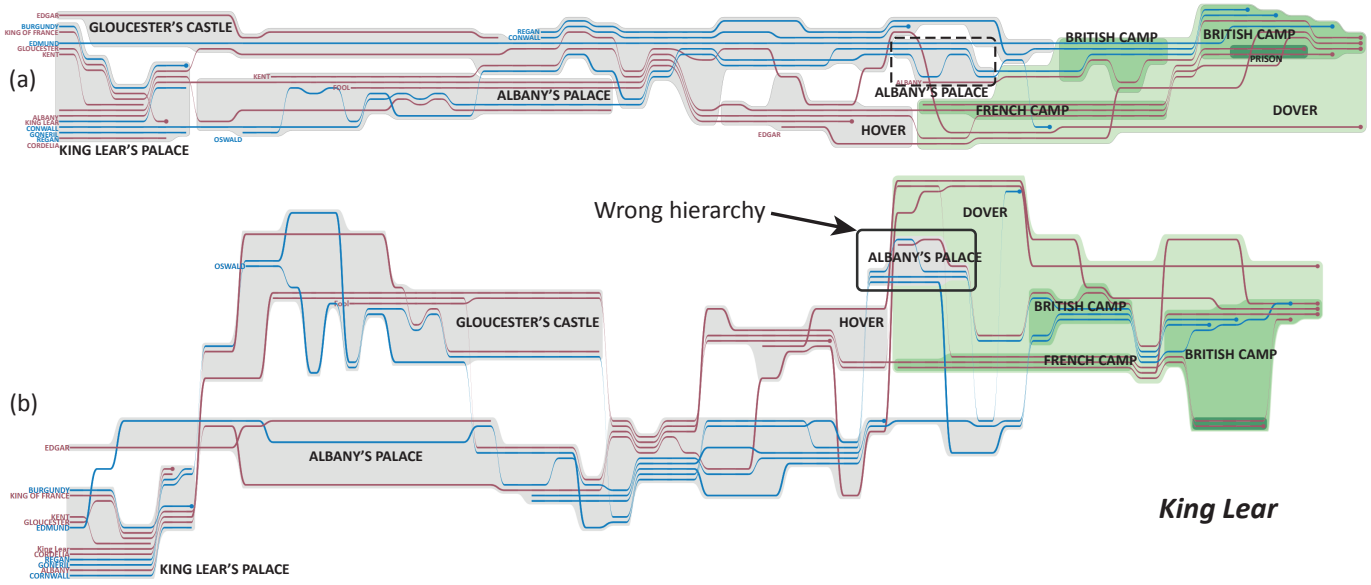

Fig. 13. Comparison of *King Lear* on encoding hierarchical locations: (a) layout by StoryFlow, (b) layout by the TM method. In this layout, *Albany's palace* (the solid rectangle) is incorrectly placed as a part of *Dover*. In our layout, the two locations are placed separately (the dashed rectangle).

were 53 and 107, while ours had 25 and 110, and the TM method had 80 and 110. In terms of line wiggles, the tree layouts are very similar. In terms of line crossings, our method outperforms the others. Compared with the original, our method removes more than half of the crossings, which implies the effectiveness of the proposed optimization method.

Next, we used the movie *Inception* to illustrate that our layout can handle entities with hierarchical locations very well. As shown in Fig. 12, although the location hierarchy is correctly encoded in both layouts, our layout is much more balanced and compact than the one generated by the TM method. For complex entity relationships, the TM method may fail to correctly represent the hierarchical locations. For example, Fig. 13 shows the storyline visualizations of the drama *King Lear*. In the layout generated by the TM method, *Albany's palace* (in the solid black rectangle) is incorrectly placed as part of *Dover*. While in our layout, these two locations are placed separately (see the dashed black rectangle). This is because hierarchical locations are well considered in our optimization approach.

To demonstrate the usefulness of StoryFlow, we conducted a case study with a sociology PhD student and a professor in media and communication studies. In this case study, we aimed to trace the evolution of the temporal cooccurrence relationships between opinion leaders based on their attention to different topics. We used a Twitter dataset related to the 2012 US presidential election. We collected 89,174,308 tweets from Twitter using query words such as "Obama," "Romney," and "election." These tweets were gathered from May 8, 2012 to Nov. 13, 2012 and covered the major events related to the presidential election. We selected 900 opinion leaders based on the number of retweets. The opinion leaders were classified into three major groups by the two experts: *political figures* (334), *media* (288), and *grassroots* (276). Five popular topics, *Welfare*, *Defense*, *Economy*, *Election*, and *Horse race*, were identified by the domain experts. Based on the classification result, we extracted 2,344 popular hashtags that were associated with the topics during the election. The topics and hashtags constitute a 2-level hierarchy.

We grouped the data by 3-day periods and obtained 63 time frames. At each time we assigned an opinion leader to a hashtag that he/she tweeted or retweeted most frequently. Thus, we created a session table for the opinion leaders at each time. With the session tables and the topic hierarchy, we visualized the temporal relationships between the opinion leaders. Fig. 14 shows the overall interactions of the three groups of opinion leaders. Each layer represents a topic. In each layer, three colored stripes were used to encode the opinion leaders involved in this topic: green for grassroots, yellow for media, and pink for political figures. The width of a layer at a time frame encodes the overall proportion of different opinion leader groups in the topic. Several patterns stand out clearly in the bundled storyline visualization.

First, all three opinion leader groups focused mainly on the *Election* topic. The grassroots group was also interested in the *Economy* topic as it was closely related to their daily lives. Their attention switched more frequently than that of the media and political figures. On the other hand, political figures were more focused and seldom switched. The media also primarily stayed on the *Election* topic but they switched to other topics occasionally. For instance, a large proportion of the media switched their focus to the *Welfare* topic from Oct. 29 to Nov. 7.

Five significant peaks emerged in the *Election* topic. Four of them were related to the presidential debates and the vice presidential debate. The last one was about voting. The visualization shows that the topics that were discussed in the debates would remain popular on Twitter for the next few days. One interesting pattern was observed after the third TV debate, which was related to foreign affairs and covered the Libya issue. During that time, grassroots attention on the *Election* topic deceased gradually and many of them switched their focus to the *Defense* topic. We expanded this region to see more information at the hashtag level (Fig. 14(b)). Clearly, the grassroots mainly discussed the issue related to the hashtag "benghazi." To discover more information about "benghazi," we browsed the related tweets. Most of them criticized the Obama administration for its handling of the terrorist attack on the U.S. diplomatic compound in Benghazi. For example, one of the tweets is 'doesn't obama owe us answers (on #benghazi)?"

Second, we identified two significant focus transitions from the *Election* to the *Economy*. On Oct. 14, a large amount of users in the grassroots group switched their attention from the *Election* to the *Economy* while fewer media or political figures switched. We expanded the *Economy* topic from Oct. 14 to 16 to examine the related hashtags (Fig. 14(a)). Obviously, grassroots attention switched to the hashtag "sensata." We used the hashtag to retrieve the related tweets for further analysis (such as the tweet "think Romney is tough on china? ask the workers of #sensata about that as they train their chinese replacements") and found that the transition was caused by the Sensata scandal invovling Mitt Romney. Then the grassroots group returned to the *Election* topic to see Romney's response to the scandal in the second debate around Oct. 17. Surprisingly, only a small proportion of media or political figures discussed the issue on Twitter. After we read the tweets related to this issue, we found that the media's opinion was similar to that of the grassroots group. They criticized Romney for outsourcing jobs to China to make profits. For political figures they discussed the impact of this scandal on Romney and Romney's response to the scandal. We speculated that they were more careful about what they said, while the grassroots group were more likely to

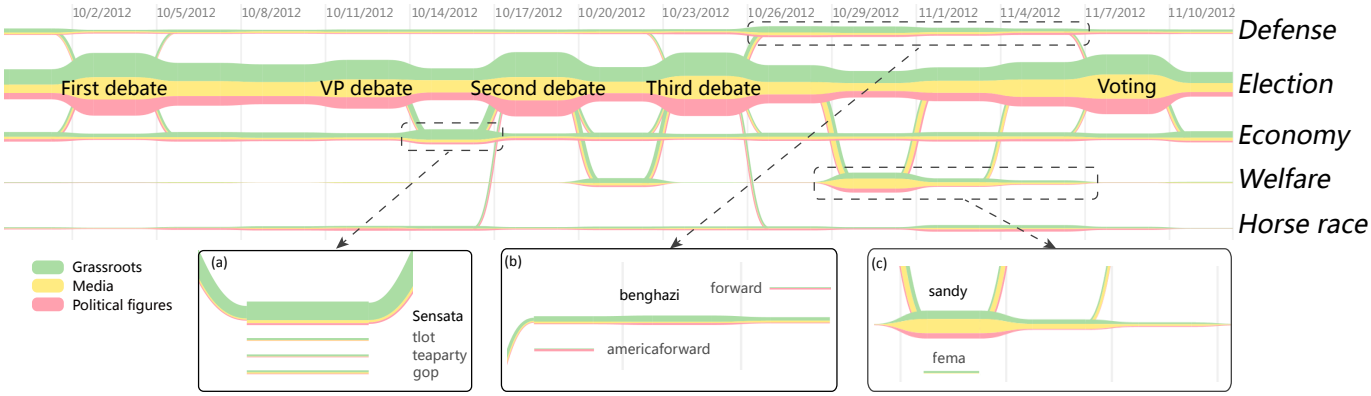

Fig. 14. Visualization of the 2012 US presidential election. It contains 900 opinion leaders and 63 time frames. The opinion leaders are organized by a 2-level topic hierarchy. The opinion leaders are bundled together by the LOD technique. Each color represents an opinion leader group.

express their opinions regarding different events/scandals. On Oct. 29, we found that a lot of opinion leaders turned from *Election* to *Welfare*. Most of them were from the media group. We expanded the related region to the hashtag level (Fig. 14(c)). We found that the transition was caused by the breaking news of hurricane "Sandy" which hit US on Oct. 29. As the hurricane weakened, the media gradually returned to the *Election* topic from Nov. 1 to 4 since the election date was approaching. The presidential election on Nov. 7 attracted a significant amount of attention from the media. After the election, they started talking about other topics, as demonstrated by the fact that the media's interest in the election declined gradually. The sociology professor commented, "This phenomena is interesting and can be interpreted through the lens of the *issue-attention cycle* theory in communication and media studies [12]."

7.3 Initial User Feedback

To evaluate the usefulness of StoryFlow, we conducted a semi-structured interview with three experts, a film professor (User *F*), a sociology PhD student (User *S*), and a professor (User *P*) in media and communication studies.

The interview of User F took 60 minutes, including 10 minutes of system demonstration, 20 minutes of case study and free exploration, and 30 minutes of post interview. Overall, StoryFlow was well received by User *F*. He commented, "I love this visualization! It is a great way to show the interactions between characters over time, which can definitely help filmmaking." He suggested several use scenarios:

- **Film directors** can use StoryFlow to understand the story evolution in a movie to make a proper plan for the film shooting. User *F* commented, "StoryFlow enables a fast review of the shooting timetable and allows the directors to make a better decision on the most advantageous shot order."
- **Script adapters** can use Storyflow to review a script quickly to decide whether to add or remove certain characters or scenes. User *F* said, "StoryFlow can also be used as an effective tool to communicate their ideas to the film directors and producers."
- **Actors** can use StoryFlow to better trace their related scenes and see immediately where and who they will interact with, so that they can better prepare for their performance.

User *F* also suggested adding a storyboard, which consists of a sequence of key frames, to show the expressive context.

After conducting the 2012 US presidential election case study with User *P* and User *S*, we also took 40 minutes to interview them. They both confirmed the usefulness of StoryFlow and the intuitiveness of the visual design. User *S* was very impressed by the visualization and commented "It is stunning and engaging. It definitely outperforms the static infographics that I have seen before." He also appreciated the interactive, level-of-detail feature of StoryFlow, which enables him to immediately see the overall patterns as well as to directly interact with the visualization to see more detail. User *P* said, "StoryFlow would be particularly useful for data-driven journalism because it not only provides a clear visual summary of events but also shows informative context for investigative analysis. He further added, "StoryFlow can be widely used since it can produce expressive visualizations with simple input." Both scholars suggested several potential applications for StoryFlow. For example, User *S* indicated that it would be interesting to see the dynamic relationships of the liberal and conservative opinion leaders over time, while User *P* suggested adding sentiment information to the StoryFlow visualization to provide richer context for further analysis.

8 Discussion and Future Work

We have presented an efficient optimization approach to generate a storyline layout with thousands of entities and hundreds of time frames. The major feature of this approach is that it divides the layout optimization into two parts: discrete optimization to minimize the number of line crossings and wiggles, and continuous optimization for minimizing the wiggle distance and white space. With this division, our method can quickly achieve a better local optimum than the state-of-the-art method. Due to the efficiency of the method, a set of rich interactions is provided to allow users to examine the story and its evolution from multiple perspectives. Accordingly, one interesting avenue for future work is to investigate which interactions are useful for what kind of analysis tasks.

Although StoryFlow has achieved certain successes in illustrating the story evolution in several domains such as movies, dynamic social networks, and blog analysis, it has some limitations in certain applications.

First, in many applications, if the time scale is coarser, an entity may occur in several locations in the same time frames. However, our layout method assumes that an entity only belongs to one session at one time. This will limit its application to some extent. One potential solution is to use several lines to represent one entity and allow them to merge and split over time. Then the current optimization approach is extended to accommodate this new representation. This is very interesting problem to investigate in the future.

Second, the timeline in StoryFlow is linear, which does not scale well with thousands of time frames. An intuitive solution is to use a non-linear timeline with some unimportant time frames merged. However, a non-linear timeline may not be compatible with the mental map of average users. As a result, the key is to find an appropriate visual metaphor that can both well present the non-linear information and preserve users' mental maps.

Finally, a story is told sequentially in StoryFlow. In addition to a sequential narrative, a flashback and a narration interspersed with flashbacks are also two widely used narrative methods in storytelling. For a simple flashback, we can still leverage the StoryFlow visualization, while for a narration interspersed with flashbacks, it is quite challenging to illustrate the story with one storyline layout. Small multiples, which display several sequential stories simultaneously, might be a solution. In the future, we plan to augment the existing storyline visualization to support more narrative methods used in various storytelling practices.

Acknowledgments

The authors would like to thank Yuzuru Tanahashi for providing the comparison data and helping generate some of the comparison examples and Stephen Lin for proofreading the paper.

REFERENCES

[1] Mosek package. http://www.mosek.com, March 2013.

[2] SIMILE Widget. http://www.simile-widgets.org/timeline/, March 2013.

[3] Wikipedia. http://en.wikipedia.org/wiki/Story, March 2013.

[4] W. Aigner, S. Miksch, B. Thurnher, and S. Biffl. PlanningLines: novel glyphs for representing temporal uncertainties and their evaluation. In *Proceedings of the International Conference on Information Visualisation*, pages 457–463, 2005.

[5] P. André, M. L. Wilson, A. Russell, D. A. Smith, A. Owens, and M. C. Schraefel. Continuum: designing timelines for hierarchies, relationships and scale. In *Proceedings of the annual ACM symposium on User interface software and technology*, pages 101–110, 2007.

[6] C. Blundell, Y. W. Teh, and K. Heller. Bayesian rose trees. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 65–72, 2010.

[7] A. A. T. Bui, D. R. Aberle, and H. Kangarloo. TimeLine: visualizing integrated patient records. *IEEE Transactions on Information Technology in Biomedicine*, 11(4):462–73, 2007.

[8] M. Burch, F. Beck, and S. Diehl. Timeline trees: visualizing sequences of transactions in information hierarchies. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 75–82, 2008.

[9] M. Burch, C. Vehlow, F. Beck, S. Diehl, and D. Weiskopf. Parallel edge splatting for scalable dynamic graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2344–2353, 2011.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.

[11] W. Cui, S. Liu, L. Tan, C. Shi, Y. Song, Z. Gao, H. Qu, and X. Tong. TextFlow: towards better understanding of evolving topics in text. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2412–2421, 2011.

[12] A. Downs. Up and down with ecology the "issue-attention cycle". *The Public Interest*, 28:38–50, 1972.

[13] P. Eades and N. C. Wormald. Edge crossings in drawings of bipartiie graphs. *Algorithmica*, 11(4):229–233, 1994.

[14] J. A. Fails, A. Karlson, L. Shahamat, and B. Shneiderman. A visual interface for multivariate temporal data: finding patterns of events across multiple histories. In *IEEE Symposium On Visual Analytics Science And Technology*, pages 167–174, 2006.

[15] D. Fisher, A. Hoff, and G. Robertson. Narratives: a visualization to track narrative events as they develop. In *IEEE Symposium On Visual Analytics Science And Technology*, pages 115–122, 2008.

[16] Y. Frishman and A. Tal. Online dynamic graph drawing. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):727–740, 2008.

[17] E. R. Gansner, E. Koutsofios, S. C. North, and K.-P. Vo. A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, 19(3):214–230, 1993.

[18] J. Hullman and N. Diakopoulos. Visualization rhetoric: framing effects in narrative visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2231–2240, 2011.

[19] M. Jensen. Visualizing complex semantic timelines. Technical report, NewsBlip, 2003.

[20] N. W. Kim, S. K. Card, and J. Heer. Tracing genealogical data with timenets. In *Proceedings of the International Conference on Advanced Visual Interfaces*, pages 241–248, 2010.

[21] M. Krstajic, M. Najm-Araghi, F. Mansmann, and D. Keim. Incremental visual text analytics of news story development. In *Proceedings of Conference on Visualization and Data Analysis, VDA*, 2012.

[22] V. Kumar, R. Furuta, and R. B. Allen. Metadata visualization for digital libraries: interactive timeline editing and review. In *Proceedings of the third ACM conference on Digital Libraries*, pages 126–133, 1998.

[23] S. Liu, M. X. Zhou, S. Pan, W. Qian, W. Cai, and X. Lian. Interactive, topic-based visual text summarization and analysis. In *CIKM*, pages 543–552, 2009.

[24] S. Liu, M. X. Zhou, S. Pan, Y. Song, W. Qian, W. Cai, and X. Lian. Tiara: Interactive, topic-based visual text summarization and analysis. *ACM Trans. Intell. Syst. Technol.*, 3(2):25:1–25:28, 2012.

[25] D. Luo, J. Yang, M. Krstajic, W. Ribarsky, and D. A. Keim. Eventriver: visually exploring text collections with temporal references. *IEEE Transactions on Visualization and Computer Graphics*, 18(1):93–105, 2012.

[26] K.-L. Ma, I. Liao, J. Frazier, H. Hauser, and H.-N. Kostis. Scientific storytelling using visualization. *IEEE Computer Graphics and Applications*, 32(1):12–19, 2012.

[27] R. Munroe. Movie narrative charts. http://xkcd.com/657/, March 2013.

[28] J. Nocedal and S. Wright. *Numerical optimization*. Springer, 1999.

[29] M. Ogawa and K.-L. Ma. Software evolution storylines. In *Proceedings of the international symposium on Software visualization*, pages 35–42, 2010.

[30] V. Ogievetsky. Plotweaver. http://ogievetsky.com/PlotWeaver/, March 2013.

[31] C. Plaisant, B. Milash, A. Rose, S. Widoff, and B. Shneiderman. LifeLines: visualizing personal histories. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 221–227, 1996.

[32] C. Plaisant, R. Mushlin, A. Snyder, J. Li, D. Heller, and B. Shneiderman. LifeLines: using visualization to enhance navigation and analysis of patient records. In *Proceedings of the AMIA Annual Symposium*, pages 76–80, 1998.

[33] H. C. Purchase. Which aesthetic has the greatest effect on human understanding? In *Proceedings of the International Symposium on Graph Drawing*, pages 248–261, 1997.

[34] K. Reda, C. Tantipathananandh, A. Johnson, J. Leigh, and T. Berger-Wolf. Visualizing the evolution of community structures in dynamic social networks. *Comp. Graphics Forum*, 30(3):1061–1070, 2011.

[35] S. Rose, S. Butner, W. Cowley, M. Gregory, and J. Walker. Describing story evolution from dynamic information streams. In *Proceedings of IEEE Symposium on Visual Analytics Science and Technology*, pages 99––106, 2009.

[36] E. Segel and J. Heer. Narrative visualization: telling stories with data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1139–1148, 2010.

[37] C. Shi, W. Cui, S. Liu, P. Xu, W. Chen, and H. Qu. Rankexplorer: Visualization of ranking changes in large time series data. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2669–2678, 2012.

[38] C. Stab, K. Nazemi, and D. W. Fellner. Sematime-timeline visualization of time-dependent relations and semantics. In *International Symposium on Visual Computing*, pages 514–523, 2010.

[39] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man and Cybernetics*, 11(2):109 – 125, 1981.

[40] R. Tamassia, G. Di Battista, and C. Batini. Automatic graph drawing and readability of diagrams. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):61–79, 1988.

[41] Y. Tanahashi. Movie interaction dataset. http://vis.cs.ucdavis.edu/~tanahashi/data_downloads/storyline_visualizations/, March 2013.

[42] Y. Tanahashi and K.-L. Ma. Design considerations for optimizing storyline visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2679–2688, 2012.

[43] I. Tazawa, S. Koakutsu, and H. Hirata. An immunity based genetic algorithm and its application to the VLSI floorplan design problem. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 417–421, 1996.

[44] T. D. Wang, C. Plaisant, A. J. Quinn, R. Stanchak, S. Murphy, and B. Shneiderman. Aligning temporal data by sentinel events: discovering patterns in electronic health records. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 457–466, 2008.

[45] K. Wongsuphasawat and D. Gotz. Exploring flow, factors, and outcomes of temporal event sequences with the outflow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2659–2668, 2012.

[46] K. Wongsuphasawat, J. A. Guerra Gómez, C. Plaisant, T. D. Wang, M. Taieb-Maimon, and B. Shneiderman. LifeFlow: visualizing an overview of event sequences. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1747–1756, 2011.

[47] P. Xu, Y. Wu, E. Wei, T. Peng, S. Liu, J. Zhu, and H. Qu. Visual analysis of topic competition on social media. *IEEE Transactions on Visualization and Computer Graphics*, 19(12), 2013.