RE-PURPOSING GNNOME: GENOME ASSEMBLY POSTPROCESSING (GAP)

Joshua Teo, Martin Schmitz
Genome Institute of Singapore, A*STAR
60 Biopolis Street, Singapore 138672
Corresponding: joshuaemmanuelteo@gmail.com

ABSTRACT

GNNome is a graph neural network based framework that aims to perform de novo genome assembly. It works by representing a genome as a graph, then assigns probability scores to edges corresponding to if it thinks those two nodes in the genome overlap. However, these graphs are imperfect, and we hypothesize that we can find this missing information by searching all pairwise overlaps between the genome's fragments. In this paper, we create a pipeline to find and introduce this missing information by re-training and re-purposing GNNome's overlap edge prediction capabilities. In doing so, we are able to consistently improve upon GNNome, and outperform other state-of-the-art genome assemblers.

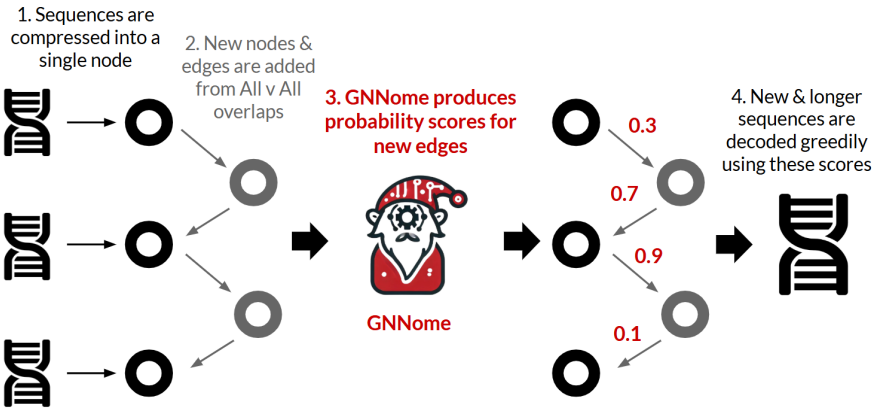

Figure 1: Using GNNome's edge probability prediction and supplementary overlap information to improve genome assemblies.

1 INTRODUCTION

De novo genome assembly, the reconstruction of genomes from short DNA fragments (reads) without the use of a reference genome, remains one of the most challenging and fundamental problems in computational biology. This process is crucial for studying organisms that have not been previously sequenced. A popular method for de novo genome assembly is known as Overlap-Layout-Consensus (OLC), which creates a graph by treating reads as nodes and overlaps between them as edges. A graph traversal algorithm is then used to decode a path which represents the final genomic sequence.

GNNome (Vrček et al., 2024) is a recently developed graph neural network (GNN) framework which automates this OLC process by training the GNN to predict the probability of an overlap edge being true. However, GNNome does not perform well on some genomes, and is also unable to handle diploid genomes as the framework does not do phasing. Moreover, overlap graphs in all such assemblers are imperfect and have missing information, primarily stemming from the nature of the sequencing data and/or the algorithmic approaches used. We hypothesize that this missing

information is available from an all vs all (AvA) overlap database which contains the pairwise overlap information of all reads in the dataset. The challenge then is how to identify and inject this information.

In this study, we create a Genome Assembly Postprocessing (GAP) pipeline to find and introduce this missing information into the assemblies, while re-purposing GNNome's overlap edge prediction capabilities in a more effective and generalisable method. We also use supplementary telomere information to enhance performance, something not done in the original GNNome paper.

This paper is structured as follows: Section 2 describes the proposed methods in the pipeline, experimental setup and configuration. Section 3 analyses the experiment's results. Finally, Section 4 concludes our study and discusses future steps.

2 METHODS

Our pipeline, GAP, requires two things: a baseline assembly and an AvA overlap database of its reads. GAP is independent from and operates without assumptions about the baseline assembly input regarding its ploidy status or sequencing technology, and delegates the choice of assembly strategy to the upstream assembler. For this study, we test haploid and diploid assemblies derived from both HiFi and ONT datasets. We use Hifiasm (Cheng et al., 2022), another state-of-the-art de novo genome assembler, to generate both the baseline assembly, and the AvA overlaps in a Pairwise mApping Format (PAF) file. Then, GAP does the following:

- **1. Compressing of Sequences**: Conventional OLC treats each individual read in the dataset as a node. Instead, we treat each sequence in the assembly as a node. The positional information of each individual read that make up these sequences is also stored.
- **2. Adding Ghost Nodes and Edges**: We then look for previously missed overlaps from the AvA overlaps, and add them into the graph as 'ghost' nodes and edges.
- **3. GNNome Overlap Edge Probability Generation**: We use a re-trained GNNome model to produce probability scores for these ghost edges.
- **4. Decoding of Sequences**: We greedily decode walks on our graph using GNNome's probability scores, then convert them into sequences for our new assembly.

2.1 ADDING GHOST NODES AND EDGES

We first create an empty graph by treating each of the assembly's sequences as a node. Then, from the AvA overlaps, we find reads that 1. overlap with at least one of these original sequences, and 2. are not already in any of them. Each of these reads are then added to the graph as a node, with an edge connecting it to the nodes of the original sequences it overlapped with. We refer to these newly added nodes and edges as 'ghosts'. When we connect a ghost node to an original sequence at position i, the bases behind i are sliced off and discarded. We also introduce a hyperparameter, **Valid Connection % (vc%)**, indicating the valid start and end percent of an original sequence that can be connected to. As vc% increases, more sections of the original sequences can be connected to, but also more will be discarded upon connection. Figure 2 illustrates this process. Additionally, we attempted to use heuristics such as coverage, overlap length, and overlap similarity to validate and filter these ghost edges. However, this was unsuccessful as these heuristics indiscriminately removed both correct and incorrect edges.

Finally, as there may be multiple connections between a ghost node and an original sequence node, we also perform a de-duplication step to pick the outermost connection (i.e. most towards the start of the sequence for connections in the starting vc%, and most towards the end of the sequence for connections in the ending vc%).

2.2 GNNOME OVERLAP EDGE PROBABILITY GENERATION

The original GNNome is a GNN framework where given an assembly's OLC graph, it predicts which edges/overlaps in the graph are valid. Accordingly, it is also trained on simulated OLC graphs with ground truth labels produced by PBSIM3 (Ono et al., 2012). Our task is almost similar to the original

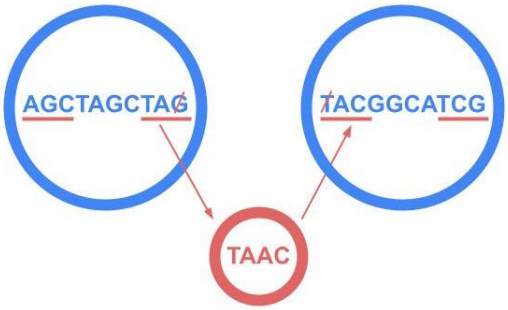

Figure 2: Adding a ghost node (red) connected to two compressed original sequence nodes (blue). vc% = 0.3, with the section of the sequence valid for connection underlined in red. The 'G' base at the end of the first blue node and the 'T' base at the start of the second blue node is discarded.

GNNome study, but with one key difference - the topology of the input graphs. We are feeding it graphs that include an additional one-hop neighbourhood of overlaps, which on average results in 100% more nodes and 300% more edges (varies by genome). As a result, we use the same model architecture as the original GNNome, but re-train the model on our larger graphs.

To create our graphs for training and validation, we first simulate HiFi Human (HG002) data (Consortium, 2023) using PBSIM3 (Ono et al., 2012). Each chromosome is separated and treated as an individual data point. We then run Hifiasm and use its assemblies to construct the conventional OLC graphs. Finally, from the AvA overlaps, we add nodes and edges with valid overlaps in a one-hop neighbourhood. This is similar to what was done in Section 2.1, with the key difference being the original nodes we are connecting to are individual reads instead of compressed sequences. During training, edges from the original OLC graph (and not from the AvA overlaps) are masked.

The learning objective is the symmetry loss used in the original paper, defined as $\text{SymLoss}(e, e') = \text{BCE}(p_e, y_e) + \text{BCE}(p_{en}, y_{en}) + 0.01 \times \|e - e'\|$, where *e* and *e'* are an edge and its virtual pair, and *p, y* and *l* denote probability, label and logit corresponding to *e* or *e'*. Hyperparameters used are latent dimension $d = 64$, latent feature dimension $f = 16$, number of GNN layers $L = 8$, dropout = 0.2. The learning rate for all models start at 0.0001, which is applied to the Adam optimiser, and then managed using PyTorch's ReduceLROnPlateau scheduler which reduces the learning rate by a factor of 0.1 with patience 20.

2.3 Decoding of Sequences

To retrieve the new sequences from our graph, we run a greedy search algorithm. Treating each original sequence node as the start, we greedily select the neighbour with the highest GNNome edge score until there are no more neighbours. We run this forwards and backwards, and track the number of original sequence nodes connected in that walk. We then select the walk that connects the most original sequence nodes, and remove it from the graph. This process then repeats until all original sequence nodes are removed from the graph.

We attempted to use other edge features such as overlap length and overlap similarity (as well as a mix of them) as heuristics when selecting neighbours. However, all experiments either matched or underperformed purely using GNNome edge scores. Another option was to use DFS to exhaustively search and retrieve the longest walk. However, this was computationally infeasible.

2.3.1 Use of telomere information

We also make use of telomere information in this step. A telomere is a region of repetitive DNA sequences called motifs demarcating the ends of eukaryotic chromosomes via repetitive DNA sequences, and can be exploited for more accurate assemblies. For all mammal genomes used in this

study, this motif is **TTAGGG**, and for all plant genomes used in this study, this motif is **TTTAGGG**. Telomere detection is done using seqtk (Li, 2021). The algorithm is outlined in Appendix A.

We use this telomere information in three ways. Firstly, original sequence nodes that start and end with telomeres are removed beforehand. Secondly, we prioritise original sequence nodes with telomeric regions when choosing start nodes for our greedy search. Lastly, during the greedy search, the telomere type of the current walk is tracked, and non-compatible walks are terminated. For example, if the walk already has a telomeric region, and it meets another identical telomeric region, that walk will terminate immediately after. The position of the telomeric region in the original sequence is also tracked, and walks are cut accordingly. For example, if an original sequence node has a telomeric region at the start of its sequence, it cannot have any incoming edges.

2.4 DATASET

All data used in this study is publicly available. We used both HiFi and ONT Simplex read data to evaluate our pipeline. HiFi datasets include Human (CHM13) (Nurk et al., 2022), Mouse (*M. musculus*) (Hon et al., 2020), Chicken (*G. gallus*) (Rhie et al., 2021), Arabidopsis (*A. thaliana*) (Wang et al., 2022), Corn (*Z. mays*) (Chen et al., 2023), Bonobo (*P. paniscus*) and Western Gorilla (*G. gorilla*) (Yoo et al., 2024) genomes. Bonobo and Western Gorilla are diploid while the rest are haploid. ONT datasets include Tomato (*S. lycopersicum*) (Su et al., 2021), Human (HG005) (Consortium, 2023), and Western Gorilla (*G. gorilla*) (Yoo et al., 2024). HG005 and Western Gorilla are diploid while tomato is haploid. HiFi Human (HG002) data (Consortium, 2023) was used to train and validate GNNome.

2.5 METRICS

To access the quality of the assemblies we use several standard metrics to measure contiguity, quality, and completeness. All metrics are computed with minigraph (Li et al., 2020).

NGx and NGAx are metrics used to measure the contiguity of the assembly, with higher values indicating longer contiguous sequences. NGAx aligns the assembly to the reference before calculation, and thus reflects accuracy as well. The 'x' refers to the percentage of the genome to account for when calculating these metrics. We use NG50 and NGA50, two commonly used metrics. When comparing the two, if NG50 is significantly higher than NGA50, it is an indication of overcorrection, where the assembly is too aggressive in connecting reads and makes connections that should not exist. We also look at AUNGA, which calculates the area under the curve when plotting NGA across all thresholds (0-100). Finally, Rdup represents the percentage of duplicate alignments or duplicated regions in the assembly.

3 RESULTS

We compare the assemblies before and after applying our method. We also perform ablation experiments with and without the use of telomere information. For haploid datasets, we also include the original GNNome's results (GNNome does not support diploid). For all runs, we set vc%=0.1. Full results of our experiments are in Table 1. We are able to consistently improve NG50, NGA50 and AUNGA and surpass the original GNNome on all genomes except for CHM13. Overcorrection (NG50 minus NGA50) already exists in most of Hifiasm's assemblies, thus we compare if our method reduces, maintains, or worsens this initial overcorrection. We can group the results on the genomes into three distinct categories:

- **Strict improvements across all metrics**: Genomes in this category are (HiFi) *A. thaliana, Z. mays*; (ONT) *S. lycopersicum, HG005 Paternal*.

  We are able to improve NG50, NGA50 and AUNGA on these genomes while reducing or maintaining overcorrection and duplication rate. The new connections introduced by our pipeline were correct.

- **Improvements in contiguity but with overcorrection**: Genomes in this category are (HiFi) *G. gallus, M. musculus, P. paniscus Paternal and Maternal, G. gorilla Paternal and Maternal*; (ONT) *HG005 Maternal, G. gorilla Paternal and Maternal*.

NG50, NGA50 and AUNGA on these genomes increased. However, so did overcorrection and/or duplication rate. This indicates that while we did introduce corrrect connections that improved the contiguity, there were also some incorrect connections introduced.

- **Overcorrection without improvement in contiguity**: Only (HiFi) *CHM13* falls in this category.

  NG50 increased with no difference in NGA50/AUNGA. This indicates that we mainly introduced incorrect connections. There is a minor increase in AUNGA, indicating that there were a small handful of correct connections introduced amongst the shorter sequences.

Table 1: Assembly results for the different genome datasets.

Experiment	NG50	NGA50	AUNGA	Rdup (%)
		(HiFi) A. thaliana		
Hifiasm	12442387	12439611	10189795	0.58
GNNome	12442359	12439583	10189885	0.64
GAP (w/o telo info)	**12749108**	**12746238**	**10437609**	0.61
GAP	**12749108**	**12746238**	**10437489**	0.58
		(HiFi) G. gallus		
Hifiasm	11487786	10858828	15181647	4.18
GNNome	11379811	10899819	13389474	**3.64**
GAP (w/o telo info)	13047147	11969028	17899402	4.20
GAP	**14092472**	**12696723**	**19085068**	4.18
		(HiFi) M. musculus		
Hifiasm	22952472	18723670	23999763	0.64
GNNome	22952576	17971948	23453739	**0.63**
GAP (w/o telo info)	**28447861**	**22371954**	**28575360**	0.68
GAP	**28447861**	**22371951**	**28593165**	0.67
		(HiFi) CHM13		
Hifiasm	111708415	111068087	100017625	0.31
GNNome	111252795	111045714	98421653	**0.26**
GAP (w/o telo info)	121736307	111068087	**102777353**	0.32
GAP	121736307	111068087	**102737998**	0.31
		(HiFi) Z. mays		
Hifiasm	102618559	79610908	94054358	7.37
GNNome	75961425	75961395	79091244	**5.40**
GAP (w/o telo info)	142826180	117558082	**106868889**	7.45
GAP	**117664086**	**117558018**	105139141	7.41
		(HiFi) P. paniscus, Paternal		
Hifiasm	3528986	3080294	4158504	1.26
GAP (w/o telo info)	5485364	4656504	5972371	1.26
GAP	**5550666**	**4708352**	**6015031**	1.26
		(HiFi) P. paniscus, Maternal		
Hifiasm	3303137	2893596	4235275	1.16
GAP (w/o telo info)	5266385	4610107	6416779	1.14
GAP	**5592962**	**4939097**	**6640755**	**1.12**
		(HiFi) G. gorilla, Paternal		
Hifiasm	12798765	10733443	12613022	**1.17**
GAP (w/o telo info)	20566508	14759912	**17192444**	1.18
GAP	**20277228**	**14815529**	17081701	1.18
		(HiFi) G. gorilla, Maternal		
Hifiasm	11778356	10212228	12022468	**1.14**

Experiment	NG50	NGA50	AUNGA	Rdup (%)
GAP (w/o telo info)	**18773065**	**15126455**	**19023744**	1.18
GAP	**18793265**	**15126455**	**19057834**	1.18
(ONT) S. lycopersicum				
Hifiasm	67122914	42403697	38311092	0.34
GNNome	67122914	43581231	38456766	0.37
GAP (w/o telo info)	67122914	**43953729**	**38476154**	0.34
GAP	67122914	**43953729**	**38476159**	0.34
(ONT) HG005, Paternal				
Hifiasm	94260768	53629323	54094963	0.60
GAP (w/o telo info)	**96449415**	**55219196**	**55746919**	**0.57**
GAP	**96449415**	**55219196**	**55764592**	**0.57**
(ONT) HG005, Maternal				
Hifiasm	87832971	53502639	56438272	0.52
GAP (w/o telo info)	**97336512**	**59512061**	**57482540**	0.52
GAP	**97336512**	**59512051**	**57484180**	0.52
(ONT) G. gorilla, Paternal				
Hifiasm	58915776	23268248	34587226	2.06
GAP (w/o telo info)	**63651705**	**24467321**	**35180192**	2.07
GAP	**63651705**	**24467321**	**35179136**	2.06
(ONT) G. gorilla, Maternal				
Hifiasm	56192691	30922572	33259053	2.37
GAP (w/o telo info)	**66744814**	**34114106**	**36074022**	**2.34**
GAP	**66744814**	**34114106**	**36073863**	**2.34**

We show that there is useful information in Hifiasm's AvA overlap database and our method presents a viable method to introduce it into the assemblies. However, it does not excel at filtering out which overlaps should be selected, as the majority of genomes face the overcorrection problem. This is directly correlated with GNNome's edge prediction capabilities, as the selection of overlaps is done via GNNome's edge scores. If we are able to improve upon GNNome, we can better filter out these incorrect edges. The use of telomere information also helps to filter out some of these incorrect edges, reducing overcorrection and improving contiguity on (HiFi) *G. gallus, Z. mays, P. paniscus Paternal and Maternal,* and *G. gorilla Maternal*. However, the effectiveness of using telomere information largely depends on the presence of these telomeric regions in the dataset as well as how accurately they are detected.

Specifically for CHM13, we suspect that since Hifiasm was designed on Human genome data, its assembly is of a very high quality. Hence, the probability of finding improved connections is low.

4 Conclusion and Future Work

There are several areas for improvement that we are currently working on, or have plans to do so in future. Firstly, in this study, the information introduced from the AvA overlaps when adding ghost nodes to our graphs was only in a one-hop neighbourhood. However, this can be extended to any n-hop number of neighbourhoods. Hifiasm's PAF itself contains up to two-hop neighbourhoods worth of alignment information. For this study, we decided against this as each additional neighbourhood adds exponential computational costs. However, this would allow connections between original sequence nodes that are n ghost nodes apart (currently they can only be one apart). This could reveal more crucial connections. Secondly, while GAP has proven its generalisability and effectiveness independent of sequencing technology and ploidy status, it would also be beneficial to tailor our methods to cater specifically to each case. One low-hanging fruit we plan to implement would be for diploid assemblies. Hifiasm only produces a single AvA overlap database/PAF. Thus, binning these overlaps beforehand could potentially reduce haplotype switches.

The methods used in GAP can be easily adapted to and integrated with any OLC assembler, with the only key piece of information needed being the AvA overlap database. In our case, this was conveniently generated by Hifiasm in the PAF. However, this can be manually achieved on any dataset. We hope that the methods and findings from this study can offer simple and practical building blocks for improving genome assembly, and on a broader scale integrate it with the domain of machine learning.

AUTHOR CONTRIBUTIONS

Teo. J. designed and developed the methods, conducted the experiments, evaluated the results, and wrote the manuscript. Schmitz. M. provided guidance and resources throughout the duration of the project.

REFERENCES

Chen, J., Wang, Z., Tan, K., et al. (2023). A complete telomere-to-telomere assembly of the maize genome. *Nature Genetics*, 55:1221–1231.

Cheng, H., Jarvis, E., Fedrigo, O., Koepfli, K., Urban, L., Gemmell, N., and Li, H. (2022). Haplotype-resolved assembly of diploid genomes without parental data. *Nature Biotechnology*, 40:1332–1335.

Consortium, H. P. R. (2023). A draft human pangenome reference. *Nature*, 615(7956):310–319.

Hon, T., Mars, K., and Young, G. (2020). Highly accurate long-read hifi sequencing data for five complex genomes. *Scientific Data*, 7, 399.

Li, H. (2021). Seqtk: a sequence toolkit for processing sequences in fasta/q formats. Accessed: 2024-10-21.

Li, H., Feng, X., and Chu, C. (2020). The design and construction of reference pangenome graphs with minigraph. *Genome Biology*, 21:1–19.

Nurk, S., Koren, S., Rhie, A., Rautiainen, M., Bzikadze, A. V., Mikheenko, A., Vollger, M. R., Altemose, N., Uralsky, L., Gershman, A., et al. (2022). The complete sequence of a human genome. *Science*, 376(6588):44–53.

Ono, Y., Asai, K., and Hamada, M. (2012). Pbsim: Pacbio reads simulator—toward accurate genome assembly. *Bioinformatics*, 29(1):119–121.

Rhie, A., McCarthy, S., Fedrigo, O., et al. (2021). Towards complete and error-free genome assemblies of all vertebrate species. *Nature*, 592:737–746.

Su, X., Wang, B., Geng, X., Du, Y., Yang, Q., Liang, B., Meng, G., Gao, Q., Yang, W., Zhu, Y., and Lin, T. (2021). A high-continuity and annotated tomato reference genome. *BMC Genomics*, 22(1):898.

Vrček, L., Bresson, X., Laurent, T., Schmitz, M., Kawagushi, K., and Šikić, M. (2024). Geometric deep learning framework for de novo genome assembly. *bioRxiv*.

Wang, B., Yang, X., Jia, Y., Xu, Y., Jia, P., Dang, N., Wang, S., Xu, T., Zhao, X., Gao, S., Dong, Q., and Ye, K. (2022). High-quality arabidopsis thaliana genome assembly with nanopore and hifi long reads. *Genomics Proteomics Bioinformatics*, 20,1:4–13.

Yoo, D., Rhie, A., Hebbar, P., Antonacci, F., Logsdon, G. A., Solar, S. J., Antipov, D., Pickett, B. D., Safonova, Y., Montinaro, F., et al. (2024). Complete sequencing of ape genomes. *bioRxiv*, pages 2024–07.

A GREEDY DECODING ALGORITHM

function GET_NEW_WALKS($original\_sequence\_nodes, graph$)
    Initialise new walks, reverse graph, and partition original sequence nodes by telomere presence
    Remove all original sequence nodes that have both start and end telomeric regions, and add them to new walks
    for each group in {telomeric, non-telomeric} original sequence nodes **do**
        while unused nodes in current group **do**
            best_walk, best_n_seq_nodes ← empty, empty
            for each $node\_id$ in current group **do**
                graph_to_use ← (graph if telomere at start of $node\_id$, else reversed graph)
                walk, n_seq_nodes ← GET_BEST_WALK($node\_id$, graph_to_use)
                if n_seq_nodes $\geq$ best_n_seq_nodes **then**
                    best_walk, best_n_seq_nodes ← walk, n_seq_nodes
                end if
            end for
            Append best_walk to new walks
            Remove all node IDs in the walk from their respective telomeric/non-telomeric groups
        end while
    end for
    return new walks
end function

function GET_BEST_WALK($start, graph$)
    walk, current_node, visited ← [$start$], $start$, empty set
    while True **do**
        Get neighbours of current_node
        Filter out neighbours that are in visited or have incompatible telomeric regions
        if there are no valid neighbours **then** break
        current_node ← neighbour with highest GNNome edge score
        Append current_node to walk
    end while
    return walk, number of key nodes in walk
end function