Addressing Directly-Follows Graphs limitations with visualization of event sequences

Luis Montana[1 [0000−0002−0511−0466)]], Manuel Resinas[2 [0000−0003−1575−406X]], and Maria-Cruz Villa-Uriol[1,3 [0000−0002−3345−539X)]]

[1] School of Computer Science, University of Sheffield, UK
[2] Universidad de Sevilla, Spain
[3] INSIGNEO Institute for in silico Medicine, Sheffield, UK

Abstract. Visual Process Analytics (VPA) is emerging as a discipline where Process Mining (PM) and Visual Analytics (VA) experts collaborate to simplify the analysis and enhance the understanding of complex processes. In this work, we focus on Directly-Follow Graphs (DFGs), a common visualization technique used in PM, and identify how some of its limitations can be overcome by complementing DFGs with a timeline-based visualization strategy used in VA systems analyzing event sequences obtained from event logs. To illustrate this, we have chosen the Road Traffic Management Process dataset, a well-known dataset to the PM community. We have identified four cases where DFGs struggle to easily convey the exact nature of the underlying processes and show how timeline-based visualization strategies help disambiguate findings.

Keywords: Visual process analytics · Directly-Follows Graph · Event sequence visualization.

1 Introduction

Data visualization plays a crucial role in analyzing datasets and uncovering meaningful insights. In process mining (PM), visualization techniques are essential to understand the execution of processes [4,8]. The Directly-Follows Graph (DFG) is a common visualization used in PM to represent control-flow behavior based on event logs. DFGs are simple to compute and generally easy to understand, though they can sometimes be misinterpreted [1].

Despite their advantages, DFGs also have limitations [1]. First, they become too complex with a high number of variants. The more variants, the larger the number of edges in the DFG, leading to Spaghetti-like DFGs difficult to interpret (**L1**). To simplify them, frequency-based thresholds are typically used, often leading to misinterpretations [1]. Second, concurrent activities may appear in varying orders across traces in the event log, leading to DFGs with loops, even when each activity occurs only once per case [1] (**L2**). Unlike more expressive modeling notations like Petri nets, DFGs often misrepresent concurrency as cyclic behavior. Third, DFGs usually have a low precision, i.e., they allow behaviors that are not observed in the event log (**L3**). Figure 1(a) shows the

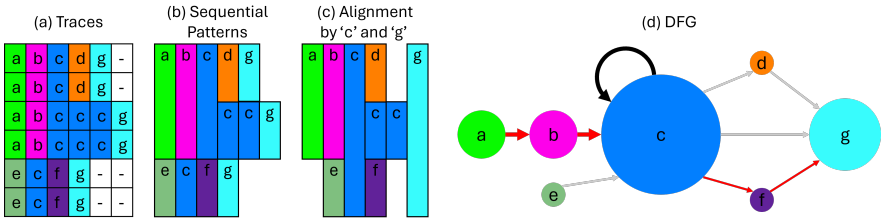

Fig. 1. (a) Event log with six traces represented as (b) sequential patterns, (c) sequential patterns with aligned activities, and (d) the corresponding DFG.

traces in an event log and its corresponding DFG in (c). This DFG allows traces like a → b → c → f → g (path in *red*), even though this behavior is not present in the event log. To address this problem, analysts use a range of manual and time-consuming strategies, such as exploring the process variants one by one, or filtering the event log to determine if a certain behavior is present or not. This task is, in some cases, unmanageable for large numbers of variants. Finally, DFGs represent repetitions of activities as loops. In Figure 1(d), the repetition of activity *c* in a → b → c → c → c → g is shown as a self-transition in node *c* (arrow in *black*). However, it is not possible to obtain the number of repetitions in the DFG, i.e., whether *c* repeats three times or, for instance, eight times (**L4**). This also happens with loops involving several activities.

In recent years, visual process analytics has emerged to emphasize the role of interactive visualizations and the multifaceted nature of the data involved in process analysis [12,14]. Visual process analytics aims to make complex process data more accessible and actionable for analysts by combining the strengths of PM with principles from visual analytics (VA). This approach goes beyond static diagrams by integrating dynamic, user-driven visual tools that support exploration, comparison, and interpretation of process behavior. In this paper, we adopt this perspective to overcome these limitations of DFG-based analysis. We argue that they can be addressed by complementing the DFGs with a timeline-based visualization technique [17,11] that uses hierarchical clustering [11] and sequence alignment [6]. Figure 1 illustrates how, for the event log in (a), the sequential patterns representation in (b) is enhanced in (c) by using alignment by activities 'c' and 'g'. First, we provide an overview of the related work. Then, we describe the four scenarios analyzed, and we finally present the conclusions.

2 Related Work

Process mining and visual analytics, particularly the literature on VA of event sequences, differ in the terminology used for core concepts in the analysis of sequential event data. In PM, *activity* refers to a categorical action within a process, and is analogous to the *event type* in VA. A single *event* in PM, an instance of an activity with a timestamp and case ID, corresponds to an *event*

occurrence (or *event*) in VA. *Traces* or *cases* are sequences of events associated with a single process instance in PM, being referred to as *individual sequences* in VA. *Variants* or *unique traces*, meaning distinct observed sequences across different cases, are *unique sequences* in VA. Finally, a *process model* in PM, which provides a high-level model of the process flow, serves a similar role to *sequential patterns* in VA, which provides an abstraction of frequent or representative event sequences across a dataset.

Visualization of event sequences. In PM, variant diagrams [14] are commonly used to visualize process variants, which represent unique sequences of activities. These diagrams arrange the variants vertically, and aligned to the left for consistency. Each activity within a variant is encoded using rectangles or horizontal chevrons, using distinct colors to enhance differentiation and readability. One limitation is that processes tend to have hundreds of variants, complicating the extraction of relevant insights. To address this, [2] proposed the use of sampling to select a representative set of variants, removing visual noise by coloring only the top 5 activities, and by ordering the traces by similarity to facilitate their reading. [13] also focus on visualizing variants using variant diagrams. However, in this case, they focus on event logs with partially ordered event data and heterogeneous temporal information per event (time intervals and time points).

In VA, existing techniques for visualizing event sequences primarily focus on encoding common pathways or sequential patterns [7,15]. In this context, van der Linden et al. [15] identified as still open challenges in the comparison of event sequences the definition of similarity metrics, the granularity of comparisons, the representation of temporal attributes, the integration of sequence attributes, and scalability. Recent surveys [7,18] provide an overview of these techniques, being particularly relevant timeline-based visualizations [10,5,16].

Table 1. Road traffic fines dataset. Acronyms representing activities and descriptions.

ID	**Description**	**ID**	**Description**
AP	Add Penalty	PAY	Payment
AJ	Appeal to Judge	RRA	Receive Result Appeal
CF	Create Fine		from prefecture
IDA	Insert Date Appeal to prefecture	SAP	Send Appeal to Prefecture
IFN	Insert Fine Notification	SF	Send Fine
NRA	Notify Result Appeal to offender	SFC	Send For Credit collection

3 Selected scenarios

We have selected four scenarios where event sequence visualizations can complement DFGs to address some of their limitations in representing general structure, parallelism, repetitions, and precision. We use the Road Traffic Fines

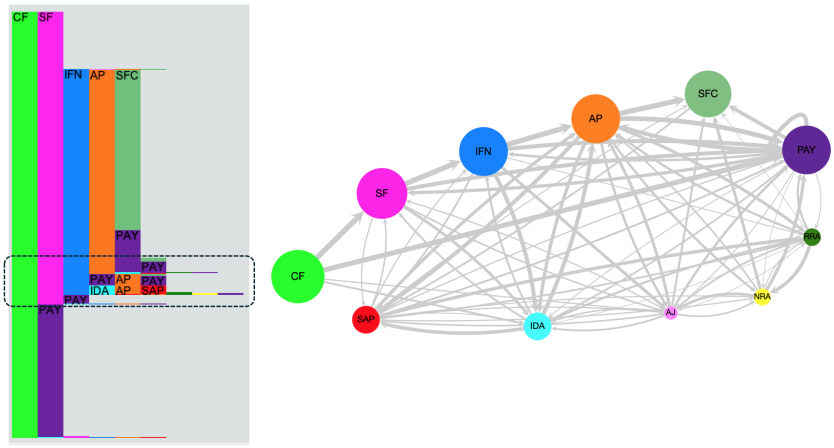

Fig. 2. Timeline-based event sequence visualization (*left*) vs. DFG (*right*) for the Road Traffic Fines dataset [9], highlighting the less frequent traces (dashed line).

dataset [9], which spans 13 years of road traffic fine management by a police force in Italy, comprising 150,370 cases and 561,480 events (Table 1).

Screenshots for the timeline-based event sequence visualizations have been obtained using Sequen-C [11], a visual analytics system for the analysis of event sequences that uses hierarchical agglomerative clustering to identify clusters of similar sequences. This strategy facilitates the simplification of complex datasets and enables the breakdown of the complexity of the processes under study [3]. DFGs have been created in Python, using `Dash`, `Cytoscape` and `distinctipy`.

General structure (L1). Event sequence visualizations and DFGs use a different visual encoding to reveal the general structure and dominant patterns in a dataset. Figure 2 (*left*) shows the complete road fine dataset, with height encoding frequency. This visualization allows for an easy identification of recurrent structures and frequent and infrequent sequences. The vertically aligned bars and stacked events reveal the key activities (CF, SF, IFN, AP, SFC). The most commonly repeated patterns are exposed (e.g., CF → SF; CF → SF → IFN → AP → SFC; CF → SF → IFN → AP → PAY; and CF → PAY), also revealing the proportion of infrequent sequences and their complexity. This allows the quick identification of those sequences that deviate the most from the most common patterns. In contrast, the corresponding DFG representation Figure 2 (*right*), despite encoding the frequency of occurrence of activities and transitions, is complex to interpret. The high connectivity and visually dense network obscure the identification of how sequences progress. While the DFG effectively captures all possible transitions between events, its capacity to reveal high-level structure is limited.

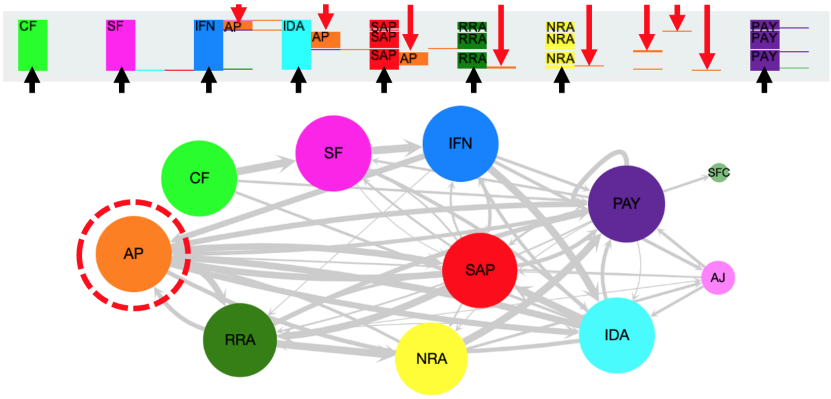

Fig. 3. Timeline-based event sequence visualization (*top*) vs. DFG (*bottom*) for 516 traces in the road traffic fines dataset [9]. The activity of interest AP is highlighted using *red arrows and circle*, and the alignment events using *black arrows*.

Parallelism (L2) Event sequence visualizations and DFGs represent differently concurrent activities, i.e., activities that can be run in parallel and may appear in varying orders across traces in the event log. Figure 3 shows an example where the activity AP occurs repeatedly over time (*red* arrows), and is not always preceded by the same activity. AP follows activities as distinct as IFN, IDA, SAT, RRA, and NRA. This suggests that AP does not exhibit a strong dependency on specific prior events and can be interleaved with a wide range of other activities. This representation is particularly effective in revealing this positional variability.In this case, AP might have been triggered after a pre-established period and after checking that a fine has not yet been paid in full. To obtain the visualization in Figure 3, event alignment (*black* arrows) has been used to align and group the activities of interest that have been identified as precursors of AP, namely, IFN, IDA, SAT, RRA and NRA. Alignment is often used in event sequence visualization as a means to explore temporal order. To achieve alignment, additional spaces are inserted between the events of interest selected as alignment events. In this case, we chose IFN, IDA, SAT, RRA and NRA to explore their relationship with AP. While alignment maintains the order of activities, it introduces visual gaps that do not represent real time. These gaps support alignment and visual comparison, but they may reduce the visibility of the broader process context by spacing activities apart.

The corresponding DFG representation in Figure 3(*bottom*) abstracts away temporal ordering in favor of aggregated transition relations. While the AP node has incoming edges from nearly all other activities, except for CF, SF, and SFC, indicating high in-degree, the graph structure cannot communicate when and how frequently AP occurs in different contexts. As a result, the DFG lacks the expressiveness needed to assess the degree to which AP is executed concurrently with other activities in the process.

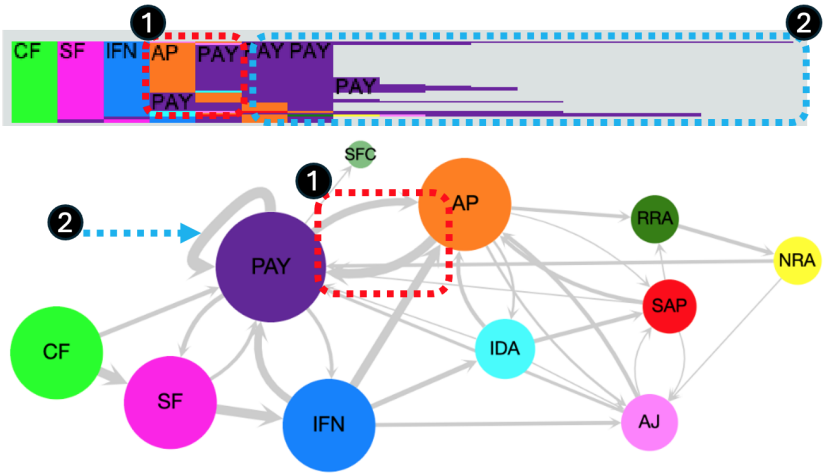

Fig. 4. Timeline-based event sequence visualization (*top*) vs. DFG (*bottom*) for 170 traces in the road traffic fines dataset [9].

Precision (L3). Event sequence visualizations visualize behaviours that occurred in the event log, whereas DFGs allow for behaviors that may have never occurred in the event log (see Section 1). This forces the analyst to use strategies to confirm whether a certain behavior in the DFG has occurred or not. Next, we show how event sequence visualizations can support the analyst in this task.

Cycles vs order. The area highlighted (*red rectangle*) in the DFG in Figure 4 illustrates a typical example of a cycle between two activities: PAY and AP. However, simply looking at the DFG does not reveal whether this is a true cycle, i.e., whether traces exist that include multiple alternating occurrences of penalties and payments, such as ... → AP → PAY → AP → PAY. In contrast, the event sequence visualization provides a clearer picture of the actual behavior recorded in the event log and makes it clear that such a cycle does occur. The highlighted area (*red rectangle*) in the event sequence visualization shows that multiple payments (PAY) can occur within the same case, while penalties (AP) are applied only once.

Focus on a single activity. When analyzing process data, a typical scenario involves characterizing the sequence of activities that lead to the execution of a particular activity. For instance, in our case, we might be interested in identifying which activity sequences typically precede or avoid the activity Send for Credit Collection (SFC), which indicates that a case has been forwarded to a credit collection agency due to non-payment of a fine. This kind of analysis is difficult to perform directly using a Directly-Follows Graph (DFG), which aggregates control-flow relations without preserving trace-level detail. For instance, the DFG in Figure 2 shows that SFC is most commonly preceded by AP and PAY, and occasionally by NRA, RRA, AJ, or SAP. However, due to the limited support of DFGs for path disambiguation, it is not possible to reconstruct

the full context or differentiate between alternative sequences that converge on SFC. In contrast, the event sequence visualization in the same figure keeps the temporal ordering of individual traces. This allows analysts to visually identify common and exceptional paths that lead to SFC. For example, it becomes apparent that SFC frequently follows the path CF → SF → IFN → AP → SFC, and less commonly, CF → SF → IFN → AP → PAY → SFC. It also becomes clear that certain paths—such as CF → PAY (without SF) or CF → SF → PAY (without IFN)—systematically avoid SFC. Such insights are possible because the event sequence visualization preserves trace-level granularity and visually encodes sequence variation.

Repetitions (L4). One key distinction between event sequence visualizations and DFGs lies in their ability to represent consecutive repetitions of events. In the event sequence visualization of Figure 4(*top*), the area highlighted (*blue rectangle*) shows clearly that the event PAY occurs repeatedly and in immediate succession. The visual encoding of sequential order along the horizontal axis facilitates the identification of repetitions but also their exact frequency and position within the temporal sequence. In contrast, the DFG representation (*bottom*) abstracts away the temporal order. The presence of a self-loop on the node PAY (*blue arrow*) indicates that PAY can transition to itself, but this fails to convey how frequently or when this transition occurs within the cases containing this activity. While the graph effectively summarizes possible transitions, it omits critical information about the temporal ordering of events.

Combining DFGs and timeline-based visualizations. Both offer complementary insights for process analysis. DFGs provide a high-level overview of all possible transitions, highlighting activities with high connectivity, identifying self-transitions and potential repetitions. Timeline-based visualizations reveal dominant patterns, highlighting frequency of events, and exposing the main paths leading to or avoiding key activities. Together, these views enable a more nuanced and complete understanding of processes.

4 Conclusions

This work contributes to the emerging field of visual process analytics. We examined four limitations of DFGs, commonly used in PM, and illustrated them through some examples in the Road Traffic Fines Management dataset. We have shown how a timeline-based visualization, commonly used in VA for event sequence analysis, offers more interpretable insights by preserving trace-level detail and highlighting patterns that are otherwise obscured by the aggregated structure of DFGs. However, these visualizations can be difficult to interpret with increasing number of sequences and events, often suffering from visual clutter.

Our paper highlights how this type of visualizations can address DFGs limitations, but they are not a replacement. DFGs can reveal infrequent yet important behaviors that may be overlooked in timeline-based visualizations due to

frequency-based scaling. Both approaches are complementary. As future work, we aim to develop interactive mechanisms to support their seamless coordination.

References

1. van der Aalst, W.M.P.: A practitioner's guide to process mining: Limitations of the directly-follows graph. Procedia Comput. Sci. **164**, 321–328 (2019)
2. Bernard, G., Andritsos, P.: VisuELs: Visualization of Event Logs (Extended Abstract). In: ICPM Conference (2021)
3. Bose, R.P.J.C., van der Aalst, W.M.P.: Trace Clustering Based on Conserved Patterns: Towards Achieving Better Process Models. In: Business Process Management Workshops. pp. 170–181. Springer, Berlin, Heidelberg (2010)
4. Capitán-Agudo, C., Salas-Urbano, M., Cabanillas, C., Resinas, M.: Analyzing How Process Mining Reports Answer Time Performance Questions. In: Business Process Management, vol. 13420, pp. 234–250. Springer International Publishing (2022)
5. Chen, Y., Xu, P., Ren, L.: Sequence Synopsis: Optimize Visual Summary of Temporal Event Data. IEEE Trans. Vis. Comput. Graph. **24**(1), 45–55 (2018)
6. Du, F., Shneiderman, B., Plaisant, C., Malik, S., Perer, A.: Coping with volume and variety in temporal event sequences: Strategies for sharpening analytic focus. IEEE Trans. Vis. Comput. Graph. **23**(6), 1636–1649 (2016)
7. Guo, Y., Guo, S., Jin, Z., Kaul, S., Gotz, D., Cao, N.: Survey on Visual Analysis of Event Sequence Data. IEEE Trans. Vis. Comput. Graph. (2022)
8. Jalali, A., Johannesson, P., Perjons, E., Askfors, Y., Rezaei Kalladj, A., Shemeikka, T., Vég, A.: dfgcompare: a library to support process variant analysis through Markov models. BMC Med. Inform. Decis. Mak. **21**(1), 356 (2021)
9. de Leoni, M., Mannhardt, F.: Road traffic fine management process (2015). `https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5`
10. Liu, Z., Kerr, B., Dontcheva, M., Grover, J., Hoffman, M., Wilson, A.: CoreFlow: Extracting and Visualizing Branching Patterns from Event Sequences. Comput. Graph. Forum **36**(3), 527–538 (2017)
11. Magallanes, J., Stone, T., Morris, P.D., Mason, S., Wood, S., Villa-Uriol, M.C.: Sequen-C: A Multilevel Overview of Temporal Event Sequences. IEEE Trans. Vis. Comput. Graph. **28**(1), 901 – 911 (2022)
12. Miksch, S., Di Ciccio, C., Soffer, P., Weber, B.: Visual Analytics Meets Process Mining: Challenges and Opportunities. IEEE Comput. Graph. **44**, 132–141 (2024)
13. Schuster, D., Zerbato, F., van Zelst, S.J., van der Aalst, W.M.P.: Defining and visualizing process execution variants from partially ordered event data. Information Sciences **657**, 119958 (2024)
14. van den Elzen, S., Jans, M., Martin, N., Pieters, F., Tominski, C., Villa-Uriol, M.C., van Zelst, S.J.: Towards multi-faceted visual process analytics. Information Systems **133**, 102560 (2025)
15. van der Linden, S., de Fouw, E., van den Elzen, S., et al.: A survey of visualization techniques for comparing event sequences. Computers & Graphics (2023)
16. Vrotsou, K., Nordman, A.: Exploratory Visual Sequence Mining Based on Pattern-Growth. IEEE Trans. Vis. Comput. Graph. **25**(8), 2597–2610 (2019)
17. Wongsuphasawat, K., Guerra Gómez, J.A., Plaisant, C., Wang, T.D., Taieb-Maimon, M., Shneiderman, B.: LifeFlow: visualizing an overview of event sequences. In: SIGCHI. pp. 1747–1756. ACM (2011)

18. Yeshchenko, A., Mendling, J.: A survey of approaches for event sequence analysis and visualization. Information Systems **120**, 102283 (2024)