

Synteny Paths for Assembly Graphs Comparison

Evgeny Polevikov

Bioinformatics Institute, Saint Petersburg, Russia
polevikovea@gmail.com

Mikhail Kolmogorov¹ 

Department of Computer Science and Engineering, University of California, San Diego, CA, USA
mkolmogo@ucsd.edu

Abstract

Despite the recent developments of long-read sequencing technologies, it is still difficult to produce complete assemblies of eukaryotic genomes in an automated fashion. Genome assembly software typically output assembled fragments (contigs) along with assembly graphs, that encode all possible layouts of these contigs. Graph representation of the assembled genome can be useful for gene discovery, haplotyping, structural variations analysis and other applications. To facilitate the development of new graph-based approaches, it is important to develop algorithms for comparison and evaluation of assembly graphs produced by different software. In this work, we introduce *synteny paths*: maximal paths of homologous sequence between the compared assembly graphs. We describe Asgan – an algorithm for efficient synteny paths decomposition, and use it to evaluate assembly graphs of various bacterial assemblies produced by different approaches. We then apply Asgan to discover structural variations between the assemblies of 15 *Drosophila* genomes, and show that synteny paths are robust to contig fragmentation. The Asgan tool is freely available at: <https://github.com/epolevikov/Asgan>.

2012 ACM Subject Classification Applied computing → Bioinformatics

Keywords and phrases Assembly graphs, Genome assembly, Synteny blocks, Comparative Genomics

Digital Object Identifier 10.4230/LIPIcs.WABI.2019.24

Supplement Material The Asgan tool is freely available at: <https://github.com/epolevikov/Asgan>.

Funding The authors received no specific funding for this work.

Acknowledgements We are grateful to Dmitry Antipov, Pavel Avdeyev, Pavel Pevzner and Giulia Guidi for their helpful comments.

1 Introduction

Genome assembly is the problem of reconstructing a DNA sequence from sequencing reads - short, overlapping substrings of the original DNA. This reconstruction is challenging because of the presence of genomic repeats - multiple copies (precise or imprecise) of the same sequence within the genome. Since the read length is limited, even simple bacterial assemblies from short Illumina reads may contain hundreds of unresolved repeats, which results in fragmented assemblies [22]. The increased read length of the Pacific Biosciences (PacBio) and Oxford Nanopore Technologies (ONT) sequencers significantly improved the contiguity of many de novo assemblies [27]. However, other genomes are still incomplete due to very long unresolved repeats, such as segmental duplications in human genomes [34].

¹ Corresponding author



To reduce the assembly fragmentation and optimally resolve repeats, genome assemblers construct various assembly graphs from input reads. One popular representation is the overlap graph [23], where each input read corresponds to a node, and directed edges represent overlaps between the corresponding reads. A limitation of this approach is that it does not reveal precise boundaries of genomic repeats as they might be hidden inside the nodes [10]. Alternatively, de Bruijn graphs [26] proved to be a useful framework that represents read information in a compact form and reveals repeat boundaries [25]. A recently introduced Flye algorithm [12] utilizes repeat graphs, which are similar to de Bruijn graphs, but are built using approximate, rather than exact sequence matches.

A typical genome assembly workflow includes building and simplifying an assembly graph, afterwards contigs are generated as unambiguous paths in this graph [32, 4]. Many studies focus only on the resulting contig fragments for downstream analysis, however it was shown that incorporating the adjacency information from assembly graphs can be useful for gene discovery [33], structural variation analysis [9], hybrid assembly [2, 35], haplotyping [30], segmental duplication analysis [12] and other applications.

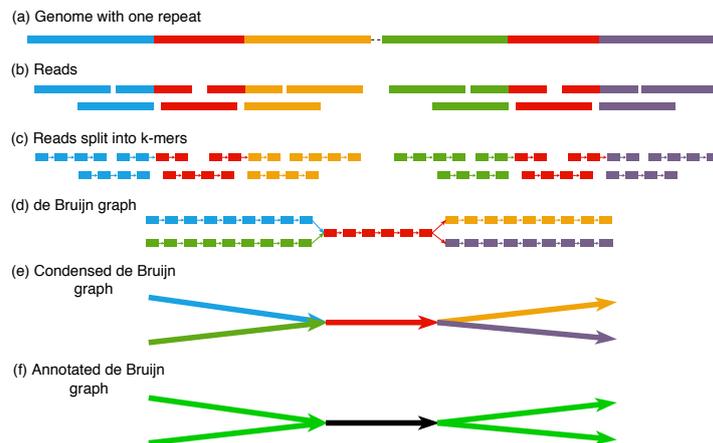
To facilitate the development of new assembly graph-based approaches, it is important to develop algorithms for comparison and evaluation of various assembly graphs produced by different approaches. Recently, a GTED distance metric [5] has been introduced as an attempt to generalize the string edit distance [14] to sequence graphs. While theoretically sound, this approach has only been applied to small viral genomes due to the computational constraints. Additionally, Earth mover's distance (EMD) was proposed as a probabilistic distance measure between de Bruijn graphs of metagenomic assemblies [18]. The authors have shown that incorporating the connectivity information from de Bruijn graphs improves the accuracy of metagenomic sample classification, comparing to the read mapping approaches. However, the described algorithm works only with de Bruijn graphs built with small values of k (less than 10), which makes it unsuitable to analyse the structure of the assembly graphs produced by the most genome assemblers. Finally, various visualization tools, such as Bandage [36] or AGB [19] allow for visual inspection of assembly graphs, but do not support automatic graphs comparison.

In this work, we propose a new method for assembly graphs analysis and comparison, which could be used for benchmarking various assembly algorithms, and for improving comparative genomics capabilities of fragmented assemblies. We introduce *synteny paths* - maximal paths of homologous sequence between the compared assembly graphs, which are inspired by synteny blocks that are commonly used for structural comparison of complete genomes [24, 8]. We formulate the *minimum synteny paths* decomposition problem, prove that its exact solution is NP-hard and provide an efficient heuristic algorithm. Then, we illustrate the application of synteny paths for evaluating assembly graphs of various bacterial genomes produced by different assemblers. Finally, we apply synteny paths to compare 15 *Drosophila* assemblies and show that our approach is robust to contig fragmentation, and reveals structural divergences between the compared genomes.

2 Background

De Bruijn graphs. Given a set of reads R and a parameter k , de Bruijn graph $DBG(R, k)$ is constructed by first representing each read of length L as a set of $L - k + 1$ overlapping k -mers (substrings of length k). Each unique k -mer k_1 from the constructed set is translated into a node v_{k_1} in the de Bruijn graph. Two nodes v_{k_1} and v_{k_2} are connected by a directed edge, if the corresponding k -mers k_1 and k_2 are adjacent in at least one of the input reads

(Figure 1a-d). Note that it also means: (i) k_1 and k_2 overlap by $k - 1$ nucleotides and (ii) edge (v_{k_1}, v_{k_2}) spells a $(k + 1)$ -mer, constructed from two overlapping k -mers. Since each read could also be represented as a sequence of consecutive $(k + 1)$ -mers: (k_1, k_2, \dots, k_n) , the corresponding edges in DBG form a path. Similarly, because every read is a substring of the original genome G , there is a path in DBG , that spells G (assuming no sequencing errors and uniform read coverage).



■ **Figure 1** De Bruijn graph construction and annotation. (a) Genome with one repeat of multiplicity two (shown in red). (b) Reads sampled from the genome. (c) Each read of length L represented as sequences of $L - k + 1$ k -mers. (d) De Bruijn graph is built by gluing k -mers that spell the same sequence. The repeat copies are collapsed into a path, with first and last nodes of this path revealing the repeat boundaries. (e) Condensed de Bruijn graph is constructed by collapsing non-branching paths into single edges. (f) Annotated de Bruijn graph, with unique edges shown in green and a repetitive edge shown in black.

For simplicity, we assume that the genome consists of one circular chromosome, thus every node in DBG has at least one incoming and outgoing edge. Because some k -mers appear multiple times in the genome, the corresponding DBG nodes might have multiple incoming or outgoing edges. We call a node *non-branching*, if it has one incoming and one outgoing edge (and *branching* otherwise). A *maximal non-branching path* is a path in which two terminal nodes are branching, and all intermediate nodes are non-branching. Many assemblers simplify the described de Bruijn graph by collapsing every maximal non-branching path into a single edge, labelled with a sequence spelled by the original path. This transformation results in a condensed de Bruijn graph (Figure 1e). Further in text, we assume that de Bruijn graphs are condensed, meaning that edge sequences might be longer than k .

Double-stranded de Bruijn graphs. De novo assembly algorithms assume that reads might originate from either forward or reverse DNA strand, therefore it is convenient to represent both strands of the assembled genome in a double-stranded de Bruijn graph. For any edge e that spells sequence S in such graph, there exists a single complementary edge e' that spells S' (reverse-complement of S). Further, a double-stranded DBG is symmetric with respect to the complement operation: for any path $P = (e_1, e_2, e_3, \dots, e_{l-1}, e_l)$ of size l that spells sequence S , there exists a complementary path $P' = (e'_l, e'_{l-1}, \dots, e'_3, e'_2, e'_1)$ of size l that spells S' .

3 Methods

Annotated de Bruijn graph (ADBG). If all k -mers in a genome G were unique, $DBG(G)$ would consist of a single non-branching cyclic path, however repetitive k -mers complicate the graph structure. If a k -mer originates from a genomic repeat (of length $\geq k$), all copies of this k -mer in the genome will be collapsed into a single node on DBG . Similarly, multiple consecutive repetitive k -mers from the genome will be collapsed into a non-branching path in the graph (or a single edge in case of the condensed DBG).

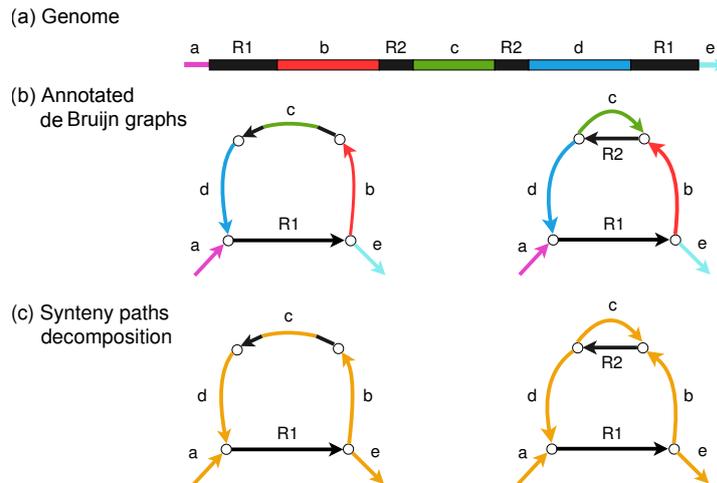
We define *annotated de Bruijn graph (ADBG)* as a de Bruijn graph with each edge labeled as either unique or repetitive. If the complete genome sequence G is available, one can classify each edge of $ADBG(G)$ as either unique or repetitive by checking how many times the corresponding sequence appears in G . If the genome sequence is unknown (as in de novo assembly), edges could be classified based on read coverage, edge length distribution and other criteria (we provide an implementation below).

Analyzing different ADBGs derived from the same genome. First, we consider a case when two ADBGs AG_1 and AG_2 are constructed from the same genome G , but with different values of k ($k_1 < k_2$). Assuming uniform read coverage and no sequencing errors, both AG_1 and AG_2 contain a path that spells G . However, AG_1 might have more repetitive edges than AG_2 corresponding to repeats varying in length from k_1 to k_2 . To reveal similarities between two graphs, we focus on their unique edges, because mapping of repetitive edges could be ambiguous. For each unique edge e^1 in AG_1 , there is a unique edge e^2 in AG_2 that has a substring that is identical to the string spelled by e^1 . We call this pair of edges e^1, e^2 *syntenic*. In the case of condensed DBG , a unique edge e^2 from AG_2 might correspond to multiple unique edges $\{e_1^1, e_2^1, \dots, e_k^1\}$ from AG_1 due to higher fragmentation level. In this case, we split e^2 into a path of syntenic edges $(e_1^2, e_2^2, \dots, e_k^2)$ that spells the original sequence of e^2 . This defines unique syntenic edge pairs between AG_1 and AG_2 . (Figure 2a-b).

Synteny paths. We say that two unique edges u and v are *compatible* in an ADBG AG if either (i) u and v are adjacent or (ii) AG contains a path between u and v such that all intermediate edges in this path are repetitive. Intuitively, we allow arbitrary insertions of repetitive sequences between the compatible unique edges, but prohibit unique sequence to be rearranged. Given two ADBGs AG_1 and AG_2 with two pairs of syntenic edges $U = \{u^1, u^2\}$ and $V = \{v^1, v^2\}$, we call pairs U and V *colinear* if u^1 and v^1 are compatible in AG_1 , and u^2 and v^2 are compatible in AG_2 . Synteny path is defined as a sequence of syntenic edge pairs $P = (E_1, E_2, \dots, E_k)$, in which every two consecutive pairs (E_{i-1}, E_i) are colinear (Figure 2c). A single syntenic edge pair is also considered a trivial synteny path.

Minimum synteny paths (MSP) decomposition. Each synteny path reveals local similarities between edges of ADBGs. To compare two graphs globally, we formulate the following decomposition problem. Given two ADBGs AG_1 and AG_2 with unique edges decomposed into the set of syntenic edge pairs $E = \{(u^1, u^2), (v^1, v^2), (w^1, w^2), \dots\}$, find the minimum number of synteny paths that cover all edge pairs from E . Intuitively, we want to find a minimal set of sequences that traverse all unique edges in AG_1 and AG_2 in the same order, while allowing arbitrary repeat insertions.

If AG_1 and AG_2 are built from the same unichromosomal genome G , they share a synteny path that spells G (with repeats removed) and solves MSP. However, because multiple MSP solutions may exist, a single path that solves MSP might not correspond to the original genome. Because each syntenic edge pair is considered a trivial synteny path, the maximum number of synteny paths in MSP equals to the number of syntenic edge pairs.

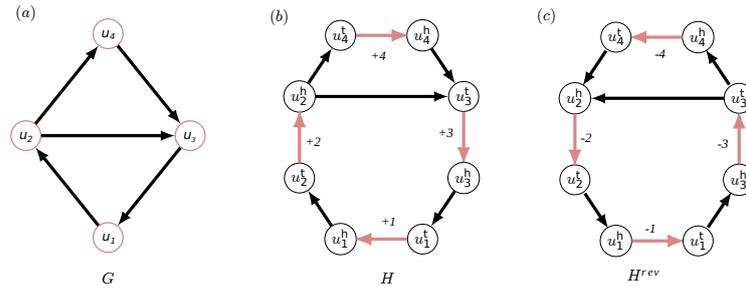


■ **Figure 2** An example of synteny paths decomposition. (a) A genome with two repeats of lengths $|R_1| > |R_2|$ shown in black. Unique sequences between repeats is shown in color. Each repeat is represented in two exact copies. (b) Two ADBGs built with different values of k . The left graph is built with $|R_2| < k < |R_1|$, so that only R_1 is collapsed in the graph. The graph on the right is built for $k < |R_2| < |R_1|$, thus both R_1 and R_2 are collapsed in the graph. Colors denote edge pairs that spell the same genome subsequences (syntenic edges). (c) Syntenic edges are joined into a single synteny path (shown in orange). The path traverses unique edges in the same order (a, b, c, d, e) and reveals the similarity between two graphs.

In case of double-stranded ADBG, we require the resulting set of synteny paths to be symmetric with respect to the complement operation: for any path P from the MSP solution, P' also belongs to the solution. Further in text we assume the symmetric version of the MSP decomposition.

MSP decomposition is NP-hard. Following we prove that the MSP decomposition is NP-hard by reducing the Hamiltonian path problem in directed graphs to the MSP decomposition. Let G be an arbitrary directed graph. We construct a new ADBG H from G , by translating nodes from G into a unique edges in H , and edges from G into repetitive edges in H . For each node u from G , we create two nodes in H : u^t and u^h (*tail* and *head* nodes, respectively), which are connected with a unique (directed) edge (u^t, u^h) . Next, for each edge (u, v) from G , we connect the nodes u^h and v^t of H with a repetitive edge (u^h, v^t) . Note that unique edges in H always start at tail nodes and end at head nodes, while repetitive edges start at head nodes and end at tail nodes (Figure 3a-b). Afterwards, we build a complementary graph H^{rev} as a copy of H with all edges reversed. Each unique edge (u_i^t, u_i^h) from H and its complement (u_i^h, u_i^t) from H^{rev} are labelled as $+i$ and $-i$, respectively. The described transformation could be performed in $O(|V| + |E|)$ time. For simplicity, we first prove that single-stranded MSP version is NP-hard using H , and then extend it to the double-stranded case with $H_{ds} = H \cup H^{rev}$ (Figure 3c). We also assume that indegree and outdegree of nodes in H is unlimited – below we show how to extend our proof to ADBGs over the DNA alphabet, in which indegree and outdegree of each node is at most four.

► **Lemma 1.** *If there is a Hamiltonian path in the initial graph G , it corresponds to a path in H that covers all unique edges.*



■ **Figure 3** Reducing the Hamiltonian path problem to the MSP problem. (a-b) Each node in the initial graph G is translated to a unique edge (shown in pink) in the new graph H , while the edges of G are translated into repetitive edges (shown in black) in H . (c) A complementary graph H^{rev} is constructed by inverting the directions of all edges in H . A union of two graphs $H_{ds} = H \cup H^{rev}$ is then used to solve the double-stranded MSP decomposition.

Proof. Let $p_G = (u_1, u_2, \dots, u_n)$ be a Hamiltonian path in G . By construction, it corresponds to the following path in H : $p_H = (u_1^t, u_1^h, u_2^t, u_2^h, \dots, u_n^t, u_n^h)$. Since each node u_i of G is covered by p_G , each unique edge (u_i^t, u_i^h) of H is covered by p_H . ◀

► **Lemma 2.** *If there is a path in H that covers all unique edges, it corresponds to a Hamiltonian path in G .*

Proof. Let $p_H = (u_1^t, u_1^h, u_2^t, u_2^h, \dots, u_n^t, u_n^h)$ be a path of length n in H that covers all unique edges (u_i^t, u_i^h) . By construction, consecutive pairs of nodes (u_i^h, u_{i+1}^t) correspond to repetitive edges in H . Thus, p_H is an alternating sequence of unique and repetitive edges in H , and is translated to the following node path of length n in G : $p_G = (u_1, u_2, \dots, u_n)$, which traverses all nodes in G . ◀

► **Theorem 3.** *Minimum synteny paths decomposition is NP-hard.*

Proof. Note that Lemmas 1-2 were formulated for paths in a single graph H . The lemmas could be trivially extended to pairwise synteny paths by duplicating H into H^{copy} and defining the duplicated unique edges as syntenic. Likewise, we can trivially extend the single-stranded version of the MSP decomposition to the double-stranded case by considering $H_{ds} = H \cup H^{rev}$, since H and H^{rev} are symmetric and independent.

Finally, to overcome the maximum node degree limit in ADBGs constructed over the DNA alphabet, we apply the following transformation to H . Each node with indegree or outdegree more than four is split into multiple nodes, and the original edges are distributed among the new nodes so as to satisfy the degree limit. The new nodes are also connected via additional repetitive edges in both directions. This forms *supernodes*, which are equivalent to the original nodes in H with respect to the connectivity of the unique edges.

The extended versions of Lemmas 1-2 prove the theorem. ◀

A heuristic algorithm for the MSP decomposition. Since the exact solution for the MSP decomposition is NP-hard, we propose an efficient heuristic algorithm. Given two double-stranded ADBGs AG_1 and AG_2 with unique edges decomposed into $2n$ syntenic pairs $\{\pm 1, \pm 2, \dots, \pm n\}$ (complementary edges have opposite sign), we construct a breakpoint graph [24, 3] as described below.

For each syntenic pair $+i$, we create two nodes in the breakpoint graph: i^t and i^h . A complementary syntenic pair $-i$ corresponds to the same pair of nodes, but in reversed order: i^h and i^t . Then, for each pair of colinear syntenic edges (u, v) , we put an undirected

edge between u^h and v^t (denoted as *adjacency* edge). Note that the complementary colinear syntenic pair $(-v, -u)$ will correspond to the same connection on the breakpoint graph, hence providing a convenient way to represent two strands of a de Bruijn graph [16] (Figure 4a-b).

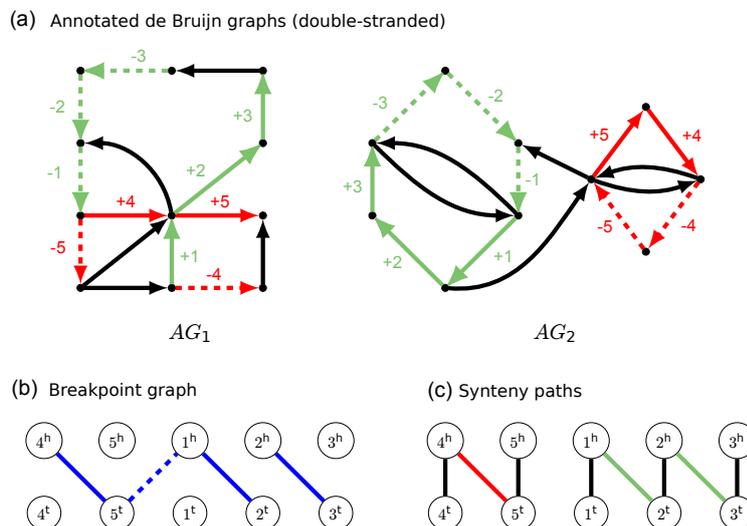


Figure 4 A heuristic algorithm for minimum syntenic paths problem. (a) Two double-stranded ADBGs AG_1 and AG_2 with edges decomposed into syntenic pairs. Repetitive edges shown in black, and unique edges are colored into green and red, highlighting two syntenic paths in the graphs. Dashed edges represent complementary edges. (b) Breakpoint graph constructed from AG_1 and AG_2 . The nodes correspond to the ends of syntenic blocks (head or tail) and edges represent the corresponding colinear blocks. Solid edges show the selected maximum matching (dashed edge was not selected). (c) Syntenic paths are revealed by removing edges that are not part of the matching, and adding trivial edges (shown in black).

The resulting breakpoint graph encodes all possible colinear syntenic pairs, therefore a matching on this graph defines a set of syntenic paths. To minimize the number of syntenic paths, we find a maximum matching on the breakpoint graph using the Blossom algorithm [7] (Figure 4b). Given the maximum matching, we reconstruct syntenic paths from the breakpoint graph by adding edges that connect nodes $\{i^h, i^t\}, i = 1, \dots, n$ (trivial edges). The resulting set of paths on the breakpoint graph defines a set of syntenic paths between AG_1 and AG_2 (Figure 4c). Note that because the complementary colinear connections are represented by the same adjacency edges in the breakpoint graph, the constructed MSP solution will automatically be symmetric.

In some cases, the reconstructed matching might not yield to the optimal MSP solution. Consider an initial breakpoint graph with only trivial edges present. When a new adjacency edge is added into the graph it either (i) connects two paths into one (reducing the number of syntenic paths by one), or (ii) transforms an existing path into a cycle (the number of paths is not reduced). After the initial maximal matching is reconstructed, we modify it using the following heuristic to minimize the number of edges of the second type and improve the MSP solution. The algorithm finds pairs of cycles in the graph that could be merged by exchanging two adjacency edges with two another adjacency edges (similarly to the 2-break operation [3]). Such pairs of cycles are then merged in a greedy manner. In practice, the described heuristic was not triggered for the most of the real datasets we describe below.

Analyzing ADBGs of two closely related genomes. For the sake of simplicity, we previously were assuming that de Bruijn graphs are built from the same genome using different values of k . Here we extend the proposed algorithm to the comparison of two closely related genomes. This comparison is more challenging as the genomes might contain small mutations or structural variations. Comparative genomics studies typically decompose genomes into sets of coarse synteny blocks to mask small-scale sequence polymorphisms [24, 8]. We apply a similar principle to decompose the assembly graph edges into syntenic segments.

Given two ADBGs AG_1 and AG_2 , we perform local pairwise alignment of all unique edges from AG_1 to all unique edges of AG_2 using `minimap2` [15]. We discard alignments that are shorter than 50 Kbp to focus on large-scale structure. Afterwards, colinear alignments (that appear in the same order and orientation in both genomes) are chained into synteny blocks [11]. Similarly to the described above, if an edge contains multiple synteny blocks, we split this edge into a path of new edges, each corresponding to a single synteny block. Since each unique edge now contains one synteny block, this defines the syntenic edge pairs. It is possible that some unique edges might not be aligned because the corresponding sequence is missing in the other genome. Such edges are flagged as *inserted*, and are logically equivalent to repetitive edges in the MSP decomposition. To be robust to possible chimeric connections in the graphs, we also require nucleotide distance between the chained alignments as well as the distance between the colinear synteny blocks to be less than 1 Mb.

Annotated repeat graphs. Recently, it has been shown how to apply repeat graphs for long-read assembly [12]. Similarly to de Bruijn graphs, repeat graphs reveal the repeat structure of the genome, which makes them suitable for the synteny paths analysis. Instead of relying on exact k -mer matches, repeat graphs are built from local sequence alignments and thus are more robust to high error rate of long reads. By design, repeat graphs can hide small sequence polymorphisms, and are well suited for the analysis of large structural variations.

The described algorithms have been implemented into a tool named **Asgan** (**A**ssembly **G**raphs **A**nalyzer). **Asgan** takes two annotated de Bruijn or repeat graphs as input (in the GFA format), and outputs synteny paths visualized using `Graphviz` along with various statistics. The `Flye` assembler [12] produces annotated repeat graphs as a part of its output. For assemblers based on the overlap graph approach, such as `Canu` [13], we construct repeat graphs by running the `Flye` graph construction algorithm on the assembled contigs (that contain flanking repeat sequences). The implementation is freely available at: <https://github.com/epolevikov/Asgan>.

4 Results

Comparing `Flye` and `Canu` assembly graphs using bacterial datasets from the NCTC collection. First, we illustrate the application of synteny paths for comparison of graphs produced by different assembly methods. We focus on the assembly graphs reconstructed from long-read sequencing data because they are typically less tangled and easier to visualize, comparing to short Illumina assemblies. We used two assemblers, `Canu` [13] and `Flye` [12] for the comparison, because they represent two different approaches for repeat resolution. We did not attempt to evaluate the `GTED` and `EMD` implementations since they were not designed for overlap / repeat graphs input (as discussed above).

We used `Flye` and `Canu` to assemble 21 bacterial genomes from the National Collection of Type Cultures (<https://www.sanger.ac.uk/resources/downloads/bacteria/nctc/>). When running `Flye`, we turned off the `Trestle` module (that resolves extra unbridged repeats) to make `Flye` graphs potentially more tangled. Each dataset contains P5C3 PacBio reads

■ **Table 1** Comparison of the assembly graphs produced by Flye and Canu using 21 bacterial genomes from the NCTC collection. Edges shorter than 50 Kb were ignored. Only one orientation (forward or reverse) of each edge and connected components was counted. Fourteen concordant datasets (where each connected component is covered by a single synteny path) are marked with the * symbol.

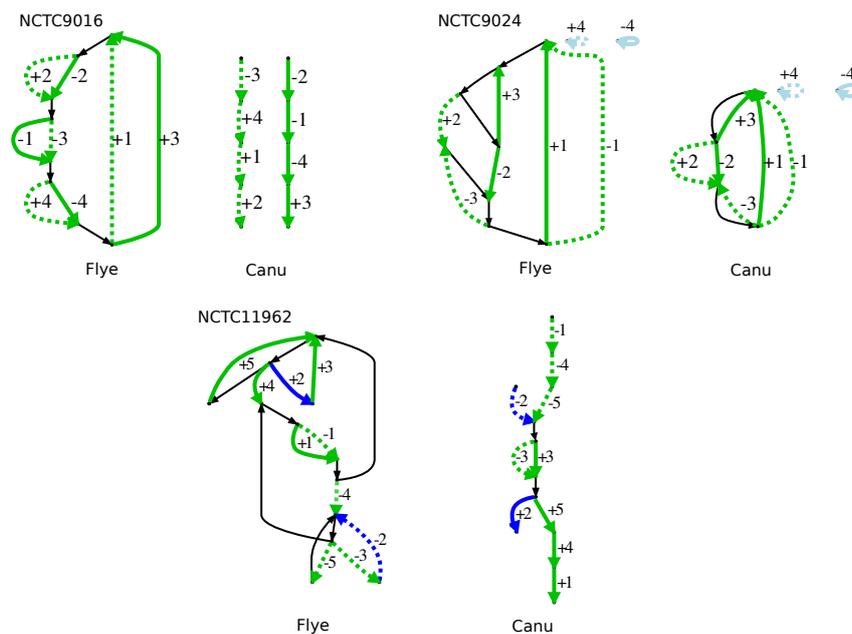
ID	Flye		Canu		Flye & Canu	
	Unique edges	Connected components	Unique edges	Connected components	Unique edges	Connected components
4450	7	1	8	6	12	8
6134	1	1	2	2	3	2
8333	8	1	3	1	8	2
8781	10	1	4	1	11	2
9657	6	2	6	6	9	7
11692	6	1	4	3	6	3
11962	4	1	3	1	5	2
5052*	7	2	5	2	7	2
7921*	3	1	1	1	4	1
9002*	1	1	1	1	2	1
9006*	7	1	2	1	7	1
9007*	2	1	2	1	2	1
9012*	4	1	2	1	4	1
9016*	4	1	1	1	4	1
9024*	4	2	4	2	4	2
9103*	3	3	3	3	3	3
9964*	4	1	1	1	4	1
10864*	6	1	1	1	7	1
11022*	4	1	1	1	4	1
12158*	3	2	2	2	3	2
12993*	2	2	2	2	2	2

with coverage varying from $70\times$ to $271\times$. For each assembly, the number of unique edges (longer than 50 Kb) ranged from 1 to 11 for Flye, and 1 to 8 for Canu. Since the assembly graphs were double-stranded, we only counted one orientation (forward or reverse) of each edge. Similarly, we counted only one orientation of each connected component, which might represent a bacterial chromosome, a plasmid, or a mixture of them.

The results are shown in the Table 1. We call two assembly graphs *concordant*, if each connected component in both graphs is covered by a single synteny path. Fourteen out of 21 datasets were concordant, which is expected for assemblies generated from the same set of reads. In 8 out of 14 concordant datasets, Flye had more unique edges than Canu. This suggests that in these 8 datasets Canu resolved more repeats than Flye (as expected due to not using Trestle). Figure 5 shows examples of concordant and discordant assembly graphs.

In five out of seven datasets that were not concordant, Canu graphs, but not Flye graphs contained “dead end” edges (disconnected from the rest of the graph from either beginning or end). These missing connections might have contributed to the increased number of synteny paths. In two remaining cases, both assemblers produced tangled graphs with complex unresolved repeats.

Asgan took less than a minute of wall clock time and less than 100 Mb of RAM to process each bacterial dataset. The running time was dominated by the minimap2 alignments (using three threads).



■ **Figure 5** (Top left) Comparison of bacterial assembly graphs produced by Flye and Canu for the NCTC9016 dataset. Synteny paths are shown in green. Complement paths are dashed. In the Canu graph, two strands of one connected component are separated, while in the Flye graph both strands are merged into one component through the unresolved repeats. Although the assembly produced by Flye is more fragmented, a single synteny path $(-2, -1, -4, +3)$ reveals the structural similarity. (Top right) Synteny paths decomposition for the NCTC9024 dataset. Both assemblies resulted into tangled repeat graphs with different number of unresolved repeats. Two synteny paths cover each connected component in full: $(+1, +3, -2)$ and $(+4)$. (Bottom) Two synteny paths (green and blue) in the NCTC11962 dataset reveal structural inconsistencies between the Flye and Canu graphs.

Comparing assembly graphs of 15 *Drosophila* genomes. Synteny paths decomposition could also be applied to structurally compare assemblies of different genomes. We used Flye to assemble 15 different *Drosophila* species from ONT data [20]. Read coverage was varying from 23x to 44x, read length N50 was varying from 7 Kb to 28 Kb. Flye produced contiguous assemblies with N50 over 1Mb for 14 out of 15 datasets (Table 2).

First, we applied synteny paths to compare each assembly against the high-quality *Drosophila melanogaster* reference genome. Table 2 shows various assembly statistics, as well as the number of synteny blocks and synteny paths for each genome. The most distant species exhibit more than 10% nucleotide divergence, and pairwise alignments were problematic to compute. Instead, we estimated the evolutionary divergence of each genome from *D. melanogaster* as the number of matched 15-mers within the synteny blocks (reported by minimap2). K -mer survival rate could be estimated from point mutation rate d as: $(1 - d)^k$. For example, 0.87 k -mer match rate of the *D. melanogaster* assembly corresponds to approximately 1% base difference, which is typical for the current ONT assemblies [12]. Synteny block coverage (total length of all synteny blocks divided by the assembly size) was varying from 94% for the *D. melanogaster* assembly to low 2% for the *D. virilis* assembly, which highlights the challenge of recovering synteny blocks between diverged genomes.

As expected, we observed an increase in the number of synteny blocks, as the distance between an assembly and the reference increases, excluding the three most distant genome which had reduced the number of blocks due to the difficulties in alignment (Table 2). On average, the number of synteny paths was 18% smaller than the number of synteny blocks within each genome (varying from 0% to 56%). Under the assumption that the breakpoint

reuse rate is low, the length of large synteny blocks should be exponentially distributed [24]. Thus, N50 contiguity metric could be used as a complement to the number of synteny blocks or paths (defined as the largest possible number L , such that all blocks/paths of length L or longer cover at least 50% of total blocks/paths length). Table 2 shows that synteny paths N50 correlates with the evolutionary distance between the compared genomes and also robust to the decreased synteny blocks coverage.

■ **Table 2** *Drosophila* assembly graphs statistics and comparison against the *D. melanogaster* reference. Genomes are sorted according to the k-mer identity, computed as the number of matching 15-mers against the *D. melanogaster* reference. Synteny blocks coverage was calculated as the total length of the blocks divided by the total assembly length. Only the sequences that are contained in a component with at least one synteny block were considered.

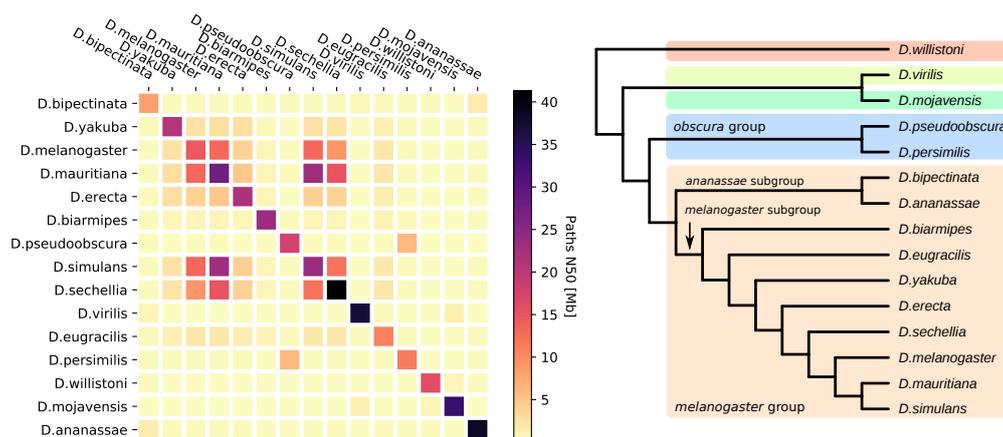
Genome	Asm. size, Mb	Asm. N50, Mb	Blocks, No.	Paths, No.	Blocks N50, Mb	Paths N50, Mb	Blocks cov.	K-mer identity
<i>D. melanogaster</i>	137.0	11.7	78	39	11.7	14.8	94%	0.87
<i>D. mauritiana</i>	128.1	13.9	32	16	13.5	20.8	88%	0.50
<i>D. simulans</i>	124.8	21.3	16	13	20.9	20.9	90%	0.48
<i>D. sechellia</i>	139.6	20.0	30	27	12.1	12.1	81%	0.47
<i>D. yakuba</i>	146.1	16.5	77	72	2.4	2.4	77%	0.30
<i>D. erecta</i>	131.8	12.4	46	39	4.3	4.8	86%	0.30
<i>D. eugracilis</i>	159.5	2.1	159	103	1.3	2.0	69%	0.19
<i>D. biarmipes</i>	168.8	4.9	194	181	0.8	0.9	64%	0.15
<i>D. ananassae</i>	176.8	3.1	286	281	0.4	0.42	40%	0.09
<i>D. bipectinata</i>	179.2	0.8	317	266	0.28	0.39	35%	0.09
<i>D. pseudoobscura</i>	151.8	2.7	238	230	0.24	0.28	28%	0.07
<i>D. persimilis</i>	156.4	2.3	237	229	0.26	0.28	26%	0.07
<i>D. willistoni</i>	187.1	2.9	27	26	0.16	0.18	2%	0.06
<i>D. mojavensis</i>	162.5	7.6	19	19	0.08	0.08	1%	0.06
<i>D. virilis</i>	161.7	10.8	82	36	0.08	0.08	2%	0.06

To further illustrate that synteny paths reveal structural variations and are robust to contig fragmentation, we computed synteny paths between all pairs of assemblies. On average, the number of synteny paths in each dataset was 20.4% smaller than the number of synteny blocks. We defined the similarity between two assemblies as $S = 1 - N50_{syn}/N50_{max}$, where $N50_{syn}$ corresponds to the synteny paths N50, and $N50_{max}$ is the maximum synteny paths N50 among all assembly pairs. Given the similarity matrix, we used Neighbor-Joining algorithm [31] to infer the phylogenetic tree of all assemblies (Figure 6). The reconstructed tree was structurally consistent with the tree reconstructed based on genomic mutation distances [6].

The running time of Asgan was less than six minutes of wall clock time for each pair of *Drosophila* assemblies. The typical RAM usage was varying from 2 Gb to 6 Gb.

5 Discussion

In this work, we presented Asgan – an algorithm for comparison and evaluation of assembly graphs produced by various assembly approaches. We introduced the concept of synteny paths, which are similar to the synteny blocks abstraction, but take advantage of assembly graph structure. The result of the minimum synteny paths decomposition problem returns the minimal set of synteny paths that traverse all unique edges in two graphs in the same order. We proved that the exact solution of the MSP decomposition is NP-hard and provided a heuristic algorithm that scales to eukaryotic assembly graphs.



■ **Figure 6** (Left) Heatmap showing synteny paths N50 between all pairs of assemblies. (Right). Phylogenetic tree reconstructed based on normalized synteny paths N50 using the Neighbor-Joining method.

We used synteny paths to compare the assembly graphs produced by Canu and Flye from the 21 bacterial datasets. In 14 out of 21 cases, each graph component was covered by a single synteny path, suggesting that both assemblers produced valid assembly graphs with no missing connections. In the remaining seven cases, some graph components were covered by multiple synteny paths, which is not expected for a unichromosomal genome. This reveals possible missing or erroneous connections in the assembly graphs, where the problematic regions are highlighted by synteny path breakpoints. Note that this type of assembly error could not be captured by the alignment of contigs to a reference genome. Thus, the proposed analysis could be useful for validation and debugging of assembly graphs produced by different approaches.

Synteny analysis is a powerful technique for comparative genomics, and a number of tools for synteny blocks decomposition has been developed [29, 28, 21, 8]. However, most of these tools were designed for comparing complete genomes, and it was recently shown [17] that their performance deteriorates when analysing fragmented assemblies. In contrast, synteny paths are taking advantage of the assembly graph structure, and are more robust to the contig fragmentation. We have demonstrated this by analysing the assemblies of 15 *Drosophila* species. As expected, the structure of the reconstructed synteny paths was correlated with mutation distances between the genomes. Using the synteny paths length distribution as a similarity measure, we reconstructed the phylogenetic tree of the analysed species, which was in agreement with the accepted *Drosophila* taxonomy.

It should be possible to extend the synteny paths approach to the comparison of multiple assembly graphs, which could be useful for assembly reconciliation [37, 1]. Intuitively, if one can prove that given assembly graphs share only one optimal MSP solution, this solution will likely correspond to the correct genomic path. However, it is currently unknown how to enumerate all optimal / suboptimal MSP solutions efficiently.

References

- 1 Sergey S Aganezov and Max A Alekseyev. CAMSA: a tool for comparative analysis and merging of scaffold assemblies. *BMC bioinformatics*, 18(15):496, 2017.
- 2 Dmitry Antipov, Anton Korobeynikov, Jeffrey S McLean, and Pavel A Pevzner. hybridSPAdes: an algorithm for hybrid assembly of short and long reads. *Bioinformatics*, 32(7):1009–1015, 2015.

- 3 Pavel Avdeyev, Shuai Jiang, Sergey Aganezov, Fei Hu, and Max A Alekseyev. Reconstruction of ancestral genomes in presence of gene gain and loss. *Journal of Computational Biology*, 23(3):150–164, 2016.
- 4 Anton Bankevich, Sergey Nurk, Dmitry Antipov, Alexey A Gurevich, Mikhail Dvorkin, Alexander S Kulikov, Valery M Lesin, Sergey I Nikolenko, Son Pham, Andrey D Prjibelski, et al. SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *Journal of computational biology*, 19(5):455–477, 2012.
- 5 Ali Ebrahimpour Boroojeny, Akash Shrestha, Ali Sharifi-Zarchi, Suzanne Renick Gallagher, S Cenk Sahinalp, and Hamidreza Chitsaz. GTED: Graph traversal edit distance. In *Research in Computational Molecular Biology*, pages 37–53. Springer, 2018.
- 6 Drosophila 12 Genomes Consortium et al. Evolution of genes and genomes on the Drosophila phylogeny. *Nature*, 450(7167):203, 2007.
- 7 Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17:449–467, 1965.
- 8 Cristina G Ghiurcuta and Bernard ME Moret. Evaluating synteny for improved comparative studies. *Bioinformatics*, 30(12):i9–i18, 2014.
- 9 Zamin Iqbal, Mario Caccamo, Isaac Turner, Paul Flicek, and Gil McVean. De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nature genetics*, 44(2):226, 2012.
- 10 Govinda M Kamath, Ilan Shomorony, Fei Xia, Thomas A Courtade, and N Tse David. HINGE: long-read assembly achieves optimal repeat resolution. *Genome research*, 27(5):747–756, 2017.
- 11 W James Kent, Robert Baertsch, Angie Hinrichs, Webb Miller, and David Haussler. Evolution’s cauldron: duplication, deletion, and rearrangement in the mouse and human genomes. *Proceedings of the National Academy of Sciences*, 100(20):11484–11489, 2003.
- 12 Mikhail Kolmogorov, Jeffrey Yuan, Yu Lin, and Pavel A Pevzner. Assembly of long, error-prone reads using repeat graphs. *Nature biotechnology*, 37(5):540, 2019.
- 13 Sergey Koren, Brian P Walenz, Konstantin Berlin, Jason R Miller, Nicholas H Bergman, and Adam M Phillippy. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome research*, 27(5):722–736, 2017.
- 14 Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10 (8), pages 707–710, 1966.
- 15 Heng Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, 2018.
- 16 Yu Lin, Sergey Nurk, and Pavel A Pevzner. What is the difference between the breakpoint graph and the de Bruijn graph? *BMC genomics*, 15(6):S6, 2014.
- 17 Dang Liu, Martin Hunt, and Isheng J Tsai. Inferring synteny between genome assemblies: a systematic evaluation. *BMC bioinformatics*, 19(1):26, 2018.
- 18 Serghei Mangul and David Koslicki. Reference-free comparison of microbial communities via de Bruijn graphs. In *Proceedings of the 7th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 68–77. ACM, 2016.
- 19 Alla Mikheenko and Mikhail Kolmogorov. Assembly Graph Browser: interactive visualization of assembly graphs. *Bioinformatics*, 2019.
- 20 Danny E Miller, Cynthia Staber, Julia Zeitlinger, and R Scott Hawley. Highly contiguous genome assemblies of 15 Drosophila species generated using nanopore sequencing. *G3: Genes, Genomes, Genetics*, 8(10):3131–3141, 2018.
- 21 Ilya Minkin, Anand Patel, Mikhail Kolmogorov, Nikolay Vyahhi, and Son Pham. Sibelia: a scalable and comprehensive synteny block generation tool for closely related microbial genomes. In *International Workshop on Algorithms in Bioinformatics*, pages 215–229. Springer, 2013.
- 22 Supratim Mukherjee, Dimitri Stamatis, Jon Bertsch, Galina Ovchinnikova, Hema Y Katta, Alejandro Mojica, I-Min A Chen, Nikos C Kyrpides, and TBK Reddy. Genomes OnLine database (GOLD) v. 7: updates and new features. *Nucleic acids research*, 47(D1):D649–D659, 2018.

- 23 Eugene W Myers, Granger G Sutton, Art L Delcher, Ian M Dew, Dan P Fasulo, Michael J Flanigan, Saul A Kravitz, Clark M Mobarry, Knut HJ Reinert, Karin A Remington, et al. A whole-genome assembly of *Drosophila*. *Science*, 287(5461):2196–2204, 2000.
- 24 Pavel Pevzner and Glenn Tesler. Transforming men into mice: the Nadeau-Taylor chromosomal breakage model revisited. In *Proceedings of the seventh annual international conference on Research in computational molecular biology*, pages 247–256. ACM, 2003.
- 25 Pavel A Pevzner, Haixu Tang, and Glenn Tesler. De novo repeat classification and fragment assembly. *Genome research*, 14(9):1786–1796, 2004.
- 26 Pavel A Pevzner, Haixu Tang, and Michael S Waterman. An Eulerian path approach to DNA fragment assembly. *Proceedings of the national academy of sciences*, 98(17):9748–9753, 2001.
- 27 Adam M Phillippy. New advances in sequence assembly. *Genome Research*, 27:xi–xiii, 2017.
- 28 Sebastian Proost, Jan Fostier, Dieter De Witte, Bart Dhoedt, Piet Demeester, Yves Van de Peer, and Klaas Vandepoele. i-ADHoRe 3.0—fast and sensitive detection of genomic homology in extremely large data sets. *Nucleic acids research*, 40(2):e11–e11, 2011.
- 29 Christian Rödelsperger and Christoph Dieterich. CYNTENATOR: progressive gene order alignment of 17 vertebrate genomes. *PloS one*, 5(1):e8861, 2010.
- 30 Yana Safonova, Anton Bankevich, and Pavel A Pevzner. dipSPAdes: assembler for highly polymorphic diploid genomes. *Journal of Computational Biology*, 22(6):528–545, 2015.
- 31 Naruya Saitou and Masatoshi Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425, 1987.
- 32 Michael C Schatz, Arthur L Delcher, and Steven L Salzberg. Assembly of large genomes using second-generation sequencing. *Genome research*, 20(9):1165–1173, 2010.
- 33 Alex Shlemov and Anton Korobeynikov. PathRacer: racing profile HMM paths on assembly graph. *BioRxiv*, page 562579, 2019.
- 34 Mitchell R Vollger, Philip C Dishuck, Melanie Sorensen, AnneMarie E Welch, Vy Dang, Max L Dougherty, Tina A Graves-Lindsay, Richard K Wilson, Mark JP Chaisson, and Evan E Eichler. Long-read sequence and assembly of segmental duplications. *Nature methods*, 16(1):88, 2019.
- 35 Ryan R Wick, Louise M Judd, Claire L Gorrie, and Kathryn E Holt. Unicycler: resolving bacterial genome assemblies from short and long sequencing reads. *PLoS computational biology*, 13(6):e1005595, 2017.
- 36 Ryan R Wick, Mark B Schultz, Justin Zobel, and Kathryn E Holt. Bandage: interactive visualization of de novo genome assemblies. *Bioinformatics*, 31(20):3350–3352, 2015.
- 37 Aleksey V Zimin, Douglas R Smith, Granger Sutton, and James A Yorke. Assembly reconciliation. *Bioinformatics*, 24(1):42–45, 2007.