Learning to Untangle Genome Assembly with Graph Convolutional Networks

Xavier Bresson
https://twitter.com/xbresson

Department of Computer Science
National University of Singapore (NUS)

Joint work with L. Vrček (GIS), T. Laurent (LMU)
M. Schmitz (GIS) and M. Šikić (GIS)

Workshops

Programs > Workshops > Artificial Intelligence and Discrete Optimization

Artificial Intelligence and Discrete Optimization

FEBRUARY 27 - MARCH 3, 2023

Overview

In recent years, the use of Machine Learning techniques to Operations Research (OR) problems, especially in the Discrete Optimization (DO) a.k.a. Combinatorial Optimization context, opens very interesting scenarios because DO is the "home" of an endless list of decision-making problems that are of fundamental importance in multitude applications.

The workshop will bring together experts in mathematics (optimization, graph theory, sparsity, combinatorics, statistics), operations research (assignment problems, routing, planning, Bayesian search, automation, scheduling), machine learning (deep learning, supervised, self-supervised and reinforcement learning) and artificial intelligence at large (including multi-agent systems, interpretability, fairness, etc.). In addition, the focus will be on

- Algorithmic challenges and potential of the interaction between AI and OR;
- Data requirements in which such an interaction can be profitable; and
- Application areas that are likely to lead to game-changing results (e.g., transportation, supply chain, public policy, energy).

IPAM Workshop on
Artificial Intelligence and
Discrete Optimization
Feb 27th 2023

Outline

- Genome Assembly
- Assembly Graphs
- Path Assembly
- Our Contribution
- Dataset
- Edge Prediction
- Graph Neural Networks
- Graph Decoding
- Numerical Experiments
- Conclusion

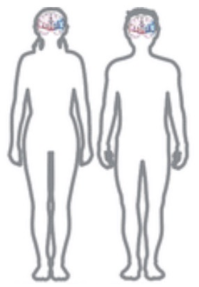
Outline

- Genome Assembly
- Assembly Graphs
- Path Assembly
- Our Contribution
- Dataset
- Edge Prediction
- Graph Neural Networks
- Graph Decoding
- Numerical Experiments
- Conclusion

Genome Assembly

- What is the task?

Human



Biological sample

$\Rightarrow$

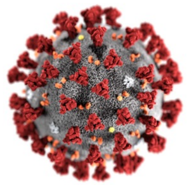
Genome Assembly

$\Rightarrow$

TGACATGGGTACACATGACGGG
||||||||||||||||||||||
ACTGTACCCATGTGTACTGCCC

3 billion of base pairs
(A-T, C-G)

Covid-19 virus



ATGGGTAC
||||||||
TACCCATG

30 thousand of base pairs

Human, animals, plants, mushroom, microbes (almost all living organisms)

Genome Sequencing Machine

Genome sequence (both strands)

Genome Assembly

- Why is this problem important?
  - Genome is the molecular code of life.
    - It is a set of instructions for the organism to develop, function and sustain.
    - Understanding the genome is critical to fight diseases.
- A quest to construct the first complete human genome started in 1990.
  - First result in 2001 but ≈210 gaps[1].
  - In 2022, 32 years later, the quest was finally achieved[2].

2001

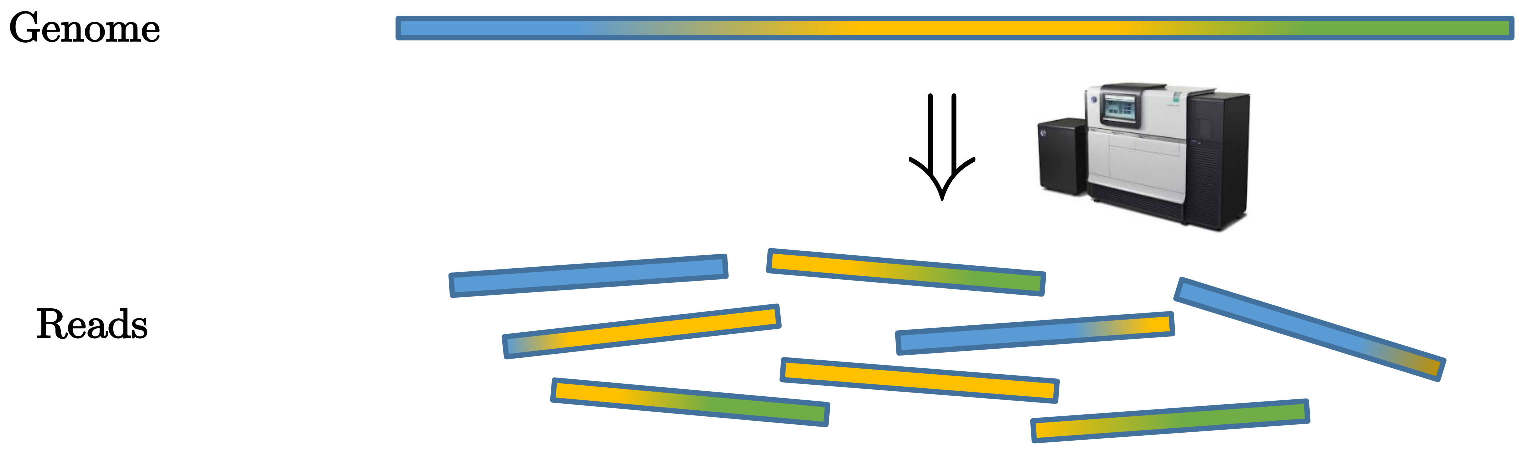
$\Rightarrow$

2022

[1] Lander et-al, Initial sequencing and analysis of the human genome, Nature 2001
[2] Nurk et-al, The complete sequence of a human genome, Science 2022

Genome Sequencing Machine

- No machine can copy the complete genome sequence in one-shot (genome breaks).
- Machines produce a collection of genome sub-sequences called reads.
- Modern machines aims at getting long reads with minimum base pair errors (A-T, G-C).
  - PacBio HiFi reads[1] : 15,000-25,000 base pairs in average with 0.5% error (≈100 errors/read)
  - Oxford Nanopore reads[2] : 50,000-100,000 base pairs in average with 5% error (≈4000 errors/read).
- Coverage depth : Each base is covered by a number of reads (typically 30 reads).

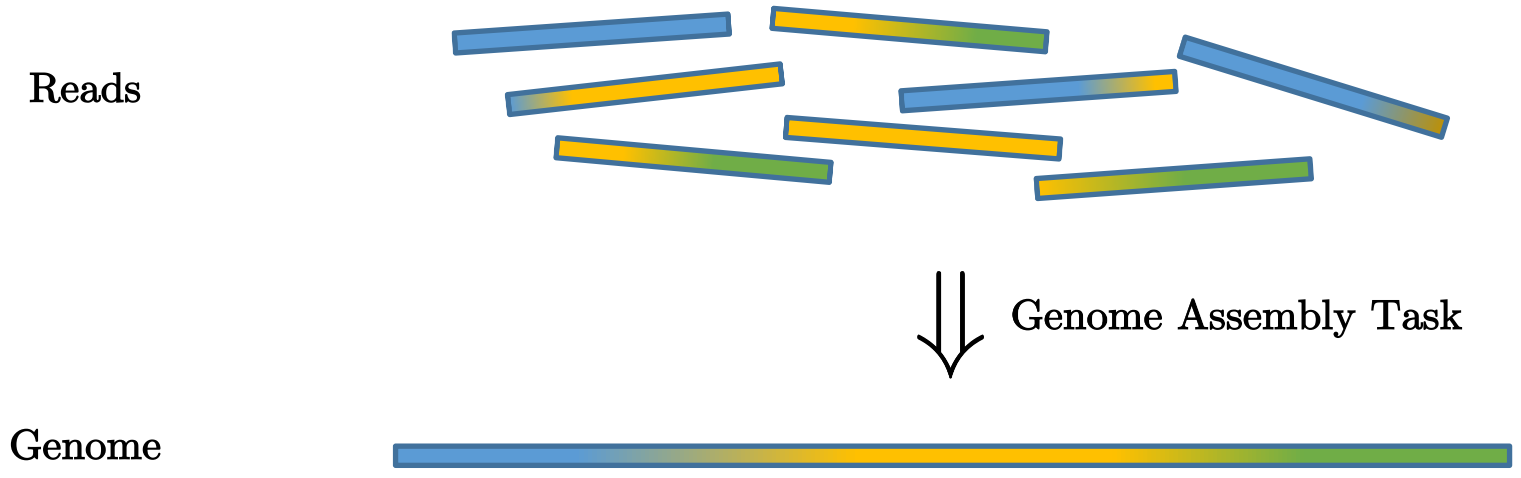


[1] Eid et-al, Real-time DNA sequencing from single polymerase molecules, Science 2009
[2] Clarke et-al, Continuous base identification for single-molecule nanopore DNA sequencing, Nature nanotechnology 2009

Genome Assembly Problem

- Combinatorial problem : Re-order overlapping reads to form the longest sequence.
  - This problem is NP-hard because complexity is O(n!), n being the number of reads.
  - n = 3B(genome len) / 20k(read len) . 30(depth) . 2(strands) . 2(haploids)

    = 18M reads

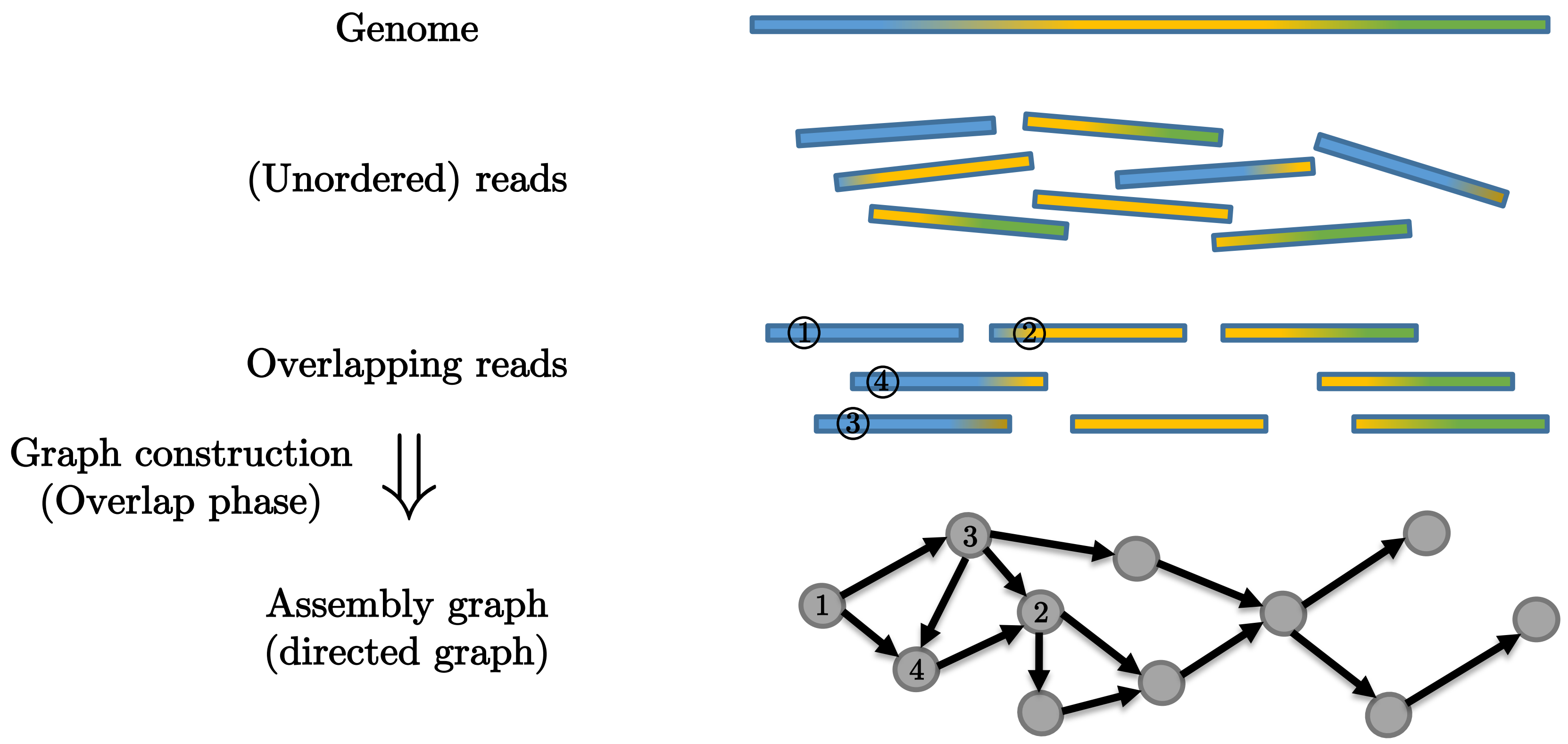


Outline

- Genome Assembly
- Assembly Graphs
- Path Assembly
- Our Contribution
- Dataset
- Edge Prediction
- Graph Neural Networks
- Graph Decoding
- Numerical Experiments
- Conclusion

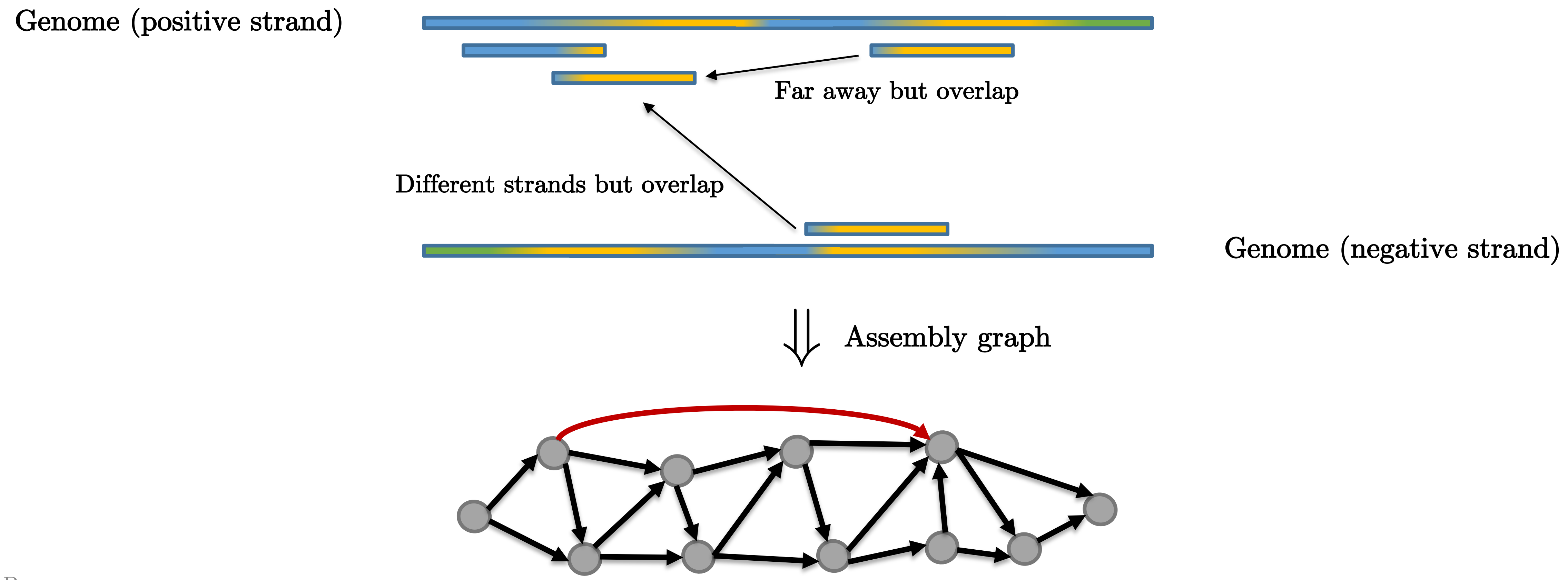
Assembly Graphs

- Reducing complexity : Only assemble from overlapping reads.
  - Overlapping reads form a graph called the assembly graph.
  - The construction of the assembly graph is called the overlap phase in genomics.



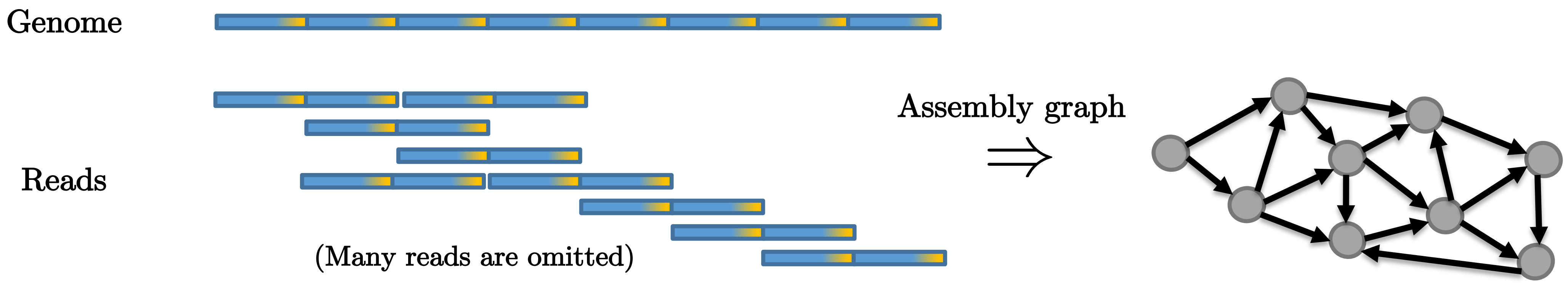
Challenges

- Interspersed regions
  - Far away reads on the genome can still overlap.
  - Reads on different strands/chromosomes/haploids can overlap.



Challenges

- Major challenge : Segment duplication
  - Some genome regions contain (lots of) repetitive patterns that are not covered by a single read.
  - These regions produce complex genome regions which are (very) hard to disentangle.
  - To this date, no genome assemblers can solve this issue.
    - We are left with fragments of genome, called contigs.
  - Solutions can come either from longer reads (better sequencing machines) or better algorithms.



Approximate Assembly Graphs

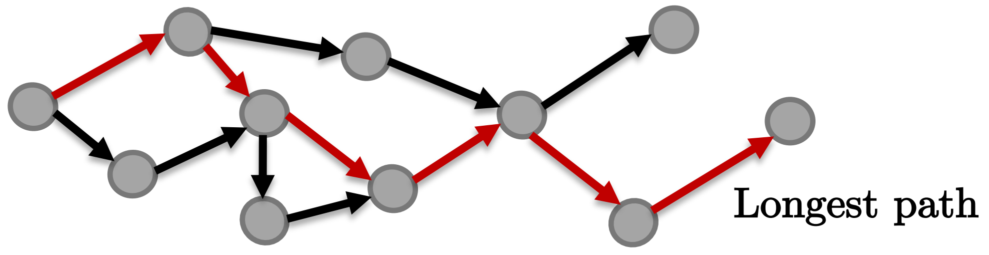
- Exact construction of assembly graph has O(n^2d^2) complexity with n number of reads and d dimension of reads.
  - For n=18M, d=20k, it would take with a GPU ≈ 3 hours(convolution) + 3 months(transfer).
- Approximation of assembly graphs is required.
  - There are as many approximations as the number of genome assemblers.
  - Genome assembler usually designs a graph constructer for each specific type of reads.
- In summary : Sequencing errors + interspersed/duplicated regions + approximate graph construction make the topology of the assembly graph challenging with multiple disconnected components, cycles, dead-ends, bubbles, transitive edges and tangles.

Outline

- Genome Assembly
- Assembly Graphs
- **Path Assembly**
- Our Contribution
- Dataset
- Edge Prediction
- Graph Neural Networks
- Graph Decoding
- Numerical Experiments
- Conclusion

Path Assembly

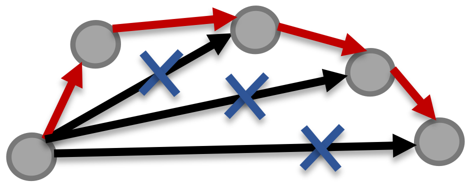
- Extracting the genome on the assembly graph reduces to solve a path routing problem on graph.
  - The problem is equivalent to the longest path problem on graphs, i.e. finding the longest path that visits each node at most once (avoiding cycles).
  - Once the longest path is found, the genome sequence is reconstructed by collating the overlapping reads along the path.
  - This decoding step is called the layout phase in genomics.
- Besides, assembly graphs are also composed of disconnected components due to approximated/error constructions.
  - This makes impossible the reconstruction of the whole genome with a single path.
  - There exist paths that reconstruct fragments of genome (contigs).
  - Existing assemblers aims at extracting the best set of paths in terms of length and reconstruction quality of contigs.

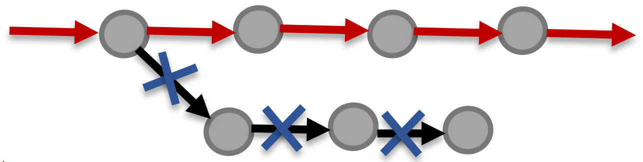


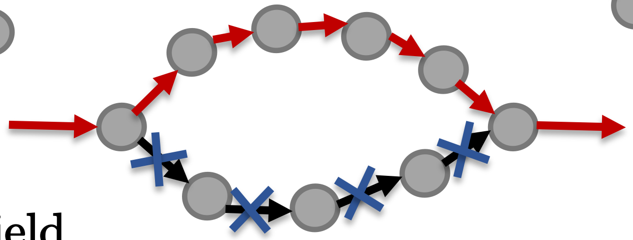
Assembly graph

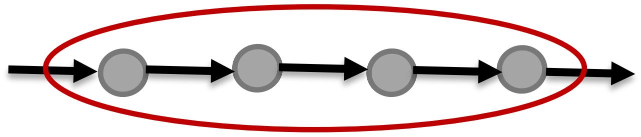
Raven[1] Genome Assembler

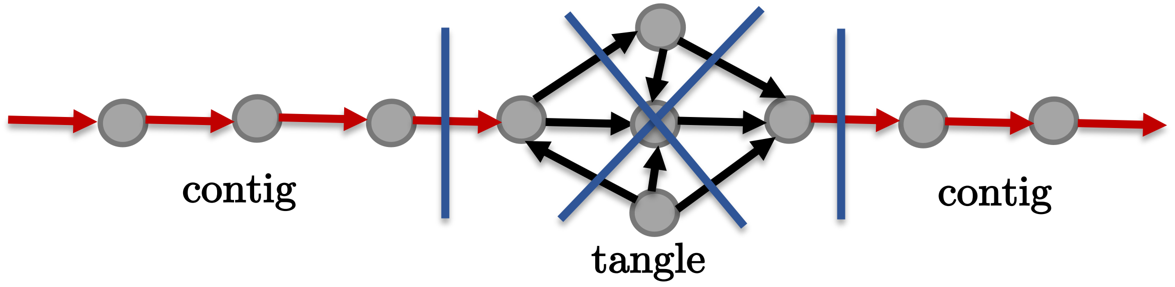
- Genome assemblers rely on human engineered heuristics that aim at simplifying the assembly graph into a set of paths representing the contigs.
- Raven's heuristics :
  - Remove transitive edges
  - Remove dead-ends
  - Remove bubbles
  - Collapse sequences into unitigs
  - Remove long connections using force directed field (FDL), a dimensionality reduction technique
  - Cut tangles and get contigs

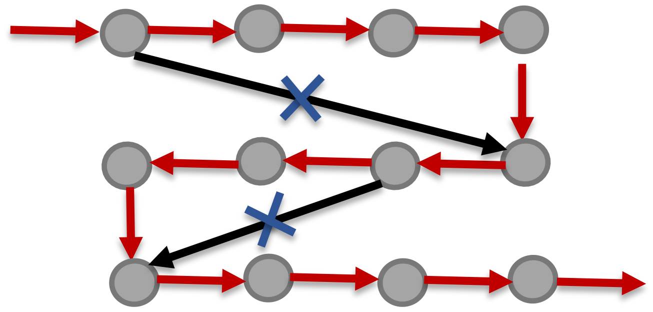












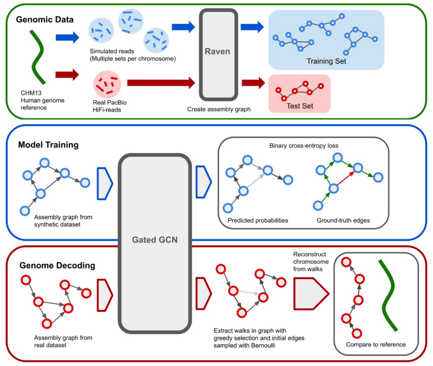
[1] Vaser, Sikic, Time-and memory-efficient genome assembly with raven, Nature Computational Science 2021

Outline

- Genome Assembly
- Assembly Graphs
- Path Assembly
- **Our Contribution**
- Dataset
- Edge Prediction
- Graph Neural Networks
- Graph Decoding
- Numerical Experiments
- Conclusion

State-of-the-Art[1]

- In 2022, the first gapless compete human genome sequence was achieved.
- What enabled this success?
  - Modern sequencing machines with longer and more accurate reads (PacBio/Nanopore).
  - Combination of multiple genome assemblers (w/ human engineered heuristics).
  - Experts perform manual inspection to resolve tangles and assemble the contigs.
- Limitation
  - Time and resource consuming (1.5 years and a large team of scientists)
  - Not generalizable
- What do we propose?
  - ML paradigm : Use deep learning to reduce/replace human heuristics $\Rightarrow$ AI-based genome assembler
  - Advantage : Solve genome assembly independently of any type of sequencing machine and no hand-crafting of genome assemblers.


Genomic Data
CHM13 Human genome reference
Simulated reads (Multiple sets per chromosome)
Real PacBio HiFi-reads
Raven
Create assembly graph
Training Set
Test Set
Model Training
Assembly graph from synthetic dataset
Gated GCN
Binary cross-entropy loss
Predicted probabilities
Ground-truth edges
Genome Decoding
Assembly graph from real dataset
Extract walks in graph with greedy selection and initial edges sampled with Bernoulli
Reconstruct chromosome from walks
Compare to reference

[1] Nurk et-al, The complete sequence of a human genome, Science 2022

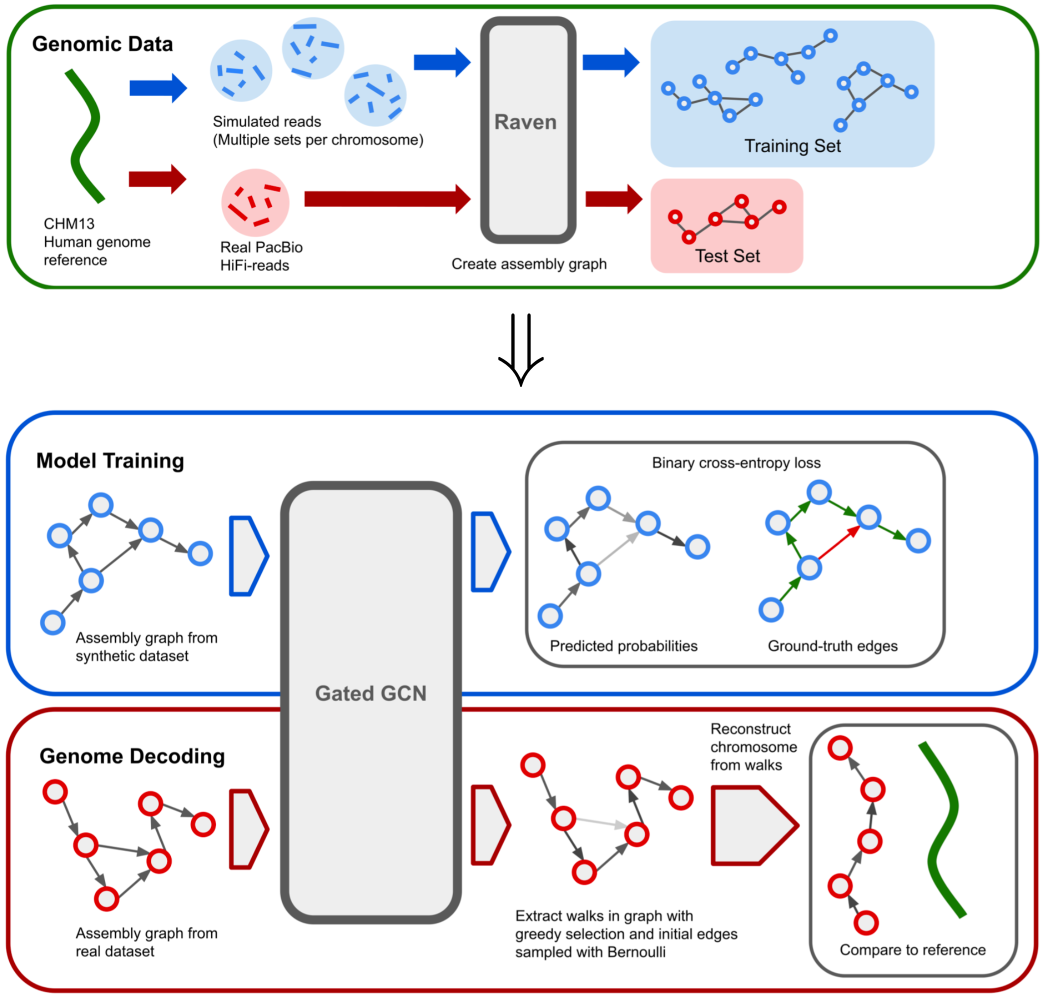
Scope of our Work

- In this work, we focus on the layout part (path extraction) of the genome assembly problem.
  - In other words, we use an existing graph assembler (computed in this project by Raven's overlap phase[1]), learn to extract long fractions of the genome, i.e. the contigs.
- We do not consider the task of graph construction which quality is obviously critical to extract the longest possible contigs.
  - Exiting graph constructers are hand-engineered for different types of reads.

[1] Vaser, Sikic, Time-and memory-efficient genome assembly with raven, Nature Computational Science 2021

Machine Learning Framework

- We propose to learn to untangle assembly graphs and reconstruct genome sequences.



Genomic Data
CHM13 Human genome reference
Simulated reads (Multiple sets per chromosome)
Real PacBio HiFi-reads
Raven
Create assembly graph
Training Set
Test Set
Model Training
Assembly graph from synthetic dataset
Gated GCN
Binary cross-entropy loss
Predicted probabilities
Ground-truth edges
Genome Decoding
Assembly graph from real dataset
Extract walks in graph with greedy selection and initial edges sampled with Bernoulli
Reconstruct chromosome from walks
Compare to reference

Overview of our framework

Outline

- Genome Assembly
- Assembly Graphs
- Path Assembly
- Our Contribution
- Dataset
- Edge Prediction
- Graph Neural Networks
- Graph Decoding
- Numerical Experiments
- Conclusion

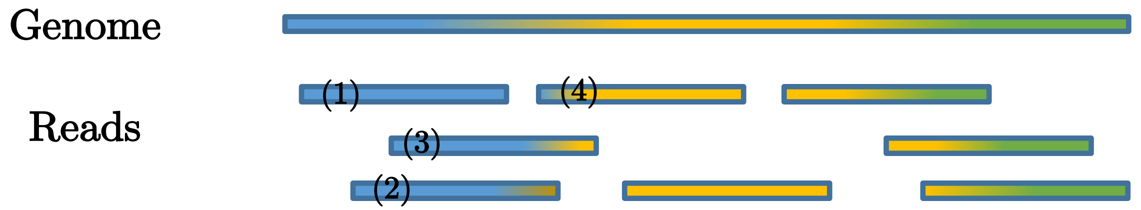
Human Genomic Dataset

- We use the 2022 CHM13 human genome sequence[1] (one female haploid, 23 chromosomes, two strands) of 3.3 billion base pairs length and a set of 5.6 million PacBio HiFi reads.
- We contribute to the dataset in two ways.
    - We correct the read errors from sequencing with hifiasm[2].
    - We map the reads to the genome sequence with minimap2[3] and resolves any gap by re-assigning similar reads while preserving the sequence.
- In this first approach, we work with individual chromosomes, not the whole genome, as there exists so far only one clean reconstructed human genome.

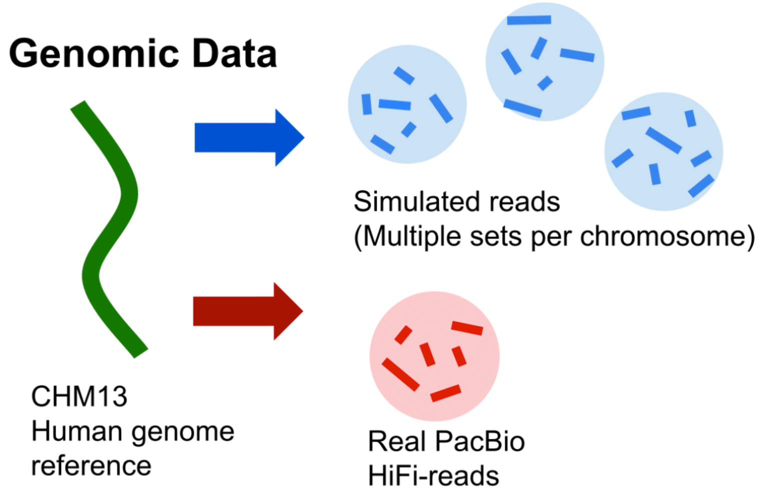
[1] Nurk et-al, The complete sequence of a human genome, Science 2022
[2] Cheng et-al, Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm, Nature methods 2021
[3] Li, Minimap2: pairwise alignment for nucleotide sequences, Bioinformatics 2018

Data Augmentation

- Data augmentation is critical to reduce overfitting and better generalization.
- We use a simulator[1] of reads with the constraint that the distribution of synthetic read lengths follows the distribution of real read lengths.
  - This allows us to simulate an arbitrary number of train/validation assembly graphs.
- We label the reads with a positional information corresponding to the ordering of the reads on the genome sequence.
  - Read positions serve as labels to train a network to reconstruct the genome exactly.
- In this work, we simulate individual chromosomes (not the whole genome).



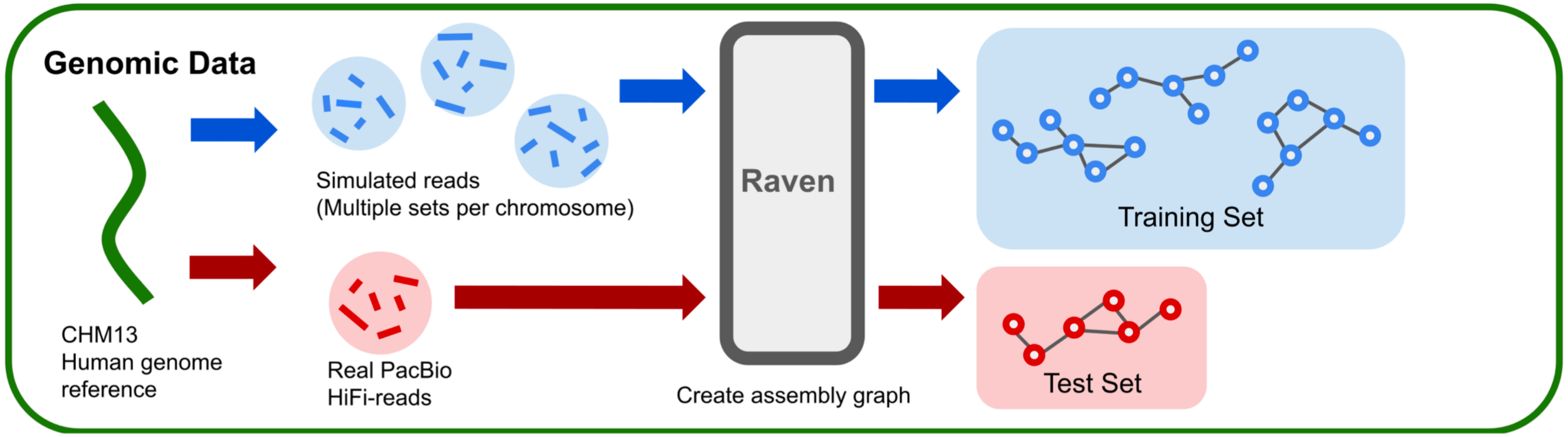
(k) corresponds to the position of the read on the genome sequence.



[1] Maryland Bioinformatics Labs, seqrequester

Assembly Graph Construction

- Raven's overlap phase[1] will be used to compute assembly graphs.
- It is composed of two steps :
  - Dimensionality reduction step : From 20k-dim reads to 512-dim "words" (hand-crafted process that identifies "words", repetitive patterns of "character" bases).
  - Pairwise matching step : Use the longest common subsequence algorithm with dynamic programming to compute the length of overlap between two reads.
  - Computational time : For n=50k reads, it takes 20min with Intel 6226R CPU and 30 threads.



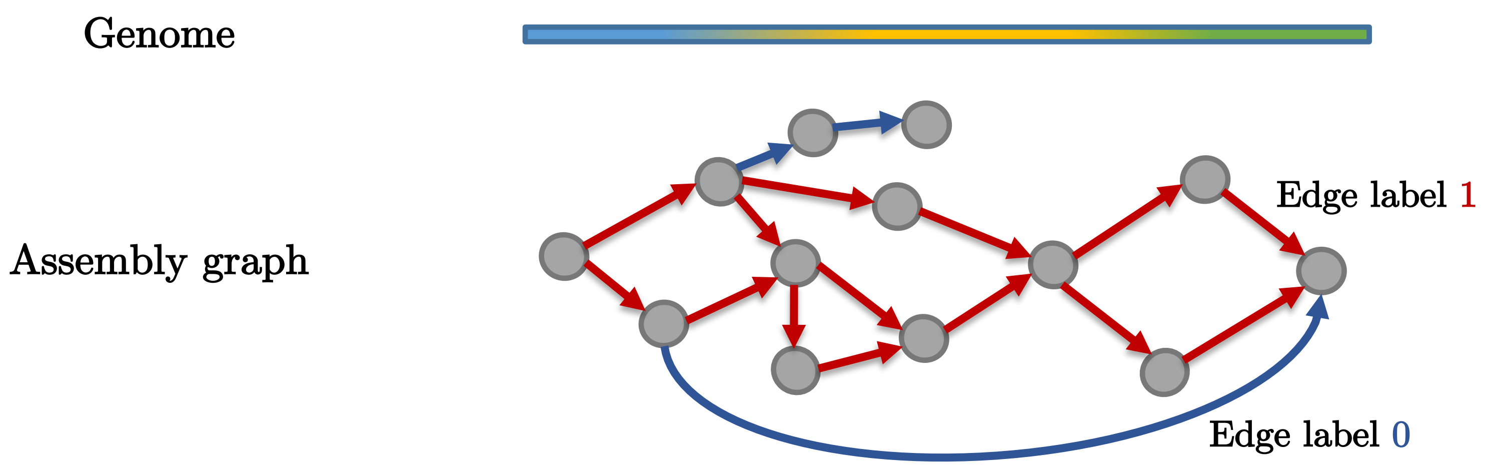
[1] Vaser, Sikic, Time-and memory-efficient genome assembly with raven, Nature Computational Science 2021

Outline

- Genome Assembly
- Assembly Graphs
- Path Assembly
- Our Contribution
- Dataset
- Edge Prediction
- Graph Neural Networks
- Graph Decoding
- Numerical Experiments
- Conclusion

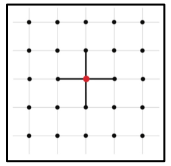
Edge Labeling

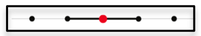
- Decoding is carried out by a path routing algorithm that follows edges that reconstruct exactly (fractions of) the genome (contigs).
- How do we get these edges?
  - They are obtained by running a depth-first search (DFS) algorithm with positional information of reads on the genome. The labeling algorithm identifies all paths/edges that lead to an optimal genome reconstruction.
  - Correct edges are labeled with value 1 and incorrect edges s.a. long-distance overlapping reads or dead-ends are assigned with value 0.
  - Note that the set of labels is unbalanced with a majority of one-value, i.e. most edges are correct but wrong edges significantly shortcut the extracted path.

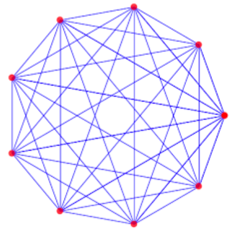


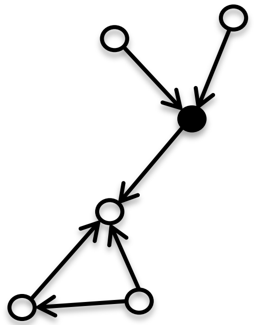
Edge Prediction

- How to predict edges that lead to optimal decoding of the genome?
  - Which network architecture do we need?
- Observe that the assembly problem is fundamentally a graph problem (overlaps of reads form a graph and decoding looks for a path in a graph).
- Can we use CNNs[1], RNNs[2] or Transformers[3]?
  - CNNs only work for grids, not for graphs.
  - RNNs only work for sequences, not for graphs.
  - Transformers only work for fully-connected graphs, not for sparse graphs s.a. assembly graphs.
- We need graph neural networks[4,5,6] (GNNs).









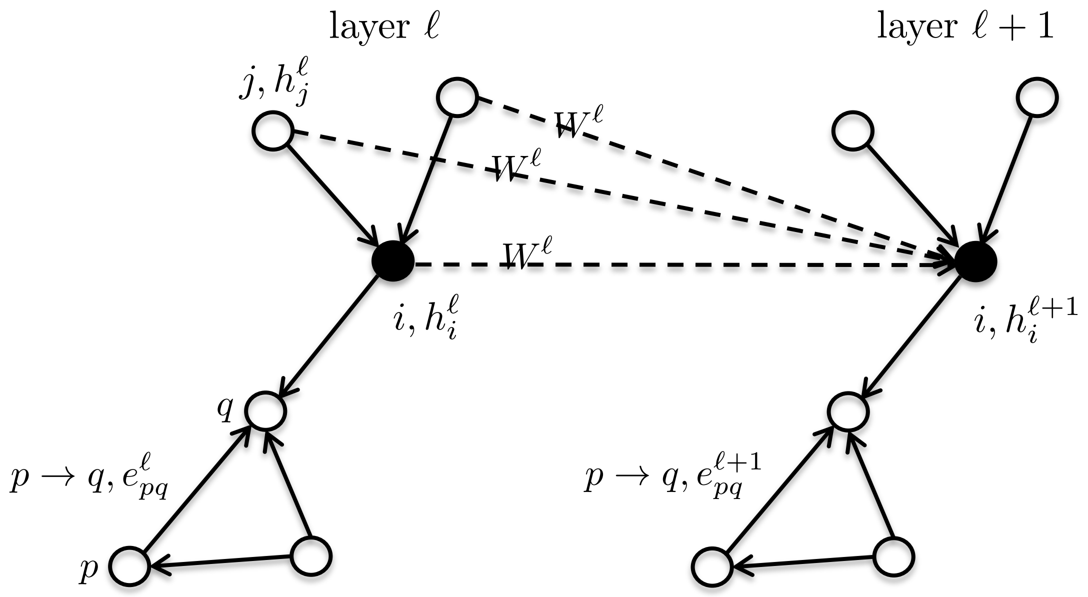
[1] LeCun, Bottou, Bengio, Haffner, Gradient-based learning applied to document recognition, 1998
[2] Hochreiter, Schmidhuber, Long short-term memory, 1997
[3] Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin, Attention is all you need, NeurIPS 2017
[4] Scarselli, Gori, Tsoi, Chung, Hagenbuchner, Monfardini, The Graph Neural Network Model, IEEE Transactions on Neural Networks 2009
[5] Bruna, Zaremba, Szlam, LeCun, Spectral Networks and Locally Connected Networks on Graphs, ICLR 2014
[6] Defferrard, Bresson, Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, NeurIPS 2016

Outline

- Genome Assembly
- Assembly Graphs
- Path Assembly
- Our Contribution
- Dataset
- Edge Prediction
- **Graph Neural Networks**
- Graph Decoding
- Numerical Experiments
- Conclusion

Designing GNNs for Assembly Graphs

- What network properties?
  - Invariant/equivariant layers
  - Independent of the size of neighborhoods and graphs
  - Anisotropic convolution on graphs
  - Directed local reception field
  - Deep architecture
  - Break node anonymity of assembly graphs



$$h^{\ell+1} = f_{\text{node}}(A, h^\ell, e^\ell) \quad \text{Node features}$$

$$e^{\ell+1} = f_{\text{edge}}(A, h^{\ell+1}, e^\ell) \quad \text{Edge features}$$

A is the adjacency matrix

GatedGCNs[1] for Assembly Graphs

- We propose the following graph network to learn expressive representation of the graph assembly :

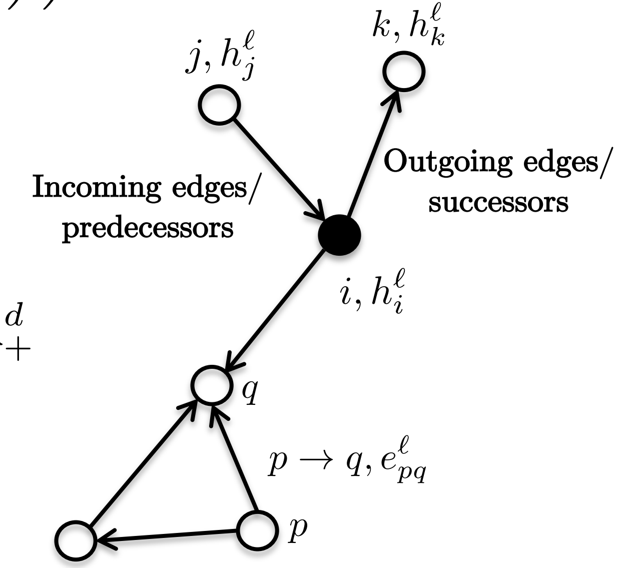
$$h_i^{l+1} = h_i^l + \mathrm{ReLU}\Big(\mathrm{BN}\Big(A_1^l h_i^l + \sum_{j\to i} \eta_{ji}^{f,l+1} \odot A_2^l h_j^l + \sum_{i\to k} \eta_{ik}^{b,l+1} \odot A_3^l h_k^l\Big)\Big) \in \mathbb{R}^d$$

$$e_{pq}^{l+1} = e_{pq}^l + \mathrm{ReLU}\Big(\mathrm{BN}\Big(B_1^l e_{pq}^l + B_2^l h_p^l + B_3^l h_q^l\Big)\Big) \in \mathbb{R}^d$$

  with the directed edge gates :

$$\eta_{ji}^{f,l} = \frac{\sigma\left(e_{ji}^l\right)}{\sum_{j'\to i}\sigma\left(e_{j'i}^l\right)+\epsilon} \in \mathbb{R}_+^d, \qquad \eta_{ik}^{b,l} = \frac{\sigma\left(e_{ik}^l\right)}{\sum_{i\to k'}\sigma\left(e_{ik'}^l\right)+\epsilon} \in \mathbb{R}_+^d$$

  where all A, B ∈ $\mathrm{R}^{d\times d}$ are learnable parameters, BN for batch normalization, $\odot$ for Hadamard product and σ is the sigmoid function.



- Anisotropic diffusion process (Perona-Malik's anisotropic PDE[2] generalized to graphs).
  - Directed edge gates can be seen as dense attention operators on graphs (actually dense attention can perform better on graphs than sparse attention[3]).

[1] Bresson, Laurent, Residual gated graph convnets, ICLR 2017
[2] Perona, Malik, Scale-space and edge detection using anisotropic diffusion, 1987
[3] Dwivedi, Bresson, A generalization of transformer networks to graphs, AAAI 2021

GatedGCNs[1] for Assembly Graphs

- This model is permutation equivariant (invariant by node re-indexing).

$$f_{\text{node}}(PA, Ph, Pe) = Pf_{\text{node}}(A, h, e)$$
$$f_{\text{edge}}(PA, Ph, Pe) = Pf_{\text{edge}}(A, h, e)$$

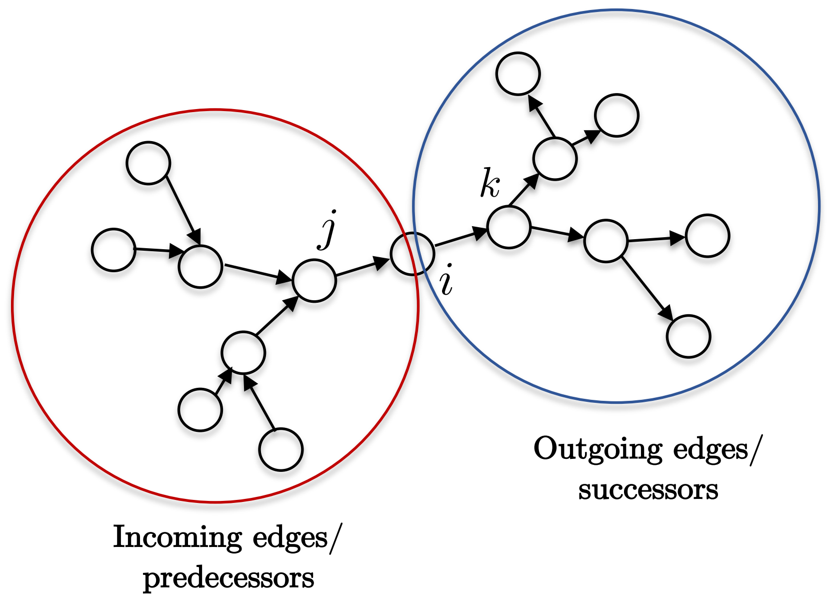
  where P is a permutation matrix.

- Independent of the size of neighborhoods and graphs (distributed computing).
  - GNN libraries s.a. DGL[2] or PyG[3]

[1] Bresson, Laurent, Residual gated graph convnets, ICLR 2017
[2] Wang-etal, Deep graph library: Towards efficient and scalable deep learning on graphs, 2019
[3] Fey, Lenssen, Fast graph representation learning with pytorch geometric, 2019

GatedGCNs[1] for Assembly Graphs

- Directed local reception fields (allow to extend the reception fields for both the node predecessors and the node successors)



- Deep architecture with Batch Normalization and Residual Connection.

- Node anonymity can be broken with graph positional encoding (next slide).

[1] Bresson, Laurent, Residual gated graph convnets, ICLR 2017

Input Features

- Edge features : $z_{ij} \in \mathbb{R}^2$
  - Length and quality of the overlap between two reads and normalized by z-scoring.
- Node features : $x_i \in \mathbb{R}^{d_n}$
  - In this work, we do not use any node features coming from the raw reads (future work).
  - In this case, GNNs perform poorly or fail in the absence of node identity[1,2].
  - This issue can be overcome with graph positional encoding (PE) s.t. Laplacian eigenvectors[3,4] for undirected graphs.
  - For directed graphs like assembly graphs, we use a k-step PageRank[5,6] diffusion vector, along with the in-degree and out-degree, which are features invariant by re-indexing permutation (essential for generalization). In summary, we have

$$x_i = d_i^{\text{in}} \,\|\, d_i^{\text{out}} \,\|\, p_i^1 \,\|\, \cdots \,\|\, p_i^K \in \mathbb{R}^{2+K}, \text{ where } \| \text{ is concatenation}$$

$$p^{k+1} = \alpha (D^{-1}A)^T p^k + (1-\alpha)\frac{1_n}{n} \in \mathbb{R}^n, \quad p^{k=0} = \frac{1_n}{n} \in \mathbb{R}^n, \ \alpha = 0.95$$

A is the adjacency matrix and
D is the out-degree matrix.

[1] Murphy, Srinivasan, Rao, Ribeiro, Relational pooling for graph representations, ICML 2019
[2] Loukas, What graph neural networks cannot learn: depth vs width, ICLR, 2020
[3] Belkin, Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural computation 2003
[4] Dwivedi, Joshi, Laurent, Bengio, Bresson, Benchmarking graph neural networks, 2020
[5] Page, Brin, Motwani, Winograd, The pagerank citation ranking: Bringing order to the web, 1999
[6] Dwivedi, Luu, Laurent, Bengio, Bresson, Graph neural networks with learnable structural and positional representations, 2021

Graph Convolutional Layers

- Input features are projected into a higher d-dimensional space with a standard MLP :

$$h_i^0 = \text{MLP}_1(x_i) \in \mathbb{R}^d$$
$$e_{ij}^0 = \text{MLP}_2(z_{ij}) \in \mathbb{R}^d$$

- The initial node/edge features are then passed to L convolutional layers :

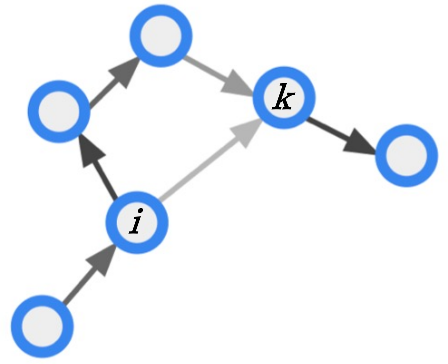
$$\begin{aligned}
&\text{for } \ell = 0, 1, \ldots, L-1 \\
&\qquad h^{\ell+1} = f_{\text{node}}(A, h^\ell, e^\ell) \in \mathbb{R}^d \\
&\qquad e^{\ell+1} = f_{\text{edge}}(A, h^{\ell+1}, e^\ell) \in \mathbb{R}^d
\end{aligned}$$

Edge Prediction Layer

- We use a MLP to predict whether a directed edge i→k can lead to an optimal decoding of the genome :

$$p_{ik} = \text{Sigmoid}\big(\text{MLP}\big(h_i^L \parallel h_k^L \parallel e_{ik}^L\big)\big) \in [0, 1]$$

with the node representations of nodes i and k, the edge representation of the directed edge i→k and L is the last GatedGCN layer.



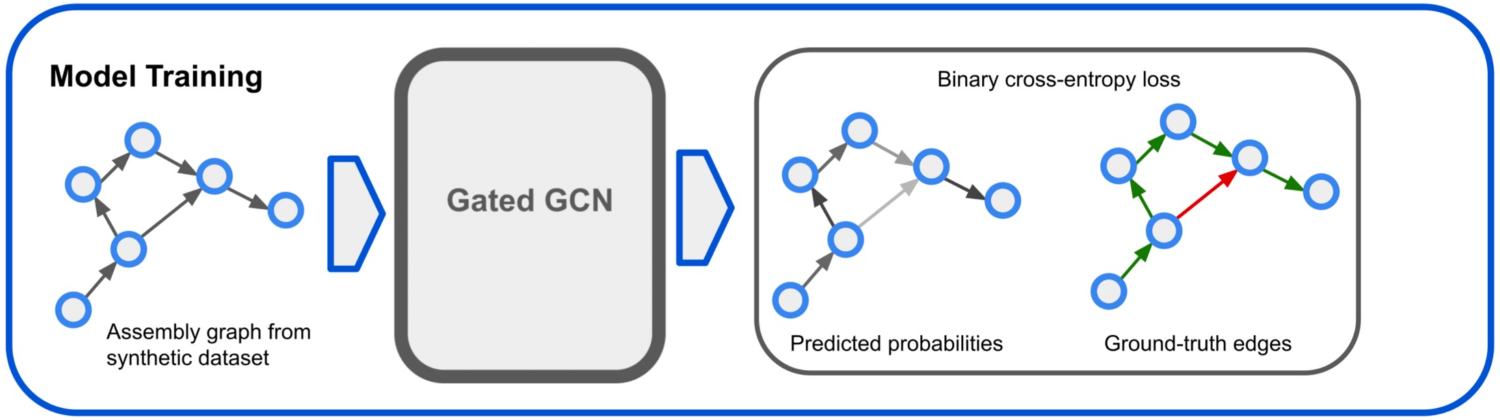
Predicted probabilities

Network Training

- Network size is 6.5M parameters with L=16 layers and d=256 hidden dimensions.
- Loss function is the binary cross-entropy using edge labels :

$$L = \frac{1}{E} \sum_{ij \in E} w_{ij} \left(\hat{p}_{ij} \log p_{ij} + (1 - \hat{p}_{ij}) \log(1 - p_{ij}) \right)$$

  where E is the set of edges, $\hat{p}_{ij}$ the ground-truth label, p_{ij} the predicted probability and w_{ij} a weight value that balances equally the number of ones and zeros in the label set.
- Optimization is done by SGD with Adam optimizer[1].



[1] Kingma, Ba, Adam: A Method for Stochastic Optimization, 2014

Training with Large Graphs

- Size of graphs (chromosomes) is [32k,184k] nodes.
    - They are too large to fit into the GPU memory.
- Graph partitioning is required.
    - We use Metis[1] clustering algorithm with a number of clusters randomly chosen in [400,600] to force different partitioning at each epoch and reduce over-fitting.

[1] Karypis, Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, scientific Computing, 1998

Outline

- Genome Assembly
- Assembly Graphs
- Path Assembly
- Our Contribution
- Dataset
- Edge Prediction
- Graph Neural Networks
- Graph Decoding
- Numerical Experiments
- Conclusion

Greedy Decoding

- We aim at solving the combinatorial path routing problem auto-regressively, i.e. selecting one node at a time (by factorizing the probability with the chain rule) :

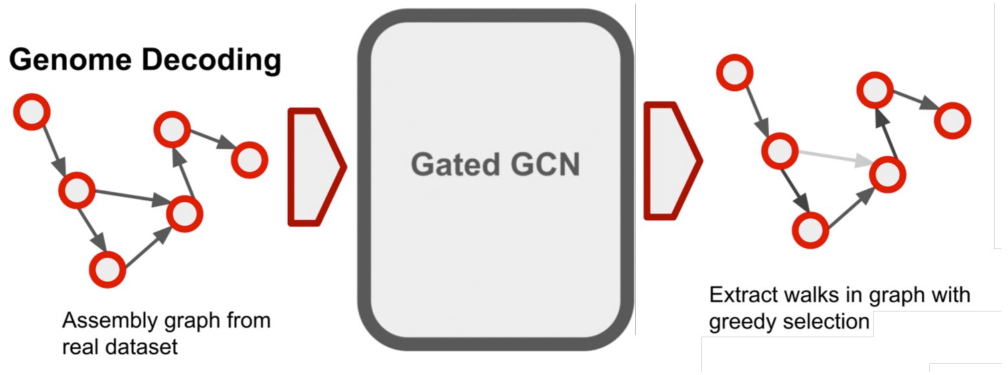
$$\max_{\text{seq}_n=\{i_1,...,i_n\}} P(\text{seq}, G)$$

$$P(i_1, ..., i_n, G) = \Pi_{t=1}^{n}\ P(i_t|i_{t-1}, i_{t-2}, ...i_1, G)$$

  where the conditional probability is estimated by the graph network.

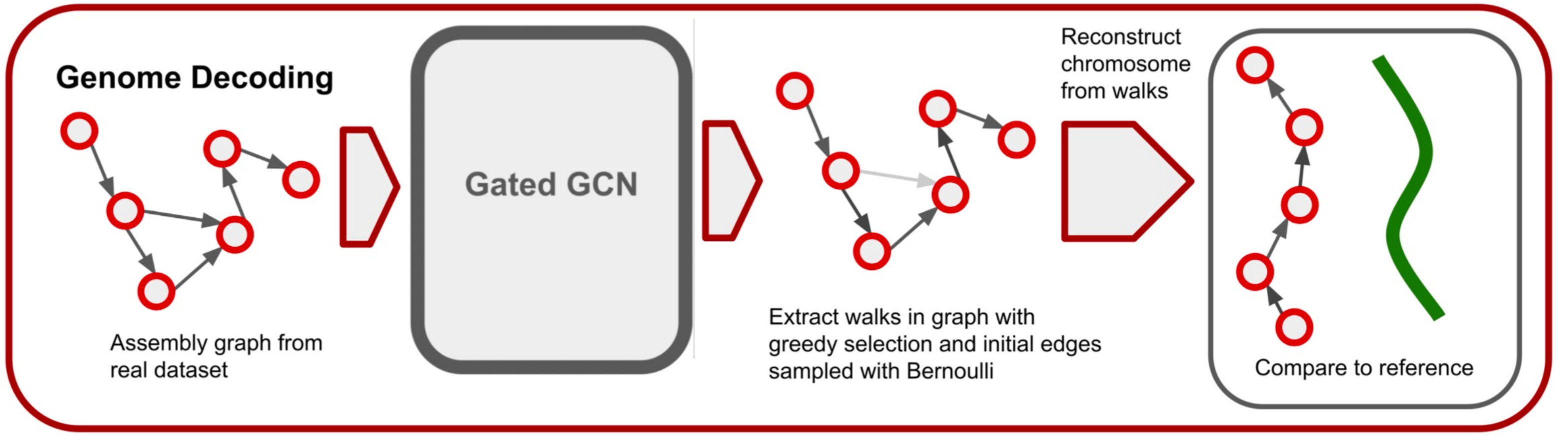
- In this work, we decode with a greedy search algorithm (O(n) complexity).
  - At each node, we select the edge $i_{t-1} \to i$ with the highest probability :

$$i_t = \arg\max_i\ P(i|i_{t-1}, i_{t-2}, ...i_1, G) = P(i|i_{t-1}, G)$$


Genome Decoding
Assembly graph from real dataset
Gated GCN
Extract walks in graph with greedy selection

Iterative Greedy Decoding

- Graph assembly is noisy (multiple connected components, dead-ends, cycles) and GNN edge-predictions are not perfect.
- We sample k paths from k initial edges selected by Bernoulli sampling and decode a path forward and path backward on the genome graph.
- We select the path with the longest sequence/contig length and marked the nodes as visited.
- We iterate the path extraction phase until the length of the extracted path is below a threshold.


Genome Decoding
Assembly graph from real dataset
Gated GCN
Extract walks in graph with greedy selection and initial edges sampled with Bernoulli
Reconstruct chromosome from walks
Compare to reference

Outline

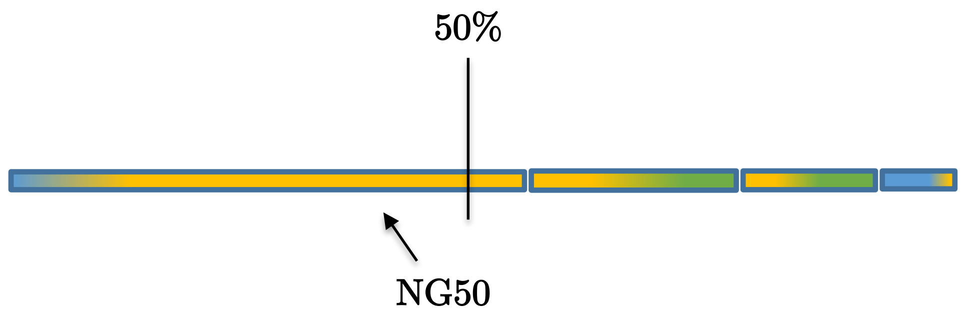
- Genome Assembly
- Assembly Graphs
- Path Assembly
- Our Contribution
- Dataset
- Edge Prediction
- Graph Neural Networks
- Graph Decoding
- **Numerical Experiments**
- Conclusion

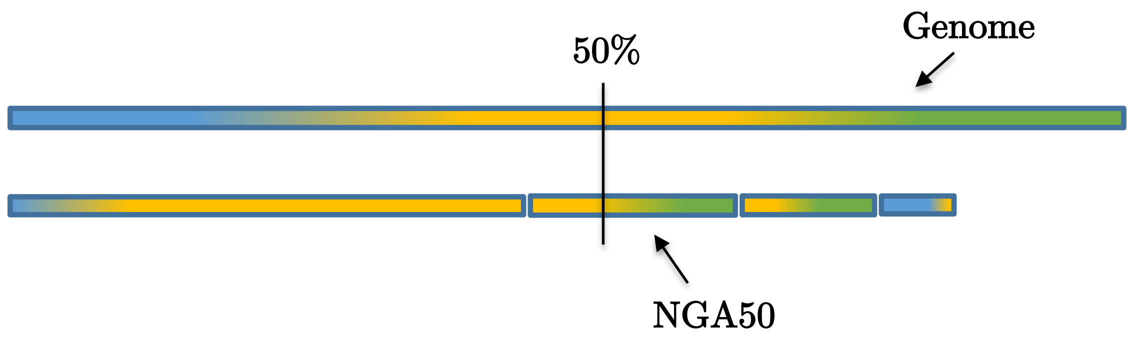
Experimental Setting

- Evaluation
  - In this work, we do not evaluate our technique on the whole genome, but on individual chromosomes.
- Training
  - We use one chromosome (chr19) for training and the remaining chromosomes for testing.
  - We generate 15 synthetic train graphs and 3 validation graphs.
  - We select the network for inference with the checkpoint having the lowest validation loss.
  - Training took 53min on Nvidia A100 GPU.
  - Note that we tried training with chromosomes 9, 19, and 22 and got slightly but not statistically better results.
- Inference
  - Forward pass + greedy decoding on the real assembly graphs of the test chromosomes

Evaluation

- Quality measures for genome assembly
  - Number of contigs : Gives an insight into how fragmented the reconstruction is (lower is better).
  - Longest contig : The length of the longest contig (higher is better).
  - Genome fraction : Fraction of the genome which is reconstructed (higher is better).
  - NG50 : Length of the contig that covers 50% of the reference genome (higher is better).
  - NGA50 : Calculated the same way as NG50, but after alignments between contigs and the reference (higher is better).
  - Base error : Number of mismatches and indels (insertions and deletions) per 100,000 base pairs (lower is better).





Experimental Results

- Evaluation of the network on assembly graphs of real human HiFi data.

	GatedGCN					**Raven**				
chr	**Num ctg**	**Longest (Mbp)**	**GF (%)**	**NG50 (Mbp)**	**NGA50 (Mbp)**	**Num ctg**	**Longest (Mbp)**	**GF (%)**	**NG50 (Mbp)**	**NGA50 (Mbp)**
1	**26**	**115.6**	**98.1**	**73.0**	**46.3**	241	86.9	97.6	44.4	44.4
2	**20**	73.1	**99.6**	**35.1**	**35.1**	56	73.1	98.9	28.1	28.1
3	**6**	**127.0**	**99.6**	**127.0**	56.0	45	90.5	99.5	56.0	56.0
4	**8**	**139.0**	99.0	**139.0**	34.8	78	67.8	99.0	34.9	**34.9**
5	**8**	**123.6**	**99.1**	**123.6**	103.5	47	103.5	99.0	103.5	103.5
6	**7**	101.0	**98.9**	101.0	**52.8**	20	**110.3**	98.7	**110.3**	25.9
7	**17**	**58.6**	**98.1**	**42.6**	**25.7**	69	29.3	98.0	25.1	17.5
8	**12**	**68.8**	**98.6**	**33.9**	28.5	33	31.6	98.4	28.5	28.5
9	**17**	**67.1**	**95.0**	**31.9**	**16.1**	139	38.9	90.2	19.7	15.8
10	**13**	**47.7**	**99.3**	**36.7**	**36.7**	43	36.7	99.2	17.2	17.2
11	**7**	**65.4**	**99.9**	**35.3**	23.2	31	35.3	99.7	32.6	23.2
12	**11**	57.2	**99.9**	31.0	31.0	33	57.2	99.8	31.0	31.0
13	**13**	**73.0**	**96.1**	**73.0**	**30.1**	116	47.5	95.9	25.5	25.5
14	**9**	82.6	**97.8**	82.6	82.6	32	82.6	97.2	82.6	82.6
15	**19**	**47.1**	**93.6**	**13.4**	**10.0**	157	29.0	93.5	9.0	8.5
16	**28**	16.0	**91.6**	**8.7**	**5.9**	164	**16.4**	90.8	5.9	5.7
17	**11**	**29.9**	**96.4**	**15.7**	**10.2**	47	12.9	96.1	9.0	9.0
18	**8**	**44.9**	97.6	**44.9**	17.4	45	43.5	**97.9**	43.5	17.4
*19	**20**	**14.0**	98.4	**5.1**	3.6	44	9.5	**98.5**	3.6	3.6
20	**9**	**32.7**	98.6	**26.7**	**17.8**	40	31.8	98.6	25.2	17.3
21	**4**	32.8	**94.6**	32.8	32.8	21	32.8	94.1	32.8	32.8
22	**11**	9.0	**94.7**	**6.7**	**4.0**	66	9.0	93.8	3.9	3.9
X	**18**	**50.6**	**98.6**	**27.1**	**13.2**	64	40.1	98.3	11.7	11.7

Our proposed learning method significantly outperforms Raven's heuristics!

Experimental Results

- Evaluation of the network on assembly graphs of real human HiFi data.

	GatedGCN		**Raven**	
chr	**Mismatch**	**Indel**	**Mismatch**	**Indel**
1	**2.54**	**0.91**	5.30	1.21
2	**1.50**	**0.64**	2.23	0.85
3	3.47	**0.69**	**2.46**	0.73
4	**1.32**	**0.65**	3.63	0.75
5	**2.65**	**0.54**	4.20	0.74
6	**0.84**	**0.50**	1.09	0.56
7	2.89	**0.99**	**2.29**	1.16
8	2.53	**0.72**	**1.79**	0.73
9	**5.22**	**1.94**	8.98	2.59
10	3.60	**0.93**	**2.85**	1.12
11	**0.65**	**0.74**	1.59	1.04
12	**0.37**	**0.53**	1.68	0.67
13	**2.16**	**0.63**	8.95	1.50
14	**1.57**	1.18	1.80	**1.14**
15	**6.02**	**1.56**	10.55	2.43
16	**8.82**	**1.98**	12.99	2.52
17	**6.01**	**1.33**	7.19	1.45
18	**4.60**	**0.69**	7.75	0.92
*19	9.45	**1.84**	**8.48**	2.09
20	**4.69**	**0.93**	9.05	1.65
21	**4.46**	**1.52**	10.00	1.82
22	**24.42**	**2.32**	27.45	4.40
X	**2.42**	**0.90**	3.42	1.18

Our proposed learning method significantly outperforms Raven's heuristics!

Outline

- Genome Assembly
- Assembly Graphs
- Path Assembly
- Our Contribution
- Dataset
- Edge Prediction
- Graph Neural Networks
- Graph Decoding
- Numerical Experiments
- Conclusion

Conclusion

- Experimental results demonstrate the potential of deep learning to solve the genome assembly grand challenge.
- Given a state-of-the-art genome assembler, we show that learned heuristics with GNN outperforms human engineered rules.
- This is a first proof-of-concept toward solving end-to-end the genome assembly task with a fast, accurate, robust, and universal algorithm.

Dataset and Code

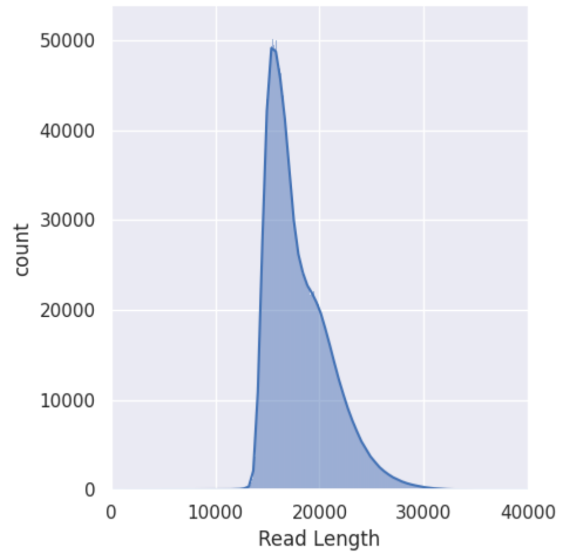
- We release the genomic dataset and GitHub repository.
  - https://github.com/lvrcek/GNNome-assembly
- Dataset (182GB)
  - CHM13 human genome (from[1])
  - Curated HiFi reads
    - Error-corrected reads
    - Positional information of reads on the genome
  - Graphs of all chromosomes
- GitHub repository
  - Reproducible results
  - Promotes research between deep learning and genome assembly.

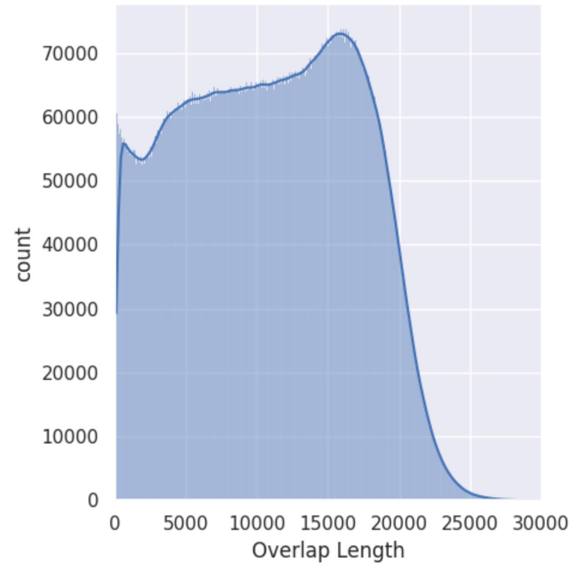
[1] Nurk et-al, The complete sequence of a human genome, Science 2022

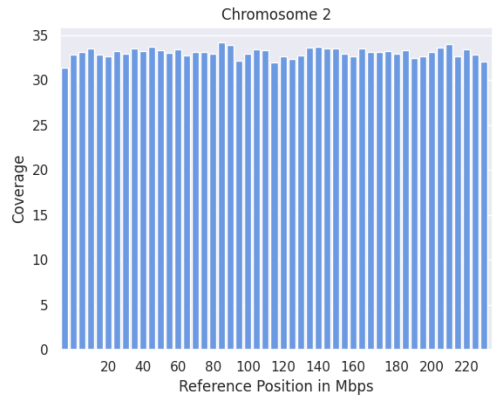
Dataset

- Statistics on real HiFi reads and assembly graphs created with Raven[1].

chr	Base pairs	Reads	Nodes	Edges
1	248,387,328	462,582	184,050	1,407,158
2	242,696,752	444,450	180,764	1,381,376
3	201,105,948	366,547	149,828	1,138,668
4	193,574,945	352,056	145,702	1,115,808
5	182,045,439	332,985	135,068	1,028,484
6	172,126,628	313,731	127,698	966,560
7	160,567,428	291,366	120,280	890,388
8	146,259,331	265,288	108,948	828,772
9	150,617,247	290,786	106,874	860,710
10	134,758,134	244,927	101,484	765,180
11	135,127,769	246,436	100,598	757,642
12	133,324,548	241,403	99,542	750,364
13	113,566,686	199,405	84,500	653,730
14	101,161,492	182,551	73,436	541,214
15	99,753,195	183,176	70,598	535,842
16	96,330,374	182,280	65,834	519,358
17	84,276,897	150,066	60,498	439,416
18	80,542,538	147,509	59,868	459,356
19	61,707,364	105,052	45,114	315,348
20	66,210,255	120,635	48,816	366,614
21	45,090,682	79,245	32,096	239,166
22	51,324,926	89,624	35,612	252,666
X	154,259,566	272,496	112,922	834,702







[1] Vaser, Sikic, Time-and memory-efficient genome assembly with raven, Nature Computational Science 2021

Next Steps

- Validate the proposed learned heuristics with other assembly graph constructers such that hifiasm[1], HiCanu[2], rust-mdbg[3] and LJA[4].
- Evaluate on the whole genome, not only individual chromosomes.
- Evaluate on two haploids simultaneously (CMH13 is a single haploid genome).
- Evaluate on other humans (different ethnicities).
- Evaluate on non-human genomes.
- Learn end-to-end the graph construction (overlap phase) along with the graph assembler (layout phase).

[1] Cheng, Concepcion, Feng, Zhang, Li. Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm, Nature methods 2021
[2] Nurk et-al, Hicanu: accurate assembly of segmental duplications, satellites, and allelic variants from high-fidelity long reads, Genome research, 2020
[3] Ekim, Berger, Chikhi, Minimizer-space de bruijn graphs: Whole genome assembly of long reads in minutes on a personal computer, Cell systems 2021
[4] Bankevich, Bzikadze, Kolmogorov, Antipov, Pevzner, Multiplex de bruijn graphs enable genome assembly from long, high-fidelity reads. Nature biotechnology 2022

Thank you

Xavier Bresson

xaviercs@nus.edu.sg

https://twitter.com/xbresson

https://scholar.google.com/citations?user=9pSK04MAAAAJ

https://www.youtube.com/channel/UCeONAtqVKCS30Xn6zy1YQ_g

https://github.com/xbresson

https://www.linkedin.com/in/xavier-bresson-738585b

https://www.facebook.com/xavier.bresson.1

Lab https://graphdeeplearning.github.io

https://www.comp.nus.edu.sg/cs/people/xaviercs