TempoBiGen: A Curated Generative Model for Healthcare Mobility Logs with Visit Duration

Hieu Vu[1], Alberto M. Segre[1], and Bijaya Adhikari[1] (✉)

University of Iowa, Iowa City IA 52242, USA
{hieu-vu,alberto-segre,bijaya-adhikari}@uiowa.edu

Abstract. Healthcare facilities, which serve vulnerable populations such as patients and the elderly, are also hotspots where pathogens drive nosocomial infections. Accurately modeling pathogen transmission within these settings is essential for understanding their dynamics and enhancing preparedness and intervention strategies. A key barrier to achieving high-fidelity models of pathogen transmission within healthcare facilities is the scarcity of fine-grained, high-quality mobility logs that capture real-world interactions. Data synthesis offers a promising solution by generating realistic mobility datasets. Existing methods can generate synthetic mobility logs while preserving the temporal evolution of the original data's structural properties. However, these approaches are limited in two key ways: (1) they typically overlook the bipartite structure inherent in mobility logs (e.g., interactions between healthcare workers and rooms), and (2) they fail to account for the duration of interactions, a critical factor in transmission dynamics. Building on top of existing work, we introduce TempoBiGen, a curated generative model designed to address these shortcomings. TempoBiGen explicitly models bipartite temporal networks and incorporates visit duration in a post-processing step, producing high-fidelity, ready-to-use synthetic mobility logs. We evaluated TempoBiGen using real-world mobility logs gathered from healthcare facilities, assessing its performance in preserving snapshot-based graph properties (e.g., degree distribution and connected components size) and replicating temporal dynamics through disease spread simulations. Our results demonstrate that the proposed approach leads to a robust and effective tool for generating synthetic mobility data, offering additional resources to enhance modeling and analyzing hospital mobility patterns.

Keywords: Deep Temporal Graph Generative Model · Hospital Bipartite Graph · Interaction duration modeling.

1 Introduction

Contact patterns between healthcare providers (HCPs) and patients observed during patient-care delivery within healthcare facilities also serve as potential pathways for *Healthcare Associated Infections* (HAIs) and *Antimicrobial Resistance Organisms* (AMROs) transmissions [10,22,2]. Moreover, these interactions

also facilitate the spread of respiratory infections such as influenza and COVID-19 [3,17]. Since healthcare facilities such as inpatient units and long-term care facilities house vulnerable populations, it is imperative to *i)* model accurate dynamics of both fomite-mediated (for HAIs and AMROs) [12] and person-person infection (for respiratory diseases) spread and *ii)* infer latent infections [13], and *iii)* design effective intervention strategies to mitigate further harm [16].

A major obstacle to this is the lack of fine-grained high-resolution mobility data collected from healthcare facilities. Such data is challenging to obtain as there are numerous technical and administrative hurdles [4]. A number of prior work circumvent the lack of data by leveraging randomized (random mixing [11] or random graphs [15]) and/or statistical (stochastic block models, Watts-Strogratz models [1,27]) contact pattern models for disease spread simulation. A different line of prior work employs wearable sensors (such as Radio-frequency identification devices, RFID) in healthcare facilities to capture the mobility patterns [5]. However, these studies are often limited to small facilities or to a unit/ward within a larger facility, as large-scale deployment still remains a challenge.

Even when fine-grained healthcare mobility data is available, sharing the data with public health experts and epidemiological modelers is challenging, as these data typically have numerous policy restrictions [8]. These administrative restrictions are primarily designed to protect patients' privacy. To enable high-fidelity epidemiological modeling within healthcare facilities while respecting these restrictions, here we ask the following question: Given a large corpus of fine-grained healthcare mobility data, is it possible to generate a *novel* yet *representative* synthetic data? Here, by novel, we mean that the generated data is only allowed a non-significant overlap with the original data, and by representative, we mean that the generated data has similar statistical properties as the original one.

An obvious solution to the question above is to represent the mobility log as a temporally dynamic network and use an existing off-the-shelf graph generative model [9,29] to generate synthetic data. However, most existing approaches blindly train a generative model (such as adversarial generative model, diffusion model, or probabilistic model) over graphs with the hope of capturing statistical properties such as degree distribution, clustering coefficients, and diameter. These approaches fail to capture the intricate relationship between the HCWs and patients, the domain constraints posed by HCWs shifts and care patterns, and other HCW behaviors such as the time between visits. To overcome the shortcomings mentioned above, we start with an existing temporal graph generative model and extend it to include bipartite constraints in our model. We then further extend to generate realistic visit duration (which prior works ignore).

Our contributions are as follows:

- We propose an extension to an existing temporal graph generation model to account for the bipartite nature of the hospital mobility data and to explicitly model visit duration for fine-grained data generation.

– We conduct extensive experiments with mobility data collected from large-scale healthcare facilities and contrast the generated data against the original data via a diverse set of metrics, including snapshot-based graph properties, HCWs shift properties, and disease spread simulations. We showed that our model is able to generate realistic mobility logs for a variety of units within healthcare facilities, each exhibiting unique care patterns.

2 Related Work

Mobility logs collection in Healthcare Facilities. Numerous efforts have been made to collect fine-grained, high-quality hospital mobility logs to improve the understanding of care patterns and facilitate early predictions of healthcare-associated infections (HAIs). Several studies have explored the use of wearable devices to track interactions among healthcare workers (HCWs) and between HCWs and specific locations [26,21,6]. While these approaches provide detailed mobility data, they are often constrained by small participant pools, short data collection periods, and the high cost of wearable devices, limiting their scalability and generalizability.

Data Synthesis for Temporal Interaction Graphs. Synthetic data generation for temporal interaction graphs focuses on preserving both structural and temporal patterns. Prior approaches fall into two categories: *motif-based methods* and *deep generative models*. *Motif-based methods* use recurring substructures to guide generation. Early work modeled the evolution of motifs over time, albeit in discretized snapshots that loses detailed timestamp information. For example, STM [24] and DyMOND [28] either utilize a frequency-based method or model the arrival rates of a fixed set of atomic static motifs (edge, wedge, triangle) over active node sequences. The MTM model [19] extended this by dynamically learning transitions among arbitrary motifs. These approaches are efficient and interpretable but limited in capturing global dynamics and rich node/edge features. *Deep generative models* offer greater expressiveness. TAGGEN [30], TG-GAN [29] and STGEN [18] utilize the GAN framework to improve fidelity by modeling edge timestamps in continuous time. However, they over rely on deep architecture to capture the temporal dynamics, which may limit interpretability and scalability. TIGGER [9] advanced this approach with a scalable recurrent model based on Temporal Point Processes, enhancing fine-grained temporal accuracy and scalability. Collectively, these methods highlight an ongoing shift towards integrating scalability and more expressive temporal modeling in generative frameworks.

In summary, existing approaches for temporal interaction graph generation range from motif-based methods emphasizing the benefit of capturing local structures to deep generative models that capture intricate temporal dependencies. However, to the best of our knowledge, no prior work has addressed the generation task specifically for bipartite temporal networks jointly with visit duration. To address this gap, we introduce TempoBiGen, which explicitly models bipartite temporal networks and incorporates visit duration in a post-processing

step, producing high-fidelity, ready-to-use synthetic mobility logs for downstream tasks such as disease simulation.

3 Our Approach

HCW mobility within healthcare facilities is best represented as a *Temporal Bipartite Graph*, $\mathcal{G}(\mathcal{H}, \mathcal{R}, \mathcal{E}, \mathcal{C})$ where the first partition $\mathcal{H}$ represents HCWs and the second component $\mathcal{R}$ represents the rooms occupied by patients. A temporal bipartite edge $e(h, r, t, d) \in \mathcal{E}_c$ describes an interaction - a visit of a HCW $h \in \mathcal{H}$ to a room $r \in \mathcal{R}$ starting at time t for a total duration of d within a class $c \in \mathcal{C}$. In our application, a class is a healthcare unit within a hospital where the logs are collected.

Problem Description. We pose the problem of generating synthetic healthcare mobility logs as a one-shot generative problem, where only one instance of temporal graph $\mathcal{G}$ is revealed to the learning algorithm with the goal of learning a generative model $P(\mathcal{G})$ that maximizes the likelihood of generating $\mathcal{G}$ by leveraging its structural and temporal properties. Once trained, one can sample a new mobility log $\mathcal{G}'$ from $P(\mathcal{G})$. Ideally, $\mathcal{G}'$ should have similar structural/temporal properties as $\mathcal{G}$ and lead to a similar dynamics of HAI spread while having as few overlap with $\mathcal{G}$ as possible.

Approach Overview. We propose a two-stage solution to our problem. In *Stage 1*, we learn and sample $\tilde{\mathcal{G}}' \sim P(\tilde{\mathcal{G}}')$ using our proposed conditional bipartite recurrent generative model, where $\tilde{\mathcal{G}}'$ is a modified version of $\mathcal{G}$ without duration information. In *Stage 2*, after estimating the probability distribution of visit durations from the training data, we sample durations for each generated visit in $\tilde{\mathcal{G}}'$ to construct the final sampled graph $\mathcal{G}'$. An overview of our proposed method is illustrated in Figure 1.

3.1 Stage 1: Learn $\mathbf{P}(\tilde{\mathcal{G}}')$

Recall that $\tilde{\mathcal{G}}'$ does not have duration and is a simple streaming temporal bipartite graph. To learn $P(\tilde{\mathcal{G}}')$, we extend an existing temporal graph generation approach, TIGGER [9], to handle bipartite edges.

Background on TIGGER and Temporal Random Walk Modeling. To model temporal interaction graphs, TIGGER first extracts temporal random walks from the graph and models their distribution using Temporal Point Processes (TPPs) [7] combined with autoregressive modeling. This approach provides flexibility in capturing temporal dynamics. Formally, given a temporal graph $\mathcal{G}$, TIGGER first samples a set of random walks $\mathcal{S}$ to define $P(\mathcal{G}) := \prod_{S \in \mathcal{S}} P_\theta(S)$, where $P_\theta(S)$ is parameterized by a recurrent generative model with a TPP-based event time modeling head. By operating on random walks, TIGGER effectively leverages the sparsity of real-world graphs, making it scalable to large networks. Additionally, its simple model design allows for extension to inductive settings by incorporating a multi-node decoder. Instead of learning a distribution over node IDs, this decoder jointly learns a mixture model of node

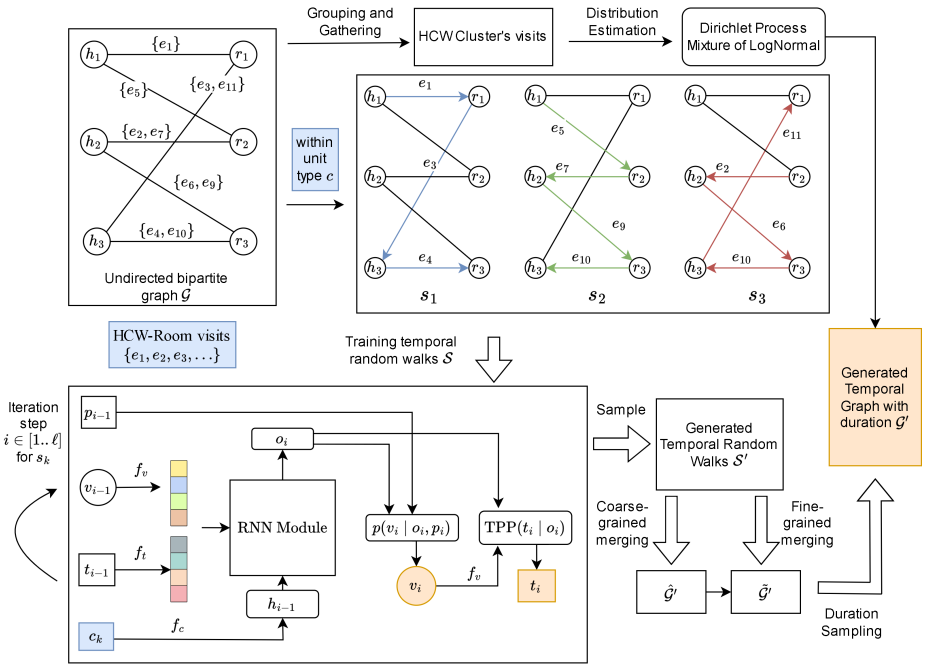

Fig. 1. Overview of proposed method.

embeddings. In this work, to mitigate the data scarcity problem in modeling a single continuous-time temporal graph, we adopt the same assumption of transferring the learning task from modeling the distribution of a temporal graph to modeling the distribution of conditional temporal random walks and only focus on the transductive generative task.

Conditional temporal random walks on Bipartite graphs. We first extract random walks in the temporal bipartite network $\mathcal{G}$. Formally, a conditional random walk on a bipartite temporal graph is defined as follows:

Definition 1. *Given a class c, we define a conditional temporal random walk (CDTR) S on a bipartite interaction graph $\mathcal{G}$ of length ℓ, starting from a node v at time t, as a sequence $S = \{s_1, s_2, \ldots, s_\ell\}$, where each tuple $s_i \in S$ is a (node, time) pair. The walk starts with $s_1 = (v, t)$, and for all $i \in [2..\ell]$, we require that $(s_{i-1}.v, s_i.v, s_i.t) \in \mathcal{E}_c$, and $s_{i-1}.v$ and $s_i.v$ belong to different node partitions.*

Conditional bipartite recurrent generative model (BiTIGGER). Note that, while TIGGER can be trained on temporal bipartite random walks, it does not explicitly model the bipartite nature of the graph, which can lead to the generation of non-bipartite walks (and eventually non-bipartite graphs). Hence, for our application, it is critical to modify TIGGER to handle the bipartite nature of our mobility logs. We start by defining the probability of a conditional

bipartite temporal graph for a given class c, $P(\mathcal{G}|c) \coloneqq P(\mathcal{S}|c)$, with $\mathcal{S}$ being the set of CTRWs extracted from $\mathcal{G}$ (Definition 1). Assuming the independence of random walks, it can be expressed as the product of the probabilities of individual random walks, i.e., $P(\mathcal{S}|c) = \prod_{S \in \mathcal{S}} p(S|c)$. As we only generate conditional RWs, to ease our notation, we will omit c in the condition and recall it when necessary. In a manner similar to [9], each individual $p(S)$ can be defined as the product of the time and the node probabilities as follows:

$$p(S) = p(s_1) \prod_{i=2}^{\ell} p(s_i.v \mid (s_1, \ldots, s_{i-1})) \times p(s_i.t \mid (s_i.v, (s_1, \ldots, s_{i-1}))) \tag{1}$$

Each of the conditional probabilities for node and time can now be modeled using a Recurrent Neural Network (RNN) [20]. The hidden state and output state of a RNN cell is updated as $\mathbf{h}_i = \text{rnn}_\theta^{\text{hidden}}(\mathbf{h}_{i-1}, s_{i-1})$ and $\mathbf{o}_i = \text{rnn}_\theta^{\text{output}}(\mathbf{h}_{i-1}, s_{i-1})$ respectively. We further incorporate the class condition into the initial hidden state $\mathbf{h}_0 = \mathbf{f}_c(c)$, where $\mathbf{f}_c(\cdot)$ is a learnable embedding function. We can simplify the conditional probabilities as follows:

$$p(S) = p(s_1) \prod_{i=2}^{\ell} p(s_i.v \mid \mathbf{o}_i) \times p(s_i.t \mid s_i.v, \mathbf{o}_i) \tag{2}$$

To explicitly model the bipartite nature of our random walks, we introduce three key modifications. First, we define a partition indicator p_i for each node v_i in the walk, which alternates between the two node partitions—$\mathcal{H}$ and $\mathcal{R}$. This ensures that consecutive nodes in a bipartite random walk always belong to different partitions.

Second, we employ dedicated prediction heads for each partition indicated by p_i. By doing so, we restrict the prediction at each step to only the appropriate partition, thereby guaranteeing that generated edges always connect nodes from different partitions. This partition-specific prediction also reduces the prediction space at each step, simplifying the learning process and improving model efficiency.

Finally, to determine when a random walk should terminate, we introduce a special end-of-walk token, denoted by $\perp$, which does not belong to either partition. At each step, the model first predicts the probability of ending the walk with $\perp$, and if not, proceeds to predict the next node from the appropriate partition as indicated by p_i. This approach ensures that the generated random walks are both valid and consistent with the bipartite structure of the underlying graph.

The overall conditional probability of generating a node $s_i.v$ at step i is thus computed as follows:

$$\begin{aligned} p(s_i.v = v \mid \mathbf{o}_i) = {} & p(\perp|\mathbf{o}_i) \times \mathbb{I}[s_i.v = \perp] \\ & + (1 - p(\perp|\mathbf{o}_i)) \times \mathbb{I}[p_i = \mathcal{H}] \times p(s_i.v = v|\mathbf{o}_i, p_i = \mathcal{H}) \\ & + (1 - p(\perp|\mathbf{o}_i)) \times \mathbb{I}[p_i = \mathcal{R}] \times p(s_i.v = v|\mathbf{o}_i, p_i = \mathcal{R}) \end{aligned} \tag{3}$$

with the probabilities $p(s_i.v = v|\mathbf{o}_i, p_i)$ and $p(\perp|\mathbf{o}_i)$ expanded as follows:

$$\begin{aligned} p(s_i.v = v|\mathbf{o}_i, p_i) &= \theta_v^{(p_i)}(\mathbf{o}_i) \\ &= \theta_v^{(p_i)}(\mathrm{rnn}_\theta^{\mathrm{output}}(\mathbf{h}_{i-1}, (s_{i-1}.v, s_{i-1}.t))) \\ &= \theta_v^{(p_i)}(\mathrm{rnn}_\theta^{\mathrm{output}}(\mathbf{h}_{i-1}, (\mathbf{f}_v(s_{i-1}.v) \parallel \mathbf{f}_t(s_{i-1}.t)))) \\ &= \frac{\exp(\mathbf{W}_v^{O^{(p_i)}}\mathbf{o}_i)}{\sum_{\forall u \in \mathcal{V}} \exp(\mathbf{W}_u^{O^{(p_i)}}\mathbf{o}_i)} \end{aligned} \tag{4}$$

where $\mathbf{W}_v^{O^{(p_i)}}$ is in $\mathbb{R}^{|\mathcal{H}|\times d_O}$ if $v \in \mathcal{H}$ and $\mathbb{R}^{|\mathcal{R}|\times d_O}$ if $v \in \mathcal{R}$, $\theta_v^{(p_i)}$ is parameters for the prediction head for partition p_i, and d_O is the dimension of vector $\mathbf{o}_i$. Similar to [9], $\mathbf{f}_v(\cdot)$ and $\mathbf{f}_t(\cdot)$ are embedding functions for the node and time, respectively, and $\parallel$ denotes concatenation. We also have $p(\perp|\mathbf{o}_i) = \mathrm{Sigmoid}(\mathbf{W}_\perp^O\mathbf{o}_i)$ with $\mathbf{W}_\perp^O \in \mathbb{R}^{1\times d_O}$.

Proposed by [25] and employed by [9], TPPs under the form of a mixture of log-normal distribution showcases strong performance in modeling inter-event time within a sequence of events. We also utilize it here and provide its formulation for the sake of completeness:

$$\begin{aligned} p(s_i.t \mid s_i.v, \mathbf{o}_i) &= p(s_i.t - s_{i-1}.t \mid s_i.v, \mathbf{o}_i) = \theta_t(\Delta t \mid s_i.v, \mathbf{o}_i) \\ &= \sum_{k=1}^{K} \phi_k^K \frac{1}{\Delta t \sigma_k^K \sqrt{2\pi}} \exp(-\frac{(\log \Delta t - \mu_k^K)^2}{2(\sigma_k^K)^2}) \end{aligned} \tag{5}$$

where Δt is time difference between $s_i.t$ and $s_{i-1}.t, p(t)$ is parameterized by θ_t and $\mu_k^K, \sigma_k^K, \phi_k^K$ are parameters of θ_t.

$$\mu_k^K = \mathbf{W}_k^{\mu K}(\mathbf{f}_v(s_i.v) \parallel \mathbf{o}_i), \quad \sigma_k^K = \exp(\mathbf{W}_k^{\sigma K}(\mathbf{f}_v(s_i.v) \parallel \mathbf{o}_i))$$

$$\phi_k^K = \frac{\exp(\mathbf{W}_k^{\phi K}(\mathbf{f}_v(s_i.v) \parallel \mathbf{o}_i))}{\sum_{j=1}^{K} \exp(\mathbf{W}_j^{\phi K}(\mathbf{f}_v(s_i.v) \parallel \mathbf{o}_i))}$$

with K is number of components in the Log-Normal Mixture distribution and $\mathbf{W}_k^{\mu K}, \mathbf{W}_k^{\sigma K}, \mathbf{W}_k^{\phi K} \in \mathcal{R}^{(d_V + d_O)}, \forall k$. Note that every component's learnable weights are shared across each time stamp in the sequence.

Similar to TIGGER, we train our proposed model to minimize the negative log-likelihood:

$$\mathcal{L}(\mathcal{S}) = \sum_{c \in \mathcal{C}} -\log p(\mathcal{S}|c) = -\sum_{c \in \mathcal{C}} \sum_{S \in \mathcal{S}} \log p(S|c) \tag{6}$$

with $\log p(S|c)$ is given in Equation 1. Once trained, we can follow Algorithm 1 in [9] to sample synthetic bipartite CTRWs for each class $c \in \mathcal{C}$.

Visits construction via Coarse-grained and Fine-grained random walks merging. After sampling a set $\hat{\mathcal{S}}$ of CTRWs, we need to construct the final temporal graph $\tilde{\mathcal{G}}'$, representing the synthetic collection of HCW-Room visits. For scalability, we employ the independent random walk sampling, which has a downside of generating redundant sets of CTRWs. By redundancy, we mean that the same HCW-Room pair appears in multiple random walks. Due to the redundancy in the sampled CTRWs, simply taking the union of the generated events from samples can lead to a noisy graph with multiple visits of the same HCW to different rooms at the same (or very close) time, which is not realistic. Previous works such as TG-GAN [29] and STGEN [18] circumvent this by using a non-parallel generation process, trading off the efficiency for event consistency. TIGGER [9] instead proposed a merging process that retains top frequent edges proportional to their occurrence in the sampled CTRWs based on the true graph statistics. However, this approach requires maintaining a distribution for each unique timestamp in the generated data, which is problematic for fine-grained temporal resolution as: 1) the number of unique timestamps becomes very large and mostly noisy due to the stochasticity of the model, and 2) the distribution of events at each timestamp is extremely sparse and insufficient to give a reliable estimation. Thus, when aggregating sampled CTRWs, we aim to achieve the following two goals: 1) reduce noise in the generated events, and 2) preserve the continuity of events up to a constant gap δ. To this end, we employ a three-step merging process: coarse-grained merging, fine-grained merging, and a final refinement step.

Step 1: Coarse-grained merging - Obtain $\hat{\mathcal{G}}^{(d)}$. Recall that the number of unique timestamps in our sampled CTRWs is larger than that in the original data. Hence, the number of events taking place in each unique time-stamp is relatively low and therefore it is unlikely that a unique HCW-Room pair appears more than once in each timestamp. Hence, frequency based filtering cannot be done at each timestamp. To address this, we first extract a collection of the most frequent HCW-Room pairs for each day from the sampled CTRWs and filter out the rest, reducing the noise in the generated events. Additionally, note that daily-snapshot graphs are also standard in evaluating graph-based properties [29,18,9].

Formally, we first obtain a distribution of the occurrence of HCW-Room pairs for each day of the synthetic data:

$$\hat{p}^d(\{h_i, r_i\}) = \frac{\alpha(h_i, r_i, d)}{\sum_{\{h_j, r_j\} \in \hat{P}^d} \alpha(h_j, r_j, d)}$$

where $\alpha(h_i, r_i, d)$ is the number of times the edge (h_i, r_i) appears in day d and $\hat{P}^d = \{(h, r) | (h, r, t) \in \hat{S}_d, t \in d\}$. Then, we keep sampling from this distribution until we get the required number of pairs for that day, which is extracted from the training data of the corresponding day.

Step 2: Fine-grained merging - Obtain $\tilde{\mathcal{G}}^{(d)}$. Coarse-grained merging is effective in preserving the sparsity of the graph. However, it cannot maintain the continuity of events. Specifically, coarse-grained merging corresponds to the

case where the minimum time gap δ between consecutive visits of an HCW to a room is set to one day (86,400 seconds), which is not realistic. To address this issue, we perform fine-grained merging to merge the visits that are within δ of each other.

We employ two strategies to ensure this: 1) Snapshot-based merging strategy, where we divide the examined period into grids of length δ and fuse edges within the same grid cell; 2) Adaptive merging strategy, where starting from the first visit, looping in sorted order of start time, we fuse the visits that are within δ period of each other. In both cases, we maintain the average start time $\bar{t}$ and the number of events that were merged together, denoted by $\beta(h_i, r_i, \bar{t})$, for each fused edge. Note that the tuples $(h_i, r_i, \bar{t})$ are unique and, at this point, the edges are not necessarily δ apart (due to averaging of the start time) and we need a refining step to ensure this.

Step 3: Refining $\tilde{\mathcal{G}}^{(d)}$.

Subsample daily visits: The fine-grained merging step may retain more than one visit for each HCW-Room pair each day, and we denote this number as $\hat{\alpha}(h_i, r_i, d)$ (note that $\hat{\alpha}(h_i, r_i, d) \leq \alpha(h_i, r_i, d)$). However, it usually results in the total number of visits being significantly higher than the actual number in the training data. Thus, to preserve the sparsity of the original data, we subsample the set of daily visits obtained from Step 2 proportionally to $\hat{\alpha}(h_i, r_i, d)$. The number of visits to be sampled for each pair $\tilde{n}^d(\{h_i, r_i\})$ is defined as follows:

$$\tilde{n}^d(\{h_i, r_i\}) = \frac{\hat{\alpha}(h_i, r_i, d)}{\sum_{(h_j, r_j) \in \tilde{P}_d} \hat{\alpha}(h_j, r_j, d)} \times n_v^d$$

where $\tilde{P}$ is the set of $(\{h_i, r_i\})$ pair that remain after Step 2 and n_v^d is the number of required visits for each day, which is extracted from the training data. Among all possible visits $(h_i, r_i, \bar{t})$ for each (h_i, r_i) pair, we sample $\tilde{n}^d(\{h_i, r_i\})$ visits proportional to $\beta(h_i, r_i, t)$ with the following distribution:

$$\tilde{p}_{h_i, r_i}(\bar{t}) = \frac{\beta(h_i, r_i, \bar{t})}{\sum_{t \in \tau_{h_i, r_i}} \beta(h_j, r_j, t)}$$

with $\tau_{h_i, r_i} = \{\bar{t} | (h, r, \bar{t}) \in \tilde{E}_d, h = h_i, r = r_i\}$.

Resolve conflicts for each HCW: An HCW usually has many visits in a single day, and those visits must be at least δ time apart. As mentioned above, this constraint may not be satisfied after the merging steps. We define a conflict as a pair of consecutive visits that violate this temporal separation requirement. To resolve such conflicts, we retain the visit associated with the higher fused-edge count, $\beta(h_i, r_i, \bar{t})$, and discard the other. Specifically, for each HCW, we iterate through their visits in chronological order and accumulate only the valid ones. The first visit is always included. For each subsequent visit, we check whether it conflicts with the last accepted visit: if no conflict occurs, it is added to the valid set; otherwise, the visit with the higher fused-edge count is retained.

3.2 Stage 2: Duration sampling

Given a set of visits for each HCW, our goal is to sample realistic durations for each visit. Examining the visit duration in our data, we noticed that the distribution of visit durations is strongly right-skewed, with multiple peaks. A straightforward baseline is to uniformly sample durations between a lower bound (the minimum observed duration) and an upper bound (the gap to the next visit). While this approach does not capture the right-skewed nature of real visit durations, it serves as a reasonable baseline, especially since our evaluation focuses on truncated snapshot graphs that depend more on aggregate duration distributions and are less sensitive to individual visit details. Below, we describe our approach for estimating and sampling from the duration distribution, which aims to better preserve the duration patterns observed in the training data.

Estimating duration distribution The most natural way of sampling visit duration is to sample from the set of observed durations based on their frequencies. However, this simple approach ignores the continuous nature of the data and may inadvertently reveal individual care patterns, thus raising privacy concerns. Additionally, some HCWs have very few visits during the examined period. Repeatedly sampling observed duration for these HCWs introduces bias towards the observed values. To address this, we aggregate data from multiple HCWs and estimate a duration distribution for each group.

However, since we are generating data conditioned on unit type, clustering HCWs within each unit separately can lead to inconsistencies. For example, we may encounter cases where an HCW absent from the training data for a unit appears in the generated samples for that unit, making it unclear which estimated distribution to use for sampling durations. To address this, we cluster HCWs across all units based on their embeddings, and then sample durations conditioned on these global clusters. This ensures consistency and applicability of the duration distributions to all generated visits. Below, we outline our step-by-step process for learning the probability distribution of visit durations.

Transform duration data into log-space: As noted, our duration distribution is right-skewed, and a straightforward way to model the duration data X is to use a mixture of log-normal distributions. Furthermore, we know that $Y = \ln(X)$ follows a mixture of Normal distributions, which is easier to model and well-supported by many off-the-shelf libraries. So the first step is to transform the duration data X into log-space $Y = \ln(X)$.

Fit a Gaussian Mixture Model to the Log-Transformed Data: Once we transform the data into the log-space, we can fit a Gaussian Mixture Model (GMM) to capture the complex mixing patterns in the data. A GMM assumes that Y is generated from K components, each with mean μ_k, standard deviation σ_k, and mixing probability π_k, where $\sum_{k=1}^{K} \pi_k = 1$. Since the optimal number of components K is unknown and the data exhibits a diverse range of mixing patterns, we employ a Dirichlet Process Mixture of Normals, which allows for automatic determination of K. Specifically, we use the `BayesianGaussianMixture` class from

`sklearn` package [23] to fit a GMM with Dirichlet Process prior to obtain the parameters π_k, μ_k, and σ_k for each remaining active component $k = 1, 2, \ldots, K$.

Sampling from the Mixture of Normal Distributions When sampling the duration for each visit, we know both the start time of the current visit and the next visit for the HCW. Therefore, the sampled duration must be bounded above by the gap to the next visit, and below by the minimum duration observed in the data to capture realistic visit patterns.

Let the bounds for a sequence of visits be $[a_1, b_1], [a_2, b_2], \ldots, [a_m, b_m]$, where a_i is the minimum allowed duration and b_i is the gap to the next visit (minus a slack, if desired). Since we model durations in log-space, $X = \exp(Y)$, a sample $X \in [a_i, b_i]$ corresponds to $Y \in [c_i, d_i]$ with $c_i = \ln(a_i)$ and $d_i = \ln(b_i)$. Note that the mixture probability density function (pdf) for variable Y is as follows:

$$f_Y(y) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}\left(y \mid \mu_k, \sigma_k^2\right),$$

Thus, the pdf for Y truncated to $[c_i, d_i]$ is given by:

$$f_{Y|\text{trunc}}(y) = \frac{f_Y(y)}{P(c_i \leq Y \leq d_i)} \text{ for } y \in [c_i, d_i],$$

where the normalization constant is:

$$P(c_i \leq Y \leq d_i) = \int_{c_i}^{d_i} f_Y(y)\, dy = \sum_{k=1}^{K} \pi_k \cdot P(c_i \leq Y \leq d_i | \text{component } k).$$

and for each component k, we have:

$$P_k = P(c_i \leq Y \leq d_i | \text{component } k) = \Phi\left(\frac{d_i - \mu_k}{\sigma_k}\right) - \Phi\left(\frac{c_i - \mu_k}{\sigma_k}\right)$$

where Φ is the standard normal CDF. The adjusted mixing probability for component k within $[c_i, d_i]$ is then:

$$\pi'_k = \frac{\pi_k P_k}{\sum_{j=1}^{K} \pi_j P_j}$$

Now, we can generate a sample by the following steps:

- Choose a component k with probability π'_k.
- Sample Y from the normal distribution $N(\mu_k, \sigma_k^2)$ truncated to $[c_i, d_i]$.
- Transform back to $X = \exp(Y)$.

This approach ensures that sampled durations are both realistic and consistent with the temporal constraints of the generated visits.

Generating duration for each visit Applying the above sampling technique, we generate duration information for each synthetic visit. For each HCW, we first compute the gap between consecutive visits to serve as the upper bound for the sampled duration. To ensure realistic transitions between rooms, we subtract a *slack gap* (e.g., 30 seconds) from this upper bound, preventing the duration from occupying the entire interval. The lower bound is set as the minimum observed duration from the training data. We then sample durations from the truncated mixture distribution within these bounds. The final output is a set of temporal edges (h, r, t, d), forming the synthetic temporal graph $\mathcal{G}'$.

4 Experiments

In this section, we present the performance of our proposed approach in capturing the daily-snapshot graph properties, disease spread characteristics, and HCW care patterns. All of our experiments were conducted on an AMD EPYC 7763 machine with 2TB memory and 8 NVIDIA A30 GPUs each with 24GB memory. Our code is available at `https://github.com/hieuvt29/TempoBiGen`.

Our experiments are designed to answer the following questions: 1) Does the generated graph capture the statistical properties of the daily snapshots of the original data? 2) Is the disease spread the same in the generated graph compared to the original network? 3) Does our model generate realistic mobility logs that capture HCW shift patterns?

Datasets: Here we use a collection of proprietary datasets obtained via a data use agreement. The data consists of more than 44 million HCP-room visits collected from 25 different healthcare facilities in the US, ranging in size from small rural facilities to large-scale tertiary-care facilities. We use a subsampled two weeks of data for the experiments below with 1939 HCWs (33 job types), 263 rooms, and a total of 118, 209 visits across 10 unit types.

Testing models: In the following experiments, we set $\delta = 120$ (2 minutes) - heuristically chosen based on the training data. We use the suffix "_CG" to denote methods that only use the coarse-grained merging, where edges are merged for the whole day. Note that this approach cannot retain the visit start time within a day. The suffix "_FG" denotes the methods that only use fine-grained merging, where edges are merged only based on δ. The suffix "_unif" refers to the uniform visit duration sampling method. The reported results are averaged over all unit types and averaged over 3 generated graphs for each method, with standard deviation in parentheses. For a concise plot, we use "TBG" to denote our proposed TempoBiGen method. The best values are in bold.

4.1 Snapshot graph properties

In this experiment, we evaluate daily-graph properties. Similar to [9], we report the median absolute error for 9 different graph properties. The results are summarized in Table 1.

Table 1. Performance based on mean absolute error (standard deviation in parentheses) between sampled and true daily-snapshot graphs.

Method	% Edge Overlap	Mean Degree	Wedge Count	PLE	Edge Entropy	LCC	NC	Mean BC	Mean CC	Has Duration?
Actual_Median	N/A	5.6269	2502.15	1.7960	0.9274	86.35	1.2	0.0246	0.3411	No
TIGGER_CG	76.1927 (0.1889)	0.3878 (0.0023)	75.50 (2.2287)	0.0740 (0.0037)	**0.0068** (0.0001)	6.75 (0.13)	**0.1** (0.0)	0.0019 (0.0001)	0.0162 (0.0007)	No
BiTIGGER_CG	75.7778 (0.2401)	**0.3690** (0.0020)	**63.30** (1.25)	0.0755 (0.0032)	0.0069 (0.0004)	**6.60** (0.09)	**0.1** (0.0)	**0.0014** (0.0001)	**0.0157** (0.0001)	No
BiTIGGER_FG	**23.4438** (0.2904)	2.5149 (0.0184)	2024.02 (6.64)	0.4916 (0.0303)	0.0292 (0.0004)	44.10 (0.23)	1.0 (0.1)	0.0300 (0.0018)	0.1201 (0.0024)	Yes
BiTIGGER_FG_unif	**23.4438** (0.2904)	2.5149 (0.0184)	2024.02 (6.64)	0.4916 (0.0303)	0.0292 (0.0004)	44.10 (0.23)	1.0 (0.1)	0.0300 (0.0018)	0.1201 (0.0024)	Yes
TBG_adapt	64.2953 (0.3571)	0.6650 (0.0096)	692.40 (13.31)	0.0761 (0.0019)	0.0087 (0.0001)	9.60 (0.09)	**0.1** (0.0)	0.0034 (0.0003)	0.0189 (0.0002)	Yes
TBG_snap	68.2677 (0.0652)	0.4688 (0.0092)	464.70 (4.66)	0.0707 (0.0003)	0.0080 (0.0004)	7.65 (0.09)	**0.1** (0.0)	0.0026 (0.0001)	0.0159 (0.0004)	Yes
TBG_unif_adapt	64.4092 (0.2257)	0.6859 (0.0043)	667.25 (11.96)	0.0831 (0.0025)	0.0091 (0.0003)	9.00 (0.22)	**0.1** (0.0)	0.0035 (0.0001)	0.0191 (0.0005)	Yes
TBG_unif_snap	68.3447 (0.0995)	0.4610 (0.0072)	484.45 (5.21)	**0.0657** (0.0029)	0.0079 (0.0003)	7.90 (0.29)	**0.1** (0.0)	0.0026 (0.0001)	0.0158 (0.0001)	Yes

From the table, we observe that methods using only coarse-grained merging perform better across most metrics. This is expected, as evaluating based on daily graphs means that merging at the day level retains the most frequently occurring HCW-Room pairs while filtering out noisy signals. However, these methods fail to preserve the precise start times of events, making them unsuitable for our downstream tasks. On the other hand, relying solely on fine-grained merging introduces excessive noise, leading to unrealistic graphs where an HCW appears to have an unreasonably high number of visits per day. Our approach, which combines both coarse-grained and fine-grained merging, strikes a balance between preserving daily graph properties and maintaining event-level details, resulting in a middle-ground performance. Notice that, due to the low number of active HCWs each day for each unit type, we are expected to see a high percentage of overlapping edges for daily-snapshots.

4.2 Disease Simulation

The core motivation of this work is to enable accurate disease modeling, which requires a generative approach that preserves disease spread patterns in the original contact network. To evaluate how well our method preserves these patterns, we start by constructing a weighted temporal graph of HCWs based on their visits. Visits are divided into 12-hour snapshot graphs, aligning with typical hospital shifts. Edges are formed between HCWs who visit the same room within a snapshot. Edge-weight for edge (h_1, h_2) is computed as $w(h_1, h_2) =$

$\sigma(d_1 + d_2 + 10 \times d_{12})$, with $\sigma(x) = 1/(1+e^{-x})$, where d_1 and d_2 are the duration visit for h_1 and h_2 respectively, d_{12} is the overlap duration. We use a factor of 10 to account for the increased likelihood of disease transmission when HCWs share a room simultaneously.

For disease simulation, we use the well-known SIR compartmental model with an edge-weight-adjusted transmission rate [14]. We run 50 simulations for each unit type with base transmission rate $\beta = 0.35$, recovery rate $\gamma = 0.2$, and number of initially infected HCWs in the first snapshot of 10. We then compare attack rates between real data and different approaches using box plots.

Figure 2 highlights the difference between considering and ignoring duration information. It clearly shows that, given the same assumption of base transmission rate, ignoring the duration information (unweighted) consistently leads to higher attack rates compared to the weighted version. This suggests that the duration information is crucial for capturing more realistic transmission scenarios.

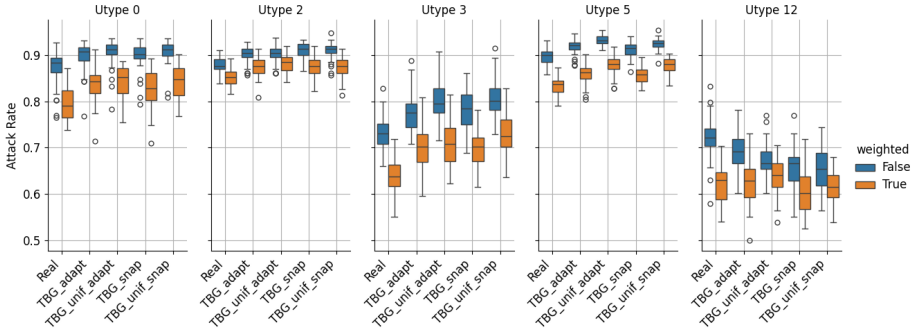

Fig. 2. Comparison with and without edge weights.

We further compare the attack rates across all unit types with weighted data. The absolute difference between attack rates of each method against that of true data averaged over all unit types is shown in legend Fig 3. The results show that TempoBiGen with the adaptive strategy performs the best, achieving the lowest average difference, followed by the snapshot-based approach, and both clearly outperform the uniform sampling methods. Additionally, having different attack rates for different unit types demonstrates that our generative model effectively captures the unique characteristics of each unit type.

4.3 Shift Efficiency and Uniformity

To assess how well the generated data follows care patterns for each unit type, we compute shift efficiency and uniformity across different methods. *Shift efficiency* is defined as the ratio of total shift visit duration to the full shift duration (a value between 0 and 1), while *shift uniformity* is the number of unique rooms

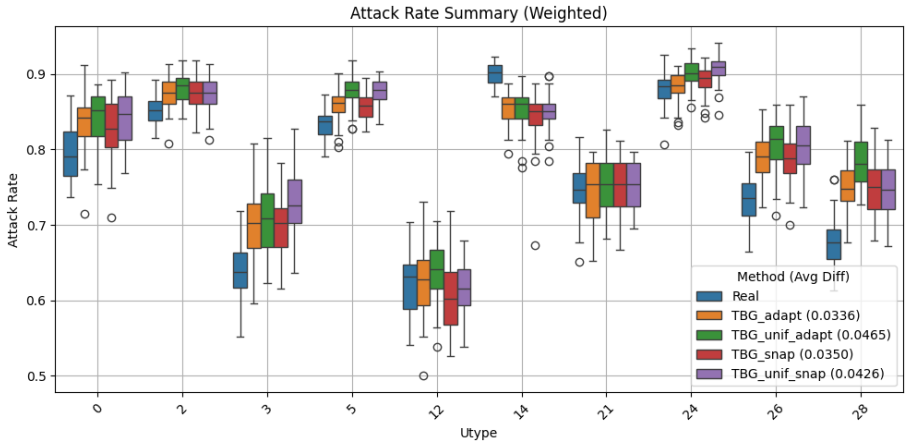

Fig. 3. Attack rates for different methods.

serviced per shift divided by the total number of shift visits (also between 0 and 1, small numbers mean more attention on fewer rooms). Table 2 shows the average absolute differences between the two values of our methods and baselines against those of the original data. The results indicate that non-trivial sampling methods significantly improve shift uniformity but are less effective in maintaining shift efficiency.

Table 2. Average Efficiency and Uniformity Differences.

Method	Avg. Efficiency Difference	Avg. Uniformity Difference
TBG_adapt	0.37098	0.29616
TBG_unif_adapt	**0.30222**	0.45497
TBG_snap	0.34981	**0.28193**
TBG_unif_snap	0.32082	0.31203

5 Conclusion

In this paper, we presented a generative model to produce realistic mobility graphs within healthcare facilities. Following the state-of-the-art temporal graph generation literature, we formulated a one-shot learning problem and extended existing methods to address conditional bipartite graph generation problem with temporal edge duration sampling. Specifically, we proposed a two-stage approach. First, we extended TIGGER to handle bipartite graphs and combine coarse-grained and fine-grained merging into our sampling procedure to

construct a bipartite temporal graph. Second, given the generated HCW-Room visits, we added a post-processing module to generate realistic visit durations. Our results demonstrate that the proposed approach effectively captures visit duration patterns, preserves the statistical properties of daily-snapshot graphs, and enables realistic disease simulations. Extending our work for the inductive setting (where new nodes arrive over time) and formally incorporating privacy (for example, via differentiable privacy) are promising future directions.

6 Acknowledgments

The authors acknowledge feedback from members of the Computational Epidemiology research group at the University of Iowa and the CDC MInD-Healthcare group. This work was supported by the CDC under cooperative agreement U01-CK000594. Its contents are solely the responsibility of the authors and do not necessarily represent the official views of CDC.

References

1. Abbe, E.: Community detection and stochastic block models: recent developments. Journal of Machine Learning Research **18**(177), 1–86 (2018)
2. Adhikari, B., Lewis, B., Vullikanti, A., Jiménez, J.M., Prakash, B.A.: Fast and near-optimal monitoring for healthcare acquired infection outbreaks. PLoS computational biology **15**(9), e1007284 (2019)
3. Capolongo, S., Gola, M., Brambilla, A., Morganti, A., Mosca, E.I., Barach, P.: Covid-19 and healthcare facilities: a decalogue of design strategies for resilient hospitals. Acta Bio Medica: Atenei Parmensis **91**(9-S), 50 (2020)
4. Casey, D.: Challenges of collecting data in the clinical setting. NT Research **9**(2), 131–141 (2004)
5. Coletti, P., Libin, P., Petrof, O., Willem, L., Abrams, S., Herzog, S.A., Faes, C., Kuylen, E., Wambua, J., Beutels, P., et al.: A data-driven metapopulation model for the belgian covid-19 epidemic: assessing the impact of lockdown and exit strategies. BMC infectious diseases **21**, 1–12 (2021)
6. Fournet, J., Barrat, A.: Contact patterns among high school students. PloS one **9**(9), e107878 (2014)
7. Goldschmidt, U.: An introduction to the theory of point processes (2016), `https://api.semanticscholar.org/CorpusID:63985456`
8. Gostin, L.O., Levit, L.A., Nass, S.J.: Beyond the hipaa privacy rule: enhancing privacy, improving health through research (2009)
9. Gupta, S., Manchanda, S., Bedathur, S., Ranu, S.: Tigger: Scalable generative modelling for temporal interaction graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 6819–6828 (2022)
10. Haque, M., Sartelli, M., McKimm, J., Bakar, M.A.: Health care-associated infections–an overview. Infection and drug resistance pp. 2321–2333 (2018)
11. Hethcote, H.W.: The mathematics of infectious diseases. SIAM review **42**(4), 599–653 (2000)

12. Jang, H., Fu, A., Cui, J., Kamruzzaman, M., Prakash, B.A., Vullikanti, A., Adhikari, B., Pemmaraju, S.V.: Detecting sources of healthcare associated infections. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 4347–4355 (2023)
13. Jang, H., Pai, S., Adhikari, B., Pemmaraju, S.V.: Risk-aware temporal cascade reconstruction to detect asymptomatic cases. Knowledge and Information Systems **64**(12), 3373–3399 (2022)
14. Kamp, C., Moslonka-Lefebvre, M., Alizon, S.: Epidemic spread on weighted networks. PLOS Computational Biology **9**(12), 1–10 (12 2013). https://doi.org/10.1371/journal.pcbi.1003352, https://doi.org/10.1371/journal.pcbi.1003352
15. Keeling, M.J., Eames, K.T.: Networks and epidemic models. Journal of the royal society interface **2**(4), 295–307 (2005)
16. Kiji, M., Hasan, D.H., Segre, A.M., Pemmaraju, S.V., Adhikari, B.: Near-optimal spectral disease mitigation in healthcare facilities. In: 2022 IEEE International Conference on Data Mining (ICDM). pp. 999–1004. IEEE (2022)
17. Lansbury, L.E., Brown, C.S., Nguyen-Van-Tam, J.S.: Influenza in long-term care facilities. Influenza and other respiratory viruses **11**(5), 356–366 (2017)
18. Ling, C., Cao, H., Zhao, L.: Stgen: Deep continuous-time spatiotemporal graph generation. In: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2022, Grenoble, France, September 19–23, 2022, Proceedings, Part III. p. 340–356. Springer-Verlag, Berlin, Heidelberg (2022). https://doi.org/10.1007/978-3-031-26409-2_21, https://doi.org/10.1007/978-3-031-26409-2_21
19. Liu, P., Sariyüce, A.E.: Using motif transitions for temporal graph generation. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 1501–1511 (2023)
20. Medsker, L.R., Jain, L., et al.: Recurrent neural networks. Design and Applications **5**(64-67), 2 (2001)
21. Monsalve, M.N., Pemmaraju, S.V., Thomas, G.W., Herman, T., Segre, A.M., Polgreen, P.M.: Do peer effects improve hand hygiene adherence among healthcare workers? Infection Control & Hospital Epidemiology **35**(10), 1277–1285 (2014)
22. Morrison, L., Zembower, T.R.: Antimicrobial resistance. Gastrointestinal Endoscopy Clinics **30**(4), 619–635 (2020)
23. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. the Journal of machine Learning research **12**, 2825–2830 (2011)
24. Purohit, S., Holder, L.B., Chin, G.: Temporal graph generation based on a distribution of temporal motifs. In: Proceedings of the 14th International Workshop on Mining and Learning with Graphs. vol. 7 (2018)
25. Shchur, O., Biloš, M., Günnemann, S.: Intensity-free learning of temporal point processes. arXiv preprint arXiv:1909.12127 (2019)
26. Vanhems, P., Barrat, A., Cattuto, C., Pinton, J.F., Khanafer, N., Régis, C., Kim, B.a., Comte, B., Voirin, N.: Estimating potential infection transmission routes in hospital wards using wearable proximity sensors. PloS one **8**(9), e73970 (2013)
27. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world'networks. nature **393**(6684), 440–442 (1998)
28. Zeno, G., La Fond, T., Neville, J.: Dymond: Dynamic motif-nodes network generative model. In: Proceedings of the Web Conference 2021. pp. 718–729 (2021)

29. Zhang, L., Zhao, L., Qin, S., Pfoser, D., Ling, C.: Tg-gan: Continuous-time temporal graph deep generative models with time-validity constraints. In: Proceedings of the Web Conference 2021. pp. 2104–2116 (2021)
30. Zhou, D., Zheng, L., Han, J., He, J.: A data-driven graph generative model for temporal interaction networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 401–411 (2020)