Interactive Strategies for Coordinated Exploration of Timeline and Directly-Follows Graph Visualizations

L. Montana [1] and M. Resinas[2] and M.-C. Villa-Uriol[1]

[1]School of Computer Science, University of Sheffield, UK
[2]SCORE Lab, Universidad de Sevilla, Spain

Abstract

Visual Process Analytics (VPA) is an emerging discipline where Process Mining (PM) and Visual Analytics (VA) experts collaborate to simplify the analysis and enhance the understanding of complex processes. In practice, process exploration is often centered around Directly-Follows Graphs (DFGs), which visualize how activities follow each other in a process. While DFGs are easy to compute and interpret, their aggregated nature and limited behavioral semantics introduce several limitations during analysis, including difficulties in understanding overall process structure, distinguishing parallelism from cyclic behavior, identifying precise execution paths, and determining repetition patterns. In previous work, we showed that timeline-based visualizations can complement DFGs by preserving trace-level detail and revealing patterns that are often obscured in DFGs. However, interaction mechanisms that allow analysts to effectively coordinate these complementary views during process exploration remain largely unexplored. In this paper, we address this gap by proposing four visual analytics strategies that coordinate DFG and timeline visualizations using interactivity. These strategies enable analysts to identify structural patterns, inspect relations between activities, and trace concrete execution paths by combining the structural overview provided by DFGs with the trace-level perspective of timeline visualizations. We demonstrate the usefulness of these strategies through their application to the Road Traffic Management Process dataset, a well-known benchmark in the process mining community.

CCS Concepts

• ***Human-centered computing*** → ***Visualization;*** *Interaction design;* • ***Information systems*** → ***Decision support systems; Data mining;***

1. Introduction

In recent years, visual process analytics has emerged as an approach that stresses the role of interactive visualizations in the analysis of complex process data [MDCSW24, vJM*25]. By integrating process mining (PM) techniques with principles from visual analytics (VA), it aims to make process data more accessible and actionable for analysts. This perspective moves beyond static process representations by introducing interactive visual strategies that enable users to explore, compare, and interpret process behavior.

In practice, process exploration is frequently centered around process maps such as Directly-Follows Graphs (DFGs), where nodes represent activities and edges encode temporal succession between them [vJM*25]. DFGs are easy to compute and relatively simple to interpret, which makes them a popular starting point for process exploration. However, their simplicity also leads to limitations that can hinder the analysis of complex processes [vdA19].

Recent work has highlighted important limitations of DFGs for process exploration and proposed timeline-based visualizations as a complement [MRVU26]. However, interaction mechanisms that support the coordinated use of both visualizations during process exploration are still missing. In this paper, we address this gap by proposing four visual analytics strategies that build on established information visualization concepts, particularly coordinated multiple views and supporting interaction techniques, to coordinate timeline and DFG visualizations during process exploration. These strategies help analysts overcome the known limitations of DFGs by combining their aggregated overview with the trace-level perspective provided by timeline visualizations. We illustrate their usefulness through an application to a well-known dataset in process mining.

2. Background and Related Work

2.1. DFGs in process mining

In PM tools, DFGs are the most commonly used visualization for exploring event logs [LPW19]. They are widely adopted [vJM*25, CASUCR22] because they are easy to compute and intuitive to read. However, their aggregated nature and limited behavioral semantics introduce limitations that become apparent during common exploration tasks [vdA19, LPW19, MRVU26]. To overcome these, analysts resort to time-consuming activities involving the use of

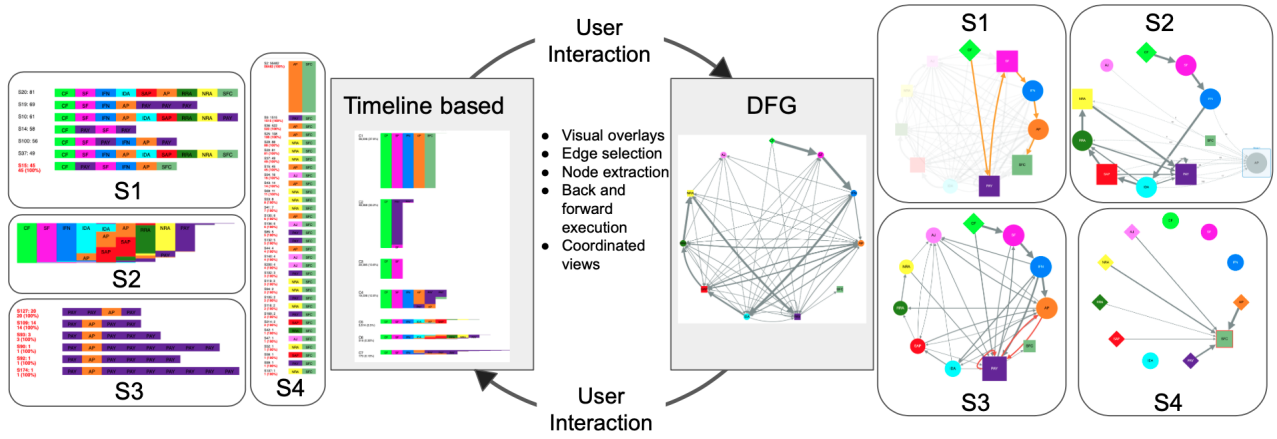
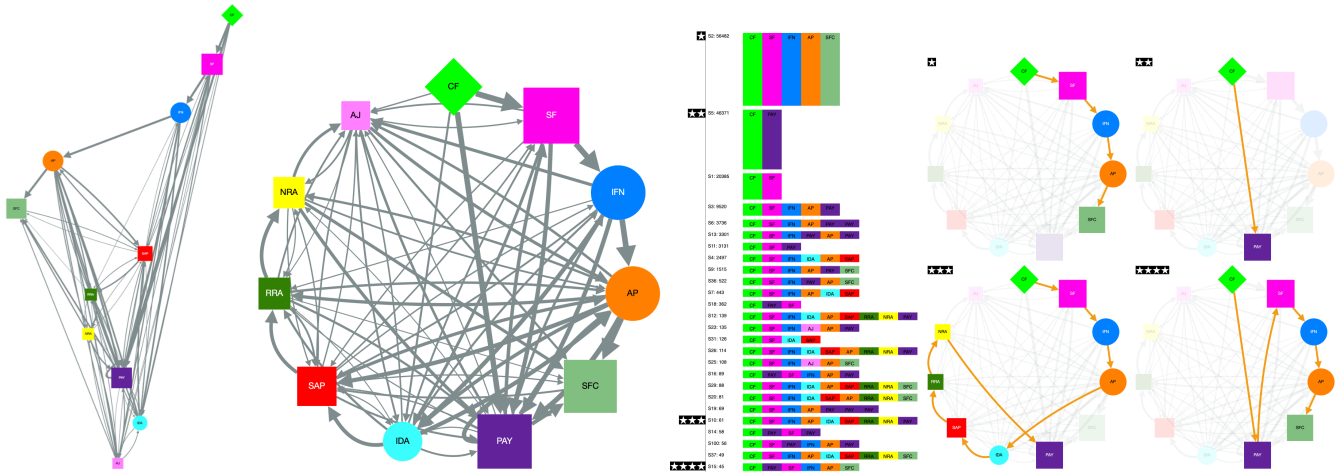


Figure 1: Visual analytics strategies (S1-S4) and how they link the timeline and DFG views.



(a) (b) (c) (d)

Figure 2: Road Traffic Fines Management dataset represented as a DFG using (a) hierarchical and (b) circular layouts, (c) timeline view with the most frequent variants. Hovering over the four variants marked with ★, produces (d) these four DFG views.

filters, access to domain-knowledge about the processes, and the manual inspection of individual process variants [CASUCR22].

A first common task is to obtain an *overview of the general structure* of the process. As the number of variants grows, the DFG accumulates edges, producing dense "spaghetti-like" graphs in which dominant execution paths become hard to identify. The use of frequency thresholds for simplification may hide relevant behavior and lead to misinterpretations [vdA19, LPW19].

A second task consists of *interpreting the behavioral relations between activities*. DFGs do no explicitly capture behavioral semantics but analysts often assume that consecutive events reflect meaningful behavioral relations, which might not be always the case. Sometimes activities occur in parallel with the main process flow as they might have been triggered by temporal constraints or external routines rather than by the immediately preceding activity. In other situations, DFG loops might not represent genuine repetitions, but instead arise from parallelism in the event log [vdA19].

Finally, analysts often attempt to *trace concrete execution sequences* from the DFG, for example to determine the paths leading to a specific activity or to verify whether a particular sequence actually occurs in the event log. Because DFGs aggregate relations across traces, the graph may permit paths that never occur in any single case [LPW19].

2.2. DFGs and timeline-based visualizations

In our earlier work [MRVU26], we showed how timeline-based visualizations [WGGP*11, MSM*22], combined with hierarchical clustering [MSM*22] and sequence alignment, can support analysts in addressing these exploration tasks. Timeline visualizations preserve trace-level detail while organizing traces to reveal patterns and similarities, enabling analysts to examine the underlying event sequences directly. This perspective helps uncover patterns that are often obscured in aggregated representations such as DFGs. In particular, event sequence visualizations allow analysts to disam-

biguate whether apparent cycles correspond to repeated activities or concurrent behavior, inspect the concrete sequences leading to a specific activity, and determine how many times an activity is repeated within a case [MRVU26] (see Figures 4c, 5a, 4b). Despite their complementarity, seamless interaction mechanisms for coordinating timeline visualizations and DFGs during process exploration remain underdeveloped.

2.3. Interactive graph operations

As graphs grow in size and connectivity, the interactions available to users become as important as the graph itself. Several surveys have organized graph interaction techniques according to whether they affect the view, the visual abstraction, or the underlying data [LKS*10, NMSL19]. Based on these, we focus on four operations relevant to understanding local structure in directed graphs: *highlighting*, *node extraction*, *edge selection*, and *path tracing*.

Highlighting is a common operation in graph visualization. Selecting a node typically emphasizes it and its immediate neighborhood [LKS*10], and more sophisticated variants allow selecting arbitrary groups using methods such as lasso or hotbox [MJ09]. Overloaded views can also serve a highlighting function, for instance by drawing convex hulls [NMSL19] or curves [ARRC11] around clusters on top of the graph. While these techniques draw attention to local structure, they do not reduce visual complexity, as all nodes and edges remain visible, making dense graphs still difficult to read.

Node extraction, which temporarily removes a node and its edges to simplify the layout, relates to the broader family of interactive graph aggregation and filtering techniques. GrouseFlocks [AMA08] lets users merge nodes into metanodes to create hierarchical abstractions, TS-Extractor [FMW*21] computes a relevant subgraph around user-selected nodes by combining topological and semantic criteria, and Envisage [WFH*26] retrieves matching instances using user-specified subgraph patterns. However, these systems do not allow users to manually extract nodes and combine multiple extractions interactively during exploration.

Edge selection goes beyond highlighting by filtering which edges are displayed. Common approaches include showing edges only for selected nodes [vdEvW14] or using spatial edge lenses that displace overlapping edges in a focus region [WCG03, TAvHS06]. While these techniques reduce local clutter, they operate on individual edges or on the edges of a single node. Selecting an edge does not tell the user anything about where that transition appears, or how often it occurs across different sequences.

Path tracing, which progressively reveals incoming or outgoing connections from a selected node, relates to bottom-up navigation strategies [LKS*10], where exploration starts from a single node and expands outward based on graph structure or a degree-of-interest function [vHP09]. However, these approaches typically do not distinguish between incoming and outgoing connections. In directed graphs, this makes it difficult to selectively trace predecessors or successors at a controlled depth, for example when asking "what led to this activity?".

Existing work supports each of these operations to some degree, but they are typically offered as isolated interactions within the graph view itself. The potential for coordinating them with a separate view, where selections in one drive the response in the other and vice versa, remains largely unexplored for DFGs.

3. Proposed Strategies

Building on our earlier work, we present how *interactive VA operations* and *coordinated views* can bring together *timeline-based* and *DFG* visualizations. We illustrate how they can help analysts overcome the problems of identifying general structure, parallelism, repetitions and precision that appear while performing the three exploration tasks identified in the literature:

T1 Identify where specific execution paths occur in a DFG.
T2 Interpret behavioral relations and interactions among activities.
T3 Determine the possible paths leading to or following a specific activity up to a specified depth within a DFG.

The four interactive VA strategies are: on demand overlay of selected patterns (**S1**), node extraction (**S2**), edge selection (**S3**), and tracing back and forward executions from a single activity (**S4**). These strategies are triggered through user interactions in either the timeline or the DFG view (Figure 1). The resulting selections and highlights are then reflected across both visualizations, allowing analysts to simultaneously explore the process structure in the DFG and the trace-level sequences in the timeline. In this way, both perspectives remain coordinated and mutually informative.

To produce the supporting figures, we have used the Road Traffic Fine dataset [dLM15], which contains information about traffic fine management by a police force in Italy during 13 years, including 150,370 cases, 561,480 events and 11 event types. Events are represented using a consistent color encoding, common to the *timeline* and *DFG* visualizations, being: AP (Add Penalty), AJ (Appeal to Judge), CF (Create Fine), IDA (Insert Date Appeal to Prefecture), IFN (Insert Fine Notification), NRA (Notify Result Appeal to Offender), PAY (Payment), RRA (Receive Result Appeal from Prefecture), SAP (Send Appeal to Prefecture), SF (Send Fine), and SFC (Send for Credit Collection).

In the *DFGs*, node shape indicates the role of the activity: diamonds (◇) mark start activities, squares (□) mark end activities, triangles (△) mark activities that appear as both start and end, and circles (○) represent all other activities. Node size and edge width are scaled logarithmically according to frequency, and we overlay the values when the user hovers the mouse over a node or an edge.

3.1. On-demand overlay of selected patterns (S1)

As the number of activities and transitions grows, identifying specific execution paths in a DFG (**T1**) becomes difficult due to overlapping edges. Filtering can help, but may discard potentially relevant information. To address this, we propose linking the timeline and DFG views: hovering over a variant or trace in the timeline view *highlights the corresponding edges in the DFG*, while the rest of the graph fades out. As this interaction does not alter the graph, it can be combined with existing filtering techniques, allowing less aggressive filters while individual paths are still present.

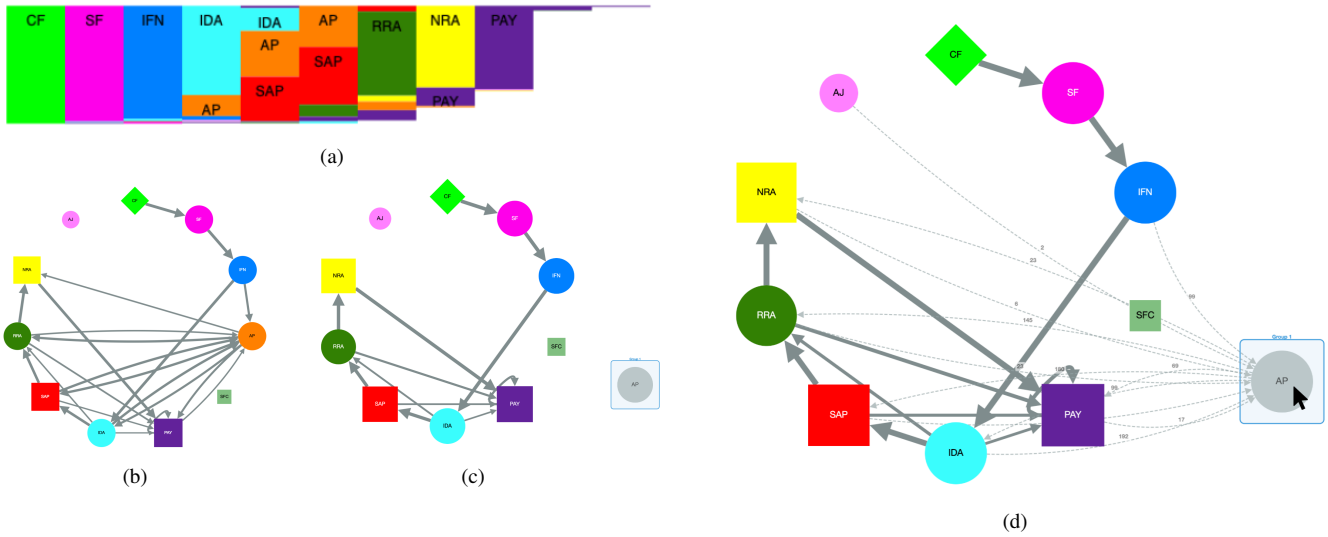


Figure 3: Node extraction example. (a) Timeline visualization for the road traffic fines dataset (516 traces).(b) Main DFG with edge filtering to remove the least frequent edges. (c) AP event is extracted from the main DFG to reduce its complexity. (d) Hovering the extracted node (AP) highlights its connections to the original DFG.

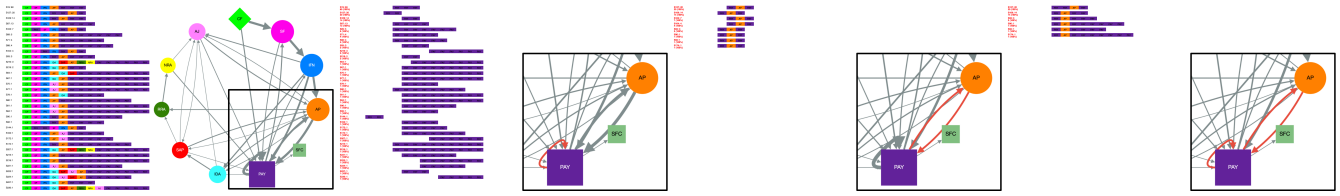


Figure 4: Using edge selection to investigate repeating activities and loops. (a) Timeline and DFG visualizations for 170 traces. (b) Selecting in the DFG the PAY self-loop (in red) shows where repetitions occur in the timeline and their length. (c) Selecting the edges between AP and PAY (edges in red) confirms that the cycle exists in the timeline view. (d) Combining (b) and (c) shows all variants with a loop between AP and PAY, and a self-loop for PAY.

Because the timeline lays out activities sequentially, recurring patterns and frequent variants are easy to spot, making it a natural starting point for exploring paths in the DFG. Figure 2 illustrates this idea with the Road Traffic Fines dataset. Figures 2a and 2b show the DFG of the entire dataset using two types of layouts and Figure 2c shows the most frequent variants. Figure 2d highlights four variants in the DFG when hovering over them in the timeline view. This interaction turns the DFG from a static and often overwhelming diagram into something the user can actively interrogate, one variant at a time.

3.2. Node extraction (S2)

Interpreting behavioral relations and interactions among activities (**T2**) can be challenging in DFGs. Although DFGs do not explicitly capture behavioral semantics, analysts often assume that consecutive events are semantically related. However, some activities occur in parallel with the main process flow and therefore challenge this assumption. As a result, these activities might appear connected to many nodes in the DFG, producing highly connected structures that obscure the dominant behavior of the process, even after removing infrequent edges.

To reduce the clutter, instead of just fading or hiding edges, we propose a *node extraction* strategy where the user can temporarily restructure the DFG. By pulling weakly coupled activities out of the main graph, the visualization relaxes the assumption that all directly-follows relations are meaningful for the process structure. This can help reveal the dominant sequential flow more clearly while still preserving access to the extracted activity and its connections when needed. This is not a purely visual strategy. Extracting nodes requires recalculating the DFG, which might include the creation of new edges bypassing the extracted node (shown in green, as seen in Figure 5b).

Figure 3 illustrates this with the AP activity in the Road Traffic Fines Management dataset. AP is triggered a fixed number of days after the IFN activity, regardless of the intermediate activities that might occur in between. As a result, the activity becomes connected to many unrelated parts of the graph, even though these connections do not necessarily represent meaningful control-flow dependencies.

Figure 3b shows the original DFG with frequency filtering applied; the graph is readable but still tangled around AP. Together, the DFG and the timeline (Figure 3a) reveal a dominant path through the process. The second graph (Figure 3c) shows the result of extracting AP; the remaining nodes settle into a much cleaner layout, and the core process flow becomes immediately apparent. The third DFG (Figure 3d) shows what happens when the user hovers over the extracted AP node, its edges reappear, revealing its high connectivity.

3.3. Edge selection (S3)

The *edge selection* strategy also supports the interpretation of behavioral relations between activities (**T2**). It links the DFG to the timeline view to allow analysts to inspect how specific transitions occur in the underlying traces. Selecting one or more edges in the DFG highlights the variants containing those transitions in the timeline, revealing where the transition appears within each sequence, how often it repeats, and which cases exhibit the behavior.

Figure 4 illustrates this with three scenarios involving the PAY and AP activities. The first selection (Figure 4b) is a self-loop on PAY; the DFG shows that PAY sometimes follows itself, but the timeline reveals the actual repetition patterns. The second selection (Figure 4c) targets the transitions between AP and PAY in both directions; the timeline shows whether individual sequences cycle between the two activities or only pass through one of the transitions. The third selection (Figure 4d) combines both, the self-loop and the back-and-forth transitions, giving a complete picture of the interplay between PAY and AP. In each case, the DFG identifies what to look for, and the timeline updates to reflect what actually happens.

3.4. Tracing back and forward from a single activity (S4)

In a dense DFG, understanding what leads to an activity of interest (**T3**) is difficult because all incoming and outgoing edges are displayed at once, making the relative importance of each path hard to assess. To address this, we propose a path exploration mechanism that lets the user select a node and progressively trace its incoming or outgoing connections by controlling the depth from immediate predecessors to the complete path. At each depth level, edges carry frequency information, making it straightforward to identify the most common paths leading to the selected activity.

This strategy can be combined with the other strategies to uncover execution patterns that would be difficult to identify using a single technique. We demonstrate this with two examples that trace back predecessors of the Send For Credit Collection (SFC) activity. In the first example (Figure 5a - 5b), we select SFC and select a depth of 1 to show the immediate predecessors. This filters the variants in the timeline to show only the subsequences that end with SFC. The timeline reveals that the majority of sequences arrive at SFC through the Add Penalty (AP) activity. However, as discussed earlier, AP is a time-dependent activity that can appear between almost any pair of nodes. To look past this intermediary, we extract AP from the main DFG (Figure 5b). This extraction triggers the formation of new edges connecting to SFC that bypass the AP node. In our example, only a new edge is formed between IFN and SFC, represented as a green dashed arrow. The timeline confirms this: with AP removed, the majority of variant subsequences now show IFN as the immediate predecessor of SFC, revealing the dominant execution path is IFN -> SFC.

In the second example (Figure 5c - 5d), we select SFC again but increase the depth to 3, exposing more edges connecting the activities leading to SFC. The DFG becomes much denser and the timeline displays longer sequences, still ending in SFC. From the timeline, we then use on-demand overlay of selected patterns to visualize them in the DFG. Hovering over a particular variant (Figure 5d) highlights a subsequence with a self-loop of Payment (PAY) followed by SFC. This pattern would be difficult to find using the timeline or the DFG individually.

4. Conclusions

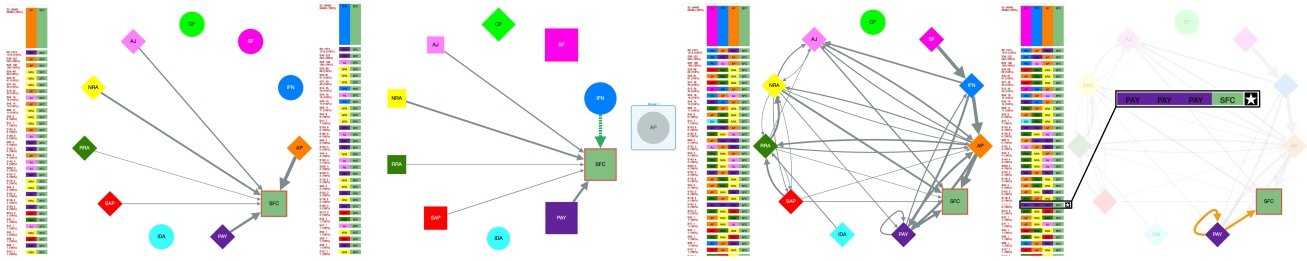
DFGs are routinely used in process mining to explore processes and obtain an overview of the relations between activities in an event log. While they effectively aggregate execution paths, their simplicity can also lead to ambiguities when attempting to understand the finer details of process behavior. In particular, analysts can encounter problems related to understanding the overall process structure, identifying parallelism and repetitions, and dealing with the limited behavioral precision of DFGs when relying on them alone.

In an earlier work [MRVU26], we showed how timeline-based visualizations can complement DFGs by preserving trace-level information that is often hidden in aggregated representations. Building on this idea, this paper proposes several visual analytics strategies that combine timeline-based visualizations and DFGs through coordinated views and interactive exploration. We introduced four interaction strategies: on-demand overlays of selected patterns, node extraction, edge selection, and forward/backward tracing from individual activities. These strategies facilitate a more complete understanding of process behavior, enabling analysts to identify where specific execution paths occur, interpret behavioral relations between activities, and trace possible paths leading to or following a given activity.

This work represents a step toward more integrated visual process analytics environments that combine complementary representations of process data. Two limitations of the current work motivate future research: we have not yet evaluated the proposed strategies with process analysts, and the scalability of the approach to large event logs remains to be examined. A systematic user study is needed to assess the effectiveness of the strategies during real analysis tasks. We also plan to explore the use of separate graphs for different behavioral relations and alignment strategies for timeline visualizations based on frequent events.

Acknowledgements

This work started at Schloss Dagstuhl (Leibniz-Zentrum für Informatik), seminars 23271 “Human in the (Process) Mines” and 25152 “Multi-Faceted Visual Process Mining” and is part of the projects PID2021-126227NB-C21/ AEI/10.13039/501100011033/ FEDER, EU, and PID2024-156482NB-I00, funded by MICIU/AEI/10.13039/501100011033 and by the ESF+.



(a) (b) (c) (d)

Figure 5: Trace-back analysis for SFC. (a) Tracing back one activity for SFC reveals that AP is the most common activity. (b) Extracting AP creates a bypass edge (in green) from IFN, and the timeline view reflects the new connections. (c) Selecting SFC with a deeper trace-back exposes longer subsequences in the timeline. (d) Hovering over a variant (★) highlights a self-loop of PAY followed by SFC in the DFG.

References

[AMA08] ARCHAMBAULT D., MUNZNER T., AUBER D.: Grouse-Flocks: Steerable Exploration of Graph Hierarchy Space. *IEEE Transactions on Visualization and Computer Graphics 14*, 4 (July 2008), 900–913. doi:10.1109/TVCG.2025.3634234. 3

[ARRC11] ALPER B., RICHE N., RAMOS G., CZERWINSKI M.: Design Study of LineSets, a Novel Set Visualization Technique. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (Dec. 2011), 2259–2267. 3

[CASUCR22] 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. Springer International Publishing, 2022, pp. 234–250. doi:10.1007/978-3-031-16103-2_17. 1, 2

[dLM15] DE LEONI M., MANNHARDT F.: Road traffic fine management process, 2015. doi:10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5. 3

[FMW*21] FU K., MAO T., WANG Y., LIN D., ZHANG Y., ZHAN J., SUN X., LI F.: TS-Extractor: large graph exploration via subgraph extraction based on topological and semantic information. *Journal of Visualization 24*, 1 (2021), 173–190. doi:10.1109/TVCG.2011.186. 3

[LKS*10] LANDESBERGER T. V., KUIJPER A., SCHRECK T., KOHLHAMMER J., WIJK J. J. V., FEKETE J.-D., FELLNER D. W.: Visual Analysis of Large Graphs. *Computer Graphics Forum* (2010). doi:10.2312/egst.20101061. 3

[LPW19] LEEMANS S. J., POPPE E., WYNN M. T.: Directly Follows-Based Process Mining: Exploration & a Case Study. In *2019 International Conference on Process Mining (ICPM)* (Aachen, Germany, June 2019), IEEE, pp. 25–32. doi:10.1109/ICPM.2019.00015. 1, 2

[MDCSW24] MIKSCH S., DI CICCIO C., SOFFER P., WEBER B.: Visual Analytics Meets Process Mining: Challenges and Opportunities. *IEEE Comput. Graph. 44* (2024), 132–141. doi:10.1109/MCG.2024.3456916. 1

[MJ09] MCGUFFIN M. J., JURISICA I.: Interaction Techniques for Selecting and Manipulating Subgraphs in Network Visualizations. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (Nov. 2009), 937–944. doi:10.1109/TVCG.2009.151. 3

[MRVU26] MONTANA L., RESINAS M., VILLA-URIOL M.-C.: Addressing directly-follows graphs limitations with visualization of event sequences. In *Business Process Management Workshops* (2026), Springer Nature Switzerland, pp. 573–581. doi:10.1007/978-3-032-13426-4_42. 1, 2, 3, 5

[MSM*22] 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 (2022), 901 – 911. doi:10.1109/TVCG.2021.3114868. 2

[NMSL19] NOBRE C., MEYER M., STREIT M., LEX A.: The State of the Art in Visualizing Multivariate Networks. *Computer Graphics Forum 38*, 3 (2019), 807–832. doi:10.1111/cgf.13728. 3

[TAvHS06] TOMINSKI C., ABELLO J., VAN HAM F., SCHUMANN H.: Fisheye Tree Views and Lenses for Graph Visualization. In *10th International Conference on Information Visualisation (IV'06)* (July 2006), pp. 17–24. doi:10.1109/IV.2006.54. 3

[vdA19] VAN DER AALST W. M. P.: A practitioner's guide to process mining: Limitations of the directly-follows graph. *Procedia Comput. Sci. 164* (2019), 321–328. doi:10.1016/j.procs.2019.12.189. 1, 2

[vdEvW14] VAN DEN ELZEN S., VAN WIJK J. J.: Multivariate Network Exploration and Presentation: From Detail to Overview via Selections and Aggregations. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (Dec. 2014), 2310–2319. doi:10.1109/TVCG.2014.2346441. 3

[vHP09] VAN HAM F., PERER A.: "Search, Show Context, Expand on Demand": Supporting Large Graph Exploration with Degree-of-Interest. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (Nov. 2009), 953–960. doi:10.1109/TVCG.2009.108. 3

[vJM*25] 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* (2025), 102560. doi:10.1016/j.is.2025.102560. 1

[WCG03] WONG N., CARPENDALE S., GREENBERG S.: Edgelens: an interactive method for managing edge congestion in graphs. In *IEEE Symposium on Information Visualization 2003* (Oct. 2003), pp. 51–58. doi:10.1109/INFVIS.2003.1249008. 3

[WFH*26] WEN X., FU Q., HAN S., GUO Y., LIU J. K., WANG Y.: Envisage: Towards Expressive Visual Graph Querying. *IEEE Transactions on Visualization and Computer Graphics 32*, 1 (Jan. 2026), 593–603. doi:10.1109/TVCG.2025.3634234. 3

[WGGP*11] 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* (2011), ACM, pp. 1747–1756. doi:10.1145/1978942.1979196. 2