

Chapter 5

Pixel-Oriented Network Visualization: Static Visualization of Change in Social Networks

Klaus Stein, René Wegener, and Christoph Schlieder

Abstract Most common network visualizations rely on graph drawing. While without doubt useful, graphs suffer from limitations like cluttering and important patterns may not be realized especially when networks change over time. We propose a novel approach for the visualization of user interactions in social networks: a pixel-oriented visualization of a graphical network matrix where activity timelines are folded to inner glyphs within each matrix cell. Users are ordered by similarity which allows to uncover interesting patterns. The visualization is exemplified using social networks based on corporate wikis.

5.1 Introduction

One important aspect of social network analysis consists in finding interaction patterns between social actors by appropriate visualization paradigms.¹ Social network visualizations offer great help in getting deeper insights in structure and relations. Most commonly graphs are used to represent social networks giving an overview especially over smaller networks, but normally do not cover network dynamics. Too little attention has been paid to networks changing over time: new users come in, old ones leave, new links between users get established, interaction between certain users declines, etc.

¹This chapter is based on a conference paper at ASONAM 2010.

K. Stein (✉) · C. Schlieder
Computing in the Cultural Sciences, University of Bamberg, Bamberg, Germany
e-mail: klaus.stein@uni-bamberg.de; christoph.schlieder@uni-bamberg.de

R. Wegener
Information Systems, Kassel University, Kassel, Germany
e-mail: wegener@uni-kassel.de

Techniques like animated graphs give some insights into network dynamics but also inherit several shortcomings: graphs tend to clutter when larger networks are being displayed, and animation requires digital media as well as more cognitive resources compared to static visualizations. A static visualization technique able to display dynamics even in large social networks is needed.

The visualization problem addressed in this chapter first arised when we analyzed user collaboration in organizational wikis where we missed a good visualization for temporal dynamics which could be used in the context of visual data mining. The social networks studied are extended coauthor networks extracted from organizational wikis using the interlocking measure.² The idea of interlocking is simple: if an user *B* edits a page previously edited by another user *A*, a directed link from *B* to *A* is established. An user *C* editing the same page afterwards establishes links to *A* and *B* and so on. Each link is associated with a time stamp, so the network holds the interaction record of users in time.

The analysis of interlocking networks can be applied to many different data sets from SVN or GIT repositories over email corpuses to newsgroups, discussion boards, CMS. Interlocking gives a directed network, as each user action is considered as an “answer” to actions of other users before. The visualizations presented in this chapter are nevertheless also useful on undirected graphs, as long as time stamps of interaction are available.

This chapter describes the following contributions to the state of the art: (1) We introduce a new kind of social network visualization based on the pixel-oriented visualization paradigm and extend it for the presentation of networks evolving in time in a way that supports visual data mining; (2) We compare different glyph layout patterns and their application to the problem domain; (3) We provide measures for arranging (sorting) nodes; and (4) We show that temporal patterns indicating cooccurrence and similar behavior are perceptually salient in our visualization even on larger networks.

The chapter is organized as follows. We discuss common visualization approaches for social networks and time series in Sect. 5.2. Section 5.3 introduces our pixel-oriented network visualization approach and describes the adequate choices of glyph layouts and node arrangement. Section 5.4 presents a case study that illustrates how to apply our approach to real-world network data. We conclude in Sect. 5.5 with a discussion of the results and an outlook on future work.

5.2 Related Work

Visual data mining techniques take advantage of the efficient perceptual grouping processes of the human visual system (see e.g. [7, 26]). Even in large data sets, perceptual saliency draws the observer’s attentions to patterns. Visualization is most

²See [32] for a detailed discussion of this measure.

useful to generate hypothesis about regularities in a data set. On the other hand visualization does not provide a proof. For instance nodes positioned close together by a layout algorithm suggest some correlation which vanishes by using another layout.

Shneiderman [29] gives an overview on visualization of large datasets, a detailed description can be found in [33].

5.2.1 Social Network Graphs

Much research has been conducted in the field of social network visualization (for an overview see [8, 20]). For a discussion of the explanatory power of network visualization see [6]. The most common way to present a social network is the network graph. The nodes are arranged by one of various graph layout algorithms which continue to be improved in their computational properties as well as their usability (see e.g. [11]). Most software packages for network analysis include graph drawing functionality. Furthermore, a variety of specialized graph drawing packages are available.³

Alternatively networks can be presented by adjacency matrices that represent the network by some kind of numbers for actors and relations. Ghoniem et al. [13] compare node-link diagrams and matrices. While probably less appealing to the user, matrix visualizations avoid common problems like cluttering. The authors conclude that node-link representations are best suited for smaller graphs while a higher number of nodes and higher degree of density are better visualized as matrix representations. Shneiderman and Aris [30] state that network graphs can be understood best if they contain between 10–50 nodes and 20–100 links. A higher number of nodes and links makes it more difficult to follow the links, count or identify nodes etc.

One way to avoid cluttering is to group related nodes. For example Shi et al. [28] and Balzer and Deussen [3] use hierarchical clustering, Peng and SiKun [25] propose a subgroup analysis layout algorithm based on attributes and Chen et al. [9] create subgraphs for subgroups. Leung and Carmichel [22] also group nodes and arrange them in a rectangular grid using horizontal edges to show relations between different actors.

Henry et al. [15] take a different approach. They combine node-link layouts and matrix representations in order to give an easy to understand overview of a

³For example UCINET (<http://www.analytictech.com/ucinet/>), JUNG (<http://jung.sourceforge.net/>), Graphviz (<http://www.graphviz.org/>), GUESS (<http://graphexploration.cond.org/>), Pajek (<http://pajek.imfm.si/>), Visone (<http://visone.info/>), and others. See also http://www.google.com/Top/Science/Math/Combinatorics/Software/Graph_Drawing/, <http://www.graphdrawing.org/> and the INSNA software list (<http://www.insna.org/software/>).

network while also revealing details that couldn't be recognized in a pure node-link visualization (see also [27]).

5.2.2 *Change in Time*

For visualizing time-oriented data a variety of methods is available, for an overview see [1]. Broadly speaking, the methods for visualizing time oriented data in social networks fall into two classes: sequences of snapshots and (interactive) animations. Moody [24] considers a third class: network summary statistics plotted as a line graph over time. Since the network topology cannot be recovered from this visualization, we will not consider it.

Two major problems arise with using a temporal sequence of snapshots as visualization: (1) The temporal resolution, i.e., the number of elements of the sequence, is severely restricted, (2) The optimization of the layout algorithm conflicts with changes. We illustrate the problems of visualizations using sequences of snapshots with a network from our own data.

The network presented in Fig. 5.1 shows students working on a larger project across three terms using a wiki as documentation platform. So in each term a bunch of new students join, others leave. The visualization uses a spring embedding layout algorithm that optimizes the length of the edges between the nodes (see [16]) in order to achieve a grouping of highly interconnected sets of nodes. Figure 5.1b–d show snapshots of the network at different times that illustrate the temporal change. While for each subgraph both spatial dimensions are used to lay out the graph on the big scale, the x-axis represents time from left (past) to right (future). The spatial extension of each “data point” (network graph) restricts the temporal resolution to few (three) time points.

We could not layout each of the graphs with spring embedding but had to keep each node at a fixed position to be able to compare the graphs. So while the change shows up very well in Fig. 5.1b–d the grouping in the single graphs is not optimal.

Moody states that the “poor job” of representing change in the network is a problem “fundamental to the media” and suggests to use animations. Network animation software is available either stand alone (e.g. SoNIA,⁴ see [4]) or as part of dedicated network analysis software (e.g. SONIVIS⁵) as most of the network visualization libraries support dynamic node and edge change. By interpolating between the graphs that represent the different intervals of the network, nodes are moved to smoothly re-layout the graph responding to changing edges. One drawback here is that this representation cannot be published in print, some kind

⁴<http://www.stanford.edu/group/sonia/>

⁵<http://sonivis.org>

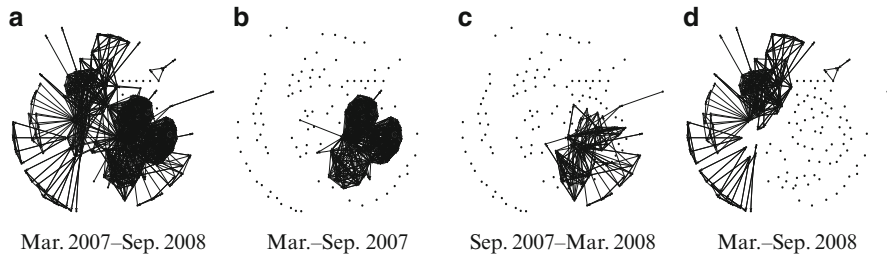


Fig. 5.1 Network evolution in time (*students wiki*). (a) Full timespan. (b) First term. (c) Second term. (d) Third term

of digital medium is needed.⁶ More important, animations are nice and striking in presentations but less useful for analysis as they do not give an overview at a glance. Animations inherit the issues of change blindness which means that some changes (maybe just because the glimpse of an eye) will not be realized by the user [7].

The static graphical presentation of data allows to externalize concepts as everything is available on paper. The visual system provides the data analyst with scanning routines that permit a very efficient processing of externalized data. Presenting all the information on one single figure instead of using an animation takes advantage of this ability. In fact, switching our attention from one area of a single figure to another will usually not only be much faster than finding the right interval of an animation with the help of a slide bar [34], it will also allow to for deeper inspection. Human’s visual working memory only stores a handful of objects at a time [34] and watching animations forces us to *remember* what we have seen while a static representation keeps everything accessible in parallel.

5.3 Pixel-Oriented Visualization of Network Data

The idea of pixel-oriented visualization was introduced by Keim et al. [18] and further developed in [17, 19] and other publications. Although the original publications do not provide a concise definition, the basic idea of the visualization methods is simple to describe. Each pixel of the screen is used to visualize one data point, representing its value by its color. This allows to visualize great amounts of data while avoiding overlapping and cluttering. Pixel-oriented visualization has been applied by Guo et al. [14] for the visualization of very large scale network matrices (BOSAM,⁷) but to the best of our knowledge it has not been tried on temporal and weighted networks.

⁶For an animated representation of the network shown in Fig. 5.1 see http://www.kinf.wiai.uni-bamberg.de/mwstat/examples/wiki_1_weekly_network.swf.

⁷Bitmap of sorted adjacency matrix.

In this section we show, how to adapt the idea of pixel-oriented visualization for a static representation of social network data in time.

5.3.1 Collaboration in Time

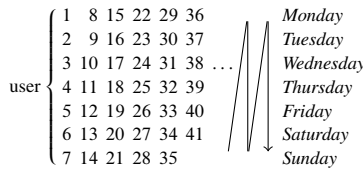
Figure 5.2a shows typical time series data in graph representation: the x-axis represents time as the independent variable, while the dependent variable (interaction activity) is displayed on the y-axis. Different colors (levels of gray) are used to distinguish the different users. Figure 5.2c gives exactly the same data in pixel-oriented representation. The x-axis again represents time, the y-axis now is used to separate the users line by line. Color (grayscale) represents interaction intensity.

The advantage of the graph-based representation consists in a visually salient rendering of extreme values (intensity spikes around June 07, August 08 and April 09) but makes it hard to distinguish the different users (Fig. 5.2a) as their graphs overlap.

In the pixel-oriented representation the activity of different users is clearly visible as there is no overlapping. The user timelines are easy to compare (we see in Fig. 5.2c which users were active in August 08 and which in April 09), but we do not get the absolute intensity values, only the relative intensity distribution is visible.

Figure 5.2b, d present the same data in higher resolution, i.e. day by day instead of month by month following the idea that “to map each data value to a colored pixel [...] allow[s] us to visualize the largest amount of data which is possible on current displays” [17]. The pixel-oriented visualization delivers additional implications: As you can see high collaboration values in 1 month are usually generated by the high collaboration rates of only a few days and not by continuing collaboration during this month.

Figure 5.2d uses the calendar metaphor which means to arrange the timeline in a weekly zigzag:



Although the single dots are too fine to be able to tell which day exactly it represents we nevertheless get the interesting patterns. We not only get an impression which users collaborate a lot at which time during the year (as every pixel column represents a week), we additionally see⁸ that only users U6 and U1 are working on Sunday while no one works on Saturday.

⁸At least I hope this is visible in the printout.

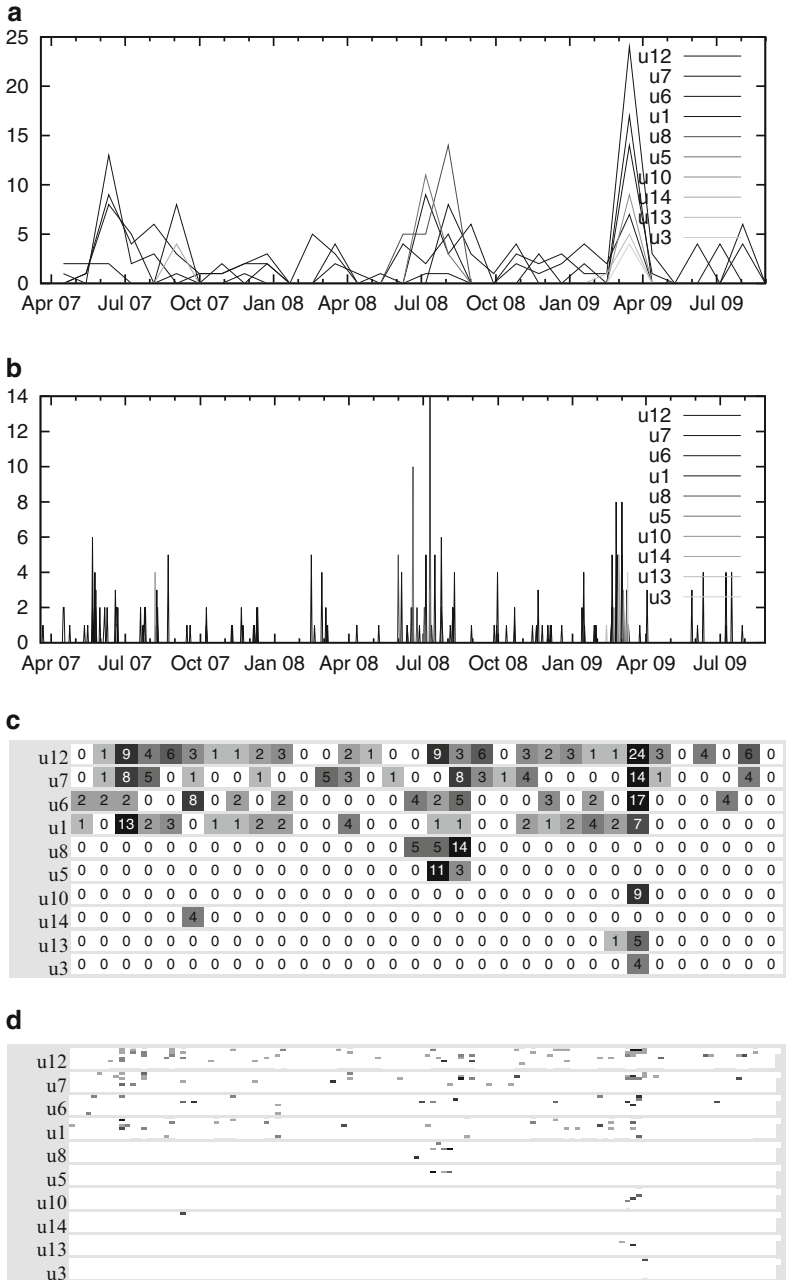


Fig. 5.2 Collaboration activity per user (*workgroup wiki*). All figures show the same timespan from April 2007 to September 2009 (x-axis), only the resolution is different. (a) Monthly. (b) Daily. (c) Monthly. (d) Daily

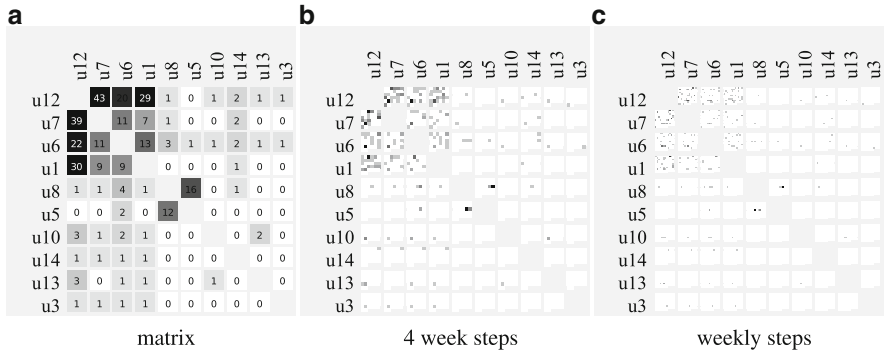


Fig. 5.3 *Workgroup wiki* network matrix. (a) Shows the adjacency matrix of the network. Each matrix element is colored according to its value. In (b) and (c) each glyph shows the folded timeline row by row. The nodes are sorted by their (weighted) degree

5.3.2 The Pixel Matrix

We adapt the idea of pixel-oriented visualization applying it to social network graphs. We present the network as an adjacency matrix with each row and each column corresponding to one node and the matrix elements giving the weight of the edge between the corresponding nodes, as shown in Fig. 5.3a.

Following the pixel-oriented visualization paradigm to present each value by one colored pixel we can inject the whole timeline of user interaction in one matrix element by folding it. Figure 5.3b gives the network matrix with each matrix element showing one glyph which holds the timeline folded row by row, each pixel representing the user-user-interaction within 4 weeks. This is somehow the inverse representation to Fig. 5.1 where time is given on the outer x-axis while now time is folded at the inside, i.e. within each glyph.

The scale is changed from monthly to 4 weeks to get timespans of equal length. While monthly data is easier to present and describe it has the disadvantage that each month covers a different *number* of days and a different number of certain weekdays: June 2010 has four Fridays, July has five. This is important when work is organized weekly, e.g. the workgroup has a weekly meeting on Friday generating extra social interaction. On a monthly raster this generates 20% more activity for July while everything is even. The other way around if there is some monthly event the 4-weeks raster would show irritating data.

We see users U12, U7, U6 and U1 interacting a lot with each other over the whole timespan. Users U8 and U5 come in after around 1 year for 3 months, working closely together with small interaction with others (U8 has few connections to the most prominent four users while U5 only meets U6). And users U10, U13 and U3 join the network even later, interacting mainly with the prominent four and not with each other.

This small example gives first insight on the potential of pixel-oriented network visualization (PONV). We do not know exactly at which time users U8 and U5

came in and we do not get a detailed analysis on the interaction between users U12 to U1, but we see gray dots sprinkled over the whole glyph telling us that there was interaction between these users all the time. And we see that U10, U13 and U3 came in at the same time even though their timelines are not layed out side by side. Our visual perception is perfect in detecting that these gray dots are on the same position within their glyph. While we are not able to read absolute data from the graph (neither exact times nor exact intensity values of collaboration) we get a good overview over the temporal behavior of each user as well as temporal cooccurrence of certain events.

And this is exactly what visual data mining is meant for: an exploratory analysis of the data to detect regularities. In a next step, all evidence in the data set is evaluated that supports the hypothetical regularity or conflicts with it. For instance, it could be checked at what time U5 did his first edit operation, at what time U8 her last edit operation and so on. In other words, the cues from visual examination are backed up with hard data.

An additional remark about U14. As we see in the U14 *row* he comes in rather early, interacts once and is gone, but in the U14 *column* there are several interactions visible even at later times. This is a result of the network creation method we used. Interlocking response graphs are directed and an edge from user *B* to user *A* is set at the time *B* edits a page *A* edited before. And in the network matrix as presented in Fig. 5.3 the row gives the tail node (*R*) and the column gives the head node (*C*), so the edge points from *R* to *C*. So what we see here is that U14 edited a page users U12 to U1 edited before, and months later users U12 to U8 “responded” by editing this page. And we can also see some closer interaction between U6 and U14 as the (U6,U14)-glyph shows us U6 must have edited this page shortly before U14.

5.3.3 Inner Glyphs

Our visual perception is good in pattern recognition and edge detection. Looking at Fig. 5.3b we detect some interesting vertical bars at (U6,U12), (U6,U1) and (U7,U12). Unfortunately these findings are arbitrary. The timeline is packed into the glyph row by row (see Fig. 5.4a) and each glyph has 6×6 pixel. This means that two pixel aligned vertically contain data that is 24 weeks apart. Looking at the same data with glyphs of width 7 would reveal other arbitrary vertical bars. On the other hand we may miss timespans of high activity which start at the end of one column and end at the beginning of the next one. The latter problem can be reduced by laying out the rows in a snake-like way (see Fig. 5.4b) as here continuous timespans are not ripped apart.

Things are different in the example before (Fig. 5.2d).⁹ Each column represents 1 week with Monday in the top and Sunday in the bottom row which allowed us to

⁹Where the timeline was presented column by column.

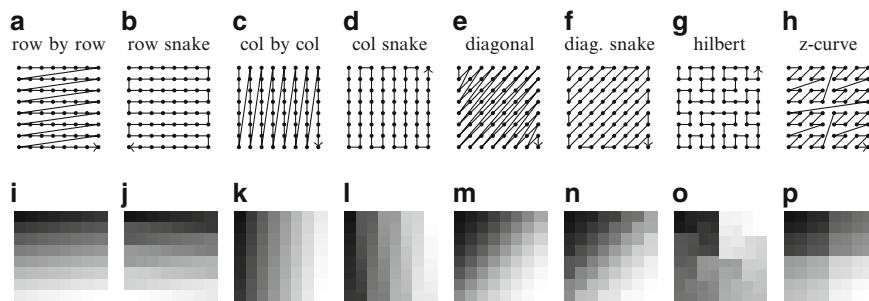


Fig. 5.4 Layout patterns for inner glyphs. (a) to (h) give the layout path while (i) to (p) show example glyphs for a color gradient from black to white in the corresponding layout. (See also [17, 19])

see who is working on weekends, and obviously a snake pattern (Fig. 5.4d) would disturb.

While Fig. 5.3b kind of allows to distinguish single pixels and – with some effort – even to see which row a pixel belongs to we are lost in Fig. 5.3c with a weekly resolution. There are darker and lighter regions and you can hardly see how many rows are involved, in other words: you see darker and lighter region patterns which may be incidental as our vision groups events which are layed out next to each other in successive rows while in fact being numbers of weeks apart.

Space-filling curves like the well-known Hilbert curve (Fig. 5.4g) are one answer to this problem as these curves lay out one-dimensional data in two-dimensional space in a locality-preserving way, i.e. data points being close together in the one-dimensional representation are kept close in the two-dimensional layout. Unfortunately it also brings data points close which were *very* far away from each other as visible in Fig. 5.4o.

A second disadvantage of this layout is that the main data order is non-linear. While the layouts Fig. 5.4a–f show a main linear direction (top→down, left→right, topleft→downright)¹⁰ the main timeline for the Hilbert curve is U-shaped which is kind of irritating not only for the uninformed reader.

The recursive z-curve Fig. 5.4h roughly maintains an overall topleft→downright direction with partly more locality than the pure diagonal layouts Fig. 5.4e, f but on the other hand shows leaps.

We found the row and column layouts least irritating to the uninformed as well as expert user, they are easy to explain and the more complicated layouts are not capable of solving all the issues described above. And finally its up to the decision of the user which layout he or she prefers.

If the data timeline is known to be structured by periods relevant to the problem domain, e.g. quarters or terms, the pixel-oriented visualization can easily adapt to

¹⁰Which could be flipped e.g. for Arabic readers who may prefer right→left.

that fact. For example, to present 10 years of data month by month use a row by row or column by column layout with 12×10 , and you may be able to see some pattern for Christmas time, for 1 year (52 weeks) day by day use 14×26 or 21×18 and so on. If no such rhythm applies choose quadratic glyphs with any layout you may like but keep in mind how to interpret it and which patterns may show up spuriously.

5.3.4 Arranging the Glyphs

Displaying the timelines of the users row by row as in Fig. 5.2c, d raises the question about the best order to arrange them. A similar issue arises with the matrix representation (Fig. 5.3).

In many cases additional background knowledge is available about the members of the social network. They belong to some part of the organization, work together in a certain project, have a certain role (manager, clerk, intern etc.). Grouping users by one of these aspects can help to identify certain patterns (are members of the same projects working together within certain timespans? Are there different patterns showing up for different roles?).

A second criterion for grouping is some node feature. For the examples up to now we used the (weighted) degree of each node, that is we sorted by a network parameter. This is reasonable as such parameters provide well-defined sorting criteria, but it is also kind of arbitrary as one would need to argue why not to base sorting on content-based criteria such as activity, timespan, or grouped nodes highly connected in the network.

The pixel-oriented network visualization supports the identification of *patterns* in the timelines. The layout algorithm can support the pattern recognition process by choosing arrangements placing similar objects side by side. Figure 5.5 shows the same user collaboration timelines as Fig. 5.2 with users grouped by similarity of their timelines. The most outstanding change is that U10, U13 and U3, the three users only working in April 2009, are grouped together and not longer disturbed by U14. We give a larger and more substantial example in Sect. 5.4 (Figs. 5.9 and 5.10).

The rest of this section focusses on ordering by these similar patterns. First an approach for computing distances (dissimilarities) on user timelines is given. Based on these distances the timelines are then ordered by multidimensional scaling (MDS). And finally we describe additional considerations for matrix ordering.

5.3.4.1 Comparing Timelines

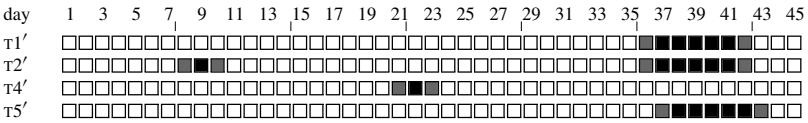
Each user timeline gives a high dimensional vector, where each data point (user activity at a certain time) is mapped to one vector component. So each timeline is represented as a point in a high dimensional space. This allows to compute the distance (i.e. the discrepancy) between each pair of timelines and to group users

But $d(T1, T5) = 12.25$, so T1 is closer to T4 than to T5. Mathematically, this is obvious as dimensions are independent to each other and T4 and T5 differ in six coordinates by $|5 - 0| = 5$, but it may not be what we want. Users T1, T4 and T5 were active in the 6 week so it may be reasonable to count their timelines as similar, even if they did not show active on the same days.

One way to solve this would be to use a coarser raster for computing the distances, we can use weekly timelines for ordering and daily timelines for presentation. So T1 and T5 show activity intensity of 15 in week 6 and no activity in other weeks giving a complete match. But it would not work for users T3 and T4, as we are unlucky with the weekly border: T3 shows activity in week 3 while T4 is active in week 4.

Therefore, we propose another solution. The basic idea consists in not computing the distances on the timelines as given above but instead to smoothen (blur) them. For each component a_i of the timeline vector the neighbors $a_{i-k}, a_{i-k+1}, \dots, a_{i-1}$ and $a_{i+1}, a_{i+2}, \dots, a_{i+k}$ are modified by the value of a_i , e.g. by addition of $\frac{1}{2}a_i$. For example with $k = 1$ by computing: $a'_i = \frac{1}{2}a_{i-1} + a_i + \frac{1}{2}a_{i+1}$ we get the ordering timeline for all timepoints¹² i .

This gives modified timelines (■ = 5, ▒ = 2.5, □ = 1, ◻ = 0):



$d(T1', T2') = 6.12$, $d(T1', T4') = 13.23$, $d(T1', T5') = 5$, $d(T2', T5') = 7.9$. So T1' and T5' are closer than T1' and T4', as requested. So in general smoothing works. How to do the smoothing exactly, e.g. how to choose the size of the neighborhood k , depends on the specific problem domain. For the daily timelines of the *workgroup wiki* (Fig. 5.5b) we used $k = 7$ which means that activities of two users are overlapping as long as they are not more than 1 week apart which is feasible for users collaborating in a wiki over several years.

5.3.4.2 Distance Measures

In the examples given the distances are computed using the euclidean distance metric. Comparing the timelines for T5, T6 and T7 gives the euclidean distances $d(T5, T6) = 6.93$ and $d(T6, T7) = 1.73$, so T6 is much closer to T7 than to T5. This holds for all minkowski (p-norm) metrics, e.g. manhattan metric gives $d(T5, T6) = 12$ and $d(T6, T7) = 3$ and maximum metric gives $d(T5, T6) = 4$ and $d(T6, T7) = 1$.

¹²For indices outside the time range (e.g. a_{-1}) the value 0 is substituted.

Whether this is the desired result depends on the scenario. Are two users which are active at the same time similar even if the activity of one is very high and of the other is very low or is low activity more similar to no activity?

The former can be achieved in several ways. Adding some δ to each value greater than 0 increases the relative gap between 0 (no activity) and other values. Alternatively transforming the activity values by $a_i' = \sqrt[m]{a_i}$ reduces the importance of high values, for very large m it is similar to setting all non-zero values to 1. Now a distance measure can be applied to the modified timelines. For example, with $m = 5$ and the euclidean metric we get $d(T5, T6) = 0.66$ and $d(T6, T7) = 1.73$ as expected.

Other measures like the Pearson correlation (known as r) or the cosine coefficient¹³ (which computes the cosine of the angle between two vectors) can be used to compare timelines by their relative intensity pattern. As they compute the similarity s of two vectors with 1 indicating maximum similarity, and not the distance, we define $d(TA, TB) = 1 - s(TA, TB)$ to get a distance measure as required for MDS. Two timeline vectors with $T_A = \lambda \cdot T_B$, ($\lambda > 0$) are considered similar by these measures, so we get $d(T5, T6) = 0$. With cosine similarity we get $d(T1', T2') = 0.11$, $d(T1', T4') = 1$ (the maximum distance possible) and $d(T1', T5') = 0.09$, i.e. sensible values.

Unfortunately both measures do not allow to compare with the zero vector (which has no direction). While this is not a problem for ordering collaboration activities as in Fig. 5.5 where users with empty timelines could simply be put at the end, it will give problems for ordering matrices (Sect. 5.3.4.4) as here a lot of glyphs with empty timelines are normal (see e.g. Fig. 5.6). Therefore comparison with the zero vector has to be handled as special case. Two empty timelines have distance 0. As absolute intensity is ignored by cosine, it seems natural to take the limes for $\lambda \rightarrow 0$ and set the distance of any to the zero vector to 0. As this does not give the desired effect a possible solution is to use a fixed maximum distance of 2 (which is larger than all other distances).

5.3.4.3 From Timeline Distances to User Arrangement

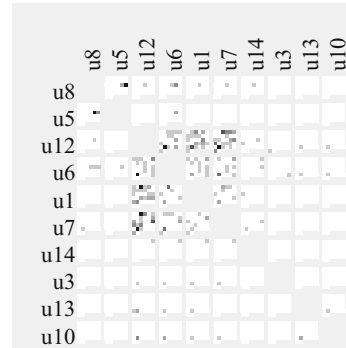
Applying distance measures as described for all pairs of timelines gives a distance matrix. Now timelines with low distances can be clustered together. Using multidimensional scaling¹⁴ down to one dimension assigns a one dimensional position to each timeline. Please note that this even works if the distance matrix does not fulfil all properties of a metric space (e.g. triangle inequality).

Sorting the users according to these positions of their corresponding timelines arranges similar ones next to each other as visible in Fig. 5.5. While the sorting

¹³Which are in fact very similar, see e.g. [23].

¹⁴Multidimensional scaling (MDS, [21]) in several variants [10].

Fig. 5.6 *Workgroup wiki* network matrix ordered by collaboration activity



is done on the smoothened timelines which, as noted above, may have coarser resolution than the original timelines, the original ones are used in visualization.

Choosing good projections for MDS is known as a difficult problem [5], especially to only one dimension. We tested a lot of distance measures and parameters (more than described above) on social network data of several wikis (examples are given in Sect. 5.4) and found the ordering of the nodes to be very sensitive even to small changes of the distance matrix while the grouping of users with very similar timelines stays rather robust. This means small parameter changes give different layouts, but within these similar users (e.g. u5 and u8 in Fig. 5.5) are grouped together. So it is important to choose the right distance measure for the general decision whether to focus on absolute intensity or on relative cooccurrence of activity, as this defines which timelines should be grouped together, and adjusting the parameters can improve the visual appearance.

5.3.4.4 Arranging Matrix Rows and Columns

For the network matrix further parameters have to be considered for sorting the nodes. While grouping by external knowledge or some node feature stays the same arranging by similarity gets more complex as now each node does not longer correspond to exactly one timeline but (for n nodes) is connected to $2(n - 1)$ timelines (the corresponding row and column).

We cannot simply arrange these $n(n - 1)$ timelines by similarity as this would destroy the matrix structure. We can only rearrange whole columns and rows so that rows/columns containing similar timelines are placed next to each other. While it is possible to have a different order for rows and columns we would not recommend this, as it is unintuitive for a network matrix.

An easy and often sufficient approach is to use the node arrangement computed on their collaboration activity timelines for the matrix (Figs. 5.5b and 5.6). Our *workgroup wiki* example will not gain any further improvements by more sophisticated measures.

For other (larger) networks things are different as we will see later (Figs. 5.9–5.11). The user timelines only show the unspecific activity pattern of the users. So two users being active at the same time are grouped together even if their activity is totally unrelated. To compute the similarity of two user rows in the network matrix we therefore compute the distances of their glyphs column by column, e.g. for the users U3 and U13 in Fig. 5.6 (third and second bottommost row) we compare the glyph distances $d(U3/U8, U13/U8)$, $d(U3/U5, U13/U5)$, \dots , $d(U3/U10, U13/U10)$. The distance of the rows U3 and U13 is the sum of the distances of their glyphs. Please note that for comparison of the single glyph timelines everything described in Sect. 5.3.4.1 (smoothing, distance measures etc.) can be used.

Pairs involving diagonal elements (in the example i.e. $d(U3/U3, U13/U3)$ and $d(U3+/U13, U13/U13)$) cannot be computed and are either skipped or the diagonal glyph is virtually set to a timeline with each point set to the maximum of all values in the matrix. The latter measure supports users working closely together to be considered more similar, e.g. $d(U1/U12, U12/U12) < d(U10/U12, U12/U12)$. Both options work and we found the effect of the latter negligible on large networks.

5.3.5 Interactive Visual Data Mining

The visualization paradigm described in this section has been implemented as part of a network analysis package. Figure 5.7 gives a screenshot of PONVA, our interactive pixel-oriented network visualization application, which allows us to experiment with different layout algorithms and settings on various networks.¹⁵

Zooming into the full plot as well as “zooming” the glyphs by changing the resolution (Fig. 5.3b, c) allows to switch between overview and detailed inspection. Selecting the active timespan within one glyph or one column or row and highlighting the corresponding pixels in the other glyphs helps comparing user activity in time. Same holds for selecting one user or a pair of users and highlighting the corresponding row(s), column(s) (and intersections). And clicking on a pixel reveals detailed data from the point in time it represents to its numeric value.

Selecting one of the different glyph layout mechanisms, color scales and some γ -correction appropriate to the network to be analyzed is done interactively. And finally rows and columns can be sorted according to various similarity measures, which are not discussed within this chapter due to limited space.

Additionally an interactive application allows to use the distance information available from MDS. Figure 5.8 shows a two-dimensional configuration of the timelines (folded to row-by-row glyphs) scaled to two dimensions. This places the users/glyphs according to the similarity relations of their timelines, so similar glyphs are visually clustered together. This representation is not too useful on a printed

¹⁵POVNA [35] is written in Java. Not all features described are included in the current stable release.

Fig. 5.7 Two-dimensional MDS representation of the *workgroup wiki* timelines

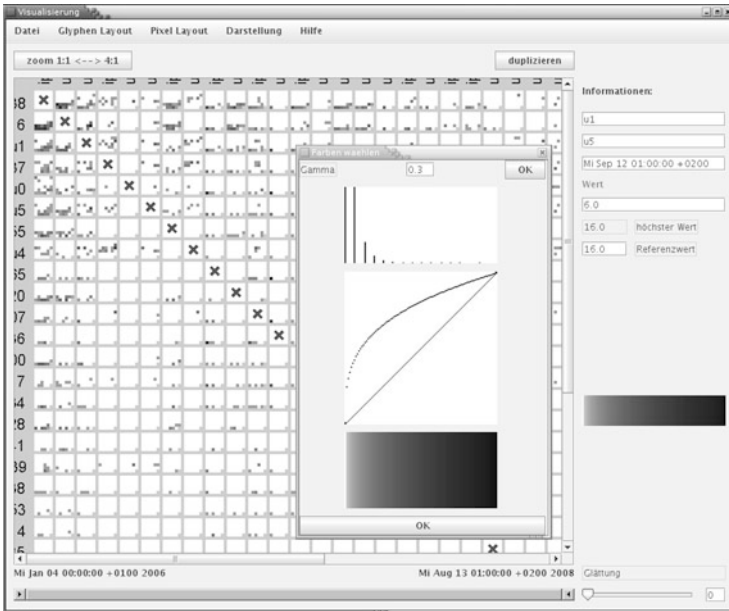
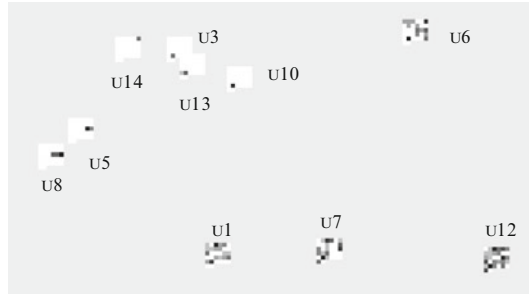


Fig. 5.8 Graphical user interface of PONVA. The main window shows a pixel-oriented visualization of the network matrix while the dialog in front allows to change γ -correction. To the right detailed information about one selected pixel is displayed

paper as it takes a larger amount of space which conflicts with the paradigm of pixel-oriented visualization that requires to use every pixel on the screen to represent one data point. The visualizations also suffers from cluttering as there are problems with overlapping glyphs and the annotation of the user names. However it is useful for interactive exploration where one can scroll and zoom in and annotations are dynamically displayed on mouseover or mouseclick.

5.4 Exploring Larger Networks

In the following we demonstrate the practical value of pixel-oriented visualization with three real world data sets. The datasets were collected during our research on organizational wikis.¹⁶ A particularity of the data set is the fact that rich background knowledge about the organization context of the social networks has been collected independently. We show how user interaction patterns in time become visually salient in the pixel-oriented visualization and demonstrate that these findings correspond to social patterns in the organization. Additionally we show how different node ordering methods improve the visualization.

5.4.1 *Students Wiki*

In Sect. 5.2.1 we introduced the *students wiki* (Fig. 5.1). Figure 5.9 shows the pixel-oriented visualization of this network sorted by (weighted) node degree: Fig. 5.9b gives the interaction timeline for each user (day by day grouped weekly) and Fig. 5.9a the corresponding network matrix. Both plots only give users with at least 50 interactions as the full matrix with 142 users would not be readable on this paper size.¹⁷

The first thing catching our eye in Fig. 5.9b is this dotted horizontal bar showing up at about one third of the timeline (users marked with ●). This is the start of the second term and obviously the new and some of the older students did a lot of work in this week. Next we see heavy traffic at the beginning of the timeline for a 2 week timespan (□). Some users continue to participate, others leave, but most of the active users within the first two terms do not show up in the third one, and the users active in the third term (○) were not present before. Exceptions are only users U8 and U105 who show up again at the end of the third term and U68 who came in the middle of the second term but without being very active. The users of the first and second term overlap but there is low connection to the third term, and this confirms what we see in Fig. 5.1b–d.

While all this is visible in Fig. 5.9b, it gets more obvious in Fig. 5.10b where the user timelines are arranged by similarity as described in Sect. 5.3.4.3. The students active in the first, second and third term are now grouped together with the ones active during a whole year (first and second term) nicely placed in between.

The network matrix in Fig. 5.9a presents a nearly regular pattern of similar glyphs interrupted by rows and columns with rather empty glyphs and some few darker ones. By looking closer we can distinguish users with different glyph patterns. The

¹⁶In-depth case studies of the wikis used here including other types of (not pixel-oriented) visualizations are provided by Stein and Blaschke [31].

¹⁷This is less a problem in interactive usage where we can scroll.

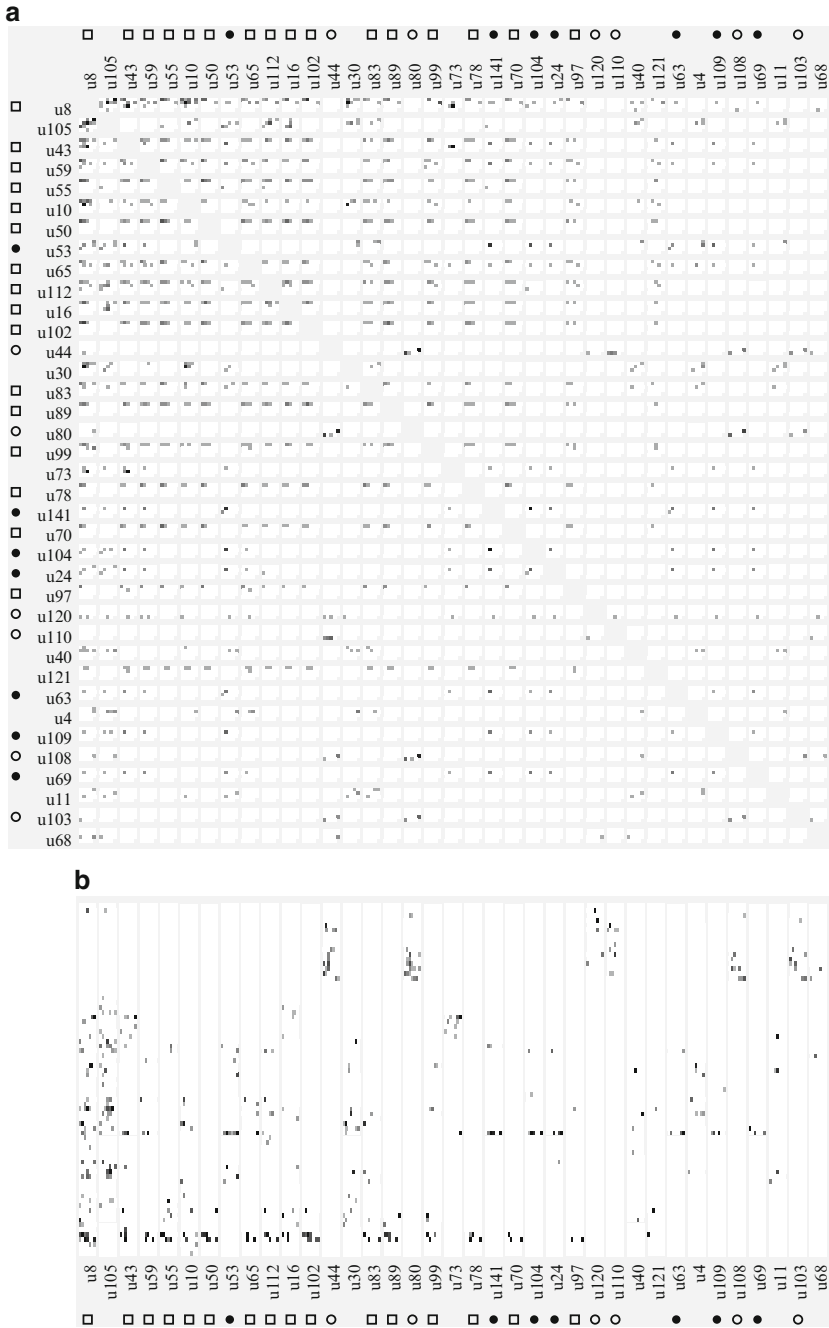


Fig. 5.9 *students wiki* (users with at least 50 links). (a) Network matrix. (b) Collaboration intensity. Time goes from *bottom* to *top*

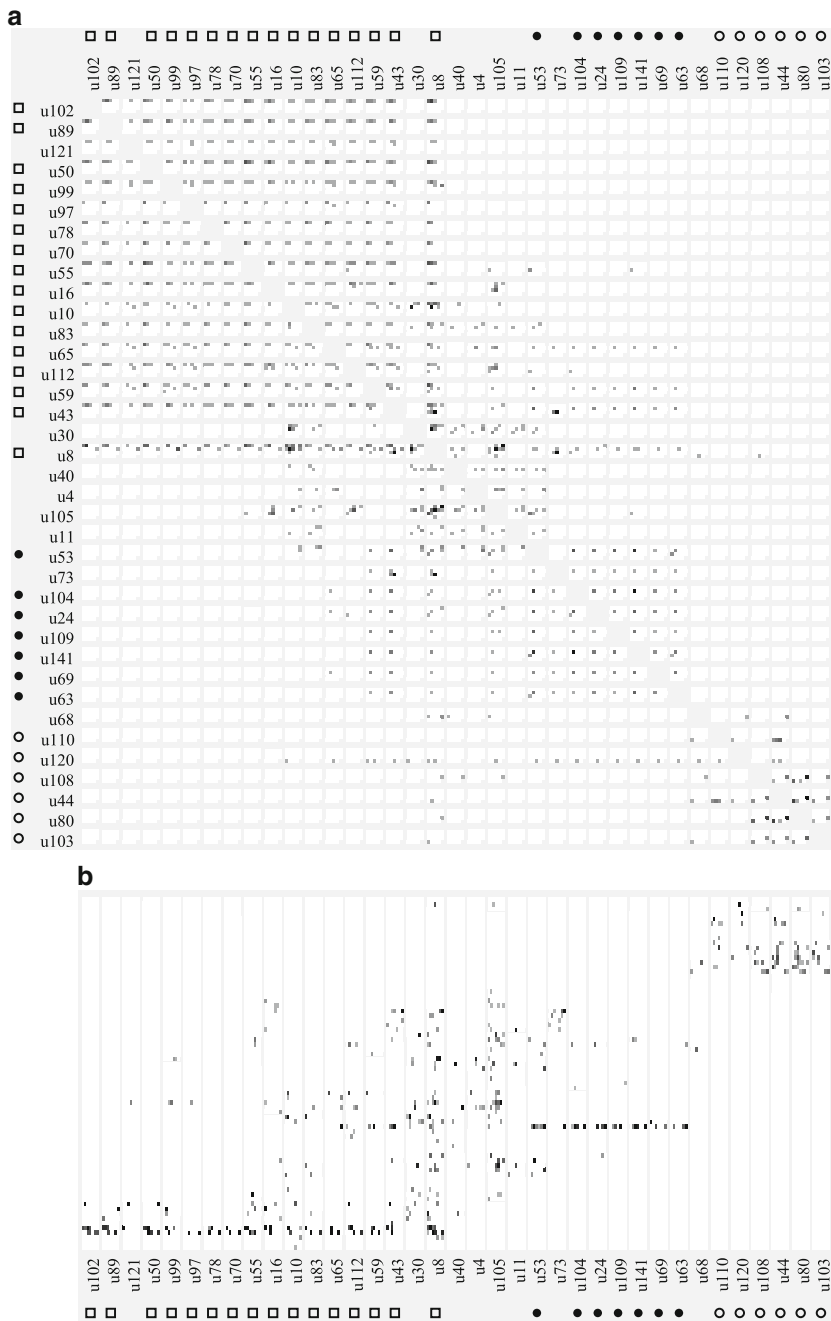


Fig. 5.10 *Students wiki* (users with at least 50 links), grouped by collaboration intensity timeline similarity. (a) Network matrix. (b) Collaboration intensity. Time goes from *bottom* to *top*

changed arrangement of rows and columns in Fig. 5.10a makes things easier by revealing patterns which give further insights: while there is collaboration between the students of the first and second term visible in the matrix the third term is nearly fully disconnected. Most of the students present in the first term collaborate with most of their fellows, for the wiki this means they all worked on the same wiki pages, they did not split up in subgroups working on different pages. Only user U120 stands out. While joining the wiki in the third term contrary to his fellows, he connects to most of the users of the first and second term, i.e. works on the pages they edited before.

In Fig. 5.11 not the collaboration intensity of the student timelines but the similarity of the glyph rows in the matrix is used to arrange the students. The difference to Fig. 5.10 is not as big as between Figs. 5.9 and 5.10 but noticeable. The three groups of students are clearly visible in the matrix, as well as the connection between the first and second group (users U83 to U43, i.e. the first six rows/columns in the matrix, plus some connection for rows/columns U112 to U16).

The exact configuration of rows and columns is highly dependent from the parameters chosen (smooth factor, distance metric, etc.) as if the distances between rows are small even small changes give different MDS to the one-dimensional space. Nevertheless the general patterns visible in the figure are rather stable.

So Figs. 5.9–5.11 are a good example for the perceptual salience of regular temporal collaboration patterns on one hand and the improvements possible by sophisticated grouping of the nodes. They not only show cooccurrence of activity of single users but also which users and groups of users are connected at which time.

5.4.2 Facilitation Wiki

Figure 5.12 introduces a new network. It shows the internal wiki of an organization that lends support for information and communication technologies to small and medium enterprises. The wiki features mostly research articles, project reports, and later publications thereof. The wiki is maintained as a dedicated project, and employees are enforced to generate a certain amount of input per anno.

The first thing catching the eye when inspecting the collaboration intensity timelines Fig. 5.12a is that users U35 (●) and U48 (○) stand out.¹⁸ U35 was the project manager of the wiki project and obviously did a lot of work until she left the company after three quarters of the timeline, where U48 had to take over this position. Both users were connected to almost everybody else as we can see in the first two rows of the network matrix (Fig. 5.12b). Only a connection from U35 to U33 is missing and as we can see in the user timelines U33 joined the wiki after U35 left.

¹⁸We may also see that they never edited the wiki on weekends.

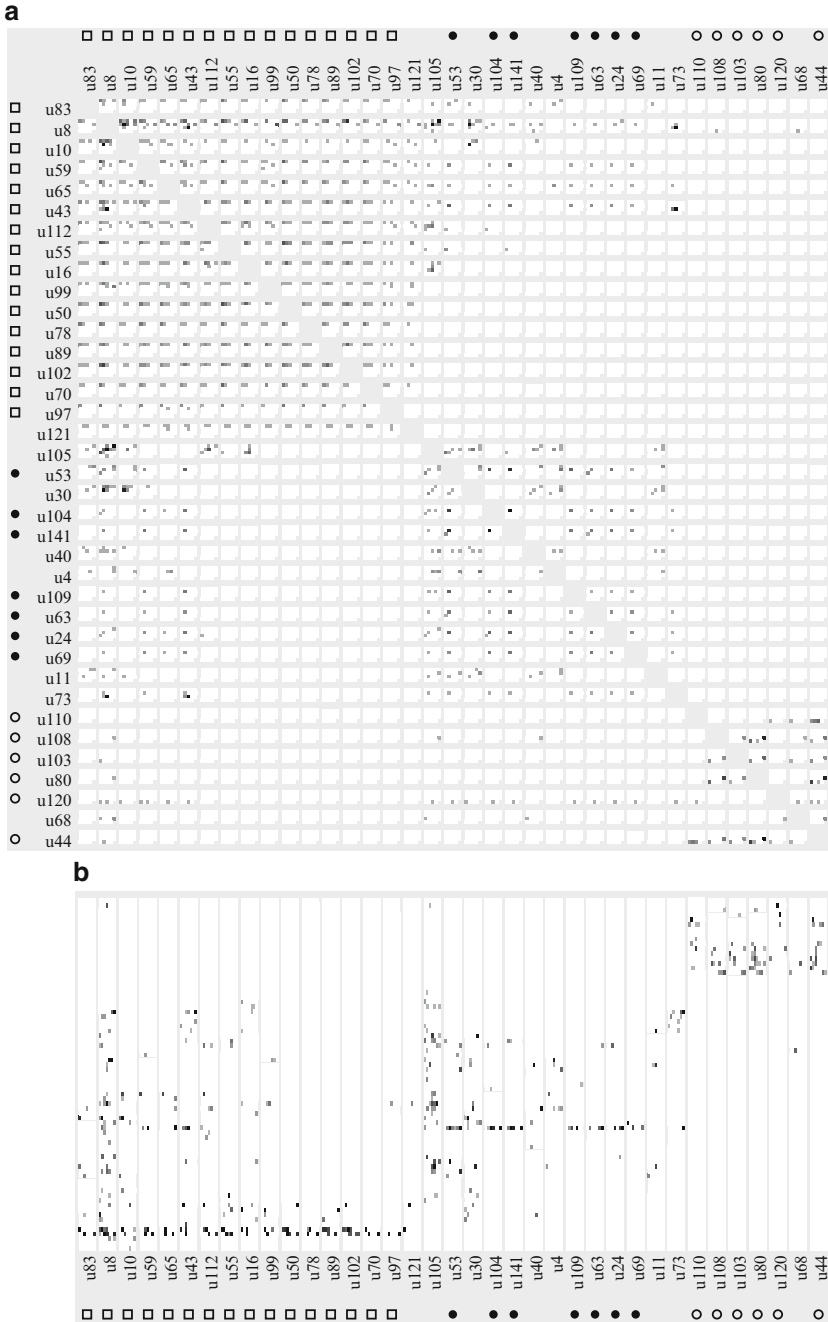


Fig. 5.11 *Students wiki* (users with at least 50 links), grouped by glyph row similarity. (a) Network matrix. (b) Collaboration intensity. Time goes from *bottom* to *top*

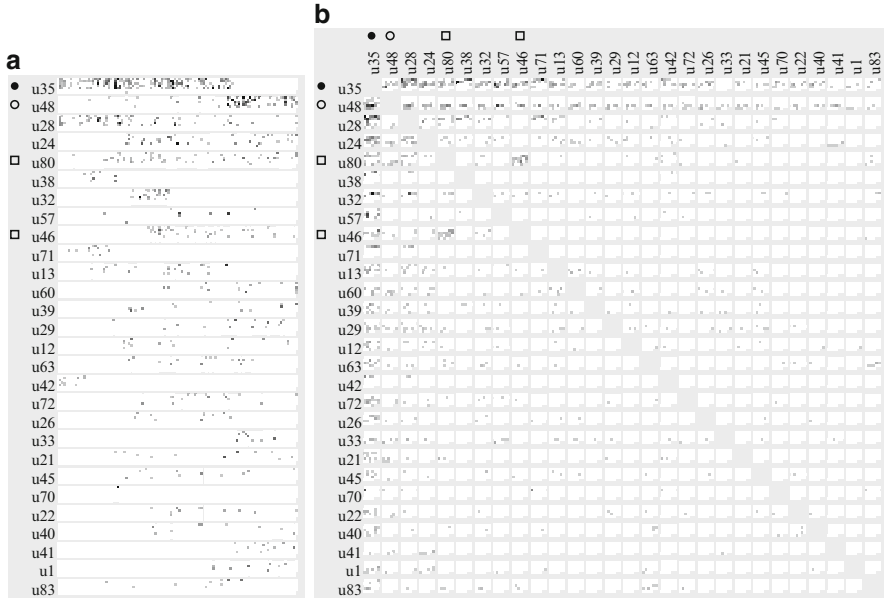


Fig. 5.12 *Facilitation wiki* (users with at least 50 links). (a) Collaboration intensity. (b) Network matrix

We could backup our findings by interviews where we learned that U35 (and later U48) was asked by all others to assist them with the wiki. However, the wiki was never really accepted by the users, and almost nobody started to use the wiki as a collaboration tool, so she had to work as a “gardener”, formatting the articles others had written. And the missing collaboration is visible in the network matrix, most of the glyphs are empty or very sparsely filled, only the pair U80 and U46 □ stands out, here direct collaboration shows up.

Contrary to the last example no uniform patterns are visible, but temporal cooccurrence as well as temporal connection draw attention when present in the fuzzy gray-spotted area of glyphs as our visual perception smoothens single dots to larger areas.

5.4.3 Startup Wiki

Our last example (Fig. 5.13) shows the *startup wiki*. It is the company wiki of an European market leader for one-stop solutions and services in mobile or proximity marketing. It was installed during the founding of the company and since then keeps its role as primary collaboration and main content management system for

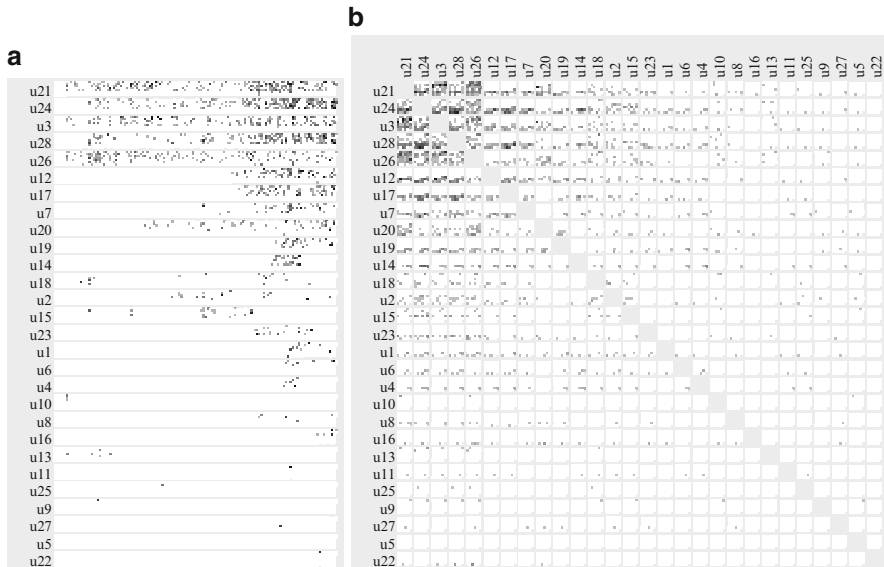


Fig. 5.13 *Startup wiki* network matrix. (a) Collaboration intensity. (b) Network matrix

the engineers. Growing from 3 to 26 users the primary users never stopped to use the wiki and new employees joining get connected to the others. Wiki growth at its best, and nicely visible in the user interlocking network matrix.

We see users with high activity as well as ones showing up only once which is not surprising as e.g. accounting uses other specialized software and other groups only need to read wiki contents without editing.

Contrary to the *facilitation wiki* users connect to each other, we do not have this one “gardener” getting all the connections but collaboration between the active users. Not everyone is using the wiki intensely but those who do are connected to each other.

5.5 Conclusions and Outlook

In this chapter we introduced pixel-oriented network visualization (PONV), a new visualization for weighted social networks changing in time. Our approach focuses on using static visualization of dynamic network data as a method of data exploration by the means of visual data mining which works best on medium-size networks where the whole matrix fits on the screen or paper at once.

We discussed different patterns for pixel as well as the glyph arrangement. Furthermore, we presented different pixel layouts from simple ones like plain rows to more complex space-filling curves. It turns out that the arrangement of the glyphs should be adapted according to the actual task and can be lead by external

knowledge, some node feature or similarity. We therefore introduced different distance measures.

The interpretation of PONV is not immediately obvious but easily learned by the interested expert. Using several interlocking coauthor networks extracted from organizational wikis we could show how pixel-oriented network visualization reveals perceptual salient patterns for temporal cooccurrence of user collaboration as well as user-user-connection. In our examples we tended to stick to the simple row by row approach for the arrangement of the pixels which we found to be least irritating. In addition a glyph arrangement according to similarity lead to the visually most appealing results.

PONV allows to detect similar collaboration patterns across users and to reveal the collaboration between a pair of users across time. It is nevertheless meant as complementary visualization besides network graphs and others and not as a substitution, as interactions between groups of more than two users only show up indirectly, it focuses on temporal patterns, not on the detection of network clusters. This also means that PONV is less useful on rather sparse networks with low traffic where no visual patterns emerge.

Topics for future work are the development of more sophisticated glyph layouts as well as combining pixel-oriented visualization with other layout algorithms like using the glyphs as nodes in a graph representation integrating both visualization techniques. Finally the understandability and usefulness of PONV shall be examined and at best improved by user studies.

The analysis and graphics in this chapter are produced using the pixvis/PONV module of our own Wiki Explorator library.¹⁹ It is available for download under an open source licence.²⁰

Acknowledgements Part of this work was supported by the Volkswagenstiftung through Grant No. II/82 509.

We thank the reviewers for their helpful comments.

References

1. Aigner, W., Miksch, S., Müller, W., Schumann, H., Tominski, C.: Visualizing time-oriented data, a systematic view. *Comput. Graph.* **31**(3), 401–409 (2007)
2. Ankerst, M., Berchtold, S., Keim, D.A.: Similarity clustering of dimensions for an enhanced visualization of multidimensional data. In: *INFOVIS '98: Proceedings of the 1998 IEEE Symposium on Information Visualization*. IEEE Computer Society, Washington, DC (1998)
3. Balzer, M., Deussen, O.: Level-of-detail visualization of clustered graph layouts. In: *APVIS'07*, pp. 133–140. IEEE, Piscataway (2007)

¹⁹Wiki Explorator is written in Ruby using R, Gnuplot, Graphviz and other open source software.

²⁰<http://wiki-explorator.rubyforge.org>. We also provide an online wiki analysis service based on this library at <http://www.kinf.wiai.uni-bamberg.de/mwstat>.

4. Bender-deMoll, S., McFarland, D.A.: The art and science of dynamic network visualization. *J. Soc. Struct.* **7**(2), 1–46 (2006). <http://www.cmu.edu/joss/content/articles/volume7/deMollMcFarland>
5. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is “nearest neighbor” meaningful? In: *Database Theory—ICDT’99*, pp. 217–235. Springer, Berlin/New York (1999)
6. Brandes, U., Kenis, P., Raab, J.: Explanation through network visualization. *Methodology* **2**(1), 16–23 (2006)
7. Chen, C.: *Information Visualization: Beyond the Horizon*, 2nd edn. Springer, London (2006)
8. Chen, I.X., Yang, C.Z.: Visualization of social networks. In: Furht, B. (ed.) *Handbook of Social Network Technologies and Applications*, pp. 585–610. Springer, New York (2010)
9. Chen, W., Ji, M., Tang, X., Zhang, B., Shi, B.: Visualization tools for exploring social networks and travel behavior. In: *2010 International Conference on Environmental Science and Information Application Technology (ESIAT)*, vol. 3, pp. 239–243. IEEE, Piscataway (2010)
10. Cox, M.A.A., Cox, T.F.: Multidimensional scaling. In: Chen, C.h., Härdle, W., Unwin, A. (eds.) *Handbook of Data Visualization*. Springer Handbooks Computational Statistics, pp. 315–347. Springer, Berlin/Heidelberg (2008). doi:http://dx.doi.org/10.1007/978-3-540-33037-0_14
11. Eppstein, D., Gansner, E.R. (eds.): *Graph Drawing: 17th International Symposium, GD 2009*. LNCS, vol. 5849. Springer, Berlin/New York (2010)
12. Everitt, B.S., Landau, S., Leese, M.: *Cluster Analysis*, 4th edn. Wiley, Chichester (2009)
13. Ghoniem, M., Fekete, J.D., Castagliola, P.: A comparison of the readability of graphs using node-link and matrix-based representations. In: *INFOVIS ’04: Proceedings of the IEEE Symposium on Information Visualization*, pp. 17–24. IEEE, Washington, DC (2004). doi:<http://dx.doi.org/10.1109/INFOVIS.2004.1>
14. Guo, Y., Chen, C., Zhou, S.: Fingerprint for network topologies. In: Zhou, J. (ed.) *Complex Sciences*, pp. 1666–1677. Springer, Berlin/Heidelberg (2009)
15. Henry, N., Fekete, J.D., McGuffin, M.J.: Nodetrix: a hybrid visualization of social networks. *IEEE Trans. Vis. Comput. Graph.* **13**(6), 1302–1309 (2007). doi:<http://dx.doi.org/10.1109/TVCG.2007.70582>
16. Kamada, T., Kawai, S.: An algorithm for drawing general undirected graphs. *Inf. Process. Lett.* **31**(1), 7–15 (1989)
17. Keim, D.A.: Designing pixel-oriented visualization techniques: theory and applications. *IEEE Trans. Vis. Comput. Graph.* **6**(1), 59–78 (2000). doi:<http://dx.doi.org/10.1109/2945.841121>
18. Keim, D.A., Keim, D.A., Kriegel, H.P., Kriegel, H.P.: Visdb: database exploration using multidimensional visualization. *IEEE Comput. Graph. Appl.* **14**, 40–49 (1994)
19. Keim, D.A., Ankerst, M., Kriegel, H.P.: Recursive pattern: a technique for visualizing very large amounts of data. In: *VIS ’95: Proceedings of the 6th Conference on Visualization ’95*. IEEE Computer Society, Washington, DC (1995)
20. Krempel, L.: Network visualization. In: Scott, J., Carrington, P.J. (eds.) *Sage Handbook of Social Network Analysis*. SAGE, London/Thousand Oaks (2009)
21. Kruskal, J.B., Wish, M.: *Multidimensional Scaling*. Sage University Paper Series on Quantitative Application in the Social Sciences. Sage, Beverly Hills/London (1978)
22. Leung, C., Carmichael, C.: Exploring social networks: a frequent pattern visualization approach. In: *IEEE International Conference on Social Computing/IEEE International Conference on Privacy, Security, Risk and Trust*, pp. 419–424. IEEE, Los Alamitos (2010)
23. Leydesdorff, L.: On the normalization and visualization of author co-citation data: Salton’s cosine versus the jaccard index. *J. Am. Soc. Inf. Sci. Technol.* **59**(1), 77–85 (2008)
24. Moody, J., McFarland, D., Bender-deMoll, S.: Dynamic network visualization. *Am. J. Sociol.* **110**(4), 1206–1241 (2005). doi:10.1086/421509
25. Peng, W., SiKun, L.: Social network visualization via domain ontology. In: *International Conference on Information Engineering and Computer Science*, pp. 1–4. IEEE, Piscataway (2009)
26. Reeve, L., Han, H., Chen, C.: Information visualization and the semantic web. In: Geroimenko, V., Chen, C. (eds.) *Visualizing the Semantic Web: XML-Based Internet and Information Visualization*. Springer, London (2006)

27. Riche, N.H., Fekete, J.D.: Novel visualizations and interactions for social networks exploration. In: Furht, B. (ed.) *Handbook of Social Network Technologies and Applications*, pp. 611–636. Springer, New York (2010). doi:http://dx.doi.org/10.1007/978-1-4419-7142-5_28
28. Shi, L., Cao, N., Liu, S., Qian, W., Tan, L., Wang, G., Sun, J., Lin, C.: HiMap: adaptive visualization of large-scale online social networks. In: *Visualization Symposium, PacificVis '09*. IEEE, Piscataway (2009)
29. Shneiderman, B.: Extreme visualization: squeezing a billion records into a million pixels. In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pp. 3–12. ACM, New York (2008)
30. Shneiderman, B., Aris, A.: Network visualization by semantic substrates. *IEEE Trans. Vis. Comput. Graph.* **12**(5), 733–740 (2006). doi:<http://dx.doi.org/10.1109/TVCG.2006.166>
31. Stein, K., Blaschke, S.: Corporate wikis: comparative analysis of structures and dynamics. In: Hinkelmann, K., Wache, H. (eds.) *Proceedings of the 5th Conference on Professional Knowledge Management. Lecture Notes in Informatics*, pp. 77–86. Gesellschaft für Informatik, Bonn (2009)
32. Stein, K., Blaschke, S.: Interlocking communication, measuring collaborative intensity in social networks. In: *Social Networks Analysis and Mining: Foundations and Applications*. Springer, New York (2010)
33. Unwin, A., Theus, M., Hofmann, H.: *Graphics of Large Datasets: Visualizing a Million*. Springer, New York (2006)
34. Ware, C.: *Visual Queries: The Foundation of Visual Thinking*. Springer, Berlin/Heidelberg (2005)
35. Wegener, R.: *Kollaborationsprozesse in Wikis: Entwurf und Umsetzung eines Analysewerkzeugs*. Master's thesis, University of Bamberg (2009)