Appendix

This is the supplementary material for the paper "Disentangled Dynamic Graph Deep Generation".

1 Model Architecture

Architecture of the generator: First, for edge generation, we use a dense layer with the inputs including z, z'' and f. Then, we utilize a graph deconvolutional network [1] to generate a matrix of edge probabilities on each time snapshot. For node generation, we use another dense layer which takes the input that is the concatenation of z', z'', and f, and output each node features on each time snapshot.

Architecture of full inference model. First, we utilize a Graph Convolutional Network [2] to produce a representation vector a_t for each snapshot graph: $a_t = GCN(E_t)$. Raw node attributes are also converted to a feature vector through a dense layer $b_t = Dense(F_t)$ first. To encode f, $q_\phi(f|E, F)$ is modeled with a neural network function $h_\phi(E, F) \mapsto f$. In details, we concatenate latent representation of topology a_t and attribute representation b_t and feed into a Bi-directional LSTM $BiLSTM$ as inputs. Then, both the last forward output m_T and the first backward output $\bar{m}_T$ are concatenated to a single vector. This vector is passed to another dense layer to get f. The function $h_\phi(E, F) \mapsto f$ is decomposed as follow:

$$a_t = GCN(E_t), b_t = Dense(F_t) \tag{1.1}$$
$$m_t, \bar{m}_t = BiLSTM(a_t||b_t),\ f = Dense(m_T||\bar{m}_T)$$

where $||$ is the concatenation operation. With similar way, we leverage another three Bi-directional LSTMs to encode z, z' and z''.

Next, to encode z, we utilize another Bi-directional LSTM which takes both a_t and encoded f as inputs, noted as $g_\phi(E, f) \mapsto z$. Instead of only using the final outputs, each step's forward output o_t and backward output $\bar{o}_t$ are passed to a dense layer to encode z_t on each time step. $g_\phi(E, f) \mapsto z$ composes of follows:

$$o_t, \bar{o}_t = BiLSTM(a_t||f)$$

$$z_t = Dense(o_t||\bar{o}_t)$$

where f is a shared copy for each time step.

Similar process is used for variable z'_t, noted as $g'_\phi(F, f) \mapsto z'$ except that we use b_t with another Bi-directional LSTM. $g'_\phi(F, f) \mapsto z'$ can be decomposed as follows:

$$o'_t, \bar{o}'_t = BiLSTM(b_t||f)$$

$$z'_t = Dense(o'_t||\bar{o}'_t)$$

where f is a shared copy for each time step.

Similar process is used for variable z''_t too, except that we input both a_t and b_t for a new bi-directional LSTM. Its function is noted as $g''_\phi(E, F, f) \mapsto z''$ with its details as follows:

$$o''_t, \bar{o}''_t = BiLSTM(a_t||b_t||f)$$

$$z''_t = Dense(o''_t||\bar{o}''_t)$$

where f is a shared copy for each time step.

Architecture of factorized inference model. The encoding structure for f is the same as the above. The only difference focuses on the encoding of z_t, z'_t, and z''_t. Different from the above full inference model, we simply use three multi-layer perception models to generate them, using the inputs E_t, F_t, and jointly E_t, F_t, respectively. No bi-directional LSTM is needed here.

2 Experimental Details

The source code and data for reproducing the results are available at here[1]. The binary cross-entropy for $log_p(E|z)$ and set the learning rate of Adam optimiser as 10^{-3}. We train 100 iterations on all datasets with a batch size of 64. All models, including the proposed D2G2 and baseline methods, employ leaky ReLU (lReLU) non-linearity as the activation function. Table 1 further specifies our model architecture. For competing methods of GraphRNN, GraphVAE, and NetGAN which were initially designed for static not dynamic graphs, we treat each time snapshot as a static graph and apply these three methods to learn separate models for each time snapshots. DSBM is a temporal graph generation model, so we directly utilize its raw model. Note that due to the excessive memory RAM requirement of DSBM, only small amount of all original graph samples within the limits of our machine's memory is used.

[1] https://github.com/vanbanTruong/D2G2

Table 1: The architecture of D2G2

Edge encoder	Node encoder	Edge decoder	Node decoder
Graph-conv ReLU	2×2 conv ReLU stride 1	FC	FC
Graph-conv ReLU	2×2 conv ReLU stride 1	FC	FC
FC	FC	Graph-deconv ReLU	deconv ReLU stride 1
FC	FC	Graph-deconv ReLU	deconv ReLU stride 1

3 Detailed Dataset Simulation Process

Protein Dataset. Protein is by nature graphically represented data where amino acids are nodes and the connectivity between them are edges. We simulate the dynamic folding process of a protein peptide with a sequence AGAAAAGA. For graph learning, this can be considered as a graph of 8 nodes with node attributes (x, y, z) corresponding to 3D coordination of the C_α atom of each amino acid. The protein contact map, which records pairwise distance between nodes, is generated based on fully atomistic molecular dynamics simulations, producing 300 temporal graphs with a sequence length of 100, i.e., each temporal graph consists of 100 graphs and formed together to simulate the morphing transformation of protein structure.

User Authentication Dataset. User authentication reflects "login" behavior from one IP address to another [3]. Each IP address is represented as a node on this authentication graph to better reflect the network activities. Each 5-hour period is used as a temporal graph, which is split into 10 snapshot graphs each with half-hour time bin. Different user authentication behavior (a connection from one IP to another) in a half-an-hour period is contributed as an out-flow count to start node and an in-flow count to end node. There is an edge between two nodes if the corresponding connections between these two IP addresses are frequent. The in-flow and out-flow counts are used as features for each node. There are totally 9 different distinct IP addresses.

Metro Dataset. Each metro station is a node on metro graph. Each day is treated as a temporal graph with each hour as a time snapshot. The directed links from one metro to another reflects the flow between them in each graph snapshot. There is a directed connection only when the flow is significant enough. The total out-flows and in-flows are used as features of nodes in this metro graphs.

Synthetic Datasets. Scale-free graphs [4, 5] are utilized to generate synthetic dataset with increasing complexity. Scale-free graphs are static graphs with a power-law degree distribution. Its generation process involves three actions, including *Action a*: creating a new edge between a new in-node and an existing out-node, *Action b*: creating a new edge between two existing nodes, and *Action c*: creating a new edge between a new out-node and an existing in-node. Choices of different actions follows a multinomial distribution with probabilities $\langle \alpha, \beta, \gamma \rangle, \forall \alpha + \beta + \gamma = 1$. Here we introduce how we utilize scale-free graphs to generate dynamic graphs.

The generation process of our dynamic scale-free graph consists of three steps: Step 1: initialize a start time 0 and three connected start node $1, 2, 3$ with edges $(1, 2), (1, 3), (2, 3)$; Step 2: do a loop where in each iteration i $(i = 1, \cdots, n_{scale})$, we do the following: sample a continuous interval time t_i from a uniform distribution $Uniform(0, 1)$. Comparing t_i with $\langle \alpha, \beta, \gamma \rangle$, if $t_i \in [0, \alpha]$, execute Action a, if $t_i \in (\alpha, \alpha + \beta]$, execute Action b, if $t_i \in (\alpha+\beta, 1]$, execute Action c. We designate this $t_1 + \cdots + t_i$ as the timestamp of the execution of this action. We just iterate Step 2 until total elapsed time exceed t_{end}. The last new edge gets timestamp of t_{end}. Now, we have obtained a continuous time dynamic graphs. Then we partition this continuous-time graph into a set of snapshot graphs with a fixed time window.

4 The Necessity of Learning Time-invariant Variable *f* as well as Time-dependent Variables *z*, *z'* and *z"*, respectively

First of all, it is dynamic networks' nature such that part of it is time-evolving while the other part could be persistent. For example, a protein can be considered as a graph of amino acids connected by bonds. During the protein folding process, many node/edge attributes do not change (e.g., the type of amino acids) while some other attributes change, the angle and strength of the bonds. Therefore, differentiating those changing patterns from those without changing is very natural and important. And this importance applies to many other dynamic networks such as social networks and brain networks (e.g., dynamic functional connectivity). What's more, such representation learning is **data efficient**. By factoring out a separate variable that encodes time-invariant factors, the dynamic variables control-

ling time-variant factors can have smaller dimensions. Therefore, z, z' and z" encode the morphing transformation of topology and node attributes from time to time, respectively while f compresses everything else. Last, such representation learning allows for **controlled generation**. We can generate random dynamics by random sampling one of f, z, z' and z" while keeping the other three fixed, and controlling the respective factor of attribute and topology statics, attribute dynamics, topology dynamics as well attribute and topology dynamics. Such controlled generation is also practically meaningful, such as analyzing how the proteins folds as the time flies. The controlled generation also indicates the learned representation by our model is indeed disentangled and can control the corresponding patterns related to them, which is confirmed by our experiments.

5 More Temporal Graph Visualization

Comparisons of real and generated graphs from D2G2 and comparison methods on Authentication dataset. We have observed numerous interesting generated graphs during our experiments and illustrate some that can further represent all of them, in addition to those included in our paper main text. For example, Figure 1 illustrates an additional graphs generated by all the methods, including the proposed D2G2 and comparison methods. In the visualization, nodes of the temporal graphs across all temporal snapshots, while arcs are edges reflecting connectivities among the nodes of the temporal snapshots. As one can see, the temporal graphs generated by our D2G2 looks closer to the ground truth when comparing with the other methods, and it indicates D2G2's capability in effectively learning the underlying structure of temporal graphs. Note that among all baselines, only GraphVAE can generate node attributes, therefore other baselines' node color is in white.

Generated graphs controlled by respective disentangled factors z, z', z'' and f. Here we provide more illustrations in addition to those in Figure 3 in our paper's main text, in order to further verify our disentangled model design. Specifically, as shown in Figures 2(a), when varying f with z, z' and z'' being fixed, both the topology and node attributes (represented by the colors of nodes) change evenly for all time snapshots, suggesting the model is capable of finding time invariant static factor. In addition, when having fixed f, z' and z'' with z varies, the generated snapshots in in Figures 2(b) reveal the same node attributes characteristics (as reflected by the stable colors) but different topology patterns. The opposite pattern can be observed in Figures 2(c) when we instead vary z' but keep f, z and z'' fixed and can see that the colors (and the corresponding node attributes values they reflect) vary instead of the topology. We can also see z'' changes both node and edges across different graphs in Figures 2(d).

6 Additional Scalability Analysis

Figure 3 also shows the scalability with respect to the sizes of sequence length (i.e., the number of snapshots) of the proposed D2G2 against other comparison methods. We report the average logarithm of the running time per epoch and the running time unit is second. We found that our model can scale well with roughly a linear time complexity, and enjoys the merit of handling graphs with increasing complexity.

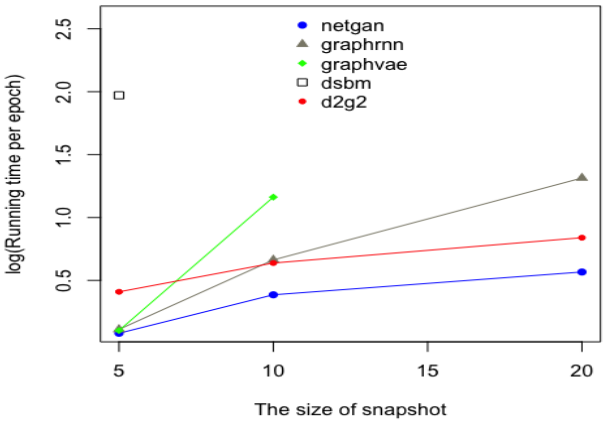


Figure 3: Scalability comparison of all the methods.

References

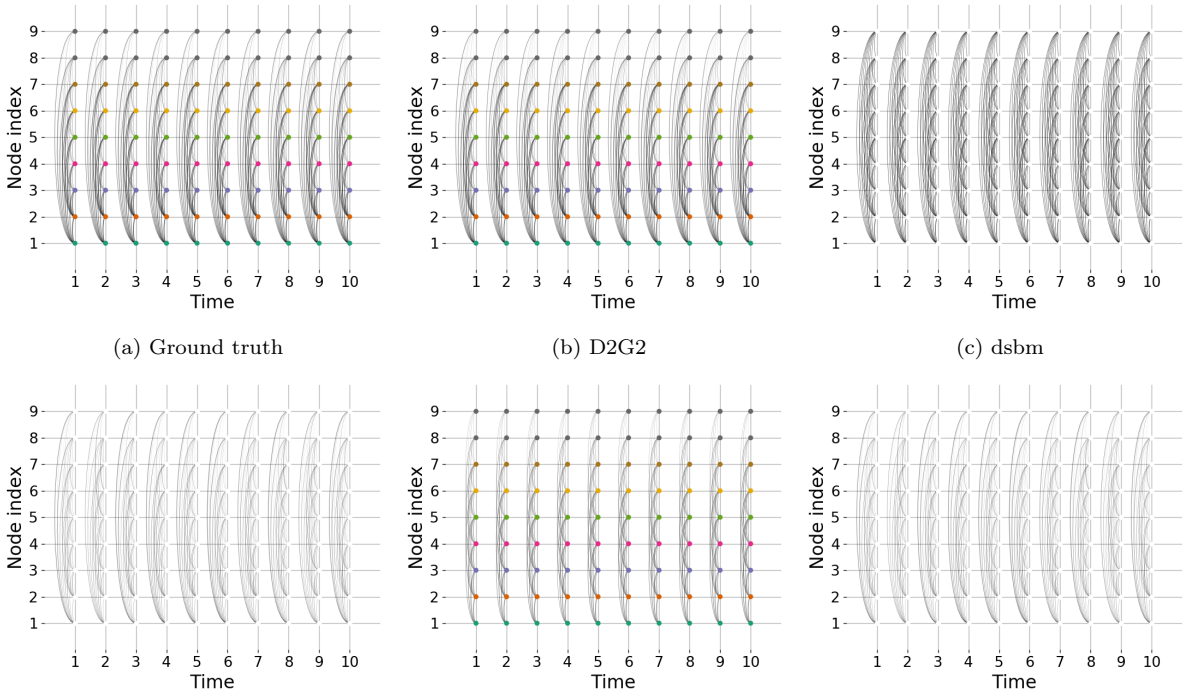
[1] X. Guo, L. Wu, and L. Zhao, "Deep graph translation," *arXiv preprint arXiv:1805.09980*, 2018.

[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[3] A. D. Kent, "Cyber security data sources for dynamic network research," in *Dynamic Networks and Cyber-Security*. World Scientific, 2016, pp. 37–65.

[4] A.-L. Barabási, R. Albert, and H. Jeong, "Mean-field theory for scale-free random networks," *Physica A: Statistical Mechanics and its Applications*, vol. 272, no. 1-2, pp. 173–187, 1999.

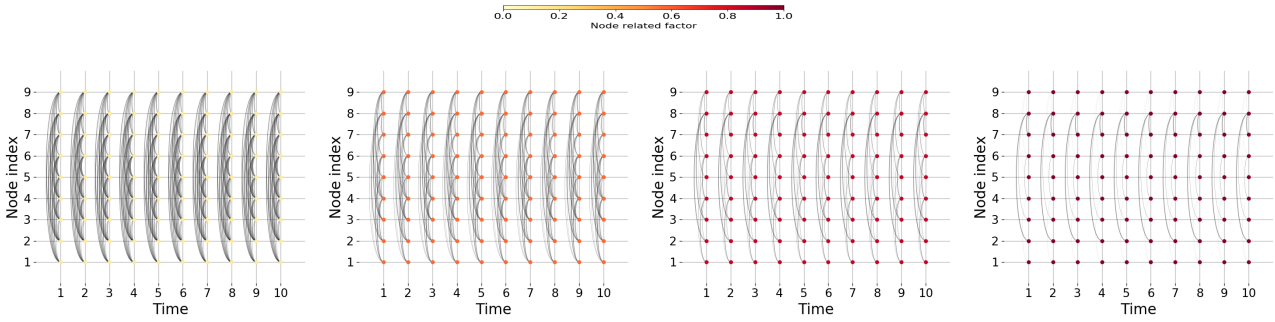
[5] B. Bollobás, C. Borgs, J. Chayes, and O. Riordan, "Directed scale-free graphs," in *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2003, pp. 132–139.



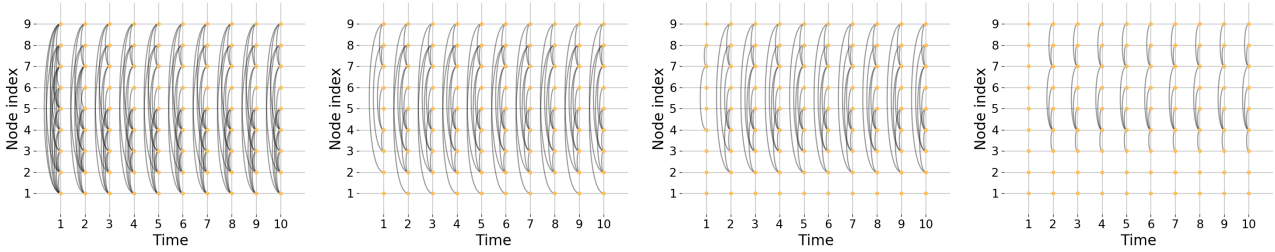
(a) Ground truth (b) D2G2 (c) dsbm

(d) graphrnn (e) graphvae (f) netgan

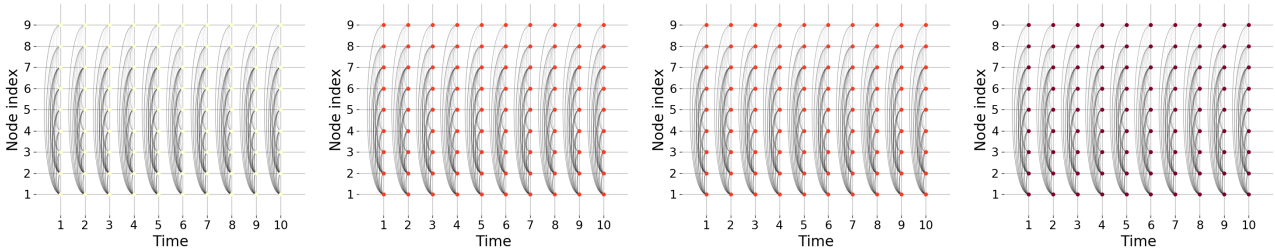
Figure 1: Visualization of graphs from Authentication dataset and generated by all methods. The color of nodes represents the normalized value of a node attribute.



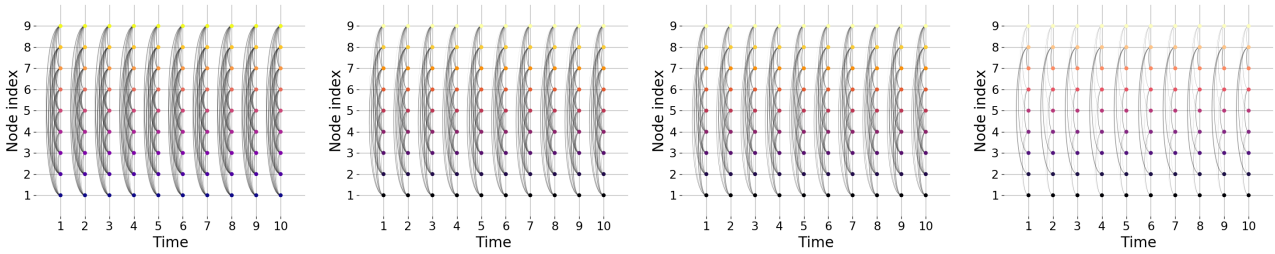
(a) Generated graphs when varying f with z, z' and z'' fixed.



(b) Generated graphs when varying z with f, z' and z'' fixed.



(c) Generated graphs when varying z' with f, z and z'' fixed.



(d) Generated graphs when varying z'' with f, z and z' fixed.

Figure 2: Authentication dataset: visualization of graphs generation controlled by respective disentangled factors. The color of nodes represents the normalized value of a node attribute.