# Structure-informed Graph Auto-encoder for Relational Inference and Simulation

**Yaguang Li** [1] **Chuizheng Meng** [1] **Cyrus Shahabi** [1] **Yan Liu** [1]

## Abstract

A variety of real-world applications require the modeling and the simulation of dynamical systems, e.g., physics, transportation and climate. With the increase of complexity, it becomes challenging to infer the true interactions solely based on observational data. In this work, we propose the Structure-informed Graph-Autoencoder for Relational inference and simulation (SUGAR) which incorporates various structural prior knowledge. SUGAR takes the form of a variational auto-encoder whose latent variables represent the underlying interactions among objects. It represents various structural prior knowledge as differentiable constraints on the interaction graph, and optimizes them using gradient-based methods. Experimental results on both synthetic and real-world datasets show our approach clearly outperforms other state-of-the-art methods in terms of both interaction recovery and simulation.

## 1. Introduction

Modeling and simulation of dynamical systems have various applications in domains including physics, transportation, climate, and social networks. These dynamical systems can be represented as groups of interacting objects. It is challenging to model dynamics in these systems, as usually we only have access to the movements of individual object, rather than the underlying interactions. Recently, many work have been done on learning the dynamic model of interacting systems using *implicit* interaction model (Sukhbaatar et al., 2016; Guttenberg et al., 2016; Scarselli et al., 2009; van Steenkiste et al., 2018), where interactions are modeled implicitly by message passing or through the attention mechanism. In Kipf et al. (2018), the authors propose the Neural Relational Inference model (NRI), an approach that infers *explicit* interactions while simultaneously learns the

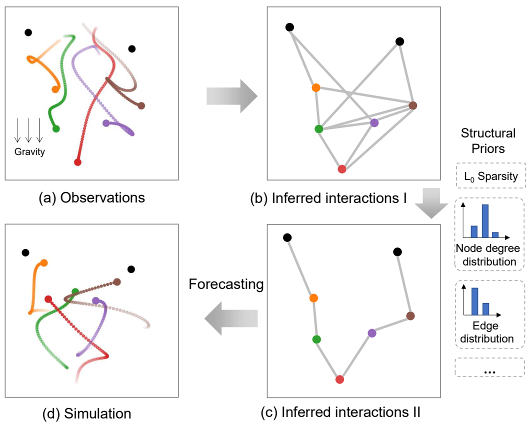


*Figure 1.* (a) Movement of a chain of connected objects under the gravity field. (b) (c) Incorporating structural prior knowledge helps find the ground truth interactions, and (d) improves simulation performance.

dynamics purely from observational data. However, with the increase of complexity, it becomes challenging to recover the true interactions solely based on observed data, and thus it is desirable to incorporate the prior knowledge about the structure of the interactions when available. Figure 1 shows a motivating example, where we observe the movements of a set of objects that are connected with springs in a chain structure. Due to the global gravity and the deeply entangled movements, NRI tends to infer redundant interactions and consequently results in degenerated simulation. To incorporate structural prior knowledge, we propose the Structure-informed Graph-Autoencoder for Relational inference and simulation (SUGAR). SUGAR takes the form of variational auto-encoder, where the latent variables represent the underlying interactions among objects. Both the encoder and the decoder employ a graph network-based architecture, with node, edge, and global features. The model can incorporate various structural priors, e.g., the node degree distribution, the interaction sparsity, and the interaction type distribution. Suppose we know the underlying interaction in Figure 1 has a chain structure, then we can recover the true interactions (Figure 1(c)) and achieve improved simulation (Figure 1(d)).

In particular, we provide the following key contributions: (1) We propose novel approaches to integrate various structural priors for better interaction recovery and simulation. (2)

[1] Department of Computer Science, University of Southern California, United States. Correspondence to: Yaguang Li <yaguang@usc.edu>.

Presented at the ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data Copyright 2019 by the author(s).

We design novel encoder and decoder that explicitly model the global features to capture global interactions and to facilitate communications between not directly connected nodes. (3) We conduct a wide range of experiments on both synthetic and real-world datasets. The results show that SUGAR clearly outperforms state-of-the-art methods in terms of both simulation and interaction recovery.

## 2. Related Work

Our work draws on several lines of previous research. In Battaglia et al. (2016); Guttenberg et al. (2016); Chang et al. (2017); Sanchez-Gonzalez et al. (2018); Scarselli et al. (2009), the author studied the problem of learning the dynamics of a physical system from simulated trajectories and from generated video data (Watters et al., 2017; van Steenkiste et al., 2018) with a graph neural network. In Li et al. (2018), the authors infer a residual graph based on the given structure. A number of recent works based on graph network (Monti et al., 2017; Velickovic et al., 2018; van Steenkiste et al., 2018; Lee et al., 2018b;a) have the ability to focus on a specific neighbor when aggregating information with the attention mechanism. These works either assume a known graph structure or infer interactions implicitly. More related work in the fields of graph generation and link prediction are discussed in the Appendix B.

We aim to infer interactions in an unsupervised manner while simultaneously learns the dynamics from observational data. The most related work is Kipf et al. (2018), where the authors propose to learn the explicit interactions among objects using variational graph auto-encoder. The main differences are that (1) we propose effective and concrete ways to encode various structured prior knowledge into the model and (2) we design an improved encoder and decoder architecture that explicitly models the global features. This helps capture global interactions and to facilitate communications between not directly connected nodes.

## 3. Methodology

**Problem definition** Given a sequence of observations $\mathbf{x} = (\mathrm{x}^{(1)}, \cdots, \mathrm{x}^{(T)}) \in \mathbb{R}^{T\times|V|\times P}$, which consists of the observations from $|V|$ objects over $T$ time steps, we want to simultaneously learn the interactions among objects and predict the future states of these objects. We use customized graph network (Battaglia, 2018) to model the movement of these objects. The *graph* consists of three components, $\mathcal{G} = (u, V, E)$. $u$ is the global variable, $V = \{v_i\}_{i=1:|V|}$ is the set of nodes, and $E = \{(e_k, r_k, s_k)\}_{k=1:|E|}$ is the set of edges, where $e_k$ is the attribute of the $k$th edge, $s_k, r_k$ are the indices of the sender and receiver nodes respectively. We use latent variable $z$ to represent the relations among objects, where $z_k$ represent the distribution of the interaction type of $e_k$. A summary of main notations used in the paper is provided in the Appendix (Table A1).

We formalize SUGAR based on the variational autoencoder (Kingma & Welling, 2013; Kipf & Welling, 2016; Kipf et al., 2018). The model consists of three components, the encoder, the decoder, and the component to incorporate structural prior knowledge. Both the encoder and the decoder are based on customized graph networks. Figure 2 shows the architecture of SUGAR. The encoder takes as input a sequence of observations, $\mathbf{x}$, and estimates the interactions $z$, while the decoder takes as input the estimated interaction graph and learns the system dynamics to predict the future state. The interaction constraint component calculates the regularizations based on various structural prior knowledge.

### 3.1. Encoder

In SUGAR, the encoder is used to infer pairwise interactions among objects based on observations $\mathbf{x}$. It employs a graph network with a fully-connected graph structure, with two round updates as follows:
Initialization: $v_i = \phi_{emb}(\mathrm{x}_i^{(1)}, \mathrm{x}_i^{(2)}, \cdots, \mathrm{x}_i^{(T)}), \quad e_k = 0$

Then each round consists of the following three steps: (1) edge update, which updates the edge based its two connected nodes and the global variable;

$$e_k^{l+1} = \phi_e^l\left(e_k^l, v_{r_k}^l, v_{s_k}^l, u^l\right)$$

(2) node update, which aggregates all the information from incoming edges;

$$v_i^{l+1} = \phi_v^l\left(v_i^l, \sum_{r_k=i} e_k^l, u^l\right)$$

(3) global update, which updates the global features with aggregated node and edge features.

$$u^{l+1} = \phi_u^l\left(\sum_k e_k^{l+1}, \sum_i v_i^{l+1}, u^l\right)$$

where $\phi_\cdot^l()$ denote the updating functions of the encoder in the layer $l$, which is usually a multi-layer perceptron.

**Interaction generation** Based on the updated edge attributes, we infer the corresponding distribution of interactions, and sample to get the interaction graph. We use the Gumbel softmax (Maddison et al., 2017) to approximate the discrete distribution of interactions and use the reparametrization trick to get the gradient from it.

### 3.2. Decoder

The decoder takes as input the observation $\mathbf{x}^{(t)}$ and the inferred interactions, and outputs $\Delta\mathbf{x}^{(t)}$ with two rounds of updates, with following processing steps.

$$e_k^{l+1} = \sum_m z_{k,m}\tilde{\phi}_e^l\left(e_k^l, v_{r_k}^l, v_{s_k}^l, u^l\right)$$

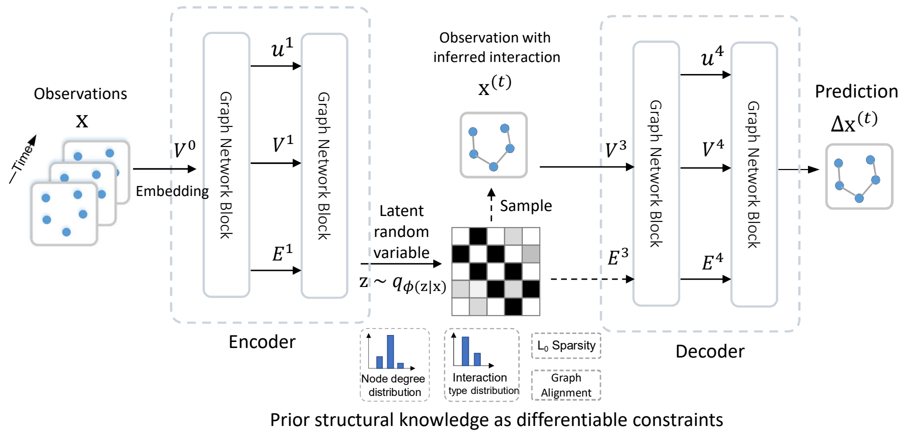


*Figure 2.* Model architecture of SUGAR. The encoder takes as input a sequence of observation, $\mathbf{x}$, and estimates the interactions $z$, while the decoder takes as input the estimated interaction graph and learns the system dynamics to predict the future state. The interaction constraint component calculates the loss function based on specified structural prior knowledge.

where $z_{k,m}$ denotes the probability of $e_k$ being the $m$-th type. Note that each type of interaction has its dedicated update function to enforce the effect of edge type. Then the decoder updates the node and the global information

$$v_i^{l+1} = \tilde{\phi}_v^l \left( v_i^l, \sum_{r_k=i} e_k^l, u^l \right)$$

$$u^{l+1} = \tilde{\phi}_u^l \left( \sum_k e_k^{l+1}, \sum_i v_i^{l+1}, u^l \right).$$

Finally, the decoder predicts the observation in the next time stamp. Here $\tilde{\phi}_\cdot^l(\cdot)$ denote the updating functions for the decoder in layer $l$.

$$\Delta \mathbf{x}_i^{(t)} = \tilde{\phi}_{\mathbf{x}}(v_i^{l+1})$$

$$q(\mathbf{x}_i^{(t+1)} | \mathbf{x}^{(t)}, z) = \mathcal{N}\left(\mathbf{x}_i^{(t)} + \Delta \mathbf{x}_i^{(t)}, \sigma^2 \mathbf{I}\right)$$

## 3.3. Incorporating Structural Prior Knowledge

For a dynamical system, we usually have prior knowledge about properties of its interactions, which can help recover the true interactions. In this work, we are particularly interested in edge/interaction-level and node/object-level structural knowledge. Examples of interaction-level structural knowledge can be the distribution of interactions types, the sparsity of interactions, while examples of object-level structural knowledge are the distribution of node degrees, the maximize/minimum interactions of a node. In this section, we show two examples of encoding the structural knowledge into differentiable graph constraints, more details are available in Appendix A. For the simplicity of illustration, we assume there are only two types of interactions, and the first type means no-edge. Thus, $z_k$ denotes the probability of there exists an interaction between $v_{s_k}$ and $v_{r_k}$, and $\hat{z}_k$ means an instance sampled from that distribution.

### 3.3.1. Interaction-level structural knowledge

With the probabilistic distribution of interactions, we can incorporate various interaction-level structural knowledge.

**Interaction Sparsity** One important example is the sparsity prior, which aims to minimize the number of interactions measured using the $L_0$ distance. $L_0$ distance is not differentiable in general, however, with the probabilistic distribution of interactions, we can minimize the *expected* number of interactions by penalizing the probability of has interactions between nodes.

$$\mathcal{L}_0(z) = \sum_k \mathbb{E}_{e' \sim z}[\mathbb{I}(e'_{k,0} \neq 1] = \sum_k z_{k,0}$$

This idea can be further generalized to "prior graph alignment", which aims to minimize the number of interactions that are different from a specified graph.

### 3.3.2. Object-level structural knowledge

We can encode object-level structural knowledge by first summarizing object-level distributions of interactions from $z$, and then minimizing the differentiable distance metric, e.g., K-L divergence, between it and the prior distribution.

**Node degree distribution** One important example of object-level structural knowledge is the distribution of the node degrees. Larger node degree means more densely related objects. Suppose, the node out-degree $d_O(v_i) \sim p_d(\cdot)$

$$d_O(v_i) = \mathbb{E}_{\hat{z} \sim z}(\sum_{s_{k'}=i} \hat{z}_{k'}) = \sum_{s_{k'}=i} z_{k'},$$

then we want the node degree distribution of the generated graph, i.e., $q_d(\cdot)$, to be close to $p_d(\cdot)$.

*Table 1.* Simulation performance w.r.t. MSE

| Dataset | Mass ($\times 10^{-2}$) | | | Skeleton ($\times 10^{-4}$) | | |
|---|---|---|---|---|---|---|
| Prediction steps | 1 | 10 | 20 | 1 | 10 | 20 |
| Static | 155 | 653 | 770 | 1.80 | 62.2 | 215 |
| VAR | 24.1 | 77.3 | 140 | 173 | 211 | 240 |
| LSTM (Single) | 85.1 | 162 | 198 | 3.64 | 38.7 | 109 |
| LSTM (Joint) | 9.04 | 25.5 | 74.1 | 2.82 | 28.2 | 75.0 |
| GN (full graph) | 68.7 | 186 | 238 | 4.35 | 47.4 | 135 |
| NRI | 11.1 | 40.2 | 104 | 3.81 | 39.3 | 109 |
| SUGAR | **2.01** | **7.09** | **31.6** | **1.72** | **15.0** | **40.3** |

$$\mathcal{L}_d(z) = D_{\mathrm{KL}}[q_d(\cdot)||p_d(\cdot)] = \mathbb{E}_{d(v)\sim q_d}(\log q_d - \log p_d) \\ = -\frac{1}{|V|}\sum_i \log p_d(\sum_{s_{k'}=i} z_{k'}) + \text{const}$$

In the case that $p_d(\cdot)$ is discrete, we can use continuous approximation, e.g., Gumbel softmax (Maddison et al., 2017) for the multinomial distribution.

## 4. Experiment

**Experimental Settings** We conduct experiments on both the physical simulation dataset, *Mass* (Sanchez-Gonzalez et al., 2018), and the real-world dataset, *Skeleton* (Kipf et al., 2018). *Mass* contains the observations of a chain of objects connected by strings moving in the gravity field generated by the simulation system in Sanchez-Gonzalez et al. (2018). *Skeleton* is the CMU Motion Capture Database used in Kipf et al. (2018). We compare the proposed method with the following approaches, including: (1) Static: which assumes a constant state, $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)}$; (2) VAR: Vector Auto-Regression model (Hamilton, 1994); (3) LSTM (single): A LSTM model whose weights are shared across different objects; (4) LSTM (joint): A LSTM model that jointly models the motion of all objects; (5) Graph Network (GN) (Sanchez-Gonzalez et al., 2018), where full graph is used; (6) Neural relational inference model (NRI) (Kipf et al., 2018): the KL divergence based sparse prior is used; (7) SUGAR-NP: the variant of SUGAR without using the structural prior knowledge. Detailed information of datasets and baselines are provided in Appendix C.

**Simulation Performance** Table 1 shows the performance comparison of different approaches on the two datasets based on Mean Squared Error (MSE)[1], and the best values are highlighted. We observe that: (1) SUGAR consistently achieves the best performance on both datasets for all prediction steps, which suggests the effectiveness of the proposed algorithm. The superiority of SUGAR becomes more significant with the increase of the number of prediction steps; (2) The performance of GN with the full graph is significantly worse than both NRI and SUGAR, which suggests the importance of inferring the interactions. Besides, in Figure 3, SUGAR performs better than SUGAR-NP which justify the importance of incorporating prior knowledge.

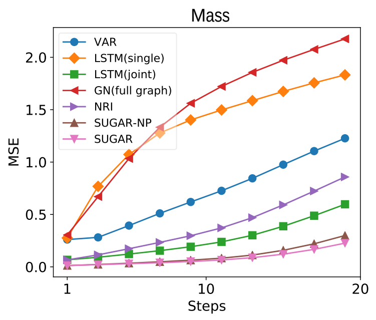


*Figure 3.* Simulation performance vs. prediction steps

*Table 2.* Interaction recovery performance

| Metric | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Corr | 63.2% | 30.2% | 67.9% | 41.8% |
| Corr (LSTM) | 57.7% | 28.2% | 76.2% | 41.2% |
| NRI | 92.7% | 72.9% | 99.3% | 84.1% |
| SUGAR-NP | 97.2% | 88.0% | **99.4%** | 93.4% |
| SUGAR | **99.2%** | **96.6%** | **99.4%** | **98.0%** |

**Interaction Recovery** Table 2 shows the performance of interaction recovery of different methods, which we compare with the baselines Corr, Corr(LSTM)[2]. We observe that: (1) SUGAR and SUGAR-NP perform clearly better than NRI and other baselines. Besides, SUGAR achieves even better performance than SUGAR-NP which justifies the importance of incorporating prior knowledge; (2) NRI usually has a high recall but relatively low precision even with the sparsity prior. There is because NRI tends to have redundant connections.

To better understand the model, we visualize the interactions learned by NRI and SUGAR (Figure A3), as well as example predictions generated by different methods (Figure A4). We observe SUGAR manages to identify the true interactions, and generates better simulations than all other baseline approaches. More prediction results and ablation studies about the effects of several types of structural information are provided in Appendix C.4.

## 5. Conclusion

In this work, we introduced SUGAR, a variational graph auto-encoder based model which effectively utilizes structural prior knowledge to better infer interactions and learn the system dynamics. In a range of experiments on both synthetic and real-world datasets, we found that with structural prior information, SUGAR achieved clearly improved performance on both interaction recovery and simulation.

[1]Evaluation w.r.t. other metrics are provided in Appendix C.3.

[2]Implementation details are provided in Appendix C.2.

## References

Battaglia, P., Pascanu, R., Lai, M., Rezende, D. J., et al. Interaction networks for learning about objects, relations and physics. In *NIPS*, pp. 4502–4510, 2016.

Battaglia, P. W. e. a. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.012*, 2018.

Bojchevski, A., Shchur, O., Zügner, D., and Günnemann, S. Netgan: Generating graphs via random walks. 2018.

Chang, M. B., Ullman, T., Torralba, A., and Tenenbaum, J. B. A compositional object-based approach to learning physical dynamics. In *ICLR*, 2017.

Franceschi, L., Niepert, M., Pontil, M., and He, X. Learning discrete structures for graph neural networks. In *ICML*, 2019.

Grover, A., Zweig, A., and Ermon, S. Graphite: Iterative generative modeling of graphs. In *ICML*, 2019.

Guttenberg, N., Virgo, N., Witkowski, O., Aoki, H., and Kanai, R. Permutation-equivariant neural networks applied to dynamics prediction. *arXiv preprint arXiv:1612.04530*, 2016.

Hamilton, J. D. *Time series analysis*, volume 2. Princeton university press Princeton, 1994.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. Neural relational inference for interacting systems. In *ICML*, 2018.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

Lee, J. B., Rossi, R., and Kong, X. Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1666–1674. ACM, 2018a.

Lee, J. B., Rossi, R. A., Kim, S., Ahmed, N. K., and Koh, E. Attention models in graphs: A survey. *arXiv preprint arXiv:1807.07984*, 2018b.

Li, R., Wang, S., Zhu, F., and Huang, J. Adaptive graph convolutional neural networks. In *AAAI*, 2018.

Liu, Q., Allamanis, M., Brockschmidt, M., and Gaunt, A. Constrained graph variational autoencoders for molecule design. In *NeurIPS*, pp. 7806–7815, 2018.

Lü, L. and Zhou, T. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*, 2017.

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M. M. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5115–5124, 2017.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Sanchez-Gonzalez, A., Heess, N., Springenberg, J. T., Merel, J., Riedmiller, M., Hadsell, R., and Battaglia, P. Graph networks as learnable physics engines for inference and control. In *ICML*, 2018.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Simonovsky, M. and Komodakis, N. Graphvae: Towards generation of small graphs using variational autoencoders. *arXiv preprint arXiv:1802.03480*, 2018.

Sukhbaatar, S., Fergus, R., et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pp. 2244–2252, 2016.

van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. In *ICLR*, 2018.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. 2018.

Wang, C., Pan, S., Long, G., Zhu, X., and Jiang, J. Mgae: Marginalized graph autoencoder for graph clustering. In *CIKM*, pp. 889–898. ACM, 2017.

Watters, N., Zoran, D., Weber, T., Battaglia, P., Pascanu, R., and Tacchetti, A. Visual interaction networks: Learning a physics simulator from video. In *Advances in Neural Information Processing Systems*, pp. 4539–4547, 2017.

You, J., Ying, R., Ren, X., Hamilton, W., and Leskovec, J. Graphrnn: Generating realistic graphs with deep autoregressive models. In *ICML*, pp. 5694–5703, 2018.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, pp. 5165–5175, 2018.

## Appendix

*Table A1.* Notation

| Name | Description |
|---|---|
| $\mathcal{G}$ | A graph $(u, V, E)$ |
| $u$ | Global variable of the graph |
| $V$ | Nodes of the graph |
| $E$ | Edges of the graph |
| $v_i$ | The $i$-th node |
| $e_k, e_{ij}$ | The $k$-th edge, the edge from $v_i$ to $v_j$ |
| $z_k, z_{ij}$ | The latent random variable representing the distribution of edge $e_k$, $e_{ij}$ |
| $\phi_v^l, \phi_e^l, \phi_u^l$ | The update functions of node, edge and global variable of encoder in layer $l$ |
| $\tilde{\phi}_v^l, \tilde{\phi}_e^l, \tilde{\phi}_u^l$ | The update functions of node, edge and global variable of decoder in layer $l$ |
| $\mathrm{x}^{(t)}$ | Observation in time $t$ |
| $\mathrm{x}_i^{(t)}$ | Observation of node $v_i$ in time $t$ |
| $\mathcal{L}.$ | Various loss functions |

Table A1 summarizes the main notations used in the paper.

## A. Detailed Technique

We incorporate prior knowledge by extending the regularization term in ELBO, i.e., $D[q(z|x)||p(z)]$, specifically:

- Customize the target probabilistic distribution $p(z)$, and minimize the KL divergence following the framework of VAE, e.g, node degree distribution and edge type distribution.
- For priors that can't easily be represented as probability distribution, e.g., graph alignment, $L_0$ sparsity, symmetric, we define customized distance metrics $D$.

**Prior graph alignment** Given a prior graph, where $e_k^*$ represents the one-hot edge type of $k$-th edge of the prior graph, we want to optimize the prediction performance using a graph that has the minimum expected number of different edges from the prior graph.

$$\begin{aligned}\mathcal{L}_{\mathcal{G}}(z) &= \sum_k \sum_m \mathbb{E}_{e' \sim z}[\mathbb{I}(e'_{k,m} \neq e^*_{k,m})] \\ &= \sum_k \sum_m z_{k,m}\; e^*_{k,m}\end{aligned}$$

**Interaction type distribution** Similar to the node degree distribution, we can enforce the interaction type distribution by first calculating the interaction type distribution from $z$, and then minimize the K-L divergence (or another differentiable distance metric of distributions).

**Symmetricity** We can enforce the symmetricity by simply setting $z_{ij} = z_{ji}$, i.e., only uses half of the latent variables.

Many priors can be specified to a particular object, e.g., L0 sparsity, as they can be written as the sum of constraints of individual object/interaction. Besides, we can also specify the node/edge dependent prior distributions to accomplish this.

## B. Additional Related Work

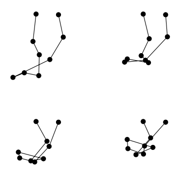
Our work also relates to literature in graph generation (You et al., 2018; Bojchevski et al., 2018; Simonovsky & Komodakis, 2018; Liu et al., 2018; Kipf & Welling, 2016; Wang et al., 2017). However, instead of generating a graph from scratch, this work focuses on inferring the interactions/edges among a set of given nodes.

Our work is different from literature in link prediction (Lü & Zhou, 2011; Zhang & Chen, 2018), as most of the link prediction method are either supervised or semi-supervised, while our task requires fully unsupervised link prediction or interaction recovery.

Some cocurrent work also investigate the problem of inferring the graph structure. Grover et al. (2019) learns the graph structure using an iterative graph refinement strategy with the low-rank approximations. Franceschi et al. (2019) propose to learn the graph structure by refining the initial KNN graph. However, none of these approaches provide a general way to incorporate prior knowledge.

## C. Detailed Experimental Results

### C.1. Dataset


(a) Mass

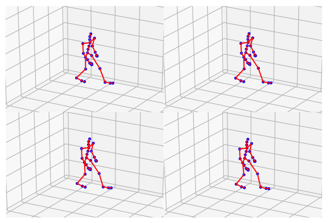

(b) Skeleton

*Figure A1.* Examples of trajectories in the experimental datasets.

*Mass*: which contains the observations of a chain of objects connected by strings moving in the gravity field. This is generated by a physical simulation system in (Sanchez-Gonzalez et al., 2018). The number of objects ranges from 5 to 8. There are 50K samples for training, 10K samples for validation and 10K samples for testing.

*Skeleton*: The CMU Motion Capture Database[3] has a large number of trajectories of different human activities, including walking, jogging, and dancing. Each sample in the dataset is the 3D trajectories of 31 nodes, each of which tracks a joint. Here we follow the data selection process in (Kipf et al., 2018): we choose 23 non-overlapping walking trials from the database and split them into training (11 trials), validation (5 trials) and test (7 trials). We use the original form of motion data (which only contains positions of each joint) in all experiments.

## C.2. Baselines

We compare the proposed method with the following approaches, including:

- Static: which assumes a constant state, $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)}$;
- VAR: Vector Auto-Regression model (Hamilton, 1994);
- LSTM (single): A LSTM based recurrent neural network. The weights are shared across different objects;
- LSTM (joint): A LSTM based recurrent neural network which takes as input the trajectories of the concatenation of all the objects in the feature dimension, and make the prediction as a whole;
- Graph Network (GN) (Sanchez-Gonzalez et al., 2018): a learnable forward and inference model with relational inductive bias. Full graph is used as the input;
- Neural relational inference model (NRI) (Kipf et al., 2018): a variational auto-encoder based inference model with graph network, the hidden dimension is 256. For the predictions on the *Mass* and *Skeleton* datasets, the KL divergence based sparse prior is used;
- Corr: We calculate a correlation matrix $R$ of all nodes, where $R_{i,j}$ is the Pearson correlation coefficient between flattened trajectories of the $i$ th node and the $j$ th node. With a threshold $\theta_1$, $(i, j)$ determined based on F1.
- Corr(LSTM): Similar with Corr, except that we use the output of the last LSTM layer at the last time step of each node to calculate the correlation matrix.
- SUGAR-NP: the variant of SUGAR without using the graph constraints derived from the structural prior knowledge.

All neural network based approaches are implemented using PyTorch (Paszke et al., 2017), and are trained using the Adam optimizer (Kingma & Ba, 2015) with learning rate annealing. The best hyperparameters are chosen based on the performance on the validation dataset. Both encoder and decoder contain two graph network blocks, with hidden dimension 64, such that it has a similar number of parameters with NRI. The initial learning rate is $5e-4$ and exponentially reduces with a ratio of $0.2$ every 50 epochs. Early stopping on the validation dataset is used. We use the multi-step prediction trick (Kipf et al., 2018), i.e., feeding the ground truth every 10 timesteps to avoid the degenerated decoder. The sparsity constraint and the node degree distribution constraint are used in the *Mass* and *Skeleton* dataset.

Note that, SUGAR and NRI share the same inputs on both Mass and Skeleton, i.e., SUGAR does not has additional input, e.g., the gravity. Instead, the global variable is a zero vector in the input layer, and the global variables is designed to facilitate communications between not directly connected nodes (in the decoder), and to additional capture global interactions.

We also conduct experiments on the *Spring* dataset Kipf et al. (2018). Both NRI and SUGAR achieve near perfect results, i.e., visually no difference from the ground truth and interaction recovery accuracy greater than 99%.

## C.3. More Simulation Performance

Table A2,A3 and A4 show the simulation performance of different baselines w.r.t. MAE, MAPE and SMAPE respectively. SUGAR consistently achieves the best performance for different prediction steps on both datasets.

*Table A2.* Simulation performance w.r.t. MAE

| Dataset | Mass ($\times10^{-2}$) | | | Skeleton ($\times10^{-2}$) | | |
|---|---|---|---|---|---|---|
| Prediction steps | 1 | 10 | 20 | 1 | 10 | 20 |
| Static | 159 | 332 | 375 | 1.75 | 8.98 | 16.5 |
| VAR | 57.6 | 111 | 147 | 16.0 | 17.4 | 18.5 |
| LSTM (Single) | 115 | 160 | 176 | 2.32 | 6.94 | 11.5 |
| LSTM (Joint) | 35.2 | 57.2 | 96.0 | 2.06 | 6.05 | 9.80 |
| GN (full graph) | 94.5 | 169 | 195 | 2.54 | 7.60 | 12.7 |
| NRI | 33.4 | 71.6 | 117 | 2.42 | 7.15 | 11.8 |
| SUGAR | **6.86** | **16.8** | **43.3** | **1.56** | **4.22** | **6.72** |

## C.4. Ablation Study

To investigate the effect of incorporating prior knowledge about the structure, we conduct experiments on the *Mass* dataset.

[3]http://mocap.cs.cmu.edu/

*Table A3.* Simulation performance w.r.t. MAPE

| Dataset | Mass (%) | | | Skeleton (%) | | |
|---|---|---|---|---|---|---|
| Prediction steps | 1 | 10 | 20 | 1 | 10 | 20 |
| Static | 47.37 | 96.23 | 119.64 | 0.82 | 4.18 | 7.77 |
| VAR | 15.74 | 32.28 | 46.61 | 7.14 | 7.71 | 8.24 |
| LSTM (Single) | 32.87 | 45.76 | 54.11 | 1.05 | 3.12 | 5.26 |
| LSTM (Joint) | 10.63 | 17.46 | 32.35 | 0.92 | 2.68 | 4.37 |
| GN (full graph) | 28.24 | 51.14 | 64.12 | 1.16 | 3.42 | 5.79 |
| NRI | 10.28 | 21.25 | 38.38 | 1.11 | 3.26 | 5.39 |
| SUGAR | **2.00** | **4.80** | **14.41** | **0.74** | **2.01** | **3.24** |

*Table A4.* Simulation performance w.r.t. SMAPE

| Dataset | Mass (%) | | | Skeleton (%) | | |
|---|---|---|---|---|---|---|
| Prediction steps | 1 | 10 | 20 | 1 | 10 | 20 |
| Static | 46.23 | 90.81 | 105.01 | 0.82 | 4.18 | 7.74 |
| VAR | 16.02 | 34.40 | 53.84 | 7.15 | 7.74 | 8.26 |
| LSTM (Single) | 34.16 | 50.69 | 64.17 | 1.05 | 3.12 | 5.24 |
| LSTM (Joint) | 10.62 | 17.49 | 33.98 | 0.92 | 2.68 | 4.37 |
| GN (full graph) | 27.99 | 50.57 | 66.48 | 1.16 | 3.42 | 5.77 |
| NRI | 10.19 | 21.42 | 40.47 | 1.11 | 3.25 | 5.38 |
| SUGAR | **1.96** | **4.61** | **13.66** | **0.74** | **2.01** | **3.22** |

### C.4.1. Effect of sparsity constraint

Figure A2 shows the effect of applying $L_0$ sparsity prior, where SUGAR-SP50 means the regularization coefficient is 50. We observe that with the increase of the regularization coefficient, the precision generally increases, while the recall first stays stable and then decreases.

### C.4.2. Effect of node degree constraint

Table A5 shows the effect of the node degree constraint. We denote as SUGAR-NDC the SUGAR model incorporating the node degree constraint. The constraint is applied when the performance becomes stable on the validation dataset. We find that applying the node degree constraint greatly increase the precision, while the recall become slightly worse, resulting a significantly improved F1 score.

## C.5. Example Simulation and Inferred Interactions

To better understand the model, we visualize the interactions learned by NRI and SUGAR (Figure A3), as well as example predictions generated by different methods (Figure A4). We observe SUGAR manages to identify the true interactions, while NRI has a couple of redundant interactions even with its sparsity prior.

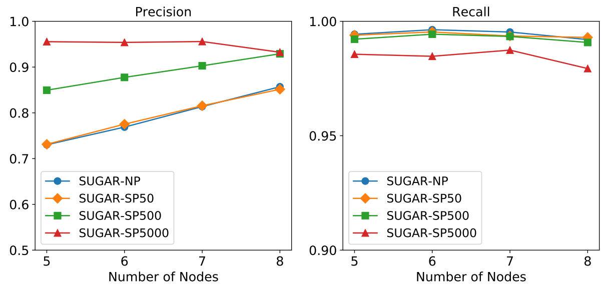


*Figure A2.* Effect of the sparsity constraint

*Table A5.* Effect of the node degree constraint on the *Mass* dataset.

| Metric | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| NRI | 92.7% | 72.9% | 99.3% | 84.1% |
| SUGAR-NP | 97.2% | 88.0% | **99.4%** | 93.4% |
| SUGAR-NDC | **99.2%** | **97.2%** | 98.8% | **98.0%** |

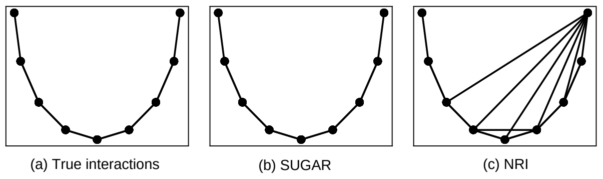


*Figure A3.* Interactions learned on the *Mass* dataset. NRI usually infers redundant interactions, while SUGAR recovers the ground truth interactions.

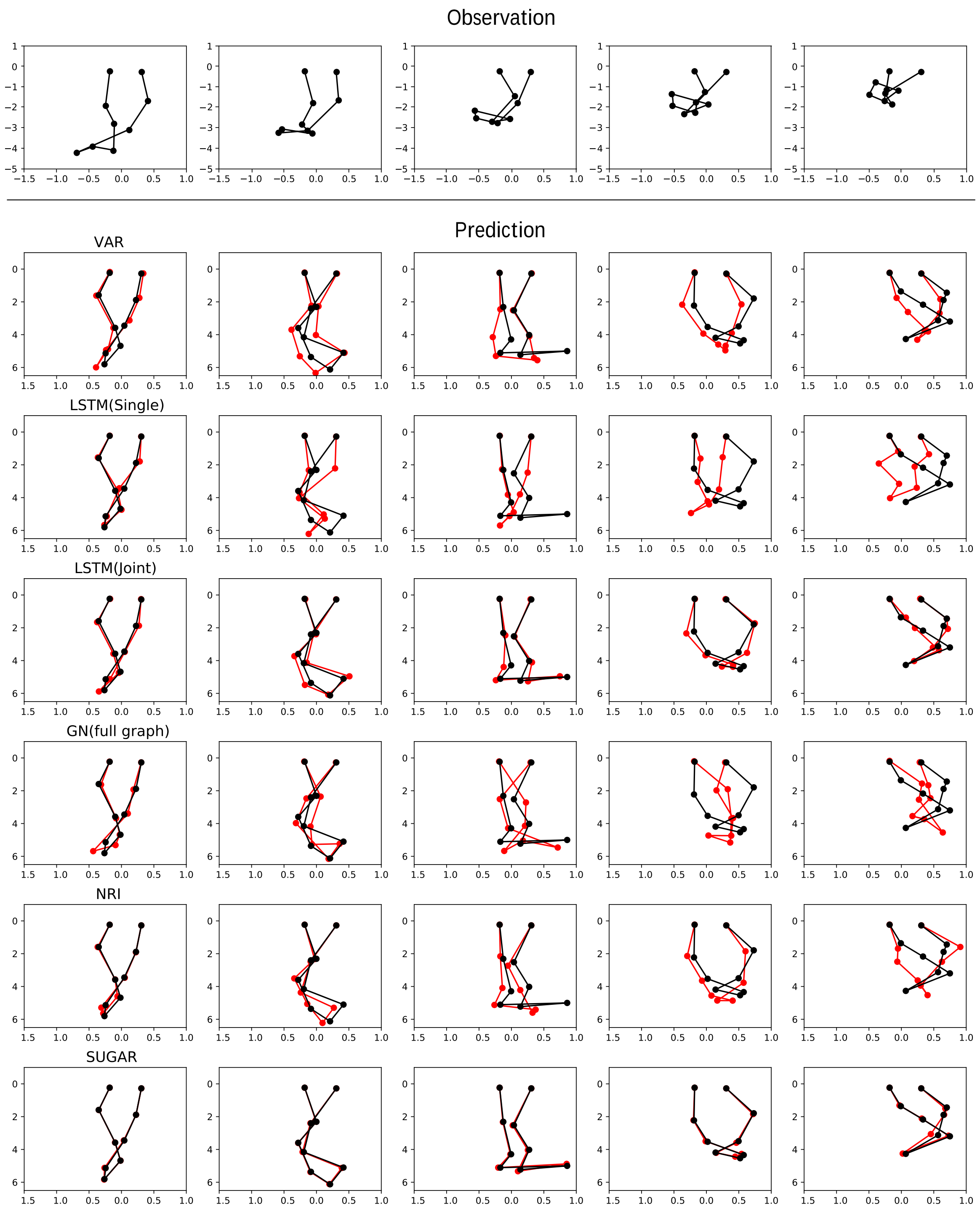


*Figure A4.* Observation (first row), simulation (black) and ground truth (red) on the *Mass* dataset