NARTI: Neural Algorithmic Reasoning for Trajectory Inference

Dobrik Georgiev [*1] **Ramon Viñas** [*1] **Sam Considine** [1] **Bianca Dumitrascu** [2] **Pietro Liò** [1]

Abstract

Trajectory inference algorithms aim to reconstruct the developmental trajectory of single cells from high-dimensional gene expression data. To solve this problem, standard techniques employ multi-step pipelines, which may have information barriers, raising the question of potential benefits in end-to-end differentiable alternatives. Here we leverage some of the latest advances in neural algorithmic reasoning to propose NARTI (Neural Algorithmic Reasoning for Trajectory Inference), an end-to-end differentiable replacement to classical algorithms. We conduct an extensive comparative analysis and show that NARTI attains competitive performance over a broad range of scRNA-seq datasets. We believe that NARTI can facilitate a seamless integration of multiple steps from standard trajectory inference techniques.

1. Introduction

Cell differentiation is a gradual process whereby a cell lineage undergoes continual specialisation, resulting in descendant cells whose gene expression is substantially different from their ancestors. However, most single-cell RNA sequencing methodologies involve cell lysis and can only offer a static snapshot of the single-cell gene expression at a single point in time (Ziegenhain et al., 2017). This restricts our ability to infer trajectories, necessitating computational approaches that can establish differentiation patterns from static expression profiles.

Trajectory analysis techniques offer a solution to this challenge. Most trajectory inference algorithms follow the pattern established by Monocle (Trapnell et al., 2014) and Slingshot (Street et al., 2018), consisting of several steps. First, they apply a dimensionality reduction technique to project high-dimensional single-cell gene expression into a low-dimensional space. Second, they cluster the low-dimensional cell representations. Third, they construct a minimum spanning tree (MST) on the cell clusters. Finally, starting from a root cluster, they order the cells based on their position along the MST, yielding *pseudotime* values that describe the relative progress of a cell along the cell lineage.

The multi-step nature of traditional trajectory inference algorithms hampers information flow between different stages. To alleviate this issue, we investigate the application of Neural Algorithmic Reasoning (NAR), an emerging research paradigm that designs neural networks with the capacity to execute algorithmic computation (Velickovic & Blundell, 2021). We propose Neural Algorithmic Reasoning for Trajectory Inference (NARTI), an approach to seamlessly combine standard trajectory inference steps in an end-to-end pipeline (Figure 1), facilitating information flow between stages in a differentiable manner. We evaluate NARTI using an extensive benchmark of 15 single-cell RNA-seq datasets (8 synthetic, 7 real) and show excellent trajectory inference capabilities. We believe that NARTI can enable information flow in multi-step lineage inference pipelines.

Neural Algorithmic Reasoning Neural Algorithmic Reasoning (NAR) is an emerging research field that focuses on constructing neural networks inspired by algorithms. The goal is to train a neural model, usually a GNN, capable of approximating and executing classical algorithms. Traditionally, NAR is employed for "breaking the scalar bottleneck", enabling algorithm execution in high-dimensional latent space. In this paper, we propose a novel application of NAR as a probabilistic and differentiable counterpart to classical algorithms.

2. Methodology

Problem formulation Let $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ be a single-cell transcriptomics dataset consisting of m cells and n genes. Denote by $\boldsymbol{x}_i \in \mathbb{R}^n$ the i-th entry of $\boldsymbol{X}$, corresponding to the gene expression values of cell i. Our goal is two-fold: 1) infer a shared backbone describing the developmental trajectory of cells and 2) infer the position of every cell on the trajectory backbone. We define the set of cluster

[*]Equal contribution [1]Department of Computer Science and Technology, University of Cambridge, Cambridge, UK [2] Department of Statistics and Irving Institute for Cancer Dynamics, Columbia University, US. Correspondence to: Bianca Dumitrascu <bmd2151@cam.ac.uk>, Pietro Liò <pl219@cam.ac.uk>.

The 2023 ICML Workshop on Computational Biology. Baltimore, Maryland, USA, 2023.

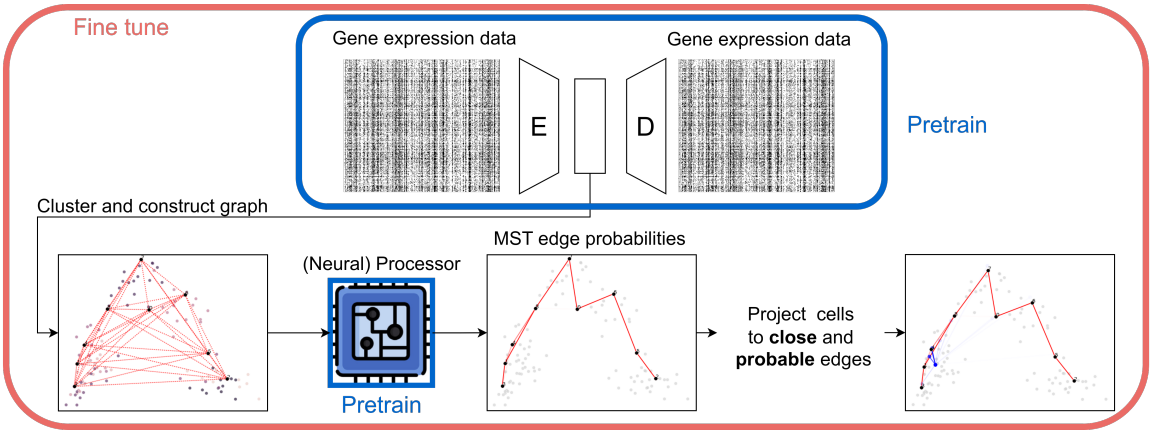

Figure 1. The NARTI pipeline training process begins with the pretraining of an autoencoder and a MST neural algorithmic reasoner. The components are then trained together within the pipeline. For further details, please refer to section 2.

centroids in the backbone as $\{\boldsymbol{\mu}_k \in \mathbb{R}^d\}_{k=1}^K$, where K, the user-definable number of cell clusters, and d, the cluster dimension, are hyperparameters.

NARTI's workflow for trajectory inference integrates multiple steps (Figure 1), including dimensionality reduction, clustering cells in the latent space, fitting a probabilistic minimum spanning tree (MST) on the cell clusters, and projecting the cells onto the MST.

Dimensionality reduction and clustering The first step of NARTI's workflow is to project the input gene expression into a low-dimensional space $\mathbb{R}^d$. For a given cell i with expression $\boldsymbol{x}_i$, we employ a multi-layer perception (MLP) to infer latent embeddings $\boldsymbol{z}_i \in R^d$, i.e. $\boldsymbol{z}_i = \text{MLP}(\boldsymbol{x}_i)$.

Let $\mathcal{C}_k$ be the set of cell indices belonging to cluster k. We initialise the centroids $\boldsymbol{\mu}_k$ by applying the k-means algorithm on the low-dimensional embeddings $\boldsymbol{z}_i$. We use the cluster centers to infer the trajectory backbone and refine them during the fine-tuning step.

Computing the trajectory backbone To compute the trajectory backbone, we consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each node $u \in \mathcal{V}$ represents a cluster centroid $\boldsymbol{\mu}_u$ and edges connect all centroid pairs. For every u and v, the edge weight $e_{uv} \in \mathcal{E}$ corresponds to the Euclidean distance between their respective centroids, i.e. $e_{uv} = \|\boldsymbol{\mu}_u - \boldsymbol{\mu}_v\|$.

The probabilistic perspective of the neural processor provides us with the backbone probabilities, denoted as p_{uv}, representing the likelihood of v being a predecessor of u where $u, v \in \mathcal{V}$. The computed probabilities can be further smoothed using a temperature parameter. This is particularly useful in the early stages of the fine-tuning process when the structural certainty may be limited.

Learning to find a MST with NAR We implement NAR as an encode-process-decode network (Ibarz et al., 2022). Given a MST instance we use linear projections to embed the algorithm inputs (e.g. edge weights) as well as an intermediate state (including current nodes in the MST and priority queue values) into high dimensional vectors ($\mathbf{u}_i$ for nodes and $\mathbf{e}_{ij}$ for edges). The inputs are then processed recurrently by a processor P, i.e. an message passing neural network (Gilmer et al., 2017) with `max` pooling:

$$\mathbf{m}_i^{(t+1)} = \max_{1 \le j \le n} f_m\left(\boldsymbol{u}_i^{(t)}, \boldsymbol{u}_j^{(t)}, \boldsymbol{h}_i^{(t)}, \boldsymbol{h}_j^{(t)}, \mathbf{e}_{ij}^{(t)}\right)$$
$$\mathbf{h}_i^{(t+1)} = f_r\left(\boldsymbol{u}_i^{(t)}, \boldsymbol{h}_i^{(t)}, \mathbf{m}_i^{(t)}\right)$$

where $\mathbf{h}_i^{(0)} = \mathbf{0}$, f_m is the message, and f_r is the readout function of P. At each timestep, we decode intermediate outputs that are then reused in subsequent steps. After $|\mathcal{V}|$ iterations, we decode the final outputs. We train NAR to mimic the behaviour of Prim's MST algorithm (Ibarz et al. (2022) and Appendix A).

Projecting cells onto the probabilistic backbone For a given cell i with embeddings $\boldsymbol{z}_i$, we compute its projection $\hat{\boldsymbol{z}}_i^{(u,v)}$ onto edge e_{uv} as follows:

$$\hat{\boldsymbol{z}}_i^{(u,v)} = \boldsymbol{\mu}_u + t(\boldsymbol{z}_i, u, v)(\boldsymbol{\mu}_v - \boldsymbol{\mu}_u)$$
$$t(\boldsymbol{z}_i, u, v) = \frac{(\boldsymbol{z}_i - \boldsymbol{\mu}_u)^\top (\boldsymbol{\mu}_v - \boldsymbol{\mu}_u)}{||\boldsymbol{\mu}_v - \boldsymbol{\mu}_u||}$$

where $t(\boldsymbol{z}_i, u, v) \in R$ is the scalar projection of $\boldsymbol{z}_i$ onto e_{uv} and we bind it in the interval [-0.10, 1.10]. We then compute the probability $p(\boldsymbol{z}_i, u, v)$ of cell i belonging to edge e_{uv} of the probabilistic backbone as:

$$p(\boldsymbol{z}_i, u, v) = \frac{1/||\boldsymbol{z}_i - \hat{\boldsymbol{z}}_i^{(u,v)}||}{\sum_{(u',v') \in \mathcal{E}} 1/||\boldsymbol{z}_i - \hat{\boldsymbol{z}}_i^{(u',v')}||}$$

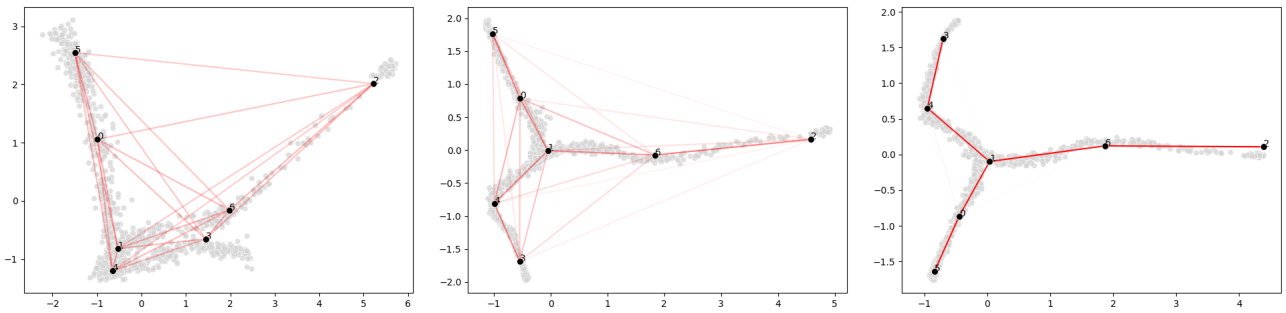

Figure 2. Evolution of cell and cluster embeddings during training. Edge probabilities are represented by opaqueness (less visible = lower probability). We used PCA for dimensionality reduction and show snapshots at epoch 0, 40, and 80 on the `bifurcating_1` dataset.

We favour this normalisation scheme over softmax because it is invariant to scale: the probabilities $p(\boldsymbol{z}_i, u, v)$ remain invariant when distances are scaled by a constant factor.

Fine-tuning Using a neural processor pre-trained for probabilistic MST inference, we fine-tune the entire NARTI architecture in an end-to-end fashion. We optimise the parameters by minimising a three component loss function.

Let f_e and f_d be the encoder and decoder networks, respectively. The first loss term $\mathcal{L}_{rec}\left(\boldsymbol{x}_i, f_d \circ f_e(\boldsymbol{x}_i)\right)$ maximises the likelihood of the data. In this work, the decoder f_d outputs the parameters of a (zero-inflated) Negative Binomial (ZINB) distribution and the loss function $\mathcal{L}_{rec}$ maximises the corresponding ZINB likelihood.

Second, we introduce a loss term $\mathcal{L}_{pr}$ that penalises the reconstruction error of the gene expression reconstructed from the projected latent variables. For every edge e_{uv} of the MST and a given cell i with projection $\hat{\boldsymbol{z}}_i^{(u,v)}$, we minimise the reconstruction loss $\mathcal{L}_{rec}\left(\boldsymbol{x}_i, f_d \circ \hat{\boldsymbol{z}}_i^{(u,v)}\right)$ weighted by the probability $p(\boldsymbol{z}_i, u, v)$ of projecting $\boldsymbol{z}_i$ onto edge e_{uv}:

$$\mathcal{L}_{pr}(\boldsymbol{x}_i, u, v) = p(\boldsymbol{z}_i, u, v)\mathcal{L}_{rec}\left(\boldsymbol{x}_i, f_d \circ \hat{\boldsymbol{z}}_i^{(u,v)}\right)$$

The final loss term $\mathcal{L}_{dist}$ considers the distances between each cell and its most likely projection:

$$\mathcal{L}_{dist}(\boldsymbol{x}_i) = \alpha||\boldsymbol{z}_i - \hat{\boldsymbol{z}}_i^{(u,v)}||$$

where $(u, v) = \arg\min_{(u',v')\in\mathcal{E}} p(\boldsymbol{z}_i, u, v)$ and $\alpha = 10^{-2}$. This term acts as an attracting force between the latent variables $\boldsymbol{z}_i$ and their most likely projection $\hat{\boldsymbol{z}}_i^{(u,v)}$, discouraging cases of cells attaining both low reconstruction and projection losses with distant representations $\boldsymbol{z}_i$ and $\hat{\boldsymbol{z}}_i^{(u,v)}$.

To balance exploration versus exploitation, we employ an edge probability annealing schedule (Figure 2) and centroid adjustment scheme (Appendix B).

Conversion to pseudotime To calculate the pseudotime of every cell, we first infer the most likely backbone from the neural processor by computing $\hat{\pi}_i = \arg\max_j \pi_{ij}$, where π_{ij} denotes the probability of j being predecessor of i in the MST. Then we obtain the backbone $\mathcal{MST}(\mathcal{V}, \mathcal{E}')$ with $\mathcal{E}' = \{(i, \hat{\pi}_i)|i \in \mathcal{V}, i \text{ is not root}\}$. We note that while algorithmic reasoners do not possess the guarantees of classical algorithms, we were always able to obtain a valid tree. This was likely due to the fact that the graph sizes during test time were in or close to our training distribution.

For a given cell i and backbone, we compute the pseudotime by first projecting its embeddings $\boldsymbol{z}_i$ onto the closest edge e_{uv} of the MST, i.e. $\hat{\boldsymbol{z}}_i^{(u,v)}$, and then finding the length of the path between the root node of the MST and $\hat{\boldsymbol{z}}_i^{(u,v)}$.

3. Results

Experimental setup and datasets We pre-trained the reasoner on synthetic datasets generated using the CLRS-30 benchmark (Veličković et al., 2022), using graphs of up to 16 nodes (1000 epochs). We used $\boldsymbol{h}_i$ of dimension 32 for the reasoner model and, similar to VITAE (Du et al., 2020), latent representations $\boldsymbol{z}_i$ of size 8. We then fine-tuned the entire pipeline on multiple single-cell datasets, including synthetic datasets generated in (Du et al., 2020) and real datasets: *aging* (Kowalczyk et al., 2015), *mesoderm* (Loh et al., 2016), and *human embryos* (Petropoulos et al., 2016) with different trajectory topologies. We trained the encoder and decoder using early stopping with a patience of 40 epochs. We used a batch size of 128 and the Adam (Kingma & Ba, 2015) optimiser with a learning rate of 0.001.

We evaluated our model on 5 different metrics (Appendix E), including the Graph Edit Distance (GED) (Abu-Aisheh et al., 2015), Ipsen-Mikhailov (IM) distance (Jurman et al., 2015), adjusted rank index (ARI) (Hubert & Arabie, 1985), and generalised adjusted rank index (GRI) and pseudotime (PTD) score (Du et al., 2020).

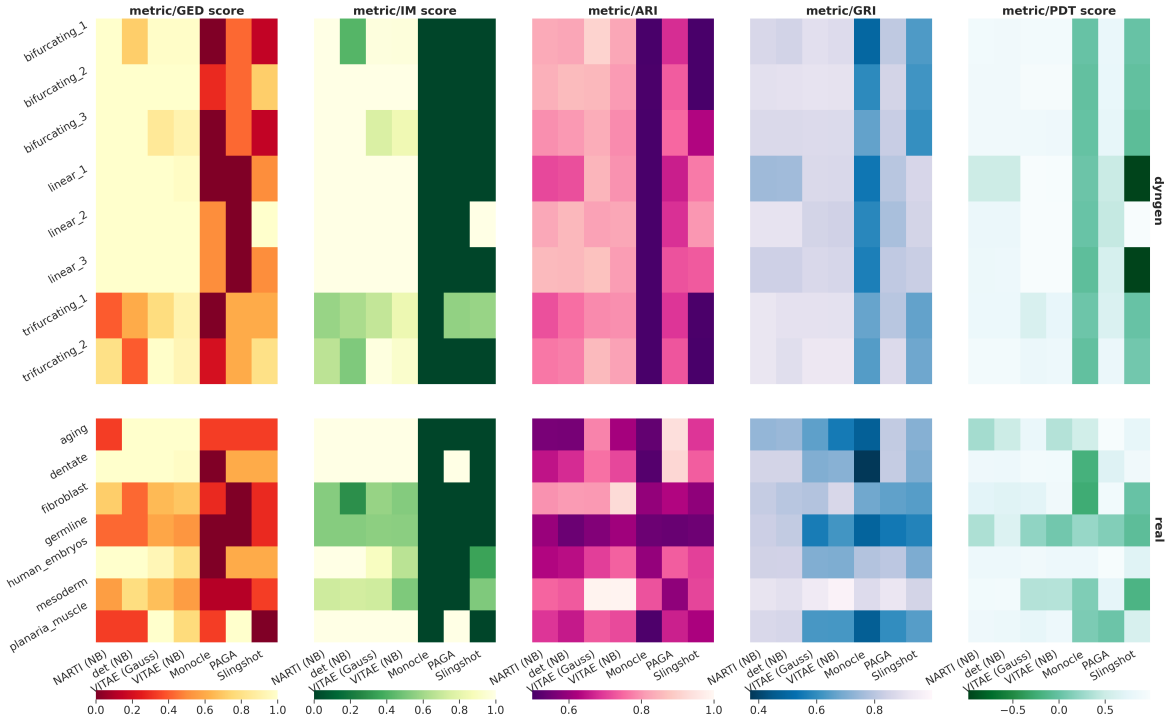

Figure 3. Comparative analysis of trajectory inference algorithms. We considered multiple baselines, including VITAE (Du et al., 2020), Monocle 3 (Cao et al., 2019), Slingshot (Street et al., 2018), PAGA (Wolf et al., 2019), and NARTI (neuralised and deterministic). For the non-NARTI baselines, we reported the results from Du et al. (2020). We evaluated performance using metrics that assess the goodness of the inferred topology (GED, IM), clustering (ARI, GRI), and cell ordering (PDT). Lighter values correspond to increased performance.

Results We benchmarked NARTI against VITAE (Du et al., 2020), Monocle 3 (Cao et al., 2019), Slingshot (Street et al., 2018) and PAGA (Wolf et al., 2019). For all baseline approaches, we reported the results from Du et al. (2020). We also compared NARTI to a non-neuralised variant that uses the classical Prim's MST algorithm without neural algorithmic reasoning.

Overall, NARTI displayed competitive performance across real and synthetic datasets (Figure 3). **On synthetic datasets**, NARTI and VITAE achieved comparable results, outperforming other baselines in terms of GED/IM, the only exception being trifurcating datasets, where NARTI achieved GED of 0.4 and 0.8 vs 0.90 and 0.93 for VITAE. **On real datasets**, NARTI attained superior GRI (cf. Appendix C) and PDT scores than most baselines including VITAE, but had comparatively lower GED (GED: 0.33 on `planaria muscle` and `aging`) and ARI (ARI: 0.75 on `mesoderm` and ARI: 0.56 on `aging`).

We performed an ablation study to determine to what extent the neuralised version of the MST algorithm improves downstream performance. We replaced the neural processor of NARTI with the deterministic counterpart *det* that produces an exact MST (i.e. with probability one for edges in the MST and zero otherwise). NARTI outperformed *det* in terms of GED score in 10 out of 15 datasets. For example, *det* scored GED: 0.71 in `bifurcating_1`, while NARTI achieved the maximum score of GED: 1. In `fibroblast` NARTI (GED: 0.71) substantially outperformed *det* (GED: 0.43).

We hypothesise that using neural algorithmic reasoning is advantageous because of its probabilistic nature, allowing to incorporate controllable uncertainty. The main benefits of using NAR over non-probabilistic algorithms are that 1) NAR allows reasoning over a larger subspace of minimum spanning trees, producing potentially smoother loss landscapes and 2) NAR avoids error propagation from early-epoch mispredictions (see Appendix D).

In summary, we presented a novel application of neural algorithmic reasoning to pseudotime trajectory inference that seamlessly combines the steps of established algorithms in an end-to-end pipeline. In spite of the promising results, we believe future work may benefit from refined clustering approaches (e.g. neuralised k-means) and improved optimisation. Further, although here we applied NAR as a probabilistic proxy of a classical algorithm, we envision that future work may also leverage the end-to-end differentiability of NAR for improved performance and explainability.

References

Abu-Aisheh, Z., Raveaux, R., Ramel, J.-Y., and Martineau, P. An exact graph edit distance algorithm for solving pattern recognition problems. In *4th International Conference on Pattern Recognition Applications and Methods 2015*, 2015.

Cao, J., Spielmann, M., Qiu, X., Huang, X., Ibrahim, D. M., Hill, A. J., Zhang, F., Mundlos, S., Christiansen, L., Steemers, F. J., et al. The single-cell transcriptional landscape of mammalian organogenesis. *Nature*, 566(7745): 496–502, 2019.

Du, J.-H., Gao, M., and Wang, J. Model-based trajectory inference for single-cell rna sequencing using deep learning with a mixture prior. *bioRxiv*, pp. 2020–12, 2020.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Hubert, L. and Arabie, P. Comparing partitions. *Journal of classification*, 2:193–218, 1985.

Ibarz, B., Kurin, V., Papamakarios, G., Nikiforou, K., Bennani, M., Csordás, R., Dudzik, A. J., Bosnjak, M., Vitvitskyi, A., Rubanova, Y., Deac, A., Bevilacqua, B., Ganin, Y., Blundell, C., and Velickovic, P. A generalist neural algorithmic learner. In Rieck, B. and Pascanu, R. (eds.), *Learning on Graphs Conference, LoG 2022, 9-12 December 2022, Virtual Event*, volume 198 of *Proceedings of Machine Learning Research*, pp. 2. PMLR, 2022. URL https://proceedings.mlr.press/v198/ibarz22a.html.

Jurman, G., Visintainer, R., Filosi, M., Riccadonna, S., and Furlanello, C. The him glocal metric and kernel for network comparison and classification. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pp. 1–10. IEEE, 2015.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Kowalczyk, M. S., Tirosh, I., Heckl, D., Rao, T. N., Dixit, A., Haas, B. J., Schneider, R. K., Wagers, A. J., Ebert, B. L., and Regev, A. Single-cell rna-seq reveals changes in cell cycle and differentiation programs upon aging of hematopoietic stem cells. *Genome research*, 25(12): 1860–1872, 2015.

Loh, K. M., Chen, A., Koh, P. W., Deng, T. Z., Sinha, R., Tsai, J. M., Barkal, A. A., Shen, K. Y., Jain, R., Morganti, R. M., et al. Mapping the pairwise choices leading from pluripotency to human bone, heart, and other mesoderm cell types. *Cell*, 166(2):451–467, 2016.

Petropoulos, S., Edsgärd, D., Reinius, B., Deng, Q., Panula, S. P., Codeluppi, S., Reyes, A. P., Linnarsson, S., Sandberg, R., and Lanner, F. Single-cell rna-seq reveals lineage and x chromosome dynamics in human preimplantation embryos. *Cell*, 165(4):1012–1026, 2016.

Prim, R. C. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6): 1389–1401, 1957.

Street, K., Risso, D., Fletcher, R. B., Das, D., Ngai, J., Yosef, N., Purdom, E., and Dudoit, S. Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics. *BMC genomics*, 19:1–16, 2018.

Trapnell, C., Cacchiarelli, D., Grimsby, J., Pokharel, P., Li, S., Morse, M., Lennon, N. J., Livak, K. J., Mikkelsen, T. S., and Rinn, J. L. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nature biotechnology*, 32(4):381–386, 2014.

Velickovic, P. and Blundell, C. Neural algorithmic reasoning. *Patterns*, 2(7):100273, 2021. doi: 10.1016/j.patter.2021.100273. URL https://doi.org/10.1016/j.patter.2021.100273.

Veličković, P., Badia, A. P., Budden, D., Pascanu, R., Banino, A., Dashevskiy, M., Hadsell, R., and Blundell, C. The clrs algorithmic reasoning benchmark. In *International Conference on Machine Learning*, pp. 22084–22102. PMLR, 2022.

Wolf, F. A., Hamey, F. K., Plass, M., Solana, J., Dahlin, J. S., Göttgens, B., Rajewsky, N., Simon, L., and Theis, F. J. Paga: graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cells. *Genome biology*, 20:1–9, 2019.

Ziegenhain, C., Vieth, B., Parekh, S., Reinius, B., Guillaumet-Adkins, A., Smets, M., Leonhardt, H., Heyn, H., Hellmann, I., and Enard, W. Comparative analysis of single-cell rna sequencing methods. *Molecular Cell*, 65(4):631–643.e4, 2017. ISSN 1097-2765. doi: https://doi.org/10.1016/j.molcel.2017.01.023. URL https://www.sciencedirect.com/science/article/pii/S1097276517300497.

A. NAR Losses

When executing the Prim's algorithm (Prim, 1957) on a graph, the network has to make a number of predictions, both at each step and as a separate final prediction. We list those below together with the loss used to optimise each prediction. Predictions prefixed with *(Step)* are made at every step and losses for them are averaged for all timesteps. Prediction marked *(Final)* is made only once after $|\mathcal{V}|$ iterations have passed.

- *(Step)* Predecessors in the partially built MST – for every node in the partial MST, the NAR model produces probabilities of each other to be its predecessor. Given the ground-truth predecessor index, we optimise this prediction using categorical cross entropy.
- *(Step)* Value in the MST priority queue – In Prim's algorithm, the MST is built by using a priority queue, where each node has an associated value. Our model is optimised to predict what this value is for each node by minimising mean-square between ground-truth and prediction.
- *(Step)* Nodes in the partial MST – at each step, the model predicts which nodes have been already added to the MST and which have not. Optimised using binary cross entropy.
- *(Step)* Nodes in Prim's algorithm queue – similar to above, at each step, the model predicts which nodes have been already added to the priority queue. Optimised using binary cross entropy.
- *(Step)* Node added to the MST at every timestep t – Prim's algorithm adds builds the MST one node per an algorithm step. Our model aims to predict which node is added at the given timestep. Optimised using categorical cross entropy.
- *(Final)* Predecessors in the final MST – after $|\mathcal{V}|$ iterations, Prim's algorithm produces the MST. We require our model to do the same. We again predict probabilities of each other to be its predecessor, but for all nodes in the graph. Optimised using categorical cross entropy.

B. Adjustment of cluster positions

After a backpropagation operation the cluster centroids may no longer be valid, as shown in Figure 4 (their valid positions are given as $+_1$ and $+_2$). However, reassigning them (dashed) results in shorter segments with every epoch, eventually collapsing the cell coordinates to a single point (singularity). To avoid this problem, we:

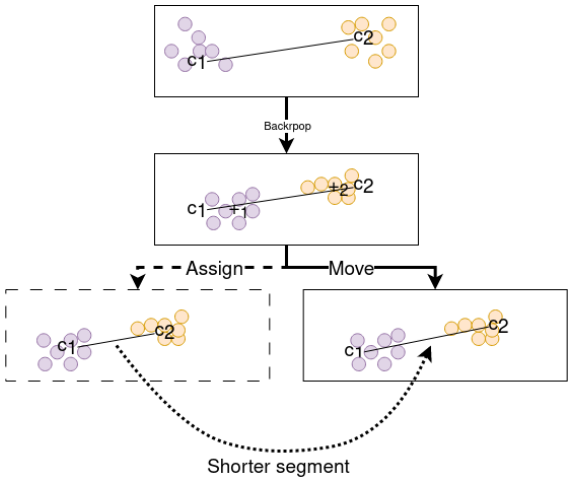

Figure 4. Our choice of cluster reassignment (solid pipeline), allows for clusters to change their positions. We chose **moving** clusters towards the new centers of masses rather than reassigning them, to avoid singularity collapse.

- Move the clusters towards the respective centers of masses (COMs), instead of reassigning.
- Decay the clusters displacement along training epochs. Similarly to the edge temperature, we start from moving the clusters by 0.5 of the distance to the new COMs and reduce that by a factor of $\times 0.95$ every epoch.

C. GRI scores

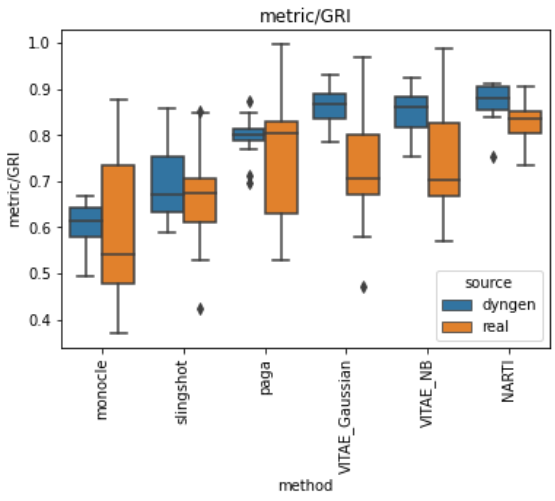

Figure 5. Our method (NARTI, rightmost) achieves comparable Generalised Rank Index on synthetic datasets and improves on previous methods on real-world data.

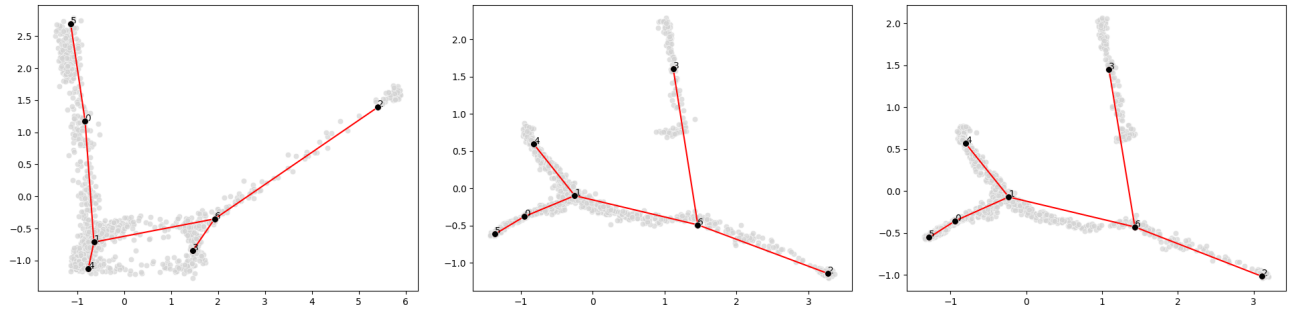

Figure 6. Evolution of cell and cluster embeddings during training using the classical, non-probabilistic Prim's MST algorithm instead of the neural reasoner. We used PCA for dimensionality reduction and show snapshots at epoch 0, 40, and 80 on the `bifurcating_1` dataset. The ground-truth topology corresponds to a bifurcating tree.

D. Optimisation evolution of a deterministic reasoner

As can be seen in Figure 6 using a deterministic MST may result in backbone estimation errors as the method has access to a single point of the MST distribution, which may propagate to later stages of training. By using NAR and controlling the temperature parameter, we can avoid committing to concrete backbone trajectory early on, essentially providing us with access to a wider distribution of MSTs.

E. Evaluation metrics

We considered the following metrics:

- Graph Edit Distance (GED): quantifies the minimum number of edit operations needed to transform the inferred backbone into the ground-truth backbone (Abu-Aisheh et al., 2015).
- Ipsen-Mihailov distance (IM): measures the spectra dissimilarity between the adjacency matrices of two graphs (Jurman et al., 2015).
- Adjusted Rank Index (ARI): Measures the similarity between the estimated clusters and the clusters given by the ground truth cell-types (Hubert & Arabie, 1985).
- Generalised Rank Index (GRI): Extension of the Rank Index that compares the similarity between the estimated and reference cell positions on the trajectory backbone (Du et al., 2020).
- Pseudotime score (PDT): Pearson correlation between the estimated and ground-truth per-cell pseudotimes.

For a detailed mathematical description of these metrics we refer the reader to Du et al. (2020).