University of Amsterdam

Towards Effective User Interfaces for Complex Agent-Based Models: Building a Criminal Network Visualisation and Simulation Tool for Law Enforcement Agencies

Individual Project 5284INPR6Y

Sándor Battaglini-Fischer

Examiner: Prof. Rick Quax

Supervisor: Laurens Stuurman

November 4, 2024

The visualisation of complex criminal networks requires tools that can handle large, dynamic datasets and provide real-time, interactive features tailored to the needs of Law Enforcement Agencies. This paper presents an improvement of *CrimeSeen*, a modular web-based platform for visualising and simulating criminal networks using agent-based models. Key enhancements include a new menu for manipulating the visual properties, enabled through a modular React-based design that ensures performance and scalability, enhanced graph layouts and better handling of nodes and edges.

Law enforcement agencies (*LEAs*) across the world are increasingly using data driven tools to aid traditional crime fighting techniques, ranging from real-time surveillance and crime scene evaluation to predictive policing and the analysis of criminal patterns and networks (1). While *LEAs* in the US tend to outsource their software development to large third-party corporations, which already have a plethora of tools to handle, analyse and visualise big datasets, in Europe and specifically The Netherlands the development of such tools is more segmented and often handled in-house or by smaller teams (2). This type of procedure creates the need for generalised frameworks and publicly available practices to construct scalable applications.

One of the essential tasks in this domain is visualising large networks of criminals. While software and packages for this exist (Gephi, NetworkX, graph-tool), they don't allow for the interactive visualization of dynamic network simulations, they run locally (which makes collaboration difficult) and are limited in their customisation capabilities, which makes them difficult to adapt the specific needs of *LEAs*. Web technologies exist, but lack the privacy options of a self-hosted setup. A custom web-based application for network visualisation offers real-time interaction, mobile accessibility, a UI/UX adapted to the specific needs of the the customer, scalability and control over the confidential data. Especially in the realm of *LEAs*, the networks are large, complex and dynamic, containing multiple layers of information. This information is sourced from real-life crime reports and expert interviews and used to build an agent-based models to inform the simulations. It is essential for efficient crime management to be able to visualise and analyse this data rapidly, as well as implement simulations for predictive policing. This calls for well thought-out tools and practices.

1. Background/Project Goals

This project is built upon the the existing *CrimeSeen* setup by the collaboration of the University of Amsterdam with the police of Amsterdam (see section *2*B). Liza Roelofsen had conducted user testing on it and conceived a list of milestones to implement, which was used as a basis here.

A. Data acquisition and model setup. When implementing data visualisations, it is imperative to understand the underlying data structures first. In the case of the criminal networks used in this project, the network is generally constructed from qualitative data. The data pipeline from unstructured interviews and case studies to the model setup and validation is developed according to the principles of the *Framework for Expert-Informed Data-Driven Agent-Based Models* (FREIDA(3)). This consists of a multi-step process: first, the information is gathered from case files and interviews to define a model design. This conceptual model is then transformed into a computational model, focusing on the agents, their behaviour and the environment. During the validation step, the model is tested to ensure accuracy to real-world scenarios and finally sensitivity analysis and uncertainty quantification is performed to refine the model. What we are left with is a network representing the criminals, their categorised or quantified properties and their connections (3).

B. The problem of visualising agent-based criminal networks. The visualisation of criminal networks specifically has to represent the underlying organisational setup. The average size of criminal networks in Europe is around 30 core members (4), but can be a lot larger for well-connected networks in hotspots such as The Netherlands. Around half of the criminals in these networks are involved in the drug trafficking business (4). Generally, these drug networks exhibit violent behaviour and

account directly or indirectly for 50% of all homicide in Europe (5). This implies that the network model must also include players such as assassins, murder brokers and organisers to capture the full picture, next to the more direct roles like dealers, cutters, transporters, retrievers, stashers etc. (3). These roles can be extracted from the collected data. The structure of the network is generally loosely interconnected due to secrecy and flexibility constraints; however, a key characteristic is the presence of central figures with high centrality measures and connectedness (6). This player is denoted the kingpin, and plays a central part in efforts to disrupt the network.

As an agent-based model, the members of the criminal network are connected to each other and have a set of unique properties. The connections are undirected and based on their mutual trust. While this is relatively straightforward to show in a graph, by mapping the trust value to the thickness and opacity of the connections, each member additionally has a unique criminal, violence and financial capital value which need to be displayed (3). There are also other static properties, such as fitness and role, and time-varying ones such as their behaviour that need to be shown.

C. Requirements of Law Enforcement Agencies. *LAEs* require an intuitive and collaborative platform for network analysis and simulation, enabling investigators to efficiently visualise these complex criminal networks and extract insights (7). The interface must adhere to the principles of good UI design (informative feedback, consistency, user-adaptability, reducing user memory load, reversible actions, offer shortcuts, structure)(8), ensuring the ease of use for non-technical law enforcement personnel. Some key requirements include the ability to identify the key players and central figures rapidly, analyse the organisational structure based on the clustering of nodes, find the weakest links and vulnerable points and analysing the effect of removing specific members or connections (9). For this, it is necessary to be able to display key metrics such as degree distributions, clustering coefficients, node centrality etc. in a more intuitive way than just quantitatively.

Additionally, options like light and dark mode should be offered. Light mode increases comprehension (10) and has a positive impact of mood, alertness and creativity (11), while dark mode has benefits when it comes to eye health, power saving and productivity and comfort at night (12). Also the ability to display the networks in 2D and 3D aids the general overview and amplifies the aesthetic of the graphs. It has been shown that a large number of overlapping edges can negatively influence a viewers perception of a graph and discourage them from further analysis. Also, maximising symmetry and showing clustering clearly aids in the viewers positive experience (13; 14). Being able to choose the preferred view significantly aids in these aspects.

2. The Citadel Web Application

A. The dataset. The data for the networks used for this project was collected by the *Nationale Politie van Nederland* and researchers from the Computational Science Lab at UVA. The final networks are comprised of up to connected 3500 nodes, that appertain to a business role (kingpin, dealer, murder broker etc.) and have a distinct set of attributes (mindset, in prison or not, fitness, capital etc.).

B. What has been done: Our setup. *CrimeSeen*, an "interactive visualisation and simulation environment for exploring criminal network dynamics using computational models" was created by Frederike Oetker, Liza A. S. Roelofsen, Rob G. Belleman, Rick Quax and Miles van der Lely, specifically as tool for the LEAs of Amsterdam to tackle the local network of cocaine criminals (7). It is made up of three components (7):

- *Citadel*, the front end React web interface,
- the *Criminal Cocaine Replacement Model* (CCRM) in the backend, which defines the rules of the Agent Based Model, and
- the *Simulator*, which combines the model with Citadel, allowing the user to select a pre-existing network and trigger interventions.

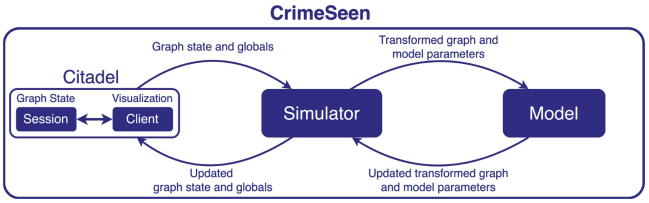

Fig. 1. CrimeSeen, from (7)

Liza Roelofsen had conducted user testing on the previous version of the software. The main issues that plagued the user experience were, in order of number of mentions:

- Confusion about the visual mapping, why the network is being displayed in a certain way
- Visual clutter: too may edges, nodes too close/large
- Slow and unstable performance
- Difficulty in selection and manipulation of nodes and edges
- Simulation is too difficult to get to run

Solving these issues requires a thorough investigation of the underlying technologies and harnessing the capabilities of React Sigma, the library Citadel is built upon.

C. Network visualisation with React Sigma. React Sigma is a JavaScript/TypeScript library that wraps Sigma.js functionality into React components. Sigma.js offers the graph visualisation capabilities, while React takes care of the rendering the UI by using a modular approach (15). A set of components represent different parts of the graph, such as SigmaContainer, Graph, Node and Edge. These components are reusable and tied together by React hooks and contexts, which manage the state and provide functionalities throughout the component tree.

3. Methods: Implementing a dynamic/flexible interface

A. Handling state. A state in React is a built-in object that holds a components data or information about a component (16). Because it is component-specific, by default, when a state changes, the components are triggered to re-render. While this is generally very useful for dynamic web interfaces, in the realm of network visualisation, this is highly problematic, as reloading the entire graph when changing one property is undesirable. For example, the node positions should not change when resizing nodes, but should remain where they were.

To solve this issue smoothly, it is essential to utilise a modular setup, where the individual dynamic functions of the interface are decoupled and can be handled independently of each other. This is achieved by using React contexts extensively. Contexts allow for the sharing of state across deeply nested components without the need to pass props through every level. In this way, every component can easily access the same state. For example, the GraphDataContext provides the state management for graph-related data. This can then be utilised by components like ForceLayout (see next section) to manage graph layouts without causing unnecessary re-renders of the entire graph (see fig. 4).

Additionally, reducers were utilised to manage complex state updates in a predictable manner. Reducers are functions that take the current state and an action as arguments, and returns a new state (17). By using the useReducer hook alongside Contexts, we can ensure that state transitions are handled efficiently and independently. For instance, actions dispatched to update node sizes or colours do not interfere with node positions, allowing for a more granular control over the different aspects of the graph visualisation.

An overview of the components, contexts and reducers can be found in the appendix (fig. 4).

B. Manipulating the layout (the buttons component). To enable interaction with the graph, a menu item ("Layout") was created. It provides the user with intuitive controls to manipulate various aspects of the graph visualisation, such as node resizing, recolouring, edge manipulation, and adjusting the global scaling factor. Additionally, a button enables easy switching between the 2D and 3D views.

The ButtonsComponent used for this is tightly integrated into the application, using the GraphDataContext to access and update the state. This ensures that any changes that are made are on a global level. By using the useEffect hook, we can immediately update the graph appearance by changing the state of the individual functions.

For example, when a user selects a resizing option, the component calculates the appropriate relative scaling factor based on the selected property (e.g., financial capital, criminal capital) and updates the node sizes accordingly. The absolute scaling can be adjusted through the handleSliderChange function, which adjusts the scaling factor in response to user input from a slider control. The updateGraphAppearance function then applies these changes uniformly across the graph.

Recoloring functions are similarly managed by the handleRevertClick and updateGraphAppearance functions. Users can select different colour schemes based on node roles or similar criteria, which allows for highlighting of specific groups within the network. The component ensures that colour changes propagate seamlessly throughout the graph without disrupting the existing layout or node positions.

Additionally, the ButtonsComponent includes functionality for edge manipulation, such as adjusting edge thickness and opacity based on trust values. This provides users with the ability to visually distinguish between the different levels of trust within the network and gives them a more profound understanding of the underlying relationships between the players in the crime network.

4. Results

A. Graph layout decisions. Effective graph visualisation relies heavily on the underlying layout algorithms, which determine the positioning of nodes and edges. In two dimensions we can utilise *Force Atlas 2*, a popular layout algorithm which relies on simulating "spring-electric" forces (edges act as springs, nodes as repulsive particles), which makes it seem very natural (18). It was found that the the LinLog mode and outboundAttractionDistribution were best turned off, as it made it harder to visualise measures such as centrality and created too many edge crossings. Using the Barnes-Hut optimisation proved useful however, because it approximates the repulsive forces of distant nodes, and creates less fragmented clustering (and has some performance benefits for large networks, $\mathcal{O}(n \log n)$ vs $\mathcal{O}(n^2)$). In the three dimensional representation, we utilise a very similar layout engine, *react-force-graph-3d*, which is optimised for the extra dimension.

While automatic layout engines do a good job of structuring the network, one of the requirements is to be able to manually manipulate the node positions. This means temporarily fixing node positions during and after dragging, while ensuring that it remains easy to return to an automatic layout if desired. To this end, the *ForceAtlas2* algorithm is temporarily disabled when a node is selected, and reactivated afterwards. This can be done, because the state updates are isolated and the layout algorithm is independent from the node position attribute.

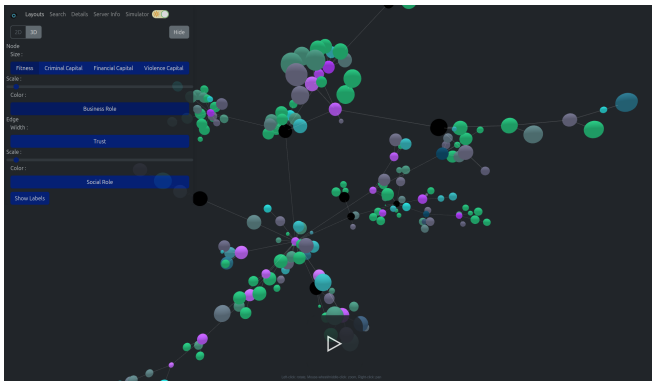

Fig. 2. The final interface with the layouts component (3D view).

B. Improvements on previous versions. The result is a clean interface with the ability to choose and visually separate distinct categories of information (see fig. 2). Compared to the previous version, the visualised network is less cluttered, more informative and is easily manipulated. The layouts section provides a clear way of interacting with the visual properties of the provided criminal network. The nodes and edges can be coloured and resized. Also, the simulation can be easily started from the buttons on the interface.

5. Discussion

The integration of the ButtonsComponent into the criminal network visualisation tool significantly enhances the user experience by providing intuitive controls for manipulating the various aspects of the graph. This modular approach, using React's state management and context APIs, ensures that updates to node sizes, colours, and edge attributes are handled efficiently without disrupting the overall layout. The

choice of *ForceAtlas 2* for the graph layout has proven effective in maintaining clarity and minimising edge crossings, which is crucial for analysing networks of complex agent-based models.

From a design perspective, future work on the UI could and should include a more unified visual experience. Additionally, more collaborative features, such as session saving and individual user accounts could be integrated. For advanced and more technical users, a section with the statistical analysis and data visualisation could prove useful, but would need to be implemented in a way as not to interfere with the main view. These further improvements would require another round of user testing.

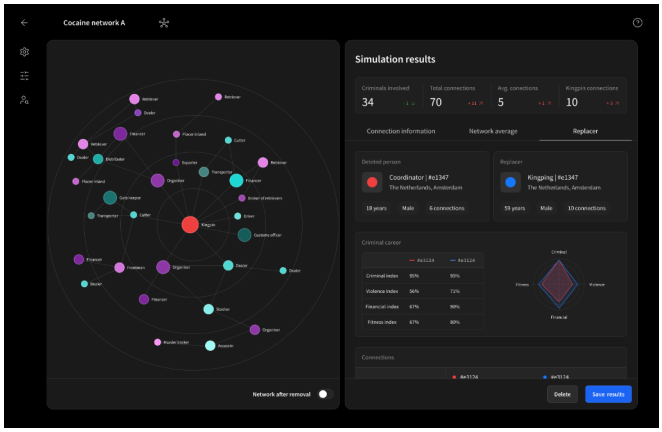

Fig. 3. Design concept for future work, with simulation statistics and graphs, as imagined by Thomas van der Linden.

6. Conclusion

The CrimeSeen software provides a robust platform for visualising and simulating criminal networks, supporting law enforcement agencies in their efforts to analyse and disrupt illicit activities. Thanks to the flexible and modular approach, it can be extended easily and used as a framework for further-reaching projects in the realm of network visualisation.

References

[1] M. A. Tayebi and U. Glässer, *Social Network Analysis in Predictive Policing*. Cham, Switzerland: Springer International Publishing, 2016.

[2] M. Schuilenburg and M. Soudijn, "Big data policing: The use of big data and algorithms by the Netherlands Police," *Policing: A Journal of Policy and Practice*, vol. 17, p. paad061, Sept. 2023. _eprint: https://academic.oup.com/policing/article-pdf/doi/10.1093/police/paad061/51736817/paad061.pdf.

[3] F. Oetker, V. Nespeca, T. Vis, P. Duijn, P. Sloot, and R. Quax, "Framework for developing quantitative agent based models based on qualitative expert knowledge: an organised crime use-case," *arXiv*, July 2023.

[4] "Decoding the EU's most threatening criminal networks | Europol," Sept. 2024. [Online; accessed 23. Sep. 2024].

[5] "Drug trafficking network active across Europe dismantled," Sept. 2024. [Online; accessed 23. Sep. 2024].

[6] G. Bichler, A. Malm, and T. Cooper, "Drug supply networks: a systematic review of the organizational structure of illicit drug trade," *Crime Sci.*, vol. 6, pp. 1–23, Dec. 2017.

[7] F. Oetker, L. A. S. Roelofsen, R. G. Belleman, and R. Quax, "CrimeSeen: An Interactive Visualization Environment for Scenario Testing on Criminal Cocaine Networks," in *Computational Science – ICCS 2024*, pp. 195–204, Cham, Switzerland: Springer, June 2024.

[8] J. Ruiz, E. Serral, and M. Snoeck, "Unifying Functional User Interface Design Principles," *International Journal of Human–Computer Interaction*, Jan. 2021.

[9] P. A. C. Duijn, "Detecting and disrupting criminal networks: A data driven approach," 2016. [Online; accessed 25. Sep. 2024].

[10] M. L. Wilson and A. J. Kinsela, "Absence of Gender Differences in Actual Induced HMD Motion Sickness vs. Pretrial Susceptibility Ratings," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 61, pp. 1313–1316, Sept. 2017.

[11] K. C. H. J. Smolders and D. Y. A. W. Kort, "Bright light and mental fatigue : effects on alertness, vitality, performance and physiological arousal," *J. Environ. Psychol.*, vol. 39, pp. 77–91, 2014.

[12] K. Kim, A. Erickson, A. Lambert, G. Bruder, and G. Welch, *Effects of Dark Mode on Visual Fatigue and Acuity in Optical See-Through Head-Mounted Displays*. Oct. 2019.

[13] G. J. Mooney, H. C. Purchase, M. Wybrow, and S. G. Kobourov, "The Multi-Dimensional Landscape of Graph Drawing Metrics," in *2024 IEEE 17th Pacific Visualization Conference (PacificVis)*, pp. 23–26, IEEE.

[14] H. Purchase, "Which aesthetic has the greatest effect on human understanding?," in *Graph Drawing*, pp. 248–261, Berlin, Germany: Springer, July 2005.

[15] "React Sigma | React Sigma," Aug. 2024. [Online; accessed 16. Sep. 2024].

[16] "State in React in React," May 2024. [Online; accessed 16. Sep. 2024].

[17] "Redux Fundamentals, Part 3: State, Actions, and Reducers | Redux," Aug. 2024. [Online; accessed 16. Sep. 2024].

[18] M. Jacomy, T. Venturini, S. Heymann, and M. Bastian, "ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software," *PLoS One*, vol. 9, p. e98679, June 2014.

Appendix

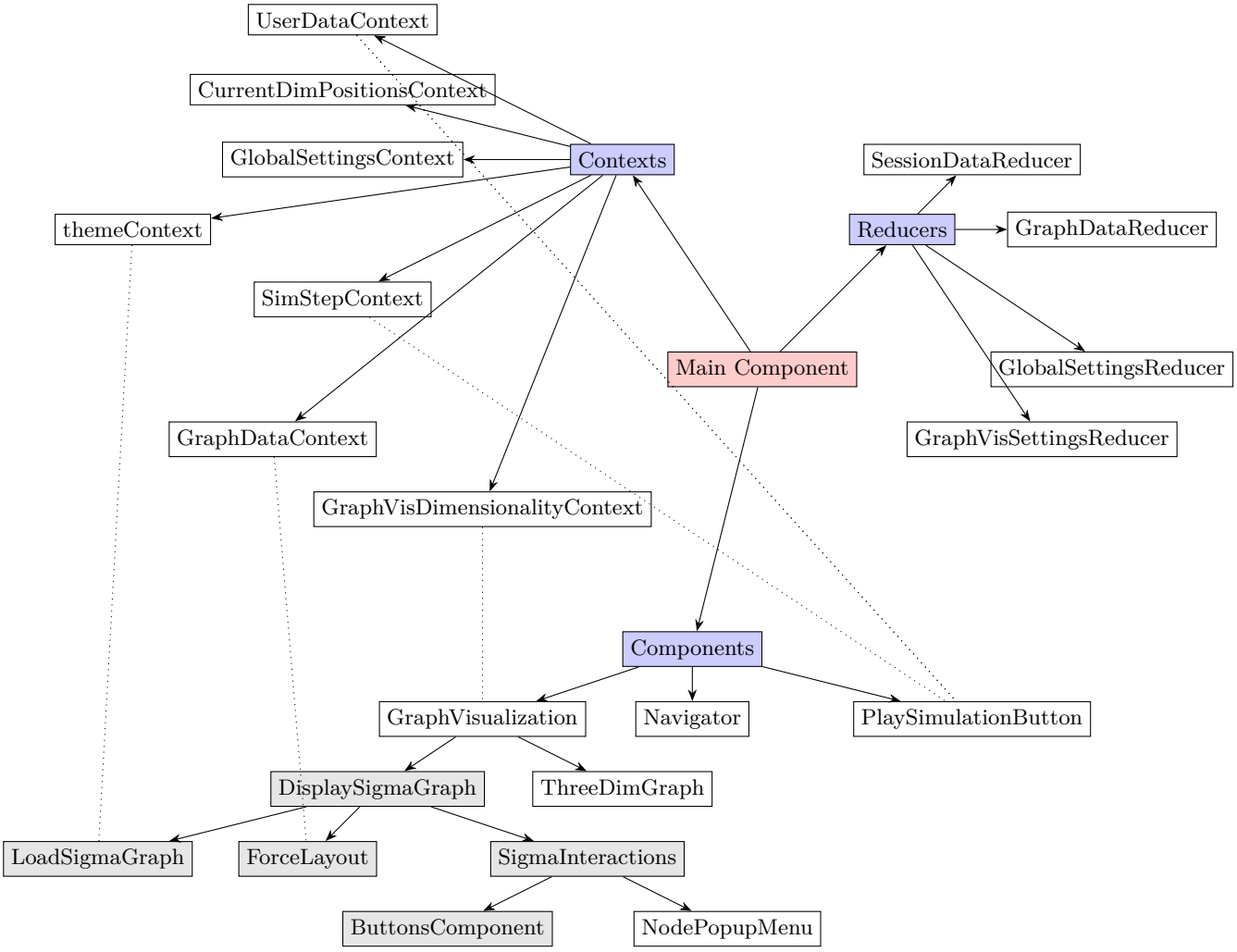

Fig. 4. Schematic, non-exhaustive overview of the relations between components, contexts and reducers used in our setup.