Group Meeting - November 20, 2020

Paper review & Research progress

Truong Son Hy *

*Department of Computer Science
The University of Chicago

Ryerson Physical Lab

Aristotle (Today is the National Teachers' Day)

1. Teachers, who educate children, deserve more honor than parents, who merely gave them birth; for the latter provided mere life, while the former ensure a good life.
2. Teaching is the highest form of understanding.

Aristotle and Alexander the Great

Papers

1. **NetGAN: Generating Graphs via Random Walks** (ICML 2018)
   `http://proceedings.mlr.press/v80/bojchevski18a/bojchevski18a.pdf`
2. **A step towards neural genome assembly** (NeurIPS 2020 workshop)
   `https://arxiv.org/abs/2011.05013`

Paper 1

NetGAN: Generating Graphs via Random Walks (ICML 2018)
Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, Stephan Günnemann
`https://arxiv.org/abs/1803.00816`

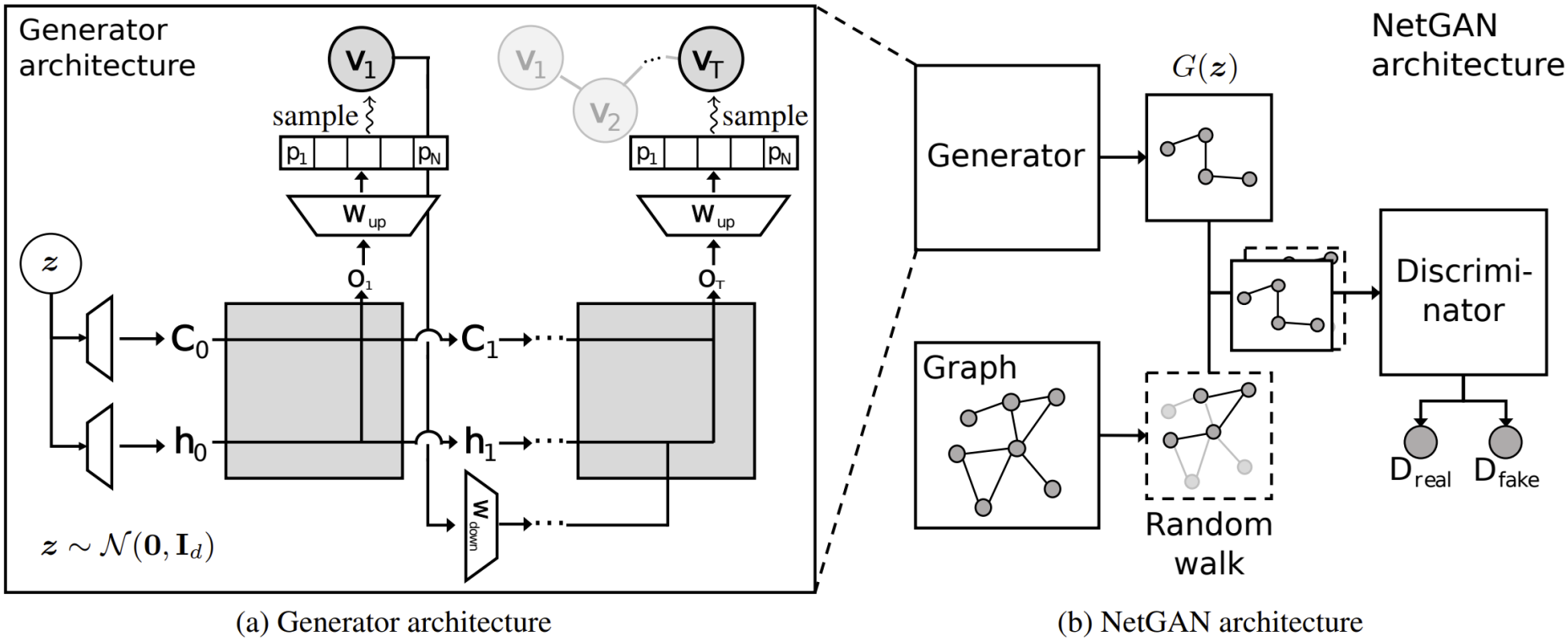
Proposals

Proposals

NetGAN:

- Graph generation as learning the distribution of biased **random walks** over the input graph.
- **Wasserstein** GAN objective (optimal transport).

Overview



(a) Generator architecture

(b) NetGAN architecture

Figure 2: The NetGAN architecture proposed in this work (b) and the generator architecture (a).

Generator & Discriminator (1)

Procedure of generating random walks:

$$
\begin{aligned}
& & \boldsymbol{z} &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d) \\
& & \boldsymbol{m}_0 &= g_{\theta'}(\boldsymbol{z}) \\
\boldsymbol{v}_1 &\sim \mathrm{Cat}(\sigma(\boldsymbol{p}_1)), & (\boldsymbol{p}_1, \boldsymbol{m}_1) &= f_\theta(\boldsymbol{m}_0, \mathbf{0}) \\
\boldsymbol{v}_2 &\sim \mathrm{Cat}(\sigma(\boldsymbol{p}_2)), & (\boldsymbol{p}_2, \boldsymbol{m}_2) &= f_\theta(\boldsymbol{m}_1, \boldsymbol{v}_1) \\
& \vdots & & \vdots \\
\boldsymbol{v}_T &\sim \mathrm{Cat}(\sigma(\boldsymbol{p}_T)), & (\boldsymbol{p}_T, \boldsymbol{m}_T) &= f_\theta(\boldsymbol{m}_{T-1}, \boldsymbol{v}_{T-1})
\end{aligned}
$$

$\boldsymbol{z}$ = latent, $\boldsymbol{m}$ = memory, f_θ = Cell, $\boldsymbol{p}$ = probability,
σ: softmax, Cat: categorical distribution, $\boldsymbol{v}$ = vertex.

Generator & Discriminator (2)

Problem in sampling the next node

Sampling from a categorical distribution is a **non-differentiable** operation it blocks the flow of gradients and precludes backpropagation.

Solution

Straight-Through Gumbel estimator [Jang et al., 2016]:

- $\boldsymbol{v}_t^* = \sigma((\boldsymbol{p}_t + \boldsymbol{g})/\tau)$, where τ is a temperature parameter, and g_i are iid samples from a Gumbel distribution with zero mean and unit scale.
- The next sample is chosen as $\boldsymbol{v}_t = \text{onehot}(\arg\max \boldsymbol{v}_t^*)$.

Generator & Discriminator (3)

Discriminator

The discriminator D is based on the standard LSTM architecture:

- At every time step t, a one-hot vector $\boldsymbol{v}_t$, denoting the node at the current position, is fed as input.
- After processing the entire sequence of T nodes, the discriminator outputs a single score that represents the probability of the random walk being real.

Assembling the Adjacency Matrix

We use the generator G to construct a score matrix $\boldsymbol{S}$ of transition counts (how often an edge appears in the set of generated random walks). We need to convert $\boldsymbol{S}$ into a binary adjacency matrix $\hat{\boldsymbol{A}}$.

1. $\boldsymbol{S}$ is symmetrized by setting $s_{ij} = s_{ji} = \max\{s_{ij}, s_{ji}\}$.
2. We ensure the every node i has at least one edge by sampling a neighbor j with probability $p_{ij} = \frac{s_{ij}}{\sum_v s_{iv}}$.
3. Repeat until we reach the desired amount of edges as the original graph.

No guaranteed to produce a fully connected graph.

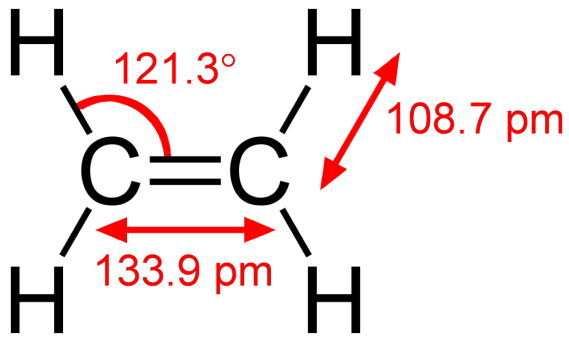
Reflection into our research

Open question

How to apply this work into our research of molecular generation?

Some ideas

1. Generate random walks on molecules. For example: **H-C=C-H** or **H-C-H** (2 random walks in Ethylene).
2. A walk on molecules contains edge/bond types also. It seems to be easier to distinguish real vs unreal walks. For example: **H-H-C=C** is unreal.
3. Assemble random walks into a single **valid** molecular graph.

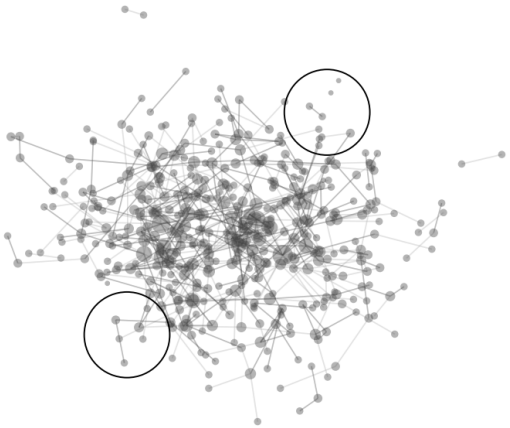


Datasets

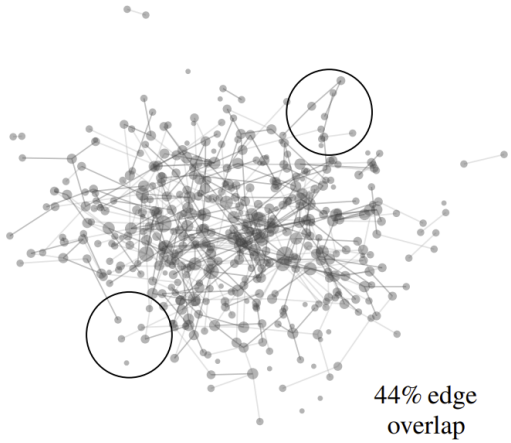
Table 2: Dataset statistics. $\mathbf{N_{LCC}}$, $\mathbf{E_{LCC}}$ - number of nodes and edges respectively in the largest connected component.

Name	$\mathbf{N_{LCC}}$	$\mathbf{E_{LCC}}$	**Reference**
CORA-ML	2,810	7,981	(McCallum et al., 2000)
CORA	18,800	64,529	(McCallum et al., 2000)
CITESEER	2,110	3,757	(Sen et al., 2008)
PUBMED	19,717	44,324	(Sen et al., 2008)
DBLP	16,191	51,913	(Pan et al., 2016)
POL. BLOGS	1,222	16,714	(Adamic & Glance, 2005)

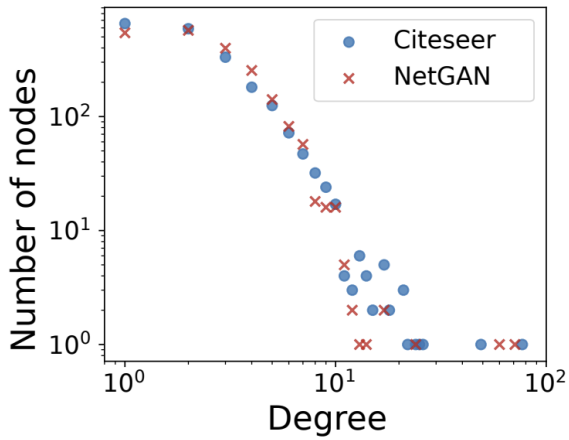
Experiments (1)



(a) Original graph



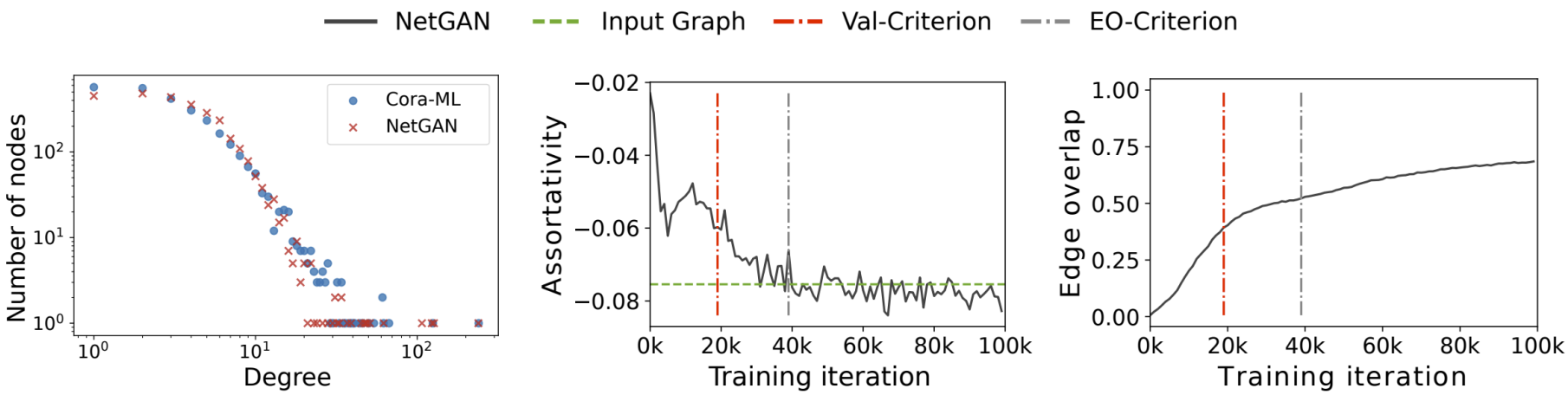
(b) Graph generated by NetGAN



(c) Degree distribution comparison

Figure 1: (a) Subgraph of the CITESEER network and (b) the respective subset of the graph generated by NetGAN. Both have similar structure but are not identical. (c) shows that the degree distributions of the two graphs are very close.

Experiments (2)



(a) Degree distribution

(b) Assortativity over training iterations

(c) Edge overlap (EO) over training iterations

Figure 3: Properties of graphs generated by NetGAN trained on CORA-ML.

Experiments (3)

Table 1: Statistics of CORA-ML and the graphs generated by NetGAN and the baselines, averaged over 5 trials. NetGAN closely matches the input networks in most properties, while other methods either deviate significantly in at least one statistic or overfit. * indicates values for the conf. model that by definition exactly match the original.

Graph		**Max. degree**	**Assort-ativity**	**Triangle count**	**Power law exp.**	**Inter-comm. unity density**	**Intra-comm. unity density**	**Cluster-ing coeff.**	**Charac. path len.**	**Average rank**
CORA-ML		240	-0.075	2,814	1.860	4.3e-4	1.7e-3	2.73e-3	5.61	
Conf. model	(1% EO)	*	-0.030	322	*	1.6e-3	2.8e-4	3.00e-4	4.38	7.50
Conf. model	(52% EO)	*	-0.051	626	*	9.8e-4	9.9e-4	6.10e-4	4.46	5.83
DC-SBM	(11% EO)	165	-0.052	1,403	1.814	6.7e-4	1.2e-3	3.30e-3	5.12	3.36
ERGM	(56% EO)	243	-0.077	2,293	1.786	6.9e-4	1.2e-3	2.17e-3	4.59	2.88
BTER	(2.2% EO)	199	0.033	3,060	1.787	1.0e-3	7.5e-4	4.62e-3	4.59	4.75
VGAE	(0.3% EO)	13	-0.009	14	1.674	1.4e-3	3.2e-4	1.17e-3	5.28	5.88
NetGAN VAL	(39% EO)	199	-0.060	1,410	1.773	6.5e-4	1.3e-3	2.33e-3	5.17	3.00
NetGAN EO	(52% EO)	233	-0.066	1,588	1.793	6.0e-4	1.4e-3	2.44e-3	5.20	1.75

Experiments (4)

Table 3: Link prediction performance (in %).

Method	CORA-ML		CORA		CITESEER		DBLP		PUBMED		POLBLOGS	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
Adamic/Adar	92.16	85.43	93.00	86.18	88.69	77.82	91.13	82.48	84.98	70.14	85.43	92.16
DC-SBM	**96.03**	95.15	98.01	97.45	94.77	93.13	**97.05**	96.57	**96.76**	95.64	95.46	**94.93**
node2vec	92.19	91.76	**98.52**	**98.36**	95.29	94.58	96.41	96.36	96.49	95.97	85.10	83.54
VGAE	95.79	**96.30**	97.59	97.93	95.11	96.31	96.38	**96.93**	94.50	**96.00**	93.73	94.12
NetGAN (500K)	94.00	92.32	82.31	68.47	95.18	91.93	82.45	70.28	87.39	76.55	95.06	94.61
NetGAN (100M)	95.19	95.24	84.82	88.04	**96.30**	**96.89**	86.61	89.21	93.41	94.59	**95.51**	94.83

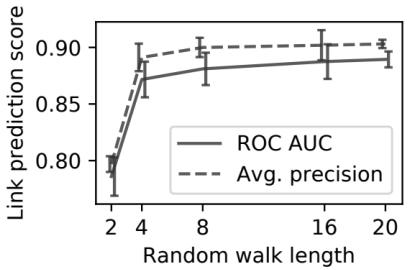


Figure 6: Effect of the random walk length T on the performance.

Paper 2

A step towards neural genome assembly (NeurIPS 2020 workshop)
Lovro Vrček, Petar Veličković, Mile Šikić
`https://arxiv.org/abs/2011.05013`

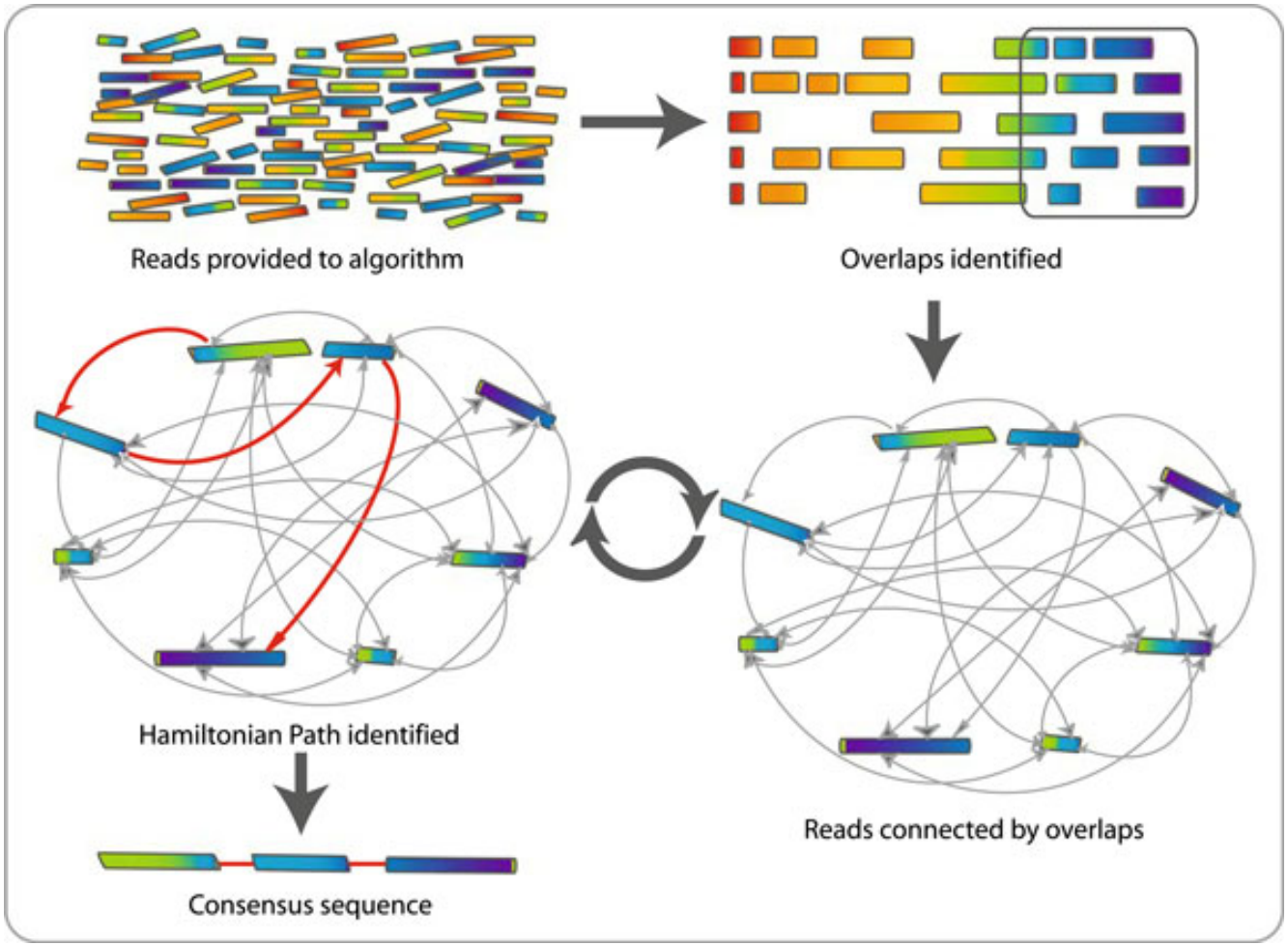
Problem definition

De novo

De novo genome assembly is on of the most difficult problems in bioinformatics:

- To **reconstruct the original genome sequence** from hundreds of thousands of relatively short sequences called **reads**, without any knowledge of what the original sequence looks like.
- Applications in biology and precision medicine.
- The problem can be reduced to finding a **Hamiltonian path** through a complex directed graph $\rightarrow$ NP-complete problem.

Overlap-Layout-Consensus (OLC) paradigm (1)



Computational Biology Methods and Their Application to the Comparative Genomics of Endocellular Symbiotic Bacteria of Insects

Jennifer Commins, Christina Toft, Mario A Fares

Overlap-Layout-Consensus (OLC) paradigm (2)

OLC paradigm:

1. **Overlap phase**: Mutually overlap all the reads in the sample.
   - Reads which are contained in the others are discarded.
   - Build the assembly graph: nodes = reads, edges = overlaps.
2. **Layout phase:** A path approximating the genome has to be found.
   - Detect and remove redundance in reads and false overlaps due to sequencing errors.
   - OLC-based SOTA assemblers Raven: transitive edges, tips and bubbles.
3. **Consensus phase:** The assembly sequence is compared to the reads in the sample and polishing of the sequence is performed.

Overlap-Layout-Consensus (OLC) paradigm (3)

Simplification algorithms:

1. **Transitive reduction:** Given a triplet of nodes (n_i, n_j, n_k), edge $n_i \rightarrow n_k$ is transitive if there exists edges $n_i \rightarrow n_j$ and $n_j \rightarrow n_k$.
2. **Tips reduction:** Removal of tips or dead ends – branches of several nodes that stem from the main chain and then cease abruptly.
3. **Bubbles reduction:** If there are multiple paths connecting two nodes, only one of these paths can be retained.

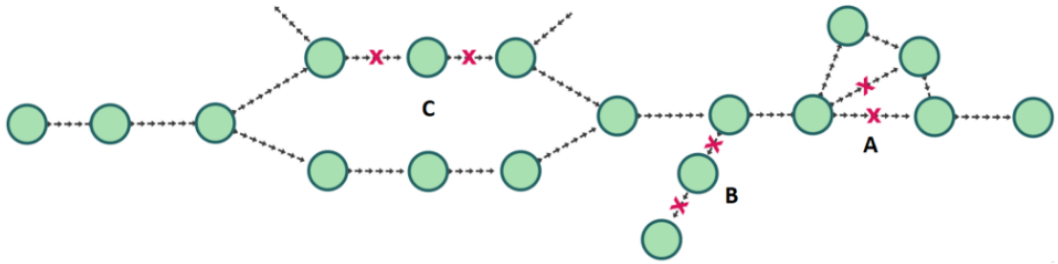


Figure 1: Example of structures in the assembly graph, before all the simplification steps. Letter **A** marks transitive edges, a short tip is marked with **B**, and a bubble which cannot be fully resolved is marked with **C**. Red crosses show which edges can be removed from the assembly graph.

Neural execution of simplification algorithms

Proposal

Use GNN/MPNN model to learn these simplification algorithms.

My opinions

- All three of the algorithms are graph traversal problems. I don't find it necessary to learn (approximate) these algorithms.
- Using GNN/MPNN might be not correct and computationally expensive.
- The real issue is to scale the computation.

Note: I skip the details of GNN/MPNN here.