Data Mining and Knowledge Discovery (2025) 39:48
https://doi.org/10.1007/s10618-025-01119-1

SimHawNet: a modified Hawkes process for temporal network simulation

Mathilde Perez[1] · Raphaël Romero[2] · Bo Kang[2] · Tijl De Bie[2] · Jefrey Lijffijt[2] · Charlotte Laclau[1]

Received: 15 April 2025 / Accepted: 8 June 2025 / Published online: 10 July 2025
© The Author(s) 2025

Abstract

Temporal networks allow representing connections between objects while incorporating the temporal dimension. While static network models can capture unchanging topological regularities, they often fail to model the effects associated with the causal generative process of the network that occurs in time. Hence, exploiting the temporal aspect of networks has been the focus of many recent studies. In this context, we propose a new framework for generative models of continuous-time temporal networks. We assume that the activation of the edges in a temporal network is driven by a specified temporal point process. This approach allows to directly model the waiting time between events while incorporating time-varying history-based features as covariates in the predictions. Coupled with a thinning algorithm designed for the simulation of point processes, SimHawNet enables simulation of the evolution of temporal networks in continuous time. Finally, we introduce a comprehensive evaluation framework to assess the performance of such an approach, in which we demonstrate that SimHawNet successfully simulates the evolution of networks with very different generative processes and achieves performance comparable to the state of the art, while being significantly faster.

Keywords Temporal networks · Temporal point process · Graph simulation

Mathilde Perez and Raphaël Romero have contributed equally to this work.

✉ Mathilde Perez
mathilde.perez@telecom-paris.fr

Raphaël Romero
raphael.romero@ugent.be

[1] LTCI, Télécom Paris, Institut Polytechnique de Paris, Palaiseau, France

[2] IDLab, Ghent University, Ghent, Belgium

1 Introduction

Motivation. The study of continuous-time temporal networks has attracted growing interest in the past decades, as time-stamped relational data emerges from various sources, including e-mail communication, e-commerce, and even neuron firing data (Goldenberg et al. 2009; Holme and Saramäki 2012). Understanding how networks evolve over time provides valuable insights into the underlying mechanisms driving their behavior. It also enables the development and testing of methods for analyzing and predicting real-world (social) networks' dynamics. Examples of use cases include the study of opinion dynamics, epidemic spreading, and recommender systems (Panzarasa et al. 2009; Machens et al. 2013).

Attempts to model such networks have traditionally relied on a discretization of time, allowing the representation of the temporal network as a sequence of static graphs[1] (Krivitsky and Handcock 2014; Kim et al. 2018; Durante and Dunson 2016). As described by Latapy et al. (2017), these attempts to read this problem through the well-documented lens of graph theory fail to capture mechanisms coming from the causal nature of temporal networks.

Moreover, temporal relational data is often available with a high level of time granularity, allowing us to consider the time stamps as continuous (Rozenshtein and Gionis 2019). The interaction patterns are often heavily localized in time, exhibiting specific properties such as *burstiness* (Masuda and Lambiotte 2016). For instance, a good model for temporal social networks should be able to give an idea of the duration that will separate two interactions between users, given a certain historical context. To this date, only a few tools are available for continuous-time temporal network analysis, e.g., methods from computational social sciences (Butts 2008) and point process modeling (Bacry et al. 2018).

To adapt these tools for temporal graph simulation, we propose incorporating features from computational social sciences into a temporal point process model. This approach, drawing inspiration from Exponential Random Graph/Maxent Models (Holland and Leinhardt 1981; Robins et al. 2007; Wasserman and Pattison 1996; De Bie 2011), elegantly and flexibly introduces inductive bias, enabling the assessment of the statistical properties of an observed temporal graph. The inter-event times are modeled using modified Hawkes processes, conditioned on both the structural nature through node embeddings and the temporal dynamics through time-varying features of the observed interactions within the graph.

For example, in the context of social networks, accurate modeling of interactions between individuals is essential for understanding network dynamics. We focus on pairwise interactions which are naturally influenced by past behavior. To capture this temporal dependency, we rely on a process that model the impact of past interactions on future interactions. Moreover, social interactions are rarely isolated; they are shaped by shared friends and local structure. We therefore incorporate temporal features that reflect the common neighborhood of two individuals.

[1] Note that we use the terms *network* and *graph* interchangeably and more specifically the terms *temporal network* and *dynamic graph* also have the same meaning here and in existing literature.

Second, we leverage the resulting model to simulate dynamic graphs that mimic the temporal evolution and interactions within the network. Our approach opens the avenue for several applications including evaluating algorithms for network analysis, studying the effects of interventions or policy changes (e.g. fairness constraints), or training and testing machine learning models for tasks such as link prediction or community detection.

Contributions. This work aims to introduce a novel framework that allows the translation of temporal point processes, specifically Hawkes processes, on a generative temporal network model. The main contributions of this paper are as follows:

1. First, we introduce the Hawkes network model (HawNet) with edge-specific inter-arrival time modeling to capture temporal network properties, and derived a Maximum Likelihood Estimation procedure for parameter inference on observed data (Sect. 4).
2. Then, in Sect. 5, we present SimHawNet, a simulation framework, that is built upon the learned model. The proposed algorithm utilizes a thinning algorithm inspired by Ogata (1981) originally proposed to simulate point processes. The full pipeline is illustrated in Fig. 1.
3. Finally, in Sect. 6, we develop an evaluation framework comprising dynamic synthetic graphs derived from established static simulation processes, alongside suitable metrics for assessing the efficacy of a dynamic graph simulation model.

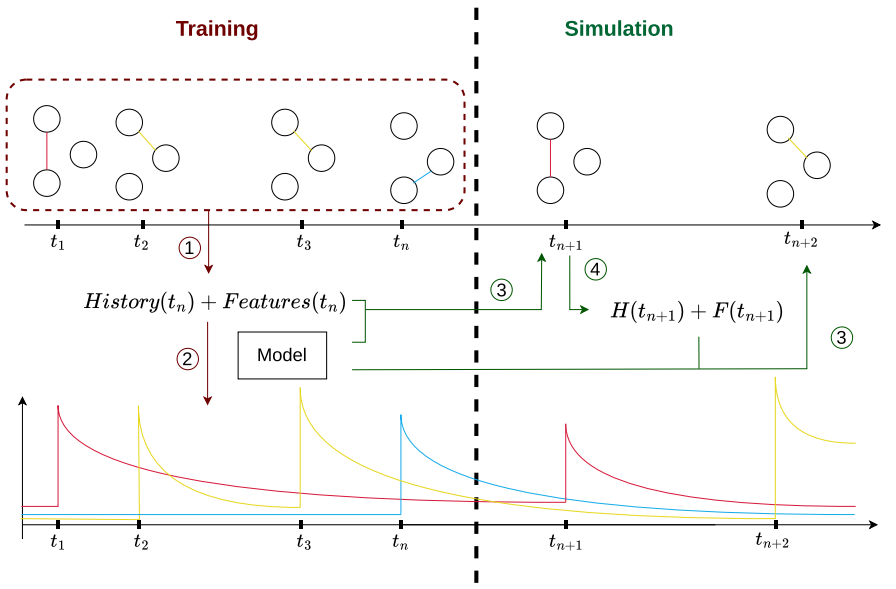


Fig. 1 SimHawNet: **1** extraction of the temporal graph's history and hand-crafted creation of features, **2** training the model to convert the graph into edge-wise intensity, **3** Ogata prediction of the next activated edge and the corresponding time based on the trained model, graph history and features, **4** updating the graph's history and features

We demonstrate the ability of SimHawNet to accurately reproduce the dynamics of temporal graphs through extensive experiments, both within our synthetic evaluation framework and on real-world datasets. The full code is publicly available.[2]

2 Related work

To the best of our knowledge, the previous work closest to the method introduced in this paper is presented by Huang et al. (2022), who propose to model relational events between nodes using mutually-exciting Hawkes processes, where the baseline intensity is influenced by node proximity in a latent space and by node-specific sender and receiver effects. However, they rely on static node effect terms to define the dynamic part of conditional intensity functions, while in our case we rely on time-varying features to model the temporal evolution of the intensity.

2.1 Point process modeling of temporal networks

More generally, Point Process modeling of Temporal Networks is a popular paradigm encompassing many previous studies. Seminal works coming from the computational social sciences and machine learning community propose to model temporal networks through their intensity functions to assess the effect of the temporal network topology on the rates of link formation (Butts 2008; Bois et al. 2013; Perry and Wolfe 2013; Farajtabar et al. 2018; Vu et al. 2011). Specifically, relational event modeling also relies on a piece-wise constant intensity hypothesis to define the likelihood of the observed network. In our case, we provide greater flexibility by allowing the insertion of a customized hazard function on the edge-inter-event times. This function can be made dependent or independent of the evolution of other edges in the network, depending on specific requirements. In the same spirit, Passino and Heard (2023) propose to posit a mutually exciting model between the different edges, such that the interactions occurring for a given edge influence the ones occurring on neighboring edges. However, they rely on time-varying node embeddings to define the conditional intensity functions and focus on parameter estimation rather than generation.

2.2 Dynamic graph embedding

Dynamic graph embedding methods such as Dai et al. (2016); Trivedi et al. (2019); Romero et al. (2023); Rastelli and Corneli (2023) also model the interaction times between edges using point processes. More precisely, they use the log-likelihood of the model to *learn time-varying node embeddings*. In the same direction, learning time-varying embeddings (not necessarily by directly using point process models) from a history of past interaction is a task attracting growing interest (Rossi et al. 2020; Xu et al. 2020; Kumar et al. 2019). In our case, however, static embeddings are used in the base rate but we separately model the influence of past interactions on the rate and do not require the embeddings to evolve through time.

[2] Code available at: https://github.com/MathouP31/SimHawNet.

2.3 Dynamic stochastic block modeling

Furthermore, extensions of the Stochastic Block Model (SBM) to the dynamic case have also considered inhomogeneous Poisson processes to model the conditional interaction between pairs of nodes (Corneli et al. 2016; Matias et al. 2018; Arastuie et al. 2020). However, these methods usually assume that some latent groups in the network drive the intensity associated with the pairs, that they aim to recover.

Remark on the connection with graph generation: it is important to highlight the significant distinction between dynamic graph simulation and (non-)autoregressive graph generation methods (Bonifati et al. 2020). While graph generation models primarily concentrate on generating static graphs, they typically only address the presence of edges without considering the possibility of edge activation multiple times. Moreover, an essential difference lies in the disregard of the temporal order by generative models, leading to an arbitrary order in the creation of edges.

3 Temporal networks and Hawkes process

This section introduces the used notation and the fundamental concept underlying our framework, namely temporal point processes. We then demonstrate how to design a first-order Markov chain model for a temporal network, based on a modified Hawkes model for the edge inter-arrival times.

3.1 Problem statement

Let $\mathcal{U}$ and $\mathcal{V}$ denote a set of source and destination nodes respectively, $\mathcal{E} \subset \mathcal{U} \times \mathcal{V}$ the set of possible edges. Temporal network data takes the form of a sequence of time-stamped network events $\mathcal{H} = \{(e_m, t_m) \| \ m = 1, ..., M\}$, where M is the number of events, $0 < t_1 < ... < t_M$ is an ordered sequence of positive time stamps, $e_m = (u_m, v_m) \in \mathcal{E}$ is the edge involved in the m-th event, and $u_m \in \mathcal{U}$ and $v_m \in \mathcal{V}$ are the source and destination nodes respectively. For instance, each event $((u, v), t)$ can represent a message sent by user u to user v at time t. For any time t we will denote $\mathcal{H}_t = \{(e_m, t_m) \in \mathcal{H} \| \ t_m < t\}$ the history of the network up to time t. Moreover, for any edge $e = (u, v) \in \mathcal{E}$, we denote $\mathcal{H}^{(e)} = \{(e_m, t_m) \in \mathcal{H} \| \ e_m = e\}$ the history of events involving the edge e. Finally, we denote by $\mathcal{H}_t^{(e)}$ the history of events involving the edge e, up to time t. In our case, we work on undirected graphs, which means $(u, v) = (v, u)$.

We now introduce the concept of a dynamic graph simulator, which serves as the foundation for our study. The primary objective of this work is to develop such a simulator–capable of generating realistic dynamic networks that reflect temporal patterns observed in real-world data. We formally define a dynamic graph simulator as follows.

Definition 1 (Dynamic Network Simulator): An algorithm that observes a dynamic network over a time interval $\mathcal{H}_{T_0} = \{(e_m, t_m) | t_m < T_0\}$ and subse-

quently attempts to generate or simulate the events for the next time interval $\mathcal{H}_{\geq T_0} = \{(e_m, t_m) | t_m \geq T_0\}$. A good simulator should verify two main properties:

1. *P1: Topological Correctness* The network formed by the simulated interactions should be *close* in distribution to the original network. That is, given a set of network statistics $\mathcal{S}$, their expected values on the simulated network should be approximately equal to their values on the original one.
2. *P2: Temporal Consistency* The timestamps of the simulated interactions should be close in distribution to those of the original network, that is the densities of events over time should match approximately.

To model a dynamic network simulator, we rely on temporal point processes, which provide a natural and flexible framework for capturing interactions in dynamic networks.

3.2 Temporal point processes

Temporal point processes are stochastic variables whose realizations are sets of positive real-valued time stamps in some time interval [0, *T*]. These variables in our case, can be represented as an event sequence of activation of edges $\mathcal{H}_t = \{(e_1, t_1), ..., (e_M, t_M)\}$ where the size of this sequence *M* is a random variable too.

A fundamental quantity in the study of temporal point processes is the *conditional intensity function*, denoted $t \mapsto \lambda(t|\mathcal{H}_t)$, such that for any $t \in [0, +\infty[$, $\lambda(t|\mathcal{H}_t)dt$ measures the average number of events occurring in the interval $[t, t + dt[$, given the event history up to time *t* (Daley and Vere-Jones 2010).

A well-known example is the Poisson process, his conditional intensity function is independent of the history, meaning that it is a *deterministic* function: $\lambda(t|\mathcal{H}_t) = \lambda(t)$. In contrast, *self-exciting* processes are governed by a *stochastic* conditional intensity function, whose value at time *t* depends on the samples previous to *t*. While Poisson Processes can be accurate when we have a clear a priori idea of the form of the intensity function, they are unable to adapt to new events. On the contrary self-exciting processes allow the conditional intensity function to take into account past events where historical events have a positive exciting influence on later events. A common self-exciting point process is the Hawkes process (Hawkes 1971), which has a conditional intensity function:

$$\lambda(t|\mathcal{H}_t) = \mu(t) + \sum_{t_i < t} \gamma(t - t_i).$$

Where $\mu(t) > 0$ represents a base intensity, γ an excitation function, and $t_i \in \mathcal{H}_t$ a previous event.

3.3 From Hawkes process to first-order Markov chain

This section will explain how to transition from a Hawkes process to a Markov chain framework by modeling the conditional intensity function using time-varying features, rather than relying on the entire history of events. This approach enables a more structured representation of the inter-arrival times of events in a temporal network.

To give an expression of the intensity function, we suppose that in the absence of any event, the point process is generated by a deterministic $\lambda(t|\mathcal{H}_t = \emptyset) = \mu(t)$. We also suppose the existence of a vector-valued stochastic function $t \mapsto \boldsymbol{x}(t)$, defined for any time $t \in [0, +\infty[$, called *time-varying features*. Then, given a non-empty history of previous events $\{t_1, ..., t_n\}$, we suppose that $\boldsymbol{x}(t_n)$ is a sufficient statistic of the history up to time t_n, for the prediction of the next event: $\lambda(t|\mathcal{H}_t = \{t_1, ..., t_n\}) = \lambda(t|\boldsymbol{x}(t_n))$.

Alternatively, the occurrence of each new event can be seen as incrementing the conditional intensity function with a time-varying value.

Definition 2 We define on $[0, +\infty[$ the *intensity increment function*:

$$t \mapsto \gamma(t|\mathcal{H}_t) = h(t - t_n|\boldsymbol{x}(t_n)) + \mu(t)$$

While h is a deterministic function of the time elapsed since the last event, γ is a stochastic function of the absolute time of the process so γ depends on the history $\mathcal{H}_t$ only through the time-varying feature vector $\boldsymbol{x}(t_n)$

Conditioned on the feature function $\boldsymbol{x}$, the procedure describes a self-exciting process, where the intensity function gets set to the value of the hazard function at every new event.

Proposition 1 *The conditional intensity function of the point process defined above is given for any $t \in [0, +\infty[$ by*

$$\lambda(t|\mathcal{H}_t) = \mu(t) + \sum_{t_n < t} \mathbf{1}_{t \in [t_n, t_{n+1})} \gamma(t|\boldsymbol{x}(t_n)) \tag{1}$$

where $\mathbf{1}_{t \in [t_n, t_{n+1})}$ is the indicator function asserting that the current time is before the next event.

To summarize, this formulation derives from Hawkes processes (Hawkes 1971), with the distinction that it employs time-varying features and considers only the most recent event to predict the next one, while Hawkes processes include a time-varying increment for each previous event in the history.

4 HawNet: Hawkes for temporal network

In this section, we show how to apply the previous procedure to temporal graphs, and we refer to this framework as HawNet.

4.1 Modeling edge-wise intensity

As in previous work (Vu et al. 2011; Perry and Wolfe 2013; Bois et al. 2013), the edges are considered as conditionally independent given the rest of the model. Thus, we start by applying the formulation from Sect. 3.3 at the edge level. This will set the stage for defining the loss as the negative log-likelihood for the entire temporal network.

In our framework, given a temporal network, for each edge, $e \in \mathcal{E}$ we define the edge-wise conditional intensity function $t \mapsto \lambda^{(e)}(t|\mathcal{H}_t^{(e)})$, associated to the point process generating the edge history $\mathcal{H}^{(e)}$:

$$\lambda^{(e)}(t|\mathcal{H}_t^{(e)}) = \mu^{(e)} + \alpha^{(e)} e^{-\beta^{(e)}(t - t_{last}^{(e)})},$$

where α determines how significant the activation of the edge is, β is a decay rate for each edge which models how long the interaction lasts, and t_{last} is the time of the last activation of the edge *e* before *t* (equivalent to t_n in Eq. 1). The global intensity of the graph is the sum of these intensities:

$$\lambda(t|\mathcal{H}_t) = \sum_{e \in \mathcal{E}} \mu^{(e)} + \alpha^{(e)} e^{-\beta^{(e)}} (t - t_{last}^{(e)})$$

A graph is an object that cannot be studied edge by edge without taking into account their mutual influence, we introduce the notion of time-varying features $F^{(e)}(t) \in \mathbb{R}_+^{(d)}$, which enables us to represent the influence of edges on each other by incorporate it in α. Defining these features allows us to describe the edge history $\mathcal{H}_t^{(e)}$ as the set of features and the timestamps of the previous activations of the edge. Finally, the model writes

$$\lambda^{(e)}(t|\mathcal{H}_t) = \mu^{(e)} + \sigma(g(F^{(e)}(t)) e^{-\beta^{(e)}(t - t_{last})} \tag{2}$$

where *g* is a linear function, σ the softplus activation function, and $F(t)$ time-varying features. An example of these features is presented in Sect. 4.3. Finally, β is a learned parameter that controls the speed at which an activation loses importance and influence over other edges. Now our goal is to learn the parameters involved in this intensity function, namely the node embeddings, β, and the parameters of *g* weighting the features.

Base intensity. The term $\mu^{(e)}$, which defines the base intensity is independent of time. It is calculated as a function of the distance between the embeddings of the two nodes that are involved in the edge, that allows us to capture the static information of the graph. Since our model operates on edges rather than nodes, we derive edge embeddings from the distance between the node embeddings implied in the edge. Without prior knowledge on the temporality of the edge base rate, a simple approach is to define the constant intensity of the edge $e = (u, v)$ as:

$$\mu^{(e)} = e^{(2c - d(z_u, z_v))}, \tag{3}$$

where for each node $u \in U \cup V$, $z_u \in R^p$ is an embedding of dimension p, $c \in R$ is an offset parameter and d is a distance function. This model is inspired by the literature about latent space models for graphs. The underlying intuition is that each node $u \in U$ in the network is equipped with a position z_u in a latent social space, and a disk centered around z_u with a radius c. c is learned and corresponds to the influence radius of a node in the latent space. So two nodes u and v have an interaction value greater than 1 whenever the intersection of their disks is not empty. In Eq. 3, the smaller $d(z_u, z_v)$ is, the higher the base intensity between u and v will be.

Remark μ is highly flexible and can be defined in other ways depending on the aspects we want to incorporate into the model. For example, if we want to add a label to the edges or prioritize certain nodes belonging to a particular community, it is easy to integrate it into μ.

4.2 Definition of the loss and optimization

Assuming a parametrization of the intensity function above, we propose to find the maximum likelihood estimates of the parameters by optimizing the negative log-likelihood.

4.2.1 Likelihood of the network

Given a point process model with conditional intensity $\lambda(t|\mathcal{H}_t)$, the general form of the likelihood of an observed event history $\{t_1, \ldots, t_n\}$ on an interval $[0, T]$ is derived inDaley and Vere-Jones (2010) and given by

$$p(\{t_1, \ldots, t_n\}|\lambda) = \left(\prod_{i=1}^{n} \lambda(t_i|\mathcal{H}_{t_i}) \right) \exp\left(- \int_0^T \lambda(s|\mathcal{H}_s)\, ds \right).$$

Since we assume independence of the edge-level interactions conditioned on the time-varying features, the likelihood of the network factorizes as a product of point process likelihoods:

$$p(\mathcal{H}|\lambda) = \prod_e \left[\left(\prod_{\tau \in \mathcal{H}^{(e)}} \lambda^{(e)}(\tau|\mathcal{H}_\tau) \right) \exp\left(- \int_0^T \lambda^{(e)}(s|\mathcal{H}_s)\, ds \right) \right].$$

Taking the negative log of this expression leads us to the following loss function:

Definition 3 The negative log-likelihood of our network point process model is given by

$$L(\mathcal{H}) = \sum_{e \in \mathcal{E}} \left[\int_0^T \lambda^{(e)}(s|\mathcal{H}_s) ds - \sum_{\tau \in \mathcal{H}^{(e)}} \log \left(\lambda^{(e)}(\tau|\mathcal{H}_\tau) \right) \right]$$

4.2.2 Making the loss tractable

The loss can be reorganized as a sum over all the events:

$$L(\mathcal{H}) = \sum_{(e_m, t_m) \in \mathcal{H}} \left[-\log \left(\lambda^{(e_m)}(t_m|\mathcal{H}_{t_m}) \right) + \sum_{e \in \mathcal{E}} \int_{t_{m-1}}^{t_m} \lambda^{(e)}(s|\mathcal{H}_s) ds \right], \quad (4)$$

where each event (e_m, t_m) contributes a positive term penalizing low intensity values at the time of the event and a negative term penalizing high cumulated intensity values for all the possible edges during the time interval $[t_{m-1}, t_m]$ where no event occurred. To mitigate the computational cost of evaluating the survival term $\sum_{e \in \mathcal{E}} \int_{t_{m-1}}^{t_m} \lambda^{(e)}(s|\mathcal{H}_s) ds$, which scales quadratically with the number of nodes in the graph, we adopt a **negative sampling** strategy. Instead of summing over all edges where no event occurred, we sample a small subset of *negative edges* $\tilde{\mathcal{E}}_m \subset \mathcal{E} \backslash \{e_m\}$ for each observed event (e_m, t_m). The survival term is then approximated by:

$$\sum_{e \in \mathcal{E}} \int_{t_{m-1}}^{t_m} \lambda^{(e)}(s|\mathcal{H}_s) ds \approx \frac{|\mathcal{E}|}{|\tilde{\mathcal{E}}_m|} \sum_{\tilde{e} \in \tilde{\mathcal{E}}_m} \int_{t_{m-1}}^{t_m} \lambda^{(\tilde{e})}(s|\mathcal{H}_s) ds.$$

This yields a tractable loss approximation:

$$L(\mathcal{H}) \approx \mathcal{L}_{\text{pos}}(\mathcal{H}) + \tilde{\mathcal{L}}_{\text{neg}}(\mathcal{H}),$$

where $\tilde{\mathcal{L}}_{\text{neg}}(\mathcal{H})$ replaces the full negative term $\mathcal{L}_{\text{neg}}(\mathcal{H})$ with a scaled mean over sampled negatives. This approach is inspired by negative sampling techniques in representation learning (Mikolov et al. 2013), where the model learns to distinguish between observed and unobserved events. This approximated loss function has a number of terms proportional to the number of events in the network history. It can be further approximated by computing it on batches, formed by taking slices of the history.

4.3 Temporal features

Hereafter, we provide examples of handcrafted features composing $F^{(e)}$. It is worth noting that these features can be learned in practice (Wang et al. 2021), but we have chosen not to explore this direction in this work, and to leave it as an open question (see Conclusion). As is common in network analysis, the choice of features is highly dependent on the nature of the network considered, and the type of analysis being

conducted. In a variety of previous work, some time-varying edge features have been introduced (Butts 2008; Vu et al. 2011; Bois et al. 2013). We propose to exploit three of them in this work, which consists of an exponentially down-weighted version of the volume (vol), the preferential attachment (pa), and common neighbors (cn).

Here we denote by γ a decay hyper-parameter controlling the rate at which the feature contributions from previous events become obsolete. In addition, $CN(u, t)$ represents the neighborhood for a given node u at time t (i.e., the set of nodes that had an interaction with node u before t). Then these features are written as follows

$$\mathbf{vol}^{(\mathbf{e})}(\mathbf{t}) = \sum_{t_i^{(e)} < t} \exp(-\gamma(t - t_i^{(e)})), \quad \mathbf{cn}^{(\mathbf{e})}(\mathbf{t}) = \sum_{w \in CN(e,t)} \exp(-\gamma(t - t_{last}^{(e)}(w)))$$

$$\text{and} \quad \mathbf{pa}^{(\mathbf{e})}(\mathbf{t}) = deg_u * deg_v, \quad \text{with} \quad deg_u(t) = \sum_{n_u \in CN(u,t)} \exp(-\gamma(t - t_{last}^{n_u})).$$

$\mathbf{vol}^{(\mathbf{e})}(\mathbf{t})$ measures the volume of recent interactions, where $t_i^{(e)}$ denotes the time of the previous activations of the edge e. We also have that $CN(e,t) = CN(u,t) \cap CN(v,t)$ is the set of neighbors of the edge $e = (u, v)$ at time t, the set of nodes that have created an edge before time t with the two nodes u and v. Moreover, $t_{last}^{(e)}(w)$ is the time of the last interaction between u or v, and the common neighbors w.

4.3.1 Feature computation example

Figure 2 shows a toy graph that illustrates the differences between the features described above. The toy graph is composed of five nodes: $\mathcal{G} = \{A, B, C, D, E\}$. Edges are denoted by e and for each edge we record their time of activation. For instance, if the graph corresponds to email communication between individuals within a company, then A and B exchange three emails in the observed period, at times 1, 3, and 7.

Focusing on e_5, we can compute the features for this particular edge along the time, so from $t = 1$ to $t = 10$. We vary the value of γ to better understand its impact. In Fig. 2, we present the volume and the preferential attachment features computed for e_5. The volume only depends on the edge itself, and not on any other edges of the graph. As such, we can see that it reaches peaks when e_5 gets activated, i.e., at times

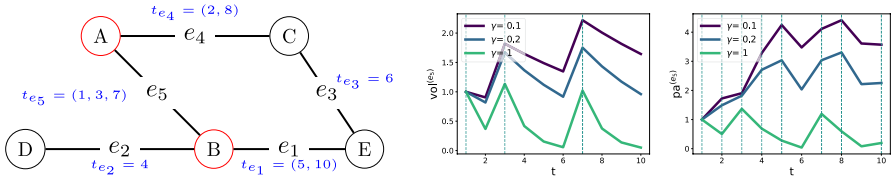


Fig. 2 (left) Example of a toy graph and temporal features computed on e_5, namely (middle): the volume, $\text{vol}^{(e_5)}$ and (right) the preferential attachment, $\text{pa}^{(e_5)}$ as functions of time $t = 1, \cdots, 10$. Vertical lines represent the time of activation of edges involved in the computation of each feature

1, 3 and 7. Between these activation times, the volume decreases at a rate that is controlled by the value of γ. Preferential attachment on the other hand is also impacted by the activations of edges that belong to either the direct neighborhood of A or B. For this reason, we observe peaks corresponding to the activation of e_5 but also to the activations of $e1$, $e2$, and e_4.

Remark γ impacts both the speed at which the values of the temporal features decreased toward 0 and the scale of the features.

5 SimHawNet: from intensity functions to graph simulation

We are now ready to use the intensity functions defined in the previous section to simulate dynamic networks. The main idea is to treat the graph as a composition of intensity functions, enabling us to use rejection sampling to generate observations. By modeling the intensity of interactions as a function of both the network structure and temporal dynamics, we can efficiently simulate realistic event sequences that mimic the behavior of the original temporal graph.

5.1 Simulation

Inspired by the thinning algorithm presented by Ogata (1981), we propose the following simulation algorithm. For every edge e, we remind that $t \mapsto \lambda^{(e)}(t|\mathcal{H}_t)$ denotes the conditional intensity of the point process of edge e. Similarly, we define λ, the total conditional intensity function, that generates the sequence of event times in the network:

$$t \mapsto \lambda(t|\mathcal{H}_t) = \sum_{e \in \mathcal{E}} \lambda^{(e)}(t|\mathcal{H}_t).$$

The full pseudocode is given in the Appendix, Algorithm 1. We describe the main steps hereafter.

Step 1: Generation of a new time. Starting from an initial time t_n, we fix the final time at which the simulation must stop and a maximum number of iterations N. At each iteration, it generates a random number that will be used to determine the time until the next event Δ_t. This time is based on the minimum of an exponential process depending on edge intensities $\lambda^{(e)}(t)$ and a variable $u \sim \mathcal{U}(0, 1)$:

$$\Delta_t = \min_e \left(-\frac{\ln(u^{(e)})}{\lambda^{(e)}(t)} \right).$$

Secondly, we compute a local upper bound, $\lambda^{(*)}$ of the total intensity on the right neighborhood of the current time at each iteration. This upper bound allows us to validate the next event time at time $t_n + \Delta_t$ across the network, in a rejection-sampling manner.

Remark We choose to stop the simulation at a specific time instead of basing it on the number of activated edges. Although this second option is feasible, it requires knowing in advance the number of activated edges, which is typically not realistic.

Step 2: Probability computation. Once a new event time is generated, we need to assign it to an edge in the network. To do so we simulate a set of activatable edges $\mathcal{E}_{\text{active}}$ at this new time using rejection sampling based on their respective intensities. We then compute the activation probability for each edge at this time. This probability is proportional to their intensity value and given by

$$p^{(e)}(t) = \frac{\lambda^{(e)}(t|\mathcal{H}_t)}{\sum_{e' \in \mathcal{E}_{\text{active}}} \lambda^{(e')}(t|\mathcal{H}_t)}. \tag{5}$$

Step 3: Activation of a new edge. We sample a top-*k* of edge candidates having the highest likelihood to be activated, such that $e_{\text{new}} \sim \text{Multinomial}(1, \{p_e | e \in \epsilon\})$. This strategy is motivated by the risk of always selecting the same edge resulting in some infinite loop. By picking *k* large enough we are forcing a more distributed activation pattern over the network.

Step 4: Updating. Finally, the activated edge is selected uniformly at random among the *k* sampled ones. The event is added to the history and updated as the activation occurs.

On the importance of time normalization The Ogata algorithm is highly sensitive to the time range over which events are observed. This sensitivity arises from the algorithm's method of calculating Δ_t the time interval until the next potential event occurrence. On one hand, if the time range is too large, Δ_t becomes excessively small, leading to an explosion in the number of generated events. On the other hand, if the time range is too small, the number of generated events becomes insufficient.

5.2 Limitations of SimHawNet

The main limit of SimHawNet is that the used features focus on the neighborhood of the edge and its historical context; however, it lacks global features that could provide additional insights into the overall trends of the graph. For instance, for some applications, factors such as whether it is a weekend or a weekday could significantly impact the overall activity on the graph. We also note that with each new event accepted into the network, the variables used to calculate time-varying features need to be updated. For example, $\mathbf{vol}^{(\mathbf{e})}(\mathbf{t})$ needs access to all timestamps of previous time events for a given edge. The complexity of this update operation depends on the feature under consideration but can also prevent the algorithm from scaling to larger graphs.

One way to address both issues could be to use a more sophisticated approach to learn the temporal features. We discuss this avenue of research in Sect. 7.

Summary We summarize the full pipeline (model training and simulation) in Fig. 1. The pipeline can be decomposed into the following steps. We start by training our model, i.e. we optimize Eq. 4 based on the observed events and the handcrafted features. This model allows us to convert the graph into edge-wise intensity functions.

Then, based on edge-wise intensity functions obtained, we predict the evolution of the graph following SimHawNet.

6 Experiments

We conducted experiments on both synthetic and real-world networks to address the following three research questions:

[RQ1]: Does SimHawNet generate events that are *topologically correct* (to evaluate **P1**) and *temporally consistent* (to evaluate **P2**)?

[RQ2]: In what ways can the parameters learned by SimHawNet be used in downstream exploratory network analysis?

[RQ3]: How does SimHawNet compare with existing methods in terms of accuracy and computational efficiency?

To answer these questions, we introduce an evaluation framework designed to assess the ability of SimHawNet to capture and reproduce network evolution. We then present the numerical results obtained through this framework.

6.1 Experimental protocol

To evaluate the ability of our approach to capture the dynamic of the network we use both real-world networks and synthetic graphs.

Synthetic networks For synthetic graphs, we create four distinct scenarios inspired by different models. The configuration model (Newman 2003), based on a given degree sequence allows us to model influence effects within a network (i.e., the more influence a person has, the higher the degree of the associated node). The second uses stream graph (Latapy et al. 2019) combined with the Erdős-Rényi Model. The two final scenarios leverage Stochastic Block Model to simulate the creation of community structures within the network. The scenario referred to as SBM-a is a temporal graph composed of two strong communities and the SBM-b extends this by modeling the dissolution of one of the communities, followed by the emergence of a new one. Additional details regarding the underlying generative process of each of these graphs are provided in Sect. B.0.1.

Real networks We use three real-world networks: the Enron Dataset[3] that represents direct messages sent between employees of the company Enron between 1999 and 2003, Reality Mining (Eagle and Pentland 2006) which is communication recorded continuously over 50 week and Militarized Interstate Disputes (MID) (Palmer et al. 2022) which records all instances of when one state threatened, displayed, or used force against another between 1816 and 2014 (more details in Appendix B.1).

Baselines We compare our model with two other continuous-time network models based on generative temporal point processes: the Latent Space Hawkes (LSH) model (Huang et al. 2022) and the Hawkes Dual Latent Space (DLS) model (Yang et al. 2017). LSH utilizes a latent space representation for nodes and models interactions through mutually exciting Hawkes processes with baseline intensities influenced

[3] https://www.cis.jhu.edu/~parky/Enron/.

by node distances and sender-receiver-specific effects. DLS combines homophily and reciprocal latent spaces to account for how users initiate communication and exchange dynamics, utilizing a mixture of exponential and periodic kernels.

General setup We simulate synthetic graphs with 100 nodes and a varying number of interactions depending on the model used. For each type of synthetic graph (configuration model, SBM, and stream graph) we generate 10 initial graphs then on each generated graph, we run our approach 10 times to evaluate the robustness of the SimHawNet simulation. For both real-world and synthetic graphs, we perform the train-test split based on time, where the first 70% of interactions are used for training and the remaining ones for testing. Statistics about the dataset are provided in the Appendix, Table 4 and Fig. 3 showcase examples of snapshots of the dynamic graphs generated by these processes.

6.1.1 Evaluation metrics

A significant challenge for our work is the evaluation of SimHawNet. Our objective is to comprehensively assess the ability of SimHawNet to capture the static component of the graph (i.e., determine the pair of nodes that will connect) and the dynamic part (i.e., in which particular order and when these connections will happen).

Static metrics (P1) Following Huang et al. (2022), we report the clustering coefficients (CC) which measures the tendency of nodes to form tightly-knit groups or triangles, a higher CC indicates that a larger proportion of possible triangles in the network are actually closed, a property known as triadic closure. We also report the number of events (number of activations) of the simulated graphs and the accuracy that is the ratio of the number of correctly predicted edges to the total number of edges simulated. This metric can be seen as a form of true positive rate for edge activation.

Dynamic metric (P2) For the dynamic component, our goal is to evaluate the temporal coherence of our simulation. We proceed as follows: once our model is trained we extract *the intensity function* computed for each edge both on the original graph

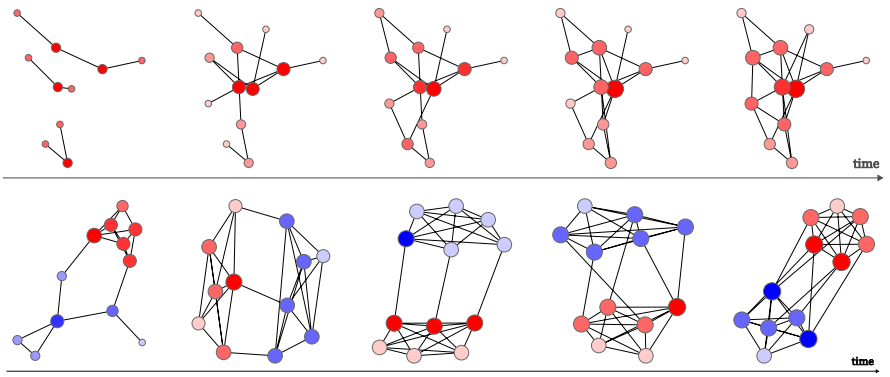


Fig. 3 Temporal snapshot of synthetic dynamic graphs generated based on the (top row) stream graph and (bottom row) SBM. The darker the node color, the higher its degree. For SBM blue and red nodes represent the community

and on the graph obtained with our fixed-parameter model. Formally, given $\lambda_{\mathcal{G}}^{(e)} \in \mathbb{R}^T$ the edge-wise intensity distribution over time for an edge e in a given graph $\mathcal{G}$, we can evaluate at a global level by computing the graph intensity as $\lambda_{\mathcal{G}} = \sum_e \lambda_{\mathcal{G}}^{(e)}$. The distance between the global intensity of two graphs $\mathcal{G}$ and $\mathcal{G}'$ is then defined by

$$\text{DTW}^{\text{global}} = DTW(\lambda_{\mathcal{G}}, \lambda_{\mathcal{G}'}).$$

where DTW corresponds to the Dynamic Time Warping (DTW) distance (Vintsyuk 1968; Sakoe and Chiba 1978), $\mathcal{G}$ is the original graph and $\mathcal{G}'$ is the graph obtained with SimHawNet. The motivation behind this proposed metric is to treat the intensity distributions over time as time sequences.

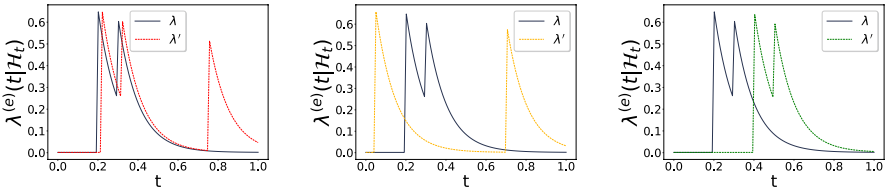
Then, we need an alignment-based metric, that relies on a temporal alignment of the sequences to assess their similarity. In this context, the popular Dynamic Time Warping (DTW) metric emerges as a natural candidate, as it can effectively accommodate minor shifts in the activation timing. An illustration of the value of the DTW between edge intensity functions in three different scenarios is provided in Fig. 4. We can see that the DTW is appropriate to assess temporal coherence as it penalizes more cases where the number of predicted activations is wrong for a given edge (Fig. 4a) and when the waiting times between activations are far from the ground truth (Fig. 4b). A DTW of 0 indicates perfect similarity.

6.2 Results

We provide the details of the optimization procedure and how we configure the hyperparameters in B.2.

6.2.1 Generation quality

Analysis of structural properties P1 Table 1 shows the results for all metrics on the synthetic networks (for more information, in Appendix B.3, we present the degree centralities of the nodes). Note that we do not present the results obtained with LSH because, despite our best efforts, we were unable to find a combination of hyperparameters that would give reasonable results. SimHawNet shows low CC for Configuration and Stream, consistent with their lack of inherent clustering, and higher CC for SBMs models, where community structure promotes triangle formation. In



(a) $DTW = 0.91$ (b) $DTW = 1.17$ (c) $DTW = 0.04$

Fig. 4 Examples of DTW values between edge intensity functions; the ground truth λ is the same across all examples and λ' represents (a) 3 activations instead of two; (b) the same number of activations but not in the same time theme and (c) the same number of activations but slightly shifted

Table 1 Mean ± std for all metrics on the synthetic networks

Network	Approach	#events	CC	$\mathrm{DTW}^{\mathrm{glob.}}$	Accuracy
Configuration	Ground truth	700 ± 20	0.1 ± 0.02	0	100
	DLS	529 ± 41	0.17 ± 0.02	129 ± 16	$\mathbf{9 \pm 1}$
	SimHawNet	$\mathbf{832 \pm 165}$	$\mathbf{0.06 \pm 0.02}$	$\mathbf{100 \pm 26}$	5 ± 1
Stream	Ground truth	577 ± 22	0.06 ± 0.01	0	100
	DLS	$\mathbf{648 \pm 84}$	0.18 ± 0.01	$\mathbf{81 \pm 6}$	$\mathbf{23 \pm 6}$
	SimHawNet	1586 ± 258	$\mathbf{0.07 \pm 0.02}$	229 ± 60	15 ± 2
SBM-a	Ground truth	489 ± 7	0.17 ± 0.01	0	100
	DLS	$\mathbf{375 \pm 17}$	0.13 ± 0.01	$\mathbf{103 \pm 12}$	$\mathbf{16 \pm 1}$
	SimHawNet	850 ± 181	0.21 ± 0.05	125 ± 37	9 ± 1
SBM-b	Ground truth	698 ± 8	0.25 ± 0.02	0	100
	DLS	499 ± 15	0.15 ± 0.01	152 ± 7	$\mathbf{15 \pm 1}$
	SimHawNet	$\mathbf{865 \pm 200}$	$\mathbf{0.20 \pm 0.08}$	$\mathbf{115 \pm 19}$	7 ± 2

CC is the clustering coefficient and Accuracy is the ratio of the number of correctly predicted edges to the total number of edges simulated. The best results are highlighted in bold.

contrast, DLS shows relatively constant CC values across all models, indicating limited sensitivity to local structure. This difference stems from SimHawNet's explicit encoding of triadic closure via features like reweighted common neighbors and interaction volume. An other key difference lies in the simulation process, DLS simulates edge activations independently, ignoring temporal dependencies between edges. In contrast, our model simulates activations sequentially, allowing each event to influence future ones (especially among neighboring edges) via updated temporal features. This enables the capture of cascading effects and local bursts, resulting in more realistic network dynamics.

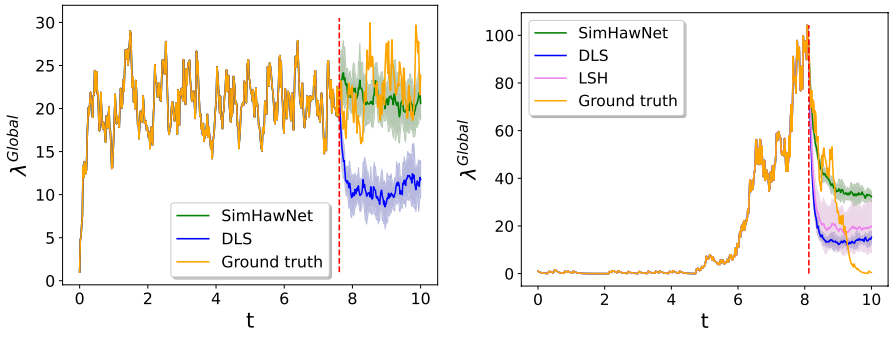
For the real-world networks discussed below, the authors provided the set of optimal hyperparameters values, so for these we do report the results of LSH. We present the results obtained on the three real-world networks in Table 2. First, we observe that LSH significantly overestimates the number of events on two networks, with discrepancies reaching a factor of 4 on Reality and exceeding a factor of 10 on MID. This issue of instability in estimating the number of events was highlighted by Huang et al. (2022) for both LSH and DLS. Interestingly, while Soliman et al. (2022) also reported that DLS is unstable and unsuitable for generating novel networks, our independent execution of their code did not confirm this limitation. Moreover, for Enron and MID datasets with a higher number of edges–our model generates over 1000 edge activations, while for Reality, which is sparser, the number of activations is significantly lower. This variation closely aligns with the structural characteristics of each dataset. In contrast, the DLS model produces a nearly constant number of events (between 600 and 700) across all three datasets, showing limited adaptability.

Analysis of temporal properties P2 Fig. 5 presents the intensities resulting from the simulation of 10 graphs using the configuration model dataset for both DLS and SimHawNet and using Enron dataset are shown including LSH. We observe that the temporal pattern of DLS remains similar across different datasets, particularly for Enron and configuration model (and across all datasets, as shown in Appendix B.4). This limitation can be traced back to the design of the DLS model, which relies on a mixture of periodic and exponential kernels that introduces regularity and smooth-

Table 2 Mean± std for all metrics on all datasets over 10 runs

Network	Approach	#events	CC	$\mathrm{DTW}^{\mathrm{glob.}}$	Accuracy
Enron	Ground truth	1133	0.23	0	100
	LSH	**1044 ± 893**	0.3 ± 0.01	309 ± 162	0.1 ± 0
	DLS	584 ± 21	**0.16 ± 0.02**	**247 ± 16**	**33 ± 2**
	SimHawNet	1467 ± 64	0.12 ± 0.02	399 ± 31	9 ± 1
MID	Ground truth	1272	0.13	0	100
	LSH	15532 ± 2253	0.46 ± 0.03	1220 ± 195	0
	DLS	698 ± 35	**0.18 ± 0.03**	223 ± 24	**45 ± 1**
	SimHawNet	**1074 ± 37**	0.29 ± 0.02	**155 ± 12**	20 ± 1
Reality	Ground truth	540	0.2	0	100
	LSH	888 ± 723	**0.2 ± 0.05**	98 ± 86	0 ± 1
	DLS	**568 ± 27**	0.19 ± 0.03	**64 ± 4**	**60 ± 1**
	SimHawNet	651 ± 25	0.25 ± 0.04	82 ± 8	29 ± 2

CC is the clustering coefficient and Accuracy is the ratio of the number of correctly predicted edges to the total number of edges simulated. The best results are highlighted in bold and the second are underlined.



(a) Config (b) Enron

Fig. 5 Global intensity function obtained. The red line marks the beginning of the simulation

ness into the temporal intensity it inherently dampens the model's ability to respond to bursts in activity. In contrast, our model demonstrates greater adaptability to temporal patterns, for the configuration model, it maintains a consistent level of edge activation when for Enron, it captures a steep decline in activation over time.

6.2.2 Parameters analysis

Now, we propose to take a closer look at the parameters learned by our model, the node embeddings learned in the base rate, the weights of handcrafted features (see Appendix B.5), and the decay rates β.

Analysis ofβ. Figure 6 shows the distribution of the β parameters learned by the model. We recall that β quantifies how the activation of an edge influences the timing of subsequent activations of the same edge. In other words, if β is high, the activation of an edge strongly increases the likelihood of another activation occurring

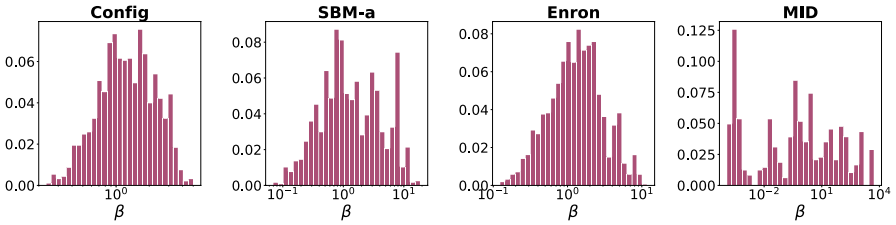


Fig. 6 Plot the percentage of the decay (β) values across the edges for the model on different learned datasets, with a logarithmic scale on the x-axis for β

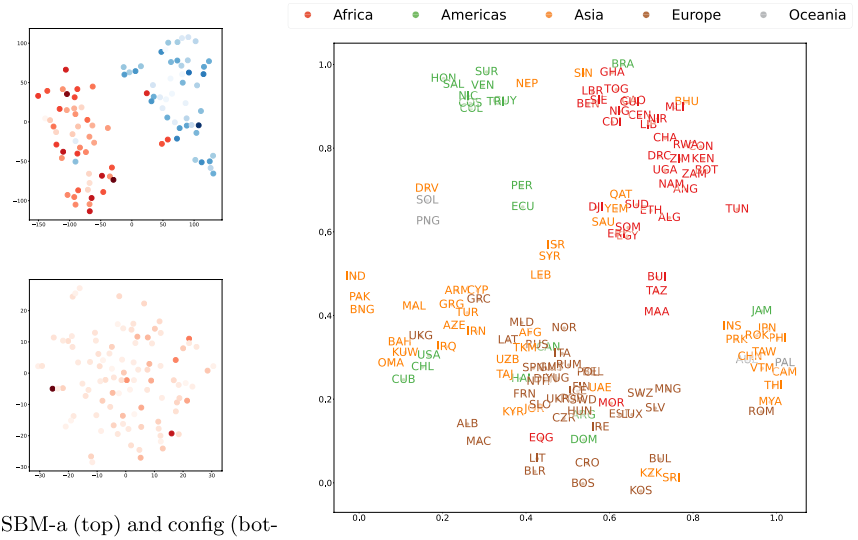
soon afterward on the same edge. In contrast, if β is low, the activation of an edge increases the likelihood of another activation occurring over a longer time interval.

For the dataset configurations SBM-a and Enron (as well as SBM-b and Stream, see Appendix B.5),we observe that the proportion of values below 1 is almost equal to the proportion of values above 1, excluding extreme cases. This indicates that the model effectively captures variations in the temporal influence of edges. This behavior is consistent with the synthetic data, where extreme edge behaviors–such as frequent repeated activations or isolated activations of an outlier edge–are not present. In the case of MID (and Reality; see Appendix B.5), we observe more extreme values (i.e., closer to 0 or greater than 10), suggesting that the model distinguishes more sharply between the impacts of different edge activations. Some activations have very short-term or negligible effects (values close to 10), while others exhibit influence over extended periods or even throughout the simulation (values close to 0). This is particularly pronounced in the MID dataset, where values reach as high as 10^3, indicating that the model interprets certain activations as singular, one-off events, while others have sustained, long-term effects. This differentiation aligns with historical observations: some conflicts are brief and isolated, while others, such as the Cold War, span much longer durations.

Analysis of the base rate Fig. 7b shows the node embeddings learned from the base rate, which are used to infer edge embeddings. For the SBM-a, the embeddings reflect the two original clusters and for config we can see disparities in the degrees of nodes (with opacity of the nodes). For the MID dataset, relevant historical conflicts are in evidence. For example, Syria, Lebanon, and Israel, with their history of enduring conflicts, cluster closely. Similarly, Peru (PER) and Ecuador (ECU), with a long-standing territorial dispute, are also close together. The United States (USA) appears near Cuba (CUB) and Iraq (IRQ), reflecting two significant events: the Cuban Missile Crisis during the Cold War and the U.S.-led invasion of Iraq in 2003. Additionally, the embedding space captures indirect relations (e.g., alliances shaped by shared adversaries) and similarities in conflict profiles, such as countries with low aggression levels compared to others.

6.2.3 Time complexity analysis

In terms of training time (material details in Appendix B.6), SimHawNet demonstrates a significantly lower computational cost compared to both DLS and LSH. Table 3 shows that it achieves a minimum speedup of 7 times over DLS, with gains



(a) SBM-a (top) and config (bottom)

(b) MID

Fig. 7 t-SNE visualization of the static node embedding learned by the model. For dynamic SBM, the two colors represent the communities. The opacity of the nodes represents their degree: from lower (light) to higher (dark)

Table 3 Training time in seconds

Model/network	Config	Stream	SBM-a	SBM-b	Enron	MID	Reality
SimHawNet	7	2	15	19	22	1087	324
LSH	NA	NA	NA	NA	566	2588	715
DLS	417	238	1174	118	6179	7450	2092

NA stands for not applicable.

reaching up to 280 times on the Enron dataset. Similarly, when compared to LSH, SimHawNet exhibits a significant efficiency advantage, being 2 to 20 times faster.

All methods share the latent space dimension as a common hyperparameter. In addition, LSH requires three decay parameters, DLS includes the number of kernels, while our model requires no extra hyperparameters beyond the latent space dimension.

6.2.4 Summary of the main results

To answer the research questions, we have demonstrated that SimHawNet can generate structurally accurate results, as shown by our results on CC, number of events

and accuracy in synthetic and real datasets. This answers **RQ1**. In addition, through analysis of SimHawNet parameters, we have shown that they are consistent with the datasets and easily interpretable, thus responding to **RQ2**. Finally, by comparing our model with state-of-the-art approaches such as DLS and LSH, we have established that our method is competitive in terms of generation quality and temporal complexity, thus responding **RQ3**.

7 Conclusion and perspectives

We introduced SimHawNet, a generative framework for temporal networks based on a modified Hawkes Process-based model coupled with a thinning simulation algorithm. Focusing on the inter-arrival times, our model is a simple yet elegant way of capturing the influence of the temporal network topology on the intensity of the dyadic point processes. Our proposed framework suggests that by encoding the history in an appropriate vector of sufficient statistics, one can very simply include inductive bias on the time-evolution of the network, and use the resulting model for simulation. We empirically demonstrated that SimHawNet can capture and reproduce the dynamic of a network on a wide range of graphs. It balances performance and efficiency, delivering results comparable to or better than state-of-the-art methods. Its ability to excel in specific scenarios while being significantly faster–by several orders of magnitude–makes it a practical tool for graph generation.

As stated before, this work is a first step towards establishing a flexible dynamic graph generation framework. For example, our model can be very useful in the study of epidemic diffusion (Machens et al. 2013), as it allows for simulating various virus spread scenarios by modifying the starting conditions. Additionally, our framework can be applied to the field of cybersecurity (Kent 2015), enabling the development of methods to detect and neutralize threats swiftly.

In further work, we plan to integrate representation learning as a substitute for the hand-crafted features (Wang et al. 2021). This will avoid the problem of choosing which features to compute, and limit the cost of updating them. In addition, it will let us study the effects of higher-order history dependency on the quality of the model.

Appendix A Algorithmic details of SimHawNet

We provide the full pseudo-code of the SimHawNet simulation algorithm.

Input: initial time t_n, upper bound T, max iterations N

Initialize: $t = t_n$, $\mathcal{H}_t = \mathcal{H}_{t_n}$, $\lambda^{(*)} = \lambda(t_n|\mathcal{H}_{t_n})$
while $t < T$ or $n < N$ **do**
  $u \sim \text{Uniform}([0,1])$
  $\Delta_t = \min_e \left(-\frac{\ln(u^{(e)})}{\lambda^{(e)(*)}}\right)$
  Set $t = t + \Delta_t$
  $\lambda_{\text{current}} = \lambda(t|\mathcal{H}_t)$
  $U \sim \text{Uniform}([0, \lambda^{(*)}])$
  if $U < \lambda_{\text{current}}$ **then**
    Define for each edge $e \in \mathcal{E}_{\text{active}}$ activated the probability

$$p^{(e)} = \frac{\lambda^{(e)}(t|\mathcal{H}_t)}{\sum_{e' \in \mathcal{E}_{\text{active}}} \lambda^{(e')}(t|\mathcal{H}_t)}$$

    Sample $e_{\text{new}} \sim \text{Multinomial}(1, \{p_e | e \in \epsilon\})$
    Add the event (e_{new}, t) in $\mathcal{H}_t$
    $t_{last} = t$ and $\mathcal{H}_{last} = \mathcal{H}_t$
    $\lambda^{(*)} = \lambda(t_{last}|\mathcal{H}_{\text{last}})$
  end if
  Set $n = n + 1$
end while

Algorithm 1 SimHawNet simulation algorithmAppendix B Additional details and results

B.0.1 Synthetic graph generation

Hereafter, we propose to derive different dynamic synthetic models, two of them being dedicated to the generation of static graphs, namely the configuration model and the Stochastic Block Model (SBM), and the third one based on stream graphs (Latapy et al. 2017). For static models, we describe below the modifications made to obtain dynamic versions. The aim is to cover a large diversity of network generation models as the literature tends to focus on SBM models only, therefore assuming a strong community structure of the network.

Dynamic configuration model The configuration model is a technique used to generate random static networks based on a given degree sequence. Frequently used as a reference for real social networks, it allows the integration of various degree distributions. We take advantage of this model to create dynamic graphs, excluding self-loops. The procedure is defined as follows. We first choose a fixed number of node $\{v_1, ..., v_n\}$ with their associated degree sequence $[d_1, ..., d_n]$ for $i \in \{1, ..., n\}$, $d_i \sim \mathcal{N}(m_i, \sigma)$ and generate our initial graph based on the configuration model. Then, for a fixed number of iterations, we repeat the following two steps: (1) modify m_i by incrementing a predefined natural integer drawn from a uniform distribution and simulate the associated graph; (2) add the edge list from the previous graph, including associated timestamps, into the final simulation.

Temporal SBM Secondly, we use a generative process based on SBM. We fix two communities, similar to the configuration model we set a fixed number of iterations. At each iteration, we generate an SBM graph with a fixed block probability and add

random timestamps to each formed edge. The final SBM multigraph is the concatenation of the different SBM graphs along with their timestamps. We also create a scenario where one of the communities disappears by merging with the other. After some time the unique community gets split again.

Stream Graph We opt for the stream graph approach (Latapy et al. 2017) to create the last type of graph. Stream graphs depict temporal data sequences alongside nodes and links, providing a dynamic perspective on evolving node relationships over time. We use Straph[4] that simulates stream graphs that follow Erdős-Rényi behavior (i.e., the activation of nodes follows a Poisson distribution). The time associated with each edge activation in the stream graph is continuous, meaning that each edge has a time interval of activation instead of a discrete time as in our framework. To convert time interval into discrete time we select n discrete times within this interval, where n is proportional to the length of the interval (i.e., for a longer interval, we will have more edge activations than for a shorter interval).

B.1 Network properties

Table 4 presents additional details regarding the synthetic and real-world networks.

B.2 Optimization procedure and hyperparameters

The convergence criterion used is early stopping to prevent overfitting. During training, the model's performance is monitored on a validation set. If the loss does not improve for a certain number of consecutive epochs (100 in our case), training is stopped early.

Other hyperparameters include the dimension of the edge embedding and the batch history size. For the former, the dimension is set to 64, which has proven sufficient to achieve satisfactory loss values (although no extensive grid search was performed). The history size corresponds to the maximum number of interactions

Table 4 Properties of the synthetic and real-world networks

Dataset	#events	#nodes	Density	CC
Config	2800 ± 75	100	0.234 ± 0.004	0.31 ± 0.01
Stream	2307 ± 88	100	0.101 ± 0.003	0.1 ± 0.01
SBM-a	1956 ± 26	100	0.288 ± 0.005	0.5 ± 0.01
SBM-b	2791 ± 32	100	0.385 ± 0.005	0.6 ± 0.01
Enron	4529	167	0.31	0.06
MID	5088	145	0.03	0.27
Reality	2159	92	0.04	0.24

We report the number of events that occurred during the time of observation, the number of nodes, the density of the network, and the clustering coefficient (CC). For simulated networks, we report mean ± std computed over 10 synthetic networks generated with the same configuration.

[4] https://github.com/StraphX/Straph?tab=readme-ov-file.

observed on a single edge in the training data. In other words, we consider the whole available history.

B.3 Degree centrality nodes

Figure 8 illustrates the distribution of degree centrality between nodes in graph simulations performed by different models: SimHawNet, DLS and LSH. Our model demonstrates superior performance in capturing node centrality for the SBM-b case compared to DLS. This dataset represents a graph undergoing dynamic transformations over time, with clusters activated sequentially.

Furthermore, we observe that our model identifies central nodes more efficiently. This is evident in datasets such as Enron, MID, SBM-b and Stream, where our model produces more elongated violin plots covering a range closely corresponding to the ground truth. This indicates better alignment with the true distribution of degree centrality.

B.4 Global intensity results

Figure 9 presents the intensities produce by our model, DLS and LSH on the other dataset.

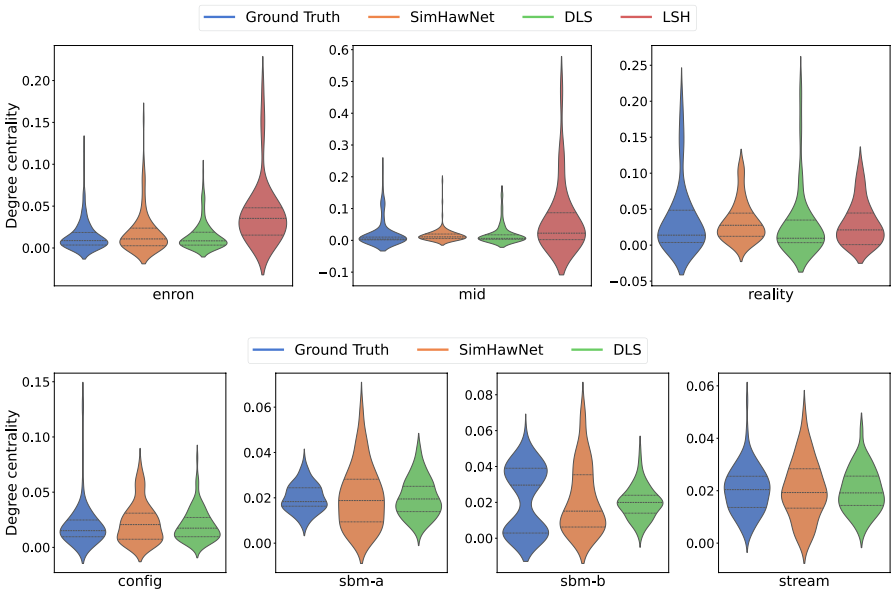


Fig. 8 Distribution of node degree centrality

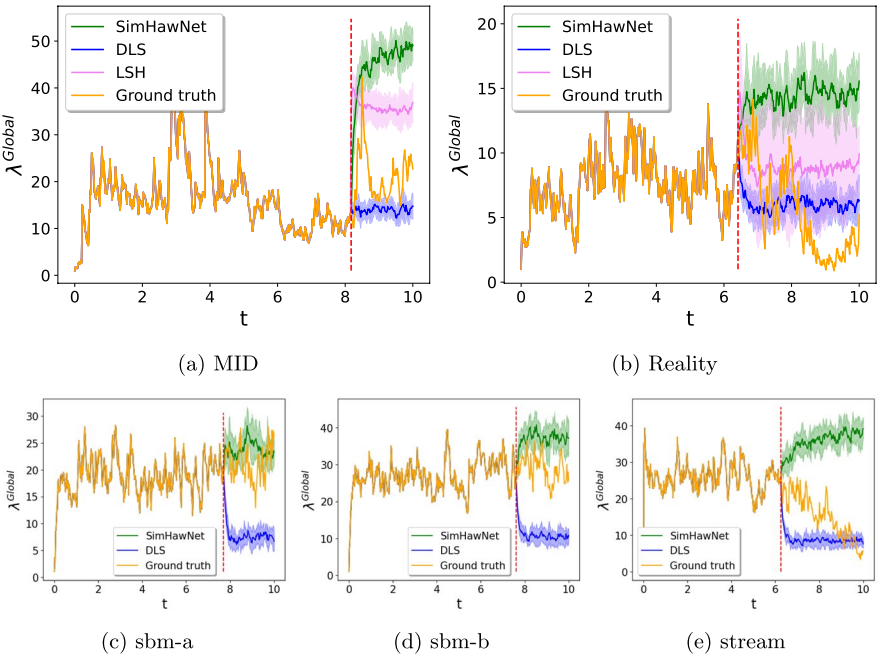


Fig. 9 Global intensity functions obtained. The red line marks the beginning of the simulation. Each subfigure corresponds to different datasets or configurations

B.5 Parameters analysis

Figure 10 shows the mean and variance of the temporal feature weights learned by our model in different graphs. We observe a consistent pattern for the SBM-a, Enron and Reality datasets: the volume and preferential attachment features have similar importance, while the common neighbors feature has a comparatively weaker influence. Moreover, we observe a significant difference in the coefficient of the feature

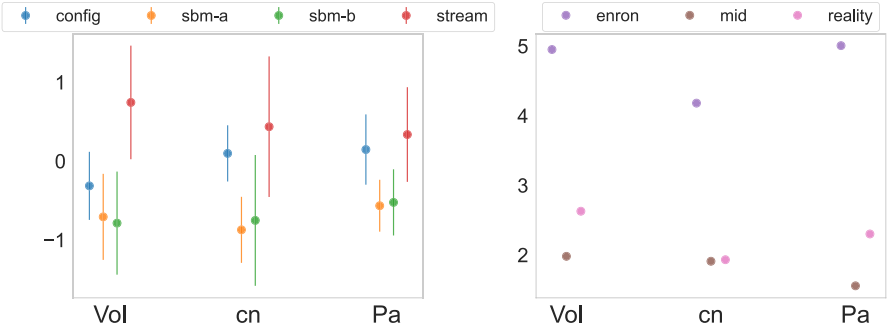


Fig. 10 Parameters of g learned by our model for (left) synthetic networks and (right) real-world networks. Features: volume (**Vol**), common neighbors (**cn**), and preferential attachment (**Pa**)

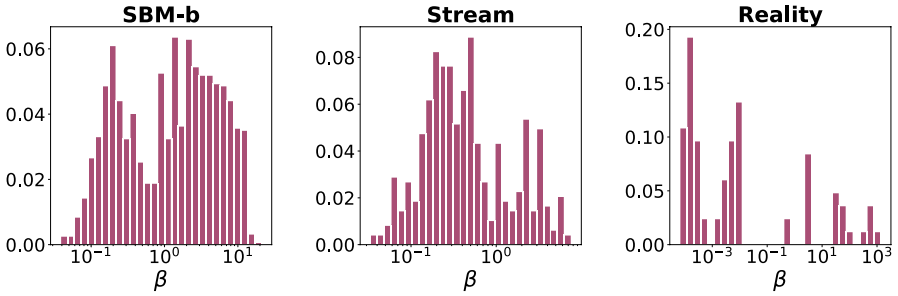


Fig. 11 Plot the percentage of the decay (β) values across the edges for the model on different learned datasets, with a logarithmic scale on the x-axis for β

Volume between the Stream dataset and the others. This higher value is consistent with the nature of the Stream model, which tends to exhibit more concentrated edge activations over short time periods. In contrast, for the Configuration model, the features Common Neighbors and Preferential Attachment have higher coefficients than Volume. This can be explained by the model's design, which relies on node degrees (i.e., the number of neighbors) to generate new edge activations. Regarding the real-world datasets, we note that Enron and Reality exhibit similar coefficient patterns. This is expected, as both datasets represent communication networks.

Figure 11 shows the distribution of the β parameters learned by the model.

B.6 Material details

We train the models on 2x Intel Xeon E5-2695 CPUs.

Acknowledgements The research leading to these results has received funding from the Special Research Fund (BOF) of Ghent University (BOF20/IBF/117), from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme, and from the FWO (project no. G0F9816N, 3G042220, G073924N). This project has also received funding from the PEPR-IA through the project FOUNDRY.

Funding Open access funding provided by Télécom Paris.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Arastuie M, Paul S, Xu KS (2020) CHIP: a Hawkes process model for continuous-time networks with scalable and consistent estimation. NeurIPS 33:16983–16996

Bacry E, Bompaire M, Deegan P et al (2018) Tick: a python library for statistical learning, with an emphasis on Hawkes processes and time-dependent models. J Mach Learn Res 18(214):1–5

Bois CD, Butts CT, Smyth P (2013) Stochastic blockmodeling of relational event dynamics. J Mach Learn Res 31:238–246

Bonifati A, Holubová I, Prat-Pérez A et al (2020) Graph generators: state of the art and open challenges. ACM Comput Surv 53(2):1–30

Butts CT (2008) A relational event framework for social action. Sociol Methodol 38(1):155–200

Corneli M, Latouche P, Rossi F (2016) Block modelling in dynamic networks with non-homogeneous Poisson processes and exact ICL. Soc Netw Anal Min 6(1):55:1-55:14

Dai H, Wang Y, Trivedi R, et al (2016) Deep coevolutionary network: embedding user and item features for recommendation. Preprint at arXiv:1609.03675

Daley D, Vere-Jones D (2010) An introduction to the theory of point processes volume I, vol I

De Bie T (2011) Maximum entropy models and subjective interestingness: an application to tiles in binary databases. Data Min Knowl Discov 23(3):407–446. https://doi.org/10.1007/s10618-010-0209-3

Durante D, Dunson DB (2016) Locally adaptive dynamic networks. Ann Appl Stat 10:2203–2232

Eagle N, Pentland AS (2006) Reality mining: sensing complex social systems. Personal Ubiquitous Comput 10(4):255–268

Farajtabar M, Gomez-Rodriguez M, Wang Y, et al (2018) COEVOLVE: A joint point process model for information diffusion and network Co-evolution. J Mach Learn Res. pp 473–477. arXiv:1507.02293

Goldenberg A, Zheng AX, Fienberg SE et al (2009) A survey of statistical network models. Sociol Methodol 2(2):129–233 arXiv:0912.5410

Hawkes AG (1971) Spectra of some self-exciting and mutually exciting point processes. Biometrika 58(1):83–90

Holland PW, Leinhardt S (1981) An exponential family of probability distributions for directed graphs. J Am Stat Assoc 76(373):33–50. https://doi.org/10.1080/01621459.1981.10477598

Holme P, Saramäki J (2012) Temporal networks. Phys Rep 519:97–125

Huang Z, Soliman H, Paul S, et al (2022) A mutually exciting latent space Hawkes process model for continuous-time networks. In: UAI, pp 863–873

Kent AD (2015) Cybersecurity data sources for dynamic network research. In: Dynamic networks in cybersecurity. Imperial College Press

Kim B, Lee KH, Xue L et al (2018) A review of dynamic network models with latent variables. Stat Surv 12(none):105–135

Krivitsky P, Handcock M (2014) A separable model for dynamic networks. J R Stat Soc Ser B Stat Methodol 76:29–46

Kumar S, Zhang X, Leskovec J (2019) Predicting dynamic embedding trajectory in temporal interaction networks. In: Proceedings of the ACM SIGKDD Conference, pp 1269–1278

Latapy M, Viard T, Magnien C (2017) Stream graphs and link streams for the modeling of interactions over time. Soc Netw Anal Min 8:1–29

Latapy M, Fiore M, Ziviani A (2019) Link streams: methods and applications. Comput Netw 150:263–265

Machens A, Gesualdo F, Rizzo C et al (2013) An infectious disease model on empirical networks of human contact: bridging the gap between dynamic network data and contact matrices. BMC Infect Dis 13:1–15

Matias C, Rebafka T, Villers F (2018) A semiparametric extension of the stochastic block model for longitudinal networks. Biometrika 105(3):665–680

Mikolov T, Sutskever I, Chen K, et al (2013) Distributed representations of words and phrases and their compositionality. arXiv:1310.4546

Masuda N, Lambiotte R (2016) A guide to temporal networks. World scientific, Singapore

Newman MEJ (2003) The structure and function of complex networks. SIAM Rev 45(2):167–256

Ogata Y (1981) On Lewis' simulation method for point processes. IEEE Trans Inf Theory 27(1):23–31

Palmer G, McManus RW, D'Orazio V et al (2022) The mid5 dataset, 2011–2014: procedures, coding rules, and description. Confl Manag Peace Sci 39(4):470–482

Panzarasa P, Opsahl T, Carley KM (2009) Patterns and dynamics of users' behavior and interaction: network analysis of an online community. J Am Soc Inform Sci Technol 60(5):911–932

Passino FS, Heard NA (2023) Mutually exciting point process graphs for modeling dynamic networks. J Comput Graph Stat 32(1):116–130
Perry PO, Wolfe PJ (2013) Point process modelling for directed interaction networks. J R Stat Soc Ser B Stat Methodol 75(5):821–849
Rastelli R, Corneli M (2023) Continuous latent position models for instantaneous interactions. Netw Sci 11(4):560–588
Robins G, Pattison P, Kalish Y et al (2007) An introduction to exponential random graph (p *) models for social networks. Soc Netw 29:173–191. https://doi.org/10.1016/j.socnet.2006.08.002
Romero R, Lijffijt J, Rastelli R et al (2023) Gaussian embedding of temporal networks. IEEE Access 11:117971–117983
Rossi E, Chamberlain BP, Frasca F, et al (2020) Temporal graph networks for deep learning on dynamic graphs. ArXiv arXiv:2006.10637
Rozenshtein P, Gionis A (2019) Mining temporal networks. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery & data mining, pp 3225–3226
Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans Acoust Speech Signal Process 1:43–49
Soliman H, Zhao L, Huang Z, et al (2022) The multivariate community Hawkes model for dependent relational events in continuous-time networks. arXiv:2205.00639
Trivedi R, Farajtabar M, Biswal P, et al (2019) Dyrep: learning representations over dynamic graphs. In: International conference on learning representations
Vintsyuk TK (1968) Speech discrimination by dynamic programming. Cybernetic 4(1):52–57
Vu DQ, Asuncion AU, Hunter DR, et al (2011) Dynamic egocentric models for citation networks. Proceedings of the 28th international conference on machine learning, ICML 2011 pp 857–864
Wang Y, Chang Y, Liu Y, et al (2021) Inductive representation learning in temporal networks via causal anonymous walks. In: International conference on learning representations, ICLR. OpenReview.net
Wasserman S, Pattison P (1996) Logit models and logistic regressions for social networks: I. An introduction to Markov graphs and p*. Psychometrika 61:401–425. https://doi.org/10.1007/BF02294547
Xu D, Ruan C, Korpeoglu E, et al (2020) Inductive representation learning on temporal graphs. In: Eighth international conference on learning representations
Yang J, Rao VA, Neville J (2017) Decoupling homophily and reciprocity with latent space network models. In: UAI

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.