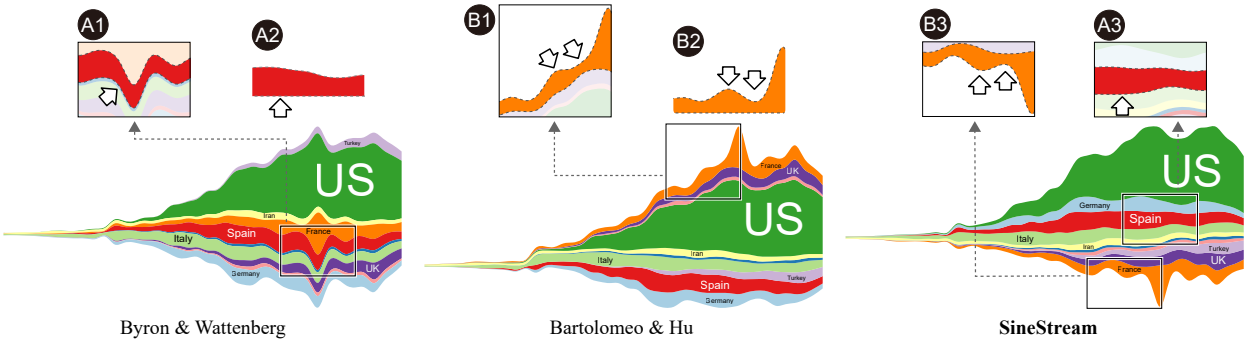
SineStream: Improving the Readability of Streamgraphs by Minimizing Sine Illusion Effects

Chuan Bu, Quanjie Zhang, Qianwen Wang, Jian Zhang, Michael Sedlmair, Oliver Deussen, Yunhai Wang



Byron & Wattenberg | Bartolomeo & Hu | **SineStream**

Fig. 1. *SineStream* minimizes sine illusion effects stemming from strong slopes and improves the readability of streamgraphs by aligning the orthogonal and vertical orientations of each layer. We show the results on the COVID dataset [23] (top) and the Bank Interest dataset [30] (bottom). Sine illusions appear because the human eye is not able to accurately estimate the vertical distance between two curves with similar slopes. As indicated by the arrows on the top, our approach (A3, B3) better reflects the real thickness of layers (A2, B2) than previous works by Byron & Wattenberg [4] (A1) and Bartolomeo & Hu [10] (B1).

Abstract—In this paper, we propose *SineStream*, a new variant of streamgraphs that improves their readability by minimizing sine illusion effects. Such effects reflect the tendency of humans to take the orthogonal rather than the vertical distance between two curves as their distance. In *SineStream*, we connect the readability of streamgraphs with minimizing sine illusions and by doing so provide a perceptual foundation for their design. As the geometry of a streamgraph is controlled by its baseline (the bottom-most curve) and the ordering of the layers, we re-interpret baseline computation and layer ordering algorithms in terms of reducing sine illusion effects. For baseline computation, we improve previous methods by introducing a Gaussian weight to penalize layers with large thickness changes. For layer ordering, three design requirements are proposed and implemented through a hierarchical clustering algorithm. Quantitative experiments and user studies demonstrate that *SineStream* improves the readability and aesthetics of streamgraphs compared to state-of-the-art methods.

Index Terms—Streamgraphs, Sine Illusion, Readability

1 Introduction

Stacked graphs are a common visualization technique for displaying multiple time-series. In a stacked graph, the values of each time-series are represented by the thickness of a colored layer. Stacking these layers on top of each other allows to display the relative amount of each time-series as well as their total sum. In 2008 the New York Times published streamgraphs [4], a variant of stacked graphs with curved baseline, and immediately these graphs gained great popularity. Since then streamgraphs have been widely used for visualizing casual information [24], such as music listening histories [3], box office revenue of movies [4], and data from social media [26].

- *C. Bu, Q. Zhang, Y. Wang are with Shandong University, Qingdao, China. E-mail: {buchuan1023, zhangquanjie.cn, cloudseawang}@gmail.com.*
- *Q. Wang is with the HongKong University of Science and Technology, Hong Kong, China. E-mail: qwangbb@connect.ust.hk.*
- *J. Zhang is with CNIC, CAS. E-mail: zhangjian@sccas.cn.*
- *M. Sedlmair is with VISUS, University of Stuttgart, Germany. E-mail: michael.sedlmair@visus.uni-stuttgart.de.*
- *O. Deussen is with Konstanz University, Germany and Shenzhen VisuCA Key Lab, SIAT, China. E-mail: oliver.deussen@uni-konstanz.de.*
- *Y. Wang is the corresponding author.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

Stacking layers, however, inevitably introduces distortions, which affect the readability and aesthetics of the result. As a streamgraph is mainly controlled by its baseline (the bottom-most curve) and the ordering of layers, variants of streamgraphs have been proposed to reduce distortions by optimizing these two factors. For example, the above-mentioned streamgraphs [4] minimize the "wiggle", the fluctuation of layers, resulting in a good compromise between aesthetics and readability. Their ordering, however, was specifically designed for displaying box office revenues of movies by using the "onset time" property, where the newly incoming layers are added at the top. To visualize more general data, Bartolomeo and Hu [10] propose to incorporate baseline and ordering into an optimization framework combined with a refined wiggle measure. This allows to further reduce distortions. While many efforts have been made to improve their readability, the design of streamgraphs still lacks a clear perceptual foundation. Byron and Wattenberg [4] speculate that the perception of such graphs is related to the principle of banking to 45 degrees [5] but provide no detailed discussion.

One elementary task for a streamgraph is to show the thickness of layers over time. Its readability, however, is hampered by the difficulties of humans to accurately estimate and compare the vertical distance between two curves whose slopes have the same sign [6]. As shown in Fig. 1, the human eye fails to perceive the actual thickness of layers, when they are slanted too much (A1, B1). This phenomenon also manifests in classical visualizations such as Playfair's trade-balance chart [22]. Recently, VanderPlas and Hofmann [32] related this phenomenon to the *sine illusion* [9] and provided a perceptual explanation: our brain prefers to take the orthogonal distance rather than the vertical

extent distance as the distance between two curves. This illusion is most prominent when the angle between the two orientations is large. Since one elementary task in reading stacked graphs (and streamgraphs) is to compare the thickness of layers (the vertical distance between the bottom and top curve), we were motivated to investigate the connection between sine illusion and streamgraphs; and we wanted to exploit the sine illusion to improve the readability of streamgraphs.

In this paper, we propose to use *SineStreams*, a variant of streamgraphs that minimizes such illusions. Therefore we revisit the mathematical model of streamgraph optimization and interpret it in terms of reducing sine illusions. This requires the slope of each layer to be close to zero as much as possible. We show that minimizing the sine illusion is equivalent to the original objective of optimizing streamgraphs, which is minimizing the sum of the squared slopes of all layers. However, to reduce the illusion over several layers stacked on top of each other, we additionally introduce a Gaussian weight to penalize layers with large variations and reduce the distortion imposed on other layers. Since a flat baseline allows viewers to read the total sum easily, we tried to order layers in a way that neighboring layers mutually cancel out their distortions to the greatest possible extent.

To arrive at an effective ordering, we propose a *compensation degree* measure, which tells us how well two layers would cancel out their wiggles. We combine this measure with the thickness and length of the layers, resulting in a pairwise distance measure between them. Based on this distance measure, we apply hierarchical clustering [17] to construct a binary tree representation for all layers. We compute the final ordering from this tree by minimizing the sum of distances between adjacent layers with a leaf ordering algorithm [1]. Once the ordering is obtained, we compute the baseline by improving Byron and Wattenbergs [4] method using a Gaussian weight.

We demonstrate the effectiveness of *SineStreams* by quantitatively comparing them with other streamgraphs w.r.t. reducing wiggles and sine illusion effects. In addition, conduct a user study to asses the readability and aesthetics of *SineStream* by following the task design of Thudt et al. [31]. The results demonstrate that *SineStreams* offer an improved readability and aesthetics. The source code is available for download on GitHub[1]. In summary, the main contributions of this paper are:

- we revisit streamgraphs and show that their baseline computation and layer ordering can be re-formulated in terms of minimizing the sine illusion effect (Sect. 3);
- we introduce three design requirements for ordering layers in a *SineStream* and propose an effective ordering algorithm (Sect. 4);
- we quantitatively compare *SineStreams* to state-of-the-art methods and conduct user studies to show the effectiveness of our approach (Sect. 5).

2 Related work

The use of streamgraphs can be traced back at least to Playfair's trade-balance chart of 1786 [22]. Later on, increased computing power enabled users to create stacked graphs that incorporate large numbers of time-series, as well as creating variants with different appeals.

When the design of streamgraphs was first discussed in ThemeRiver [15], the authors introduced a baseline algorithm that produced a result symmetrical along the direction of the timeline. No specific ordering algorithm was proposed here, even though the authors mention the possibility of putting related layers close to each other. Because in a ThemeRiver layers are stacked upwards and downwards from the baseline, its outer layers are less distorted than the outer layers of the original stacked graphs and therefore it displays a higher readability and aesthetic appeal.

As mentioned above, Streamgraphs [4], a stacked graph with a curved baseline, are a variant of stacked graphs that is a widely-used visualization technique for displaying multiple time-series. The authors propose a baseline algorithm that reduced the distortion of each layer, thus further improving the readability with regard to ThemeRiver. By finding a baseline that minimizes the distortions of the layers, streamgraphs enable the user to easily read the thickness of each layer at the cost of perceiving the overall thickness of the whole graph less well. The authors discuss a set of distortion functions for baseline optimization, including *silhouette, deviation, wiggle,* and *weighted_wiggle*. They conclude that *weighted_wiggle* leads to the best balance between readability and aesthetics. An ordering algorithm was developed specifically for visualizing the box office revenues of movies. Since such revenues usually have a sudden increase and then a rapid decrease, the authors ordered the layers based on their "on-set" time so that layers with the biggest bursts are put at the outside of the graph and do not influence the layout of other layers. The authors also discuss the possibility of designing an ordering algorithm that allows to minimize the distortion function, but leave that as a promising direction for future research.

Since the "on-set" ordering was tailored for displaying box office revenues, Bartolomeo and Hu [10] propose a new ordering algorithm, aiming to generalize streamgraphs to other types of data. Their ordering algorithm first generates an initial ordering in a greedy manner, and then iteratively refines the ordering to minimize the distortion value. This ordering scheme not only generalizes streamgraphs for more than box office revenues, but also further improves the readability of streamgraphs. For baseline optimization, the authors present an alternative definition of the wiggle distortion function based on the L1-norm, which allows to reduce distortions caused by sudden jumps in time-series.

Apart from optimizing baseline and ordering, interaction can also improve the readability of streamgraphs. TouchWave [2] introduces interactive layout adjustments and data queries to streamgraphs, aiming to mitigate perceptual issues and improve readability. Thudt et al. [31] propose an interactive baseline straightening for streamgraphs to improve the readability of a selected layer. By clicking on one individual layer, the baseline of this layer is straightened so that the thickness can be easily observed without distortion. Even though this is an effective operation, these interactions can not be applied to a wide range of statistic visualizations (*e.g.*, posters), where optimizing the baseline and ordering is still the only solution to improve the readability.

In spite of these optimizations, streamgraphs still face considerable problems for their readability. Studies have been conducted to better examine the graphical perception of streamgraphs. In his blog post, Kirk [19] reviewed a set of streamgraphs published on the web, investigated their usage scenarios, and compared their design trade-offs. He concluded that a streamgraph is "a fantastic solution to displaying large datasets to a mass audience". A user study conducted by Thudt et al. [31] validated the effectiveness of these optimizations and interactions in improving the readability of streamgraphs. The authors compare the relative readability among different variants of streamgraphs with 16 participants in four types of tasks. The results show that baseline optimization improves the readability for value comparison tasks.

A number of applications for streamgraphs in formal visual analysis emerged recently, especially, but not exclusively, in the field of visual text analysis. In these applications, such graphs have been proposed to visualize complex relationships in time-series, such as merging and splitting of layers [7], their hierarchical structures [8, 11], and the competition among layers [26, 34]. Even though new ordering and baseline algorithms are proposed in these applications, they mainly aim to adapt streamgraphs to specific domain problems or scenarios, rather than addressing key perceptual issues. For example, Topic Streams [13] visualize online conversations about large-scale events using modified streamgraphs. Layers are ordered here based on a global measure of the novelty of a topic, which is similar to the "on-set" ordering proposed in [4]. Tiara [20] adds word clouds inside layers of streamgraphs to show visual summarizations of large collections of text. In Tiara, layers are ordered based on several constraints, including semantic similarity among topics and their starting time. These applications demonstrate the usefulness of streamgraphs in formal analysis, but paid little attention to address key perceptual issues involved in this form of visualization.

In sum, while various baseline and ordering optimizations as well as

[1] https://github.com/VisLabWang/SineStream

interactions have effectively improved the readability of streamgraphs, these optimizations are mainly based on empirical observations. The design of such graphs still lacks a solid conceptual foundation. In this study, we revisit the optimization goal of streamgraphs and interpret it in terms of the sine illusion effect. New algorithms are proposed to improve the readability of streamgraphs by minimizing sine illusions.

3 Sine Illusion in Streamgraphs

In this section, we introduce basic concepts of streamgraphs and afterwards connect them to the sine illusion.

3.1 Streamgraphs

Unlike traditional stacked charts, which plot values along a straight x-axis, the layers of a streamgraph are stacked upon a curved baseline, thus resembling natural streams, which leads to an aesthetically pleasing effect.

The geometry of a streamgraph is determined by two factors: the **order** of the layers and the shape of the **baseline**. In a typical algorithmic pipeline to construct a streamgraph, we first order the layers, then compute the shape of the baseline, upon which the ordered layers are stacked. Following the notation of previous studies, the time-series that form a streamgraph are modeled as lists of n real-valued, non-negative, differentiable functions $[f_1,...,f_n]$ that determine the thickness of each layer. This assumption is satisfied in many real world datasets and widely used in previous data analyses [4, 10]. For simplicity, we assume that the order of the list of time series indicates the order of the layers in the streamgraph. Its baseline is denoted as g_0 and describes the y-coordinate of the bottom. Therefore, the top of the i_{th} layer can be expressed as $g_i = g_0 + \sum_{j=1}^{i} f_j$.

In pursuit of producing an aesthetic and legible streamgraph, different baseline and ordering algorithms have been proposed. The core of these algorithms is to optimize a metric that quantifies the quality of a streamgraph, *i.e.*, *"what is the definition of a good streamgraph?"* A set of metrics, including silhouette, deviation, wiggle, have been proposed [4, 10, 15]. Among these metrics the wiggle metric, which describes the distortion of layers, leads to the best balance between readability and aesthetics [4, 31]. There are two common definitions of the wiggle metric. Byron and Wattenberg [4] defined it as

$$Wiggle = \sum_{i=1}^{n} (\frac{1}{2}(g_i' + g_{i-1}'))^2 \cdot w_i \tag{1}$$

where g' is the derivative of g and w_i is a positive weight for each layer.

Bartolomeo and Hu [10] criticize that this definition only works for smooth time-series, *i.e.*, time-series whose values do not change dramatically over time. To tackle this issue, they proposed an alternative definition of the wiggle metric based on the L1-norm:

$$Wiggle_{n_1} = \sum_{i=1}^{n} \frac{|g_i'| + |g_{i-1}'|}{2} \cdot w_i \tag{2}$$

Ordering and baseline algorithms proposed by Bartolomeo and Hu [10] are considered as the state of the art for streamgraph production. However, even though it is widely used, using the wiggle metric to optimize a streamgraph layout is mainly based on empirical observation and lacks a clear perception foundation. This is why we want to introduce the sine illusion at this point.

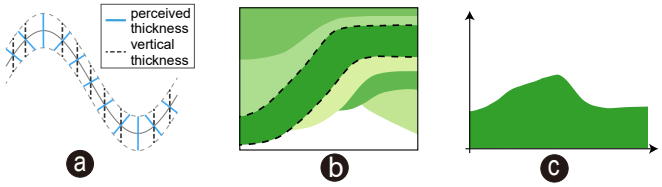


Fig. 2. (a) A line with uniform thickness is drawn along a sinusoidal curve. Perception leads us to using the orthogonal distance rather than the vertical distance to determine its thickness. (b) The green layer with dotted border in the streamgraph seems to have a constant thickness. However, a peak occurs in its vertical thickness (c).

3.2 From Wiggles to the Sine Illusion

The sine illusion reflects the difficulty humans have in visually measuring the distance between two curves. In the cognitive literature, the sine illusion was first documented in the context of sinusoidal curves [9]. In Fig. 2(a), a line with uniform thickness is centered along a sinusoidal curve. All line segments are of equal length (dashed lines) but viewers have the illusion that the line thickness varies along the curve.

The sine illusion is not restricted to sinusoidal curves. It has been reported and discussed extensively for other curves in the cognition literature but also in graphics [6]. The strength of the sine illusion is influenced by two factors: the **slope** of the curve and the **distance** between the two borders of the curve [32]. Increasing the slope, or decreasing the vertical distance leads to an increase of the sine illusion.

The illusion originates from our preference in evaluating the curve width as an orthogonal width rather than the difference along the vertical axis (Fig. 2(a)). This preference is based on our perceptual experience with the surrounding three-dimensional world: tilting an object does not change its orthogonal width. Misapplying this preference to two-dimensional charts leads to the sine illusion. Therefore, minimizing sine illusion requires orthogonal and vertical orientations to be aligned as much as possible. In other words, the slope of each layer should be as close to zero as possible.

Streamgraphs, consisting of many layers with curved boundaries $(g_0, g_1, ...g_n)$, commonly produce sine illusions for the above reason. As shown in Fig. 2, the peak in the data (c) is barely visible in the layer of the streamgraph shown in (b). Thus, viewers can easily be misled by their visual perception and obtain a wrong understanding of the underlying data. Since an elementary task for understanding streamgraphs is to read and compare the thickness of layers, the sine illusion dramatically threatens their readability. It is therefore natural to measure the quality of a streamgraph in terms of the sine illusion, which can provide a solid perceptional foundation for their design.

As mentioned above, to minimize the effect of the sine illusion in streamgraphs, the slope of each layer $\frac{1}{2}|g_i' + g_{i-1}'|$ needs to be minimized. Interestingly, the wiggle based optimization (Eq. 1) can therefore be reinterpreted from the perspective of reducing the sine illusion:

$$\begin{aligned} Illusion = Wiggle &= \sum_{i=1}^{n} (\frac{1}{2}(g_i' + g_{i-1}'))^2 \cdot w_i \\ &= \sum_{i=1}^{n} (g_0' + \frac{1}{2} f_i' + \sum_{j=1}^{i-1} f_j')^2 \cdot w_i \end{aligned} \tag{3}$$

4 SineStream

Even though Byron and Wattenberg [4] unintentionally reduced the sine illusion with minimizing their wiggle value during baseline optimization, they did not take the sine illusion into account when ordering the layers. Therefore, in the obtained streamgraphs the illusion was not minimized. Moreover, the authors choose an inappropriate weight $w_i = f_i$ in their optimization that ignored the stronger sine illusion for thin layers. As shown in Eq. 3, the corresponding distraction of a streamgraph is determined by both g_0' and the order of $[f_1', f_2', ..., f_n']$. *SineStreams* (Sine Illusion Minimized Streamgraphs), therefore aim to minimize the sine illusion in streamgraphs by using a refined baseline computation and an optimization algorithm for ordering layers appropriately.

4.1 Baseline Optimization

As shown in Eq. 3, the sine illusion is a function of g_0' and can be minimized when g_0' is properly chosen. As the w_i are positive, we can get $\frac{\partial^2 Illusion}{\partial^2 (g_0')} = \sum_{i=1}^{n} 2w_i > 0$ and minimize the sine illusion when $\frac{\partial Illusion}{\partial (g_0')} = 0$. Therefore, we share the solution of g_0 with Byron and Wattenberg [4], since their choice of g_0 simultaneously minimizes the value of the L2-norm-based *wiggle* measure in Eq. 1 and the value of

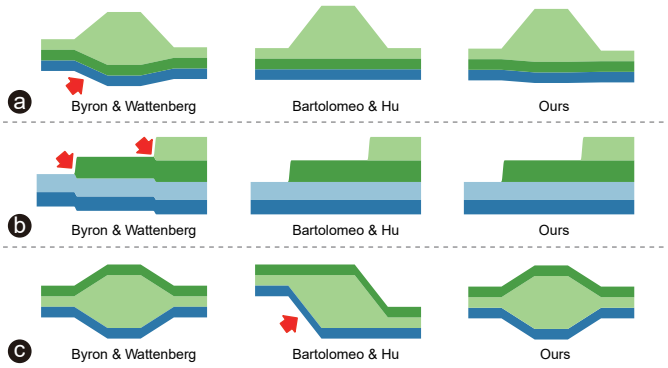


Fig. 3. Each row represents streamgraphs with the same data and layer order, but different baseline computations. (a) Byron & Wattenberg [4] encourages $[g'_i, g'_{i-1}]$ to have equal absolute values and opposite signs, regardless of the increased sine illusion it causes to other layers. (b) This leads to unpleasant distortions when layers have sudden changes in thickness. (c) Bartolomeo & Hu [10] prefer a single distortion of a large layer rather than two smaller ones, which also leads to a larger illlusions.

sine illusion in Eq. 3:

$$g'_0 = -\frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=0}^{n} \frac{1}{2} \left(\sum_{j=1}^{i} f'_j + \sum_{j=1}^{i-1} f'_j \right) w_i$$
$$= -\frac{\sum_{i=1}^{n} w_i \left(f'_1 + \sum_{j=1}^{i-1} (f'_j + f'_{j+1}) \right)}{2 \sum_{i=1}^{n} w_i} \tag{4}$$

This choice of g_0 has been widely been used since it was proposed, but recently it was challenged by Bartolomeo and Hu [10], who claimed that an L2-norm-based optimization only works well for relatively smooth time-series.

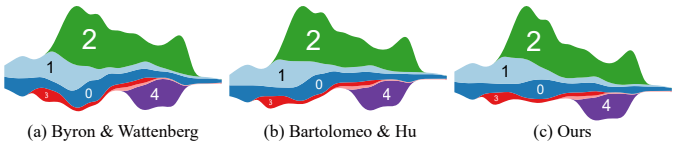
Limits of Existing Techniques. According to Bartolomeo and Hu [10], an L2-norm-based optimization will lead to two main problems: a) it encourages $[g'_i, g'_{i-1}]$ with equal absolute values and opposite signs, regardless of the increased distortion this might cause to other layers; b) it leads to unpleasant distortions when layers have sudden and big changes in thickness (*i.e.*, data jumps). To eliminate these two issues, Bartolomeo and Hu propose an alternative L1-norm-based baseline optimization (Eq. 2).

Even though these two shortcomings do exist in Byron and Wattenbergs L2-norm-based optimization, we find that Bartolomeo and Hu's criticism is not accurate, since such unpleasant distortions are mainly caused by inappropriately chosen weights $w_i = f_i$ instead of the L2-norm-based definition. In other words, the distortion of thinner layers is treated as less important than the one of the thicker layers during baseline optimization.

As shown in Fig. 2(a), the thickness f_i of the light green layers grows, which leads to a large weight $w_i = f_i$. Therefore, the L2-norm-based wiggle value of the thick layer is reduced to a great extent even though this significantly increases the 2-norm wiggle value of the two thin layers (dark green and dark blue). The same goes for the distortions caused by data jumps (Fig. 3(b)). If less weight is given to layers when they have a sudden data jump, *i.e.*, the start of the dark green and the light green layer, the distortion of the other layers can be minimized during baseline optimization.

More importantly, an L1-norm-based optimization, even though solving the aforementioned two shortcomings, potentially introduces severe distortions and increases the effect of the sine illusion, as shown in Fig. 3(c). Compared to the L2-norm-based optimization, it prefers a large distortion over several small ones. Thus, the baseline optimization by Bartolomeo and Hu increases the slope of the dark blue and the dark green layer. Since the sine illusion increases with the slope of the curves, an L1-norm-based optimization results in stronger illusions.

Gaussian Weight for Baseline Optimization. To solve these issues with existing techniques, we propose to use a modified weighting scheme for the baseline optimization in *SineStreams*. First, we propose to use the L2-norm, rather than the L1-norm as it allows to measure the effect of the sine illusion and avoids big slopes that might be introduced



(a) Byron & Wattenberg (b) Bartolomeo & Hu (c) Ours

Fig. 4. Three streamgraphs that use our ordering algorithm but different baseline optimizations: $Wiggle_{norm2}$, $Wiggle_{norm1}$, and sine illlusion.

by the L1-norm (Fig. 3(c)). We modify the original weight $w_i = f_i$ by a Gaussian weight to reduce the influence of a layer when its thickness undergoes big changes:

$$w_i = \exp\left(-\frac{f'^2_i}{2c^2}\right) \cdot f_i$$

where c can be either the median, arithmetic mean, harmonic mean, or geometric mean of the $|f'_1|, |f'_2|, ..., |f'_n|$. By adjusting the weight with the change of the thickness, streams with large fluctuations are penalized so that the ones with small fluctuations will have lower slopes (Fig. 3(a)). In doing so, streams with large fluctuations can be perceived more easily. Based on our experiments, we choose to set c to be the median. When $c = 0$, we define $w_i = f_i$.

When the weight of a layer f_i is lowered according to this formula, its neighboring layers gain a relatively larger weight and thus distortions caused by f_i can be removed more effectively during baseline optimization. As shown in Fig. 3(a), the dark blue and the dark green layers suffer from distortions caused by the thickness change of the light green layer. Our Gaussian weight increases their relative importance, thus flatting these two layers during baseline optimization and reducing the effect of the sine illusion. The same goes for Fig. 3(b). At the beginning, the dark green and light green layers have a sudden thickness change and accordingly introduce distortions to the neighboring layers. By reducing the weights of these two layers at their start time, the Gaussian weight effectively removes the distortions of the neighboring layers.

Meanwhile, using a Gaussian weight also increases the relative weight of thin vs. thick layers. In real word datasets, a layer with large thickness changes f'_i is usually a thick layer. The term $\exp(-f'^2_i/2c^2)$ is a monotonously decreasing function of f'_i and thus increases the relative weight of thin layers. As a result, our baseline optimization pays more attention to reducing the distortions of thinner layers. As the effect of the sine illusion increases with decreasing vertical distance between curves [32], a Gaussian weight helps minimizing the total amount of sine illusion in a *SineStream*. It is determined by the following function:

$$Illusion = \sum_{i=1}^{n} \left(\frac{1}{2} (g'_i + g'_{i-1}) \right)^2 \cdot \exp\left(-\frac{f'^2_i}{2c^2}\right) \cdot f_i \tag{5}$$

As shown in Fig. 4, using the same ordering, the method of baseline optimization substantially influences the quality of the resulting streamgraph. Our proposed baseline optimization leads to the most satisfying layout.

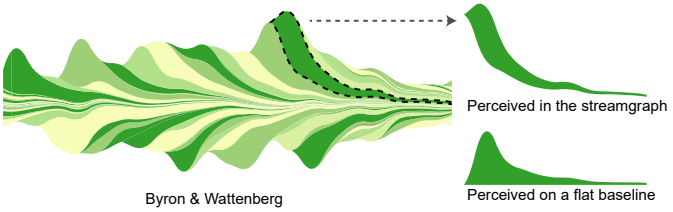


Fig. 5. The Ebb and Flow of Movies [28], produced using the method of [4]. A dramatic difference exists between the perceived thickness of the streamgraph and the actual thickness perceived with a flat baseline.

4.2 Optimizing the Layer Order

In streamgraphs, layers are stacked on top of each other. In some cases, they have a semantically intrinsic order, *e.g.*, alphabetical [33]. But in all other cases, layers can be reordered to enhance the readability.

Limits of Existing Techniques. Two main techniques exist for the ordering in streamgraphs: 1) LateOnset ordering [4]; 2) TwoOpt

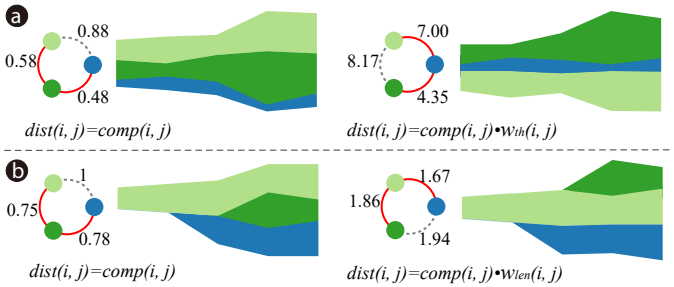


Fig. 6. Examples about how the *thickness weight* and *length weight* help improve layer ordering. In the distance graph, each node represents a layer and the number on the edge indicates the distance value $dist(i,j)$ between the two layers. Layers with low distances values (red edges) are put together in the streamgraph. For reducing the sine illustion effect, the *thickness weight* helps moving thin layers to the middle (a) while the *length weight* helps moving short layers to the outside (b).

ordering [10]. Other ordering implementations are mainly based on modifications of these two techniques [13, 20].

As mentioned above, *LateOnset* ordering was specifically tailored for displaying trends within box office revenues of movies in streamgraphs. In this ordering, layers are first sorted by their "on-set" time. Then the layers are stacked alternatively on either the bottom or the top of the existing streamgraph in order to achieve an even visual effect. The main purpose of LateOnset ordering is to pursue a better aesthetic appeal when displaying the revenues, which typically have a sudden increase and then decrease over time. LateOnset ordering can effectively move the maximum "burst" of each layer to the outside of the graph and thus achieves a smooth layout. While this leads to a smooth streamgraph that is visually pleasing, it tends to place layers on slanted edges, thus potentially introducing severe sine illusion effects, as shown in Fig. 5. Moreover, it is difficult to apply this algorithm to general time-series, for example when all time-series in a dataset have the same start time.

Contrary to LateOnset, *TwoOpt* ordering is designed for general time series. In TwoOpt ordering, an initial ordering is obtained by sorting all layers in a greedy manner. At each time step, a layer that introduces the lowest wiggle value to the streamgraph is selected and stacked on the graphs. Then, the initial ordering is iteratively refined by swapping adjacent layers if the swapping reduces the wiggle value. Swapping stops after a given number of iterations. Such iterative swapping is able to cancel out the total wiggle value of neighboring layers. This idea is already discussed in [4], but left to future work.

While successfully allowing streamgraphs to be applied to general time series, TwoOpt has several limits. First, being a greedy strategy, TwoOpt cannot always produce the same ordering. The method requires users to define the number of swapping operations and can generate different layer orderings for the same set of input data. During the refinement, iterative swapping also only considers canceling out the wiggle values of the two neighboring layers, wiggle canceling through merging more than two layers is not discussed. Moreover, TwoOpt ordering also suffers from inappropriate weights $w_i = f_i$, which give higher priority to reducing the wiggle of thick layers, as discussed above. As a result, during optimization thick layers are more likely to be put in the middle of a streamgraph, where layers are flatted and have small distortions. But as thick layers in practice tend to have larger thickness changes than thin layers, putting them in the middle introduces more distortions to the neighboring layers and the overall visualization. For example, in Fig. 8(b), the thickest layer (6) in light green is put to the middle of the streamgraph. Its thickness changes will distort all neighboring layers.

Design Considerations for Layer Ordering. In *SineStreams*, we try to avoid thickness changes and examine layer ordering from the perspective of reducing sine illusions. We propose an ordering algorithm based on hierarchical clustering to produce a layer ordering that is better than the results of greedy neighbor swapping.

In the original streamgraph algorithm, the ordering of layers not only influences the sine illusion for individual layers (whose sum is represented by Eq. 3) but also the readability of the whole stream, which is related to the value of g'_0. As shown in Eq. 4, the solution of g'_0 can be re-formulated as the sum of differences between the slopes of every two adjacent layers $f'_i + f'_{i-1}$.

It has been reported that a flat baseline can help alleviating sine illusions for the whole stream [4, 6] and increase its aesthetic quality [31]. This observation motivates us to order layers in a way that neighboring layers mutually compensate their thickness changes. In other words, the absolute value of $f'_i + f'_{i-1}$ should be minimized to ensure an as-flat-as-possible baseline g_0. We define the *compensation degree* $comp(i,j)$ to describe the mutual compensation for every two layers f_i, f_j as:

$$comp(i,j) = \frac{1}{L}\sum_{t=0}^{T}\frac{|f'_i(t)+f'_j(t)|}{|f'_i(t)|+|f'_j(t)|} \tag{6}$$

where L indicates the length of the combined layer. We define $\frac{|f'_i(t)+f'_j(t)|}{|f'_i(t)|+|f'_j(t)|} = 0$ when $|f'_i(t)|+|f'_j(t)| = 0$.

Apart from the amount of compensation, thickness and length of layers have also to be considered when ordering them. Since the sine illusion increases with decreasing vertical distance between curves [32], it is better to merge a pair of thinner layers if two pairs of layers have similar compensation degrees. The rationale behind this is that, compared to thicker layers with the same compensation degree, thinner layers have fewer changes in their thickness, and therefore introduce fewer distortions to their neighboring layers, and thus should be given higher priority during merging.

We introduce a *thickness weight* to describe our preference for the compensation of relatively thinner layers:

$$w_{th}(i,j) = max\{f_i(t) + f_j(t) : t \in [1,m]\}$$

where $f_i(t)$ denotes the value of f_i at the time point t and m indicates the number of time points. Apart from computing the maximum, we experimented with different functions for computing the *thickness weight*, including arithmetic mean, harmonic mean and geometric mean. The maximum, however, lead to the best results in our experiments.

Meanwhile, we observed that the merging of shorter layers will create a slanted baseline for the later merging of longer layers, but not the other way round (Fig. 6(b)). Since a slanted baseline will enhance sine illusions within surrounding layers, we give priority to the merging of longer layers if two pairs of layers have similar amounts of compensation. We use a *length weight* to describe our preference for the compensation of relatively longer layers:

$$w_{len}(i,j) = max(\frac{m}{L_i}, \frac{m}{L_j})$$

where L_i is the length of layer f_i. Sometimes, a layer can be too thin to be observed. Therefore, when calculating the length of a layer, we only consider their parts with a thickness exceeding a threshold value (1/9 of the maximum thickness of each layer based on our experiments).

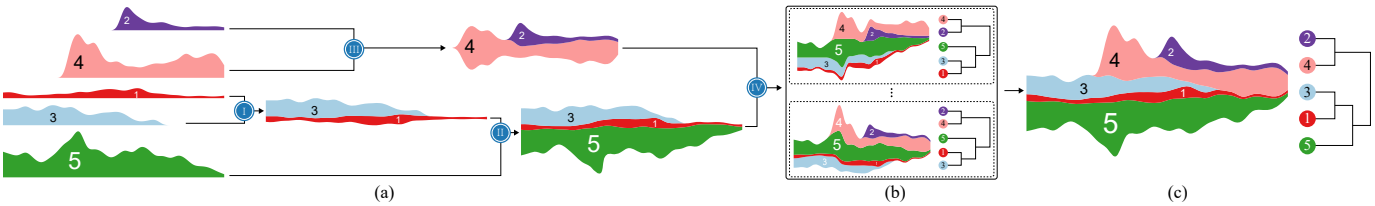
Fig. 6 gives examples about how *thickness weight* and *length weight* can help improving layer ordering. Each node represents a layer and the number of an edge indicates the distance $dist(i,j)$ between the corresponding two layers f_i, f_j. Layer ordering is now obtained by minimizing the distance between neighboring layers. By adding w_{th} to the distance function (a), the layer ordering tends to place thinner layers in the middle. By adding w_{len} to the distance function (b), layer ordering is prone to place short layers on the outside.

So far, we have translated the three design considerations, *compensation*, *thickness*, and *length*, into a quantitative form. Within our optimization we multiply the corresponding values to get a distance function that reflects the relative importance of putting two layers together.

$$dist(i,j) = comp(i,j)\cdot w_{len}(i,j)\cdot w_{th}(i,j) \tag{7}$$

The smaller $dist(i,j)$, the higher the priority that should be given to ensure that layer f_i and layer f_j are adjacent to each other.

Hierarchical-Clustering-Based Ordering. We employ a hierarchical clustering algorithm [17] for layer ordering, see Fig. 7 At each time step (numbered in the blue circle), the two layers f_i, f_j with the shortest distance are merged to obtain a new combined layer $f_k = \{f_i + f_j\}$. We then calculate the distances between this new layer f_k and all other



(a) (b) (c)

Fig. 7. Layer ordering based on hierarchical clustering: (a) we hierarchically cluster layers with shortest distances and build a binary tree. Each node of the tree is a layer. The two children of an internal node are the two layers that are put together in the ordering. (b) Flipping internal nodes leads to different layer orderings, which incorporate different amounts of sine illusions. (c) We compute our ordering from the binary tree by minimizing the sum of distances between adjacent layers.

layers, and repeat merging (see Fig. 7(a)). Once the clustering is done, a hierarchical clustering tree is formed (see Fig. 7(b)), however, multiple orderings among layers can be generated by flipping internal nodes in this tree. Some orderings, however, might result in large sine illusions, see the red layer of the top streamgraph in Fig. 7(b).

To guarantee an ordering that minimizes sine illusions, we create the final order by minimizing the sum of distances between adjacent layers:

$$\arg\min \sum_{i=1}^{n-1} dist(i, i+1) \tag{8}$$

where the i th and $i+1$th layers correspond to two adjacent leaf nodes of the hierarchical clustering tree. An leaf ordering algorithm [1] is used to quickly obtain this final order. Most layers have smaller slopes in Fig. 7(c) than in Fig. 7(b). It is worth noting that our goal is not to find the ordering that minimizes the sum of distances of adjacent layers but to enforce a compensation-based ordering.

A comparison of different ordering algorithms is illustrated in Fig. 8. Using LateOnset, layers are added to the streamgraph based on their start time. New layers (*e.g.*, Layer 2 in Pink) are usually put on a slanted baseline, which introduces distortions and sine illusion effects to these new layers. TwoOpt tends to put thick layers (Layer 6 in Light Green) in the middle, resulting in large distortions and strong sine illusion effects at the neighboring layers. Compared to LateOnset and TwoOpt, our ordering algorithm (c) leads to a visually pleasing streamgraph. Orthogonal and vertical orientations are aligned in most layers, thus sine illusions are minimized.

(a) LateOnset (b) TwoOpt (c) Ours

Fig. 8. Three streamgraphs using the same baseline optimization but different ordering algorithms: LateOnset, TwoOpt, and our algorithm.

4.3 Time Complexity

SineStream has a comparable time complexity to previous methods. The following discussion assumes that each of the n time-series has m time points, *i.e.*, the total data size is nm.

Like the baseline optimization proposed by Byron et al. [4] and Batholomeo et al. [10], our baseline optimization has a time complexity of $O(nm)$. For each of the m time points it needs $O(n)$ steps to calculate g_0' in Eq. 4.

The ordering optimization in *SineStream* consists of four parts: computing the compensation degree, length weight, and thickness weight as well as the hierarchical clustering. Calculating the first three parts requires going through all m time points, resulting in a time complexity of $O(m)$. The original hierarchical clustering proposed by Johnson [17] has a time complexity of $O(n^3)$. Therefore, the total time complexity of ordering would be $O(n^3m)$. However, the hierarchical clustering can be improved to $O(n^2)$ using the methods presented in [18, 35]. In other words, the total time complexity of the ordering can be reduced to $O(n^2m)$, but in practice n will not be a large number, typically below 20.

LateOnset has a time complexity of $O(mn + n\log n)$. Layers are first sorted based on their "on-set" time ($O(n\log n)$) and then added to the graph to balance the "weight" for summing up the whole time series ($O(mn)$). LateOnset has a lower time complexity than *SineStream*, but has an inferior performance for reducing sine illusions, see Fig. 3.

The time complexity of TwoOpt is $O(r(n+snm+nm))$, where s and r are custom parameters: s denotes the scanning time and r the time needed for iterative refinement. In practice, we set $s=n$, which leads to a time complexity of $O(rn^2m)$. The value of r depends on the data properties, therefore the optimal setting varies from dataset to dataset. In sum, our ordering algorithm is not slower than TwoOpt , while significantly reducing the sine illusion effects.

5 Evaluation

In this section, we compare *SineStream* with other existing streamgraph algorithms, first through quantitative measures and then through a controlled user study. Note that we only focus on streamgraphs in this evaluation. The comparison between streamgraphs and other multiple time-series visualizations is beyond the scope of this study. We refer the interested reader to Javed et al. [16] for more details.

5.1 Quantitative Evaluation

Experimental Design. We evaluated the effectiveness of the algorithms (LateOnset [4], TwoOpt [10], and Ours) in reducing layer distortion through three numerical metrics: the L2-norm wiggle defined in Eq. 1, the L1-norm wiggle defined in Eq. 2, and the sine illusion defined in Eq. 5. Since ordering algorithms are not specifically designed for certain baselines, we computed three baselines for each ordering algorithm and recorded the respective metric value for each baseline, generating 3 values for each ordering, that is, 3×3 combinations in total.

To achieve a comprehensive assessment, we collected 36 real world datasets from a variety of data sources:

- Agriculture statistics provided by the Food and Agriculture Organization of the United Nations [14].
- GDP and population statistics provided by the National Bureau of Statistics of China [21]
- Finance statistics provided by the World Bank [30]
- Disease statistics provided by the World Health Organization [27]
- Sports statistics provided by Sports-Reference.com [25]

For each dataset, we selected multiple time-series with a varying number of layers and time points, m ranging from 12 to 265 and n ranging from 6 to 35. For each dataset, the metric value was first calculated for each time point, then summed over time, and finally normalized to [0, 1]. We then ordered the layers in streamgraphs using the three methods LateOnset, TwoOpt, and our algorithm. Then, for each ordering, we computed baselines that optimize the L1-norm wiggle, L2-norm wiggle and the sine illusion. Since TwoOpt is not a deterministic algorithm, we calculated average metric values after running the ordering process 20 times, following the practices of the original paper [10]. In this experiment, all algorithms were implemented in Javascript using Node.js and all experiments were executed on a computer with an i5-3320 core, 8G RAM and Windows 10 operating system.

Results. Fig. 9 summarizes the results of the quantitative evaluation. Except computation time, all values were normalized to the range of $[0,1]$. The lower the values, the better the performance of an algorithm. We used a violin plot to show the distribution of the results. The white dot in the middle indicates the median value and the black bar represents the interquartile range. The width of the curved shape corresponds to the approximate frequency of data points in each region.

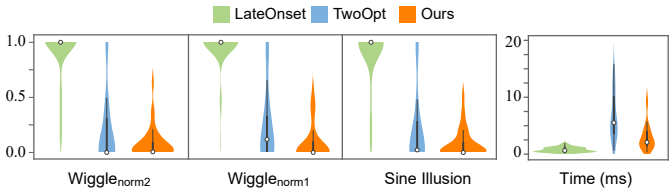


Fig. 9. Performance of three algorithms in terms of L1-norm wiggle, L2-norm wiggle and sine illusion. The violin plots summarize the value distribution for 36 datasets, smaller values are better.

The streamgraph layouts of all 36 datasets, computed with the three algorithms (LateOnset, TwoOpt and Ours), together with their respective metric values, are provided in the supplementary material.

For all three metrics, TwoOpt and *SineStream* clearly outperform LateOnset. This scenario was expected since LateOnset is designed specifically for box office revenues and does not perform well for general time series.

Our algorithm outperforms TwoOpt for all three metrics. For the L2-norm wiggle measure, the median values of our algorithm and TwoOpt are zero. For the L1-norm wiggle and the sine illusion measure, the median values of TwoOpt are slightly higher than that of our algorithms (L1-norm wiggle: 0.117 vs. 0, sine illusion: 0.023 vs. 0). In terms of value distribution, our algorithm led to a more concentrated distribution than TwoOpt for all three metrics. This is caused by the fact that TwoOpt is not a deterministic algorithm. Sometimes it produces unsatisfactory orderings that lead to large values for all three metrics.

The time complexity of the three algorithms has already been discussed in Sect. 4.3. The time complexity of LateOnset is $O(mn + n\log n)$, for TwoOpt it is $O(rn^2m)$, and for our algorithm $O(n^3m)$. We employed the default setting for the source code of TwoOpt and set $s = n, r = 5$ in this experiment. LateOnset has, as expected, the lowest computation time, but at the cost of increasing layer distortions. With a small mean value and a more concentrated distribution, our algorithm is more efficient than TwoOpt, but all three algorithms are able to compute reasonable streamgraphs within less than 20ms, allowing interactive computation rates.

5.2 User Study

We conducted a user study to compare the readability of *SineStream* with other variants of streamgraphs. The design and analysis of the user study follow the practices of assessing stacked graphs by Thudt et al. [31]. Here we report on the main findings. More information can be found in the supplementary material.

Participants & Apparatus. We recruited 24 participants (20 male and 4 female, age 18-26, μ = 23.375, δ = 1.64) through university mailing lists. All participants had no prior experience in designing streamgraphs.

Conditions. We again compared the readability of *SineStream* with the LateOnset-based streamgraphs proposed by Byron et al. [4] (called *LateOnsetStream*), and the TwoOpt-based ones by Bartolomeo et al. [10] (called *TwoOptStream*). *LateOnsetStream* is the most widely used type of streamgraph; *TwoOptStream* is considered to be the state of the art. To reduce the impact of colorization on the results, all streamgraphs used the same color scheme that was also used in [31]. The names *LateOnsetStream* and *TwoOptStream* were only used during recording and analyzing of results. In the user study, we did not use any names for streamgraphs to avoid a potential user bias. Instead, they were randomly ordered and named based on their positions such as "left/right graph".

Datasets. Following the experimental design in [31], the study was conducted on six datasets, including two real world datasets and four randomly generated datasets:

- A Call dataset [29] that contains 10 time-series over 35 time points ($n = 10, m = 30$). Each time-series represents the number of complaint calls made to the New York 311 line on a topic over 35 days. All time-series have non-zero values over the whole time period and share a similar weekly pattern.
- A Movie dataset that contains 30 time-series with 20 time points ($n = 30, m = 20$). Each time-series represents the revenue of a single movie over 20 weeks. The time-series in this dataset have non-zero values over an average of 36.35% of the time points (weeks).
- Four datasets that are randomly generated using the data generator by Byron et al. [4], where $n = 15, 20, 25, 30$ and $m = 30, 35, 35, 40$. The datasets show varying temporal patterns.

Tasks. All participants were required to complete three tasks designed by Thudt et al. [31] to assess the readability of the streamgraphs.

T_{ind} This task evaluates the ability of the participants to read the thickness of individual layers. Participants were asked to compare the thickness of two individual layers at two given time points. Compared with comparing two timepoints of the the same layer (or two layers at the same timepoint), this task is more commonly performed in streamgraphs [16, 31]. We thus selected this task to help us better assess readability.

T_{tre} This task evaluates the ability of the participants to perceive the trend of an individual layer. Given an area chart on a straight baseline, participants were asked to select a layer whose thickness is presented by the line chart.

T_{agg} This task evaluates the ability of the participants to read the thickness of the whole streamgraph. Participants were asked to compare the thickness of the whole streamgraph at two given time points.

Procedure. A within-subject design was employed for the user study. The independent variables were the streamgraph technique, the task, and the dataset and the dependent variables were the accuracy and the completion time. Each participant had to perform tasks for all independent variables, resulting in $3(technique) * 3(task) * 6(dataset) = 54$ trials. With 24 participants, this user study produced a total of 1296 trials. The three types of streamgraphs were randomly associated with the six datasets using a balanced $3 * 6$ Latin square in order to mitigate learning effects.

Before each task, participants familiarized themselves with the task and the question format through two training questions, whose correct answers were available to the participants. During each task, participants answered $3 * 6 = 18$ single choice questions, covering 3 types of streamgraphs and 6 datasets. Each question for T_{ind} and T_{agg} had two options. For T_{tre}, the participants answered a question by clicking the corresponding layer in the streamgraph. Participants were not allowed to use any measuring device and completed all tasks only based on their observation with the naked eye. After answering a question, participants had to move their mouses and click a "next" button to move to the next question. The "next" button was designed to ensure that the position of the cursor was reset for each question, reducing the possibility that the participants to just repeat the answer of the last question. We recorded the completion time and the participants' answer for each question.

After the tasks, we asked participants to score the three streamgraphs based on their aesthetic preference and perceived readability using a 7 point seven point likert scale. The user study took around 25 minutes for each participant.

Hypothesis. We have the following hypothesis:

H1 For the accuracy in readability tasks (T_{ind} & T_{tre} & T_{agg}), *SineStream* leads to higher accuracy than the other two streamgraph variants.

H2 For the completion time in readability tasks (T_{ind} & T_{tre} & T_{agg}), *SineStream* cost less time than the other two streamgraph variants.

We formulated H2 & H1 because *SineStream*, with a reduced effect of sine illusion, should facilitate participants in reading the thickness of layers, thus reducing the completion time and improving the accuracy.

Results & Analysis. The analysis follows the practices in [31] and is based on effect sizes with bootstrapped confidence intervals. As the limits of null hypothesis significance testing (NHST) are concerned in various studies (summarized in [12]), this approach is an alternative to NHST and is recommended by the American Psychological

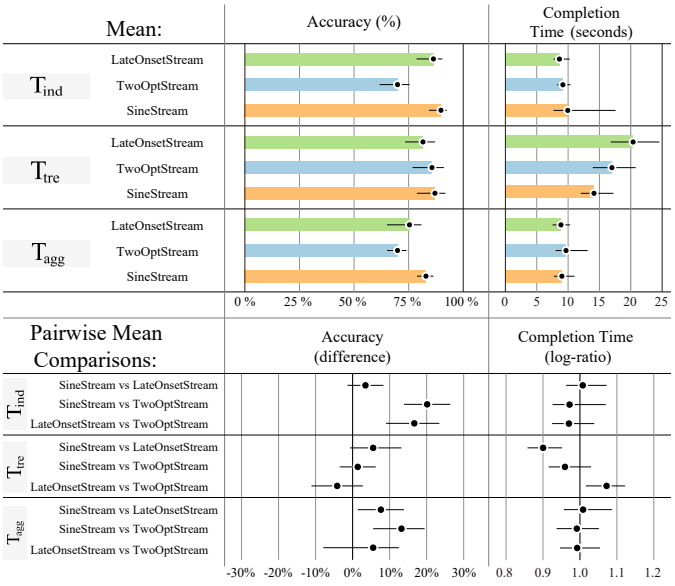


Fig. 10. Mean comparison and pairwise comparison in terms of accuracy and completion time. Error bars show 95% bootstrapped confidence intervals.

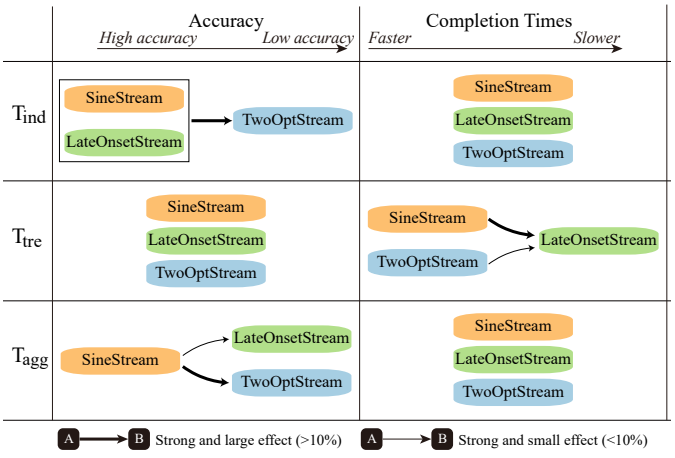


Fig. 11. Summary of differences among the three variants of streamgraphs in accuracy and completion time. Within each cell, the horizontal position of the icons encode their performance, ordered from best (left) to worst (right). The same horizontal position indicates a similar performance.

Association.

The readability for each type of streamgraphs is evaluated based on accuracy and completion time. Fig. 10 shows the mean comparison and the pairwise comparison. The pairwise comparisons for A vs. B are computed by $A-B$ for accuracy and $log(A)/log(B)$ for completion time. When performing pairwise comparisons, the values are computed for each participant. In terms of accuracy, *SineStream* has the highest mean accuracy in all three tasks. Among the three tasks, T_{tre} has the highest mean accuracy, indicating the task is relatively easy. In terms of completion time, all three streamgraphs show similar times except at T_{tre}, where *SineStream* $<$ *TwoOptStream* $<$ *LateOnsetStream*. Among the three tasks, T_{tre} has the longest completion time. This is expected as the questions in T_{tre} have more options (n options) than the questions for T_{ind} and T_{agg} (2 options).

We interprete the pair-wise difference to be small if it is less than 10% and to be large if it is greater than 10%; the difference is treated as weak if the confidence interval crosses the 0/1 vertical line and otherwise is treated as strong. Fig. 11 summarizes our findings, the thickness of the lines to represent whether the effect was small or large.

For the readability task (H1), on average, *SineStream* outperformes *LateOnsetStream* and *TwoOptStream*. The advantage is most obvious

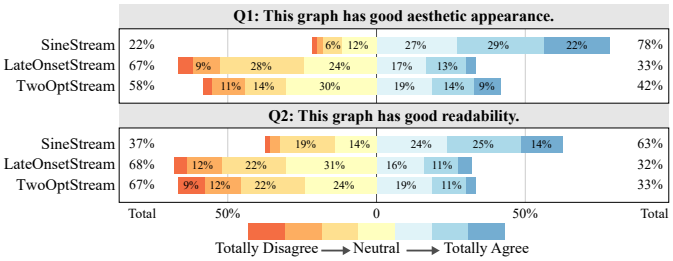


Fig. 12. Comparison of aesthetics and readability for three streamgraph versions from a post-study questionnaire (using a 7-point Likert scale). Agreeing scores are placed on the right, disagreeing on the left.

for T_{agg}. For T_{ind}, *SineStream* and *LateOnsetStream* have higher accuracy than *TwoOptStream*,with a strong and large effect, which is surprising. *TwoOptStream*, as the current state-of-the-art and an improvement of *LateOnsetStream*, actually has a lower accuracy than *LateOnsetStream* in representing the true thickness of individual layers (T_{ind}). We believe this is because the L2-norm-based optimization in *LateOnsetStream* also reduces the effect of sine illusions in each layer. For T_{tre}, all three types of streamgraphs have a similar accuracy. We suspect this is caused by the easiness of the task, which has the highest mean accuracy among three readability tasks. For T_{agg}, *SineStream* has higher accuracy than *LateOnsetStream* (strong the small effect) and *TwoOptStream* (strong and large effect). This is expected since we flatten g_0 to improve the readability of the whole streamgraph during layer ordering.

Another aspect is the completion time for the readability tasks (H2). Overall, the three streamgraphs require similar completion times, except at T_{tre}, where *LateOnsetStream* requires more time. For T_{ind} and T_{agg}, the similar completion time among the three streamgraph versions might indicate that users are equally certain about their answers across different versions, even though their answers have low accuracy for a certain version. For T_{tre}, *SineStream* (with a strong and large effect) and *TwoOptStream* (with a strong and small effect) require less completion time than *LateOnsetStream*. We suspect the advantage of *SineStream* and *TwoOptStream* is caused by their ordering algorithms, which work well for general time-series in conrast to the specially-tailored ordering algorithm in *LateOnsetStream*.

Self-reported preferences of the participants for aesthetics and readability are presented in Fig. 12. Overall, the participants reported that *SineStream* is more aesthetically pleasing and readable than *TwoOptStream* and *LateOnsetStream*. Participants also said that *TwoOptStream* is more aesthetically pleasing than *LateOnsetStream*. We suspect this is caused by the unpleasant distortions that appear in *LateOnsetStream* when layers have sudden jumps. There is no significant difference between the self-reported readability of *LateOnsetStream* and *TwoOptStream*. In general, the self-reported readability of the three variants agrees with the results of the tasks.

5.3 Case Study

To further demonstrate the effectiveness of *SineStream*, we conducted a case study and compared it with *LateOnsetStream* and *TwoOptStream* for four real world datasets (Fig. 1 and Fig. 13 illustrate the comparison):

- Call dataset: the number of complain calls made to New York 311 line about 10 topics over 35 days ($n = 10, m = 35$) [29].
- Bank Interest dataset: the bank interest rate of 10 developing countries from 1972 to 2019 ($n = 10, m = 48$) [30].
- Assistance dataset: Net official development assistance received by 11 countries over 59 years ($n = 11, m = 59$) [30].
- COVID dataset: the top 10 number of daily confirmed COVID-19 cases per country from Mar 1st to Apr 13th ($n = 10, m = 34$) [23].

LateOnsetStream orders the layers to balance the weight (*i.e.*, the sum of time series) between the top and bottom and the baseline is calculated to minimize the weighted L2-norm wiggle value (Eq. 1). Since $w_i = f_i$ in the weighted wiggle measure, *LateOnsetStream* mainly reduces the distortions in thick layers, *e.g.*, the "Street light condition" layer in the Call dataset, the "UKR" layer in the Bank Interest dataset,

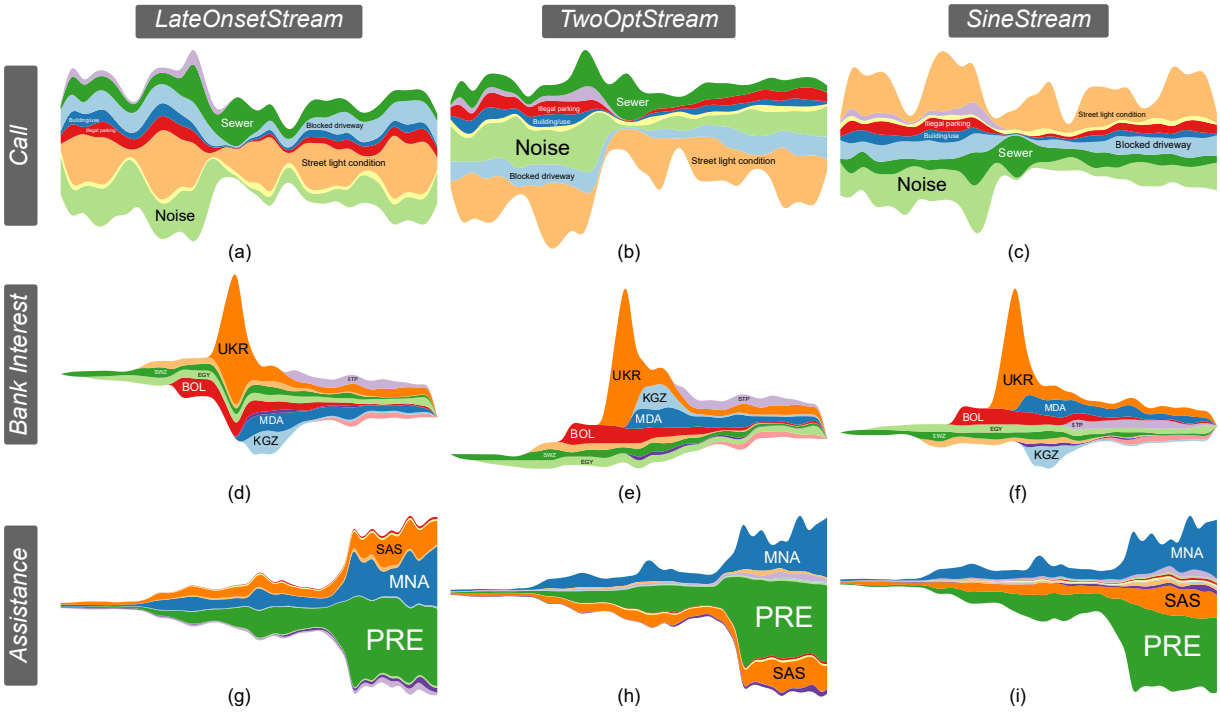


Fig. 13. Comparing *LateOnsetStream* [4], *TwoOptStream* [10], and *SineStream* for the Call dataset [29] (top), the Bank Interest dataset [30] (middle) and the Assistance dataset [30] (bottom). The streamgraphs are drawn using different baselines and layer orderings.

the "PRE" and "MNA" layers in the Assistance dataset. Thus, the thickness of these thick layers can be read accurately. The thinner layers, however, suffer from strong effects of sine illusion, which are caused by the large thickness changes of the thick layers. These thin layers include the "Blocked driveway" layer in Call dataset, the "BOL" and "EGY" layer in the Interest dataset, and the "SAS" layer in the Assistance dataset. The top and bottom boundaries of these thin layers have the same sign, thus misaligning the orthogonal and vertical orientations and making it difficult to visually estimate the actual thickness.

TwoOptStream, employs a heuristic algorithm, "TwoOpt", to order layers; the baseline is calculated to minimize the weighted L1-norm wiggle value (Eq. 2). As discussed in Sect. 4.2, "TwoOpt" tends to place thick layers in the middle, such as the "Noise" layer in Call dataset and the "PRE" layer in the Assistance dataset. The thickness changes of these thick layers will inevitably distort their neighboring layers. Meanwhile, the L1-norm-based baseline optimization in *TwoOptStream* prefers one big distortion over several small distortions. As a result, the distortion caused by the middle thick layers is mainly imposed to layers on one side and causes strong sine illusions. For example, in the Call dataset, the rapid thickness decrease in "Noise" introduces huge distortions to the layers below, *i.e.*, the "Blocked driveway" and "Street light condition" layer.

Compared to *LateOnsetStream* and *TwoOptStream*, *SineStream* minimizes the effect of sine illusion and improves the readability through an improved baseline and ordering algorithms. In all four datasets, *SineStream* decreaes sine illusion effects for thin layers without significantly increasing such illusion for the thick layers. For example, in the Assistance dataset, *SineStream* significantly improves the readability of the "SAS" layer and the purple layer by flatting their baselines. The "PRE" layer is slightly distorted in *SineStream*, but the degree of this distortion is much smaller than the distortion of "SAS". Furthermore, the thick layers are more resistant to the sine illusion. This improvement is mainly cause by two factors: Compared to *LateOnsetStream*, *SineStream* introduces a Gaussian weight for the baseline computation and penalizes layers with large thickness changes. Compared to *TwoOptStream*, *SineStream* introduces a *thickness weight* in the ordering and tends to place thin layers in the middle. Moreover, our efforts to minimize $\sum_{i=1}^{n-1} dist(i, i+1)$ let the slope of the baseline g'_0 (Eq. 4) be closer to zero than the other two methods. For example, the *SineStream* result for the Call dataset is more flat than *TwoOptStream*; for the Interest dataset, *SineStream* is more flat than both *LateOnsetStream* and *TwoOptStream*. This allows an easier comprehension of the thickness of the whole graph.

6 Conclusion & Future Work

In this paper, we relate the optimization of streamgraph to the sine illusion and propose *SineStream*, a new version of streamgraphs with improved readability. The sine illusion provides a cognitive foundation for quantifying the readability of a streamgraph, *i.e.*, how easy users can accurately read the layer thicknesses. Therefore, we re-formulate baseline computation and layer ordering in terms of minimizing the sine illusion effect. Apart from individual layers, *SineStream* also considers sine illusions for the whole stream when ordering layers. A quantitative comparison and a user study demonstrate that *SineStream* effectively improves the readability and aesthetics of streamgraphs compared to the state-of-the-art methods.

In the future, we plan to conduct large-scale user studies and a comprehensive evaluation to better understand *SineStream* and sine illusion effects in streamgraphs. *SineStream* models sine illusion using Eq. 5 and achieves good results in improving readability. However, how to precisely quantify the effect of sine illusion is still an open question. VanderPlas and Hofmann [32] conclude that the sine illusion effect increases with the slope of curves but provided no further details, *e.g.*, whether the relationship is linear or nonlinear. Cognitive studies are needed to help us better understand and capture the relationships between sine illusion effects and other visual variables in streamgraphs.

Acknowledgments

This work is supported by the grants of the NSFC (61772315, 61861136012) and Deutsche Forschungsgemeinschaft (DFG) Project-IDs DE 620/26-1, as well as 251654672 (TRR 161 Quantitative methods for visual computing).

REFERENCES

[1] Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17(suppl_1):S22–S29, 06 2001. doi: 10.1093/bioinformatics/17.suppl_1.S22

[2] D. Baur, B. Lee, and S. Carpendale. Touchwave: Kinetic multi-touch manipulation for hierarchical stacked graphs. In *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '12, p. 255–264, 2012. doi: 10.1145/2396636.2396675

[3] L. Byron. Last. fm listening history–what have i been listening to, 2008.

[4] L. Byron and M. Wattenberg. Stacked graphs – geometry aesthetics. *IEEE Trans. Vis. & Comp. Graphics*, 14(6):1245–1252, Nov 2008. doi: 10.1109/TVCG.2008.166

[5] W. S. Cleveland, M. E. McGill, and R. McGill. The shape parameter of a two-variable graph. *Journal of the American Statistical Association*, 83(402):289–300, 1988. doi: 10.1080/01621459.1988.10478598

[6] W. S. Cleveland and R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 79(387):531–554, 1984. doi: 10.1080/01621459.1984.10478080

[7] 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 Trans. Vis. & Comp. Graphics*, 17(12):2412–2421, Dec 2011. doi: 10.1109/TVCG.2011.239

[8] W. Cui, S. Liu, Z. Wu, and H. Wei. How hierarchical topics evolve in large text corpora. *IEEE Trans. Vis. & Comp. Graphics*, 20(12):2281–2290, Dec 2014. doi: 10.1109/TVCG.2014.2346433

[9] R. H. Day and E. J. Stecher. Sine of an illusion. *Perception*, 20(1):49–55, 1991. doi: 10.1068/p200049

[10] M. Di Bartolomeo and Y. Hu. There is more to streamgraphs than movies: Better aesthetics via ordering and lassoing. *Computer Graphics Forum*, 35(3):341–350, 2016. doi: 10.1111/cgf.12910

[11] W. Dou, L. Yu, X. Wang, Z. Ma, and W. Ribarsky. Hierarchicaltopics: Visually exploring large text collections using topic hierarchies. *IEEE Trans. Vis. & Comp. Graphics*, 19(12):2002–2011, Dec 2013. doi: 10.1109/TVCG.2013.162

[12] P. Dragicevic. *Fair Statistical Communication in HCI*, pp. 291–330. Springer International Publishing, Cham, 2016. doi: 10.1007/978-3-319-26633-6_13

[13] M. Dörk, D. Gruen, C. Williamson, and S. Carpendale. A visual backchannel for large-scale events. *IEEE Trans. Vis. & Comp. Graphics*, 16(6):1129–1138, Nov 2010. doi: 10.1109/TVCG.2010.129

[14] Food and Agriculture Organization of the United Nations. Fao stat. `http://www.fao.org/`. Accessed: 2020-03-20.

[15] S. Havre, E. Hetzler, P. Whitney, and L. Nowell. Themeriver: visualizing thematic changes in large document collections. *IEEE Trans. Vis. & Comp. Graphics*, 8(1):9–20, Jan 2002. doi: 10.1109/2945.981848

[16] W. Javed, B. McDonnel, and N. Elmqvist. Graphical perception of multiple time series. *IEEE Trans. Vis. & Comp. Graphics*, 16(6):927–934, Nov 2010. doi: 10.1109/TVCG.2010.162

[17] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, Sep 1967. doi: 10.1007/BF02289588

[18] G. Karypis, Eui-Hong Han, and V. Kumar. Chameleon: hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, Aug 1999. doi: 10.1109/2.781637

[19] A. Kirk. Making sense of streamgraphs. `www.visualisingdata.com/2010/08/making-sense-of-streamgraphs/`, 2010. Accessed: 2020-01-30.

[20] 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), Feb. 2012. doi: 10.1145/2089094.2089101

[21] National Bureau of Statistics of China. National data. `http://data.stats.gov.cn/`. Accessed: 2020-03-20.

[22] W. Playfair. The commercial and political atlas (london). 1786.

[23] R. Pombo. time-series of coronavirus cases. `https://github.com/pomber/covid19`. Accessed: 2020-04-06.

[24] Z. Pousman, J. Stasko, and M. Mateas. Casual information visualization: Depictions of data in everyday life. *IEEE Trans. Vis. & Comp. Graphics*, 13(6):1145–1152, Nov 2007. doi: 10.1109/TVCG.2007.70541

[25] S. References. Sports references. `https://www.sports-reference.com/`. Accessed: 2020-03-20.

[26] G. Sun, Y. Wu, S. Liu, T. Peng, J. J. H. Zhu, and R. Liang. Evoriver: Visual analysis of topic coopetition on social media. *IEEE Trans. Vis. & Comp. Graphics*, 20(12):1753–1762, Dec 2014. doi: 10.1109/TVCG.2014.2346919

[27] The New York Times. Disease. `https://www.who.int/emergencies/diseases/en/`. Accessed: 2020-03-20.

[28] The New York Times. The ebb and flow of movies: Box office receipts 1986 — 2008. `http://archive.nytimes.com/www.nytimes.com/interactive/2008/02/23/movies/20080223_REVENUE_GRAPHIC.html`, 2008. Accessed: 2020-01-30.

[29] The New York Times. Nyc open data. `https://nycopendata.socrata.com/`, 2008. Accessed: 2020-01-30.

[30] The World Bank. Global development data. `https://data.worldbank.org.cn`. Accessed: 2020-03-20.

[31] A. Thudt, J. Walny, C. Perin, F. Rajabiyazdi, L. MacDonald, R. Vardeleon, S. Greenberg, and S. Carpendale. Assessing the readability of stacked graphs. In *Proceedings of Graphics Interface*, pp. 167–174, 2016. doi: 10.20380/GI2016.21

[32] S. VanderPlas and H. Hofmann. Signs of the sine illusion—why we need to care. *Journal of Computational and Graphical Statistics*, 24(4):1170–1190, 2015. doi: 10.1080/10618600.2014.951547

[33] L. Wattenberg. Babyname wizard: Namevoyager. *Online: `http://www.babynamewizard.com/voyager`*, 2005.

[34] P. Xu, Y. Wu, E. Wei, T. Peng, S. Liu, J. J. H. Zhu, and H. Qu. Visual analysis of topic competition on social media. *IEEE Trans. Vis. & Comp. Graphics*, 19(12):2012–2021, Dec 2013. doi: 10.1109/TVCG.2013.221

[35] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. *SIGMOD Rec.*, 25(2):103–114, June 1996. doi: 10.1145/235968.233324