Kermit: Guided Long Read Assembly using Coloured Overlap Graphs

Riku Walve[1]
Department of Computer Science, Helsinki Institute for Information Technology HIIT, University of Helsinki, Helsinki, Finland
riku.walve@helsinki.fi
https://orcid.org/0000-0003-0397-003X

Pasi Rastas[2]
Institute of Biotechnology, University of Helsinki, Helsinki, Finland
pasi.rastas@helsinki.fi

Leena Salmela[1]
Department of Computer Science, Helsinki Institute for Information Technology HIIT, University of Helsinki, Helsinki, Finland
leena.salmela@helsinki.fi
https://orcid.org/0000-0002-0756-543X

Abstract

With long reads getting even longer and cheaper, large scale sequencing projects can be accomplished without short reads at an affordable cost. Due to the high error rates and less mature tools, *de novo* assembly of long reads is still challenging and often results in a large collection of contigs. Dense linkage maps are collections of markers whose location on the genome is approximately known. Therefore they provide long range information that has the potential to greatly aid in *de novo* assembly. Previously linkage maps have been used to detect misassemblies and to manually order contigs. However, no fully automated tools exist to incorporate linkage maps in assembly but instead large amounts of manual labour is needed to order the contigs into chromosomes. We formulate the genome assembly problem in the presence of linkage maps and present the first method for guided genome assembly using linkage maps. Our method is based on an additional cleaning step added to the assembly. We show that it can simplify the underlying assembly graph, resulting in more contiguous assemblies and reducing the amount of misassemblies when compared to *de novo* assembly.

2012 ACM Subject Classification Applied computing → Sequencing and genotyping technologies, Mathematics of computing → Graph theory

Keywords and phrases Genome assembly, Linkage maps, Coloured overlap graph

Digital Object Identifier 10.4230/LIPIcs.WABI.2018.11

1 Introduction

High-throughput, second generation, sequencing technologies made large scale *de novo* assemblies possible and commonplace. Their short read lengths however still pose a major problem to this day. Third generation, long read sequencing technologies, such as single-molecule real-time sequencing (SMRT) and Oxford Nanopore (ONT) are promising but their

[1] Supported by Academy of Finland (grants 308030 and 314170).
[2] Supported by Jane and Aatos Erkko Foundation.

error rates have made assemblies difficult in practice. Therefore most long read assemblers include an error correction step to reduce the error rate.

Recently introduced Minimap-Miniasm workflow [12] has given new insight towards an error correction-free pipeline for long read assemblies. Minimap finds useful overlaps in long reads with high error rates and makes long read-only assembly projects practical and highly efficient. However finding the overlaps between reads with high error rates becomes impractical in very large data sets and splitting the reads into smaller sets is not possible without additional information on the reads.

Compared to *de novo* assembly, where the only available information for the assembly are the reads, *guided genome assembly* has additional data that gives information on the positions of the reads. Normally this additional data is a reference genome of a closely related species. The reads can be aligned to the reference genome, which results in a linear ordering of the reads. This clearly makes it easier to then assemble the reads.

Directly guiding the assembly this way makes it hard to get an assembly that is higher quality than the reference. This becomes an issue when no high quality reference genome exists, or if the donor genome deviates too far from the reference genome. In this paper, we guide the assembly using linkage maps.

Linkage maps (also called genetic linkage maps or genetic maps) [17] are a useful technique to orient and place contigs within a chromosome and to detect misassembled contigs. The linkage maps themselves consists of variable genetic markers, typically called relative to some draft assembly. The markers are derived from a set of variations, such as single-nucleotide variations (SNVs), found from a sequenced *cross*, a population of related individuals. SNVs that are close to each other in the genome are more likely to be inherited together than SNVs that are more distant from each other. Linkage maps can therefore be constructed by genotyping the individuals in the cross and examining the probabilities of SNVs being inherited together. With a large enough set of variations in population, the linkage map becomes dense enough to potentially place long reads directly.

In this paper, we formulate the genome assembly problem in the presence of linkage maps. We propose the first method to directly use linkage maps in genome assembly. Our method disentangles the overlap graph by removing false edges based on the linkage map. Our experimental results show that the method is able to simplify the overlap graph and we further show that our method decreases the number of misassemblies and improves the N50 statistic as compared to *de novo* assembly without linkage maps.

This linkage map-guided genome assembly can be seen as a generalisation of reference-guided genome assembly. With a hypothetical linkage map of evenly spaced markers, the linkage map-guided assembly becomes essentially reference-guided assembly as each read can be placed on the genome unambiguously.

We have implemented our method in a tool called Kermit which is freely available at `https://github.com/rikuu/kermit`.

2 Related Work

Despite the emergence of long read sequencing technologies like PacBio and ONT and the development of long read assemblers [11, 12, 7, 10], auxiliary long range data is still needed to organise the resulting scaffolds or contigs to chromosomes for large eukaryotic genomes. Depending on the characteristics of the species of interest such data may include optical mapping data, Hi-C data, or linkage maps.

Linkage maps consist of a set of markers, typically SNVs, on a genome. The location of the markers with respect to each other is at least approximately known. Typically a linkage map successfully assigns the markers to chromosomes and a partial order of the markers within each chromosome is known. Markers can be localised on contigs or scaffolds and the linkage map can be used to correct the scaffolds and to order them into chromosomes [1, 18, 4]. However, currently the ordering of scaffolds based on a linkage map is a time-consuming, error-prone and mostly manual process as no fully automated tools exist [8]. Furthermore, all current tools use linkage maps as a post processing step after assembly, whereas our work integrates the linkage maps already in the assembly phase.

Chromonomer [5] attempts to correct and orient scaffolds based on a linkage map. It first finds a non-conflicting set of markers in the linkage map. It then assigns orientation to scaffolds containing more than one recombination point and splits scaffolds if they conflict with the linkage map. Finally, Chromonomer produces a visualisation of the linkage map and the scaffolds. Another tool that facilitates the visualisation of linkage maps and corresponding genome assemblies is ArkMAP [15]. However, ArkMAP focuses on visual exploration including cross species comparisons and as such does not support automatic ordering of scaffolds into chromosomes.

Similarly to linkage maps also optical mapping data can be used for improving genome assemblies. Also for optical mapping data the main focus has been to use the optical map in a post processing step after genome assembly. AGORA [14] is one of the few methods integrating optical map data with genome assembly. It is a de Bruijn graph based assembler that uses optical map data to eliminate alternative paths that are not consistent with the optical map. The method by Alipanahi et al. [2] for disentangling the de Bruijn graph using optical map data is more related to our work. They map the long reads to the genome wide optical map and use this mapping to produce a positional de Bruijn graph which resolves most ambiguities in the de Bruijn graph. In our work we similarly first get a preliminary ordering of long reads based on a linkage map and then use this ordering to disentangle the overlap graph.

3 Definitions

3.1 De Novo Assembly

De novo assembly, the problem of assembling a genome given only a set of reads, has historically been given solutions in two different categories. The de Bruijn graph-based (DBG) assembly algorithms, such as SPAdes [3], and Overlap-Layout-Consensus (OLC) algorithms, such as Canu [11] and Miniasm [12].

OLC-assemblers attempt to first find pair-wise overlaps between reads which is the bottleneck of this approach. From these overlaps, a directed graph of the set of reads, called an overlap graph, is laid out by assigning edges between reads if an overlap is observed. The graph is then simplified by removing all transitive edges. Ideally the graph would be simple enough to unambiguously spell a genome assembly by following the edges from one side of the graph to the other. This is actually never the case and some sophisticated method for transforming the graph as close as possible to such a case is the goal of the layout step.

DBG-assemblers attempt to simplify the entire process by not looking for overlaps between the reads. Instead they take all possible $k-1$ length overlaps between substrings of length k from the set of reads to construct a de Bruijn graph. The methods for finding the genome from the overlap graph in the OLC setup mostly apply here too.

Though clearly simpler than the OLC assembly algorithms, the effectiveness of further splitting the reads somewhat diminishes the usefulness of using long reads for genome assembly. In this paper we are only looking at the OLC category of assembly algorithms, more specifically, we are looking at Miniasm [12], a very simplistic and highly efficient implementation of the ideas in OLC-assemblers.

Unlike traditional OLC-assemblers, Miniasm only implements the overlap and layout steps. This reduces the base-level quality of the resulting assembly but does not greatly affect the large scale structure. It also does not use any overly sophisticated method for finding the genome in the overlap graph.

To choose an assembly path, Miniasm finds *unitigs* which are maximal non-branching paths in the overlap graph. Such paths are simple to find and intuitively give the maximal unambiguous and nonrepetitive sequences that the overlap graph can spell without changing the graph. The problem of finding the unitigs from the overlap graph can be stated as follows:

▶ **Definition 1.** *Unitig assembly problem.* Given a directed graph $G = (V, E)$, find all maximal paths $P = v_1 v_2 \cdots v_n$ such that

$$\forall v_i \in \{v_1, \ldots, v_{n-1}\}, \deg^+ v_i = \deg^- v_{i+1} = 1.$$

Unitigs can be efficiently found by first looking for a vertex with either zero or more than one incoming edge and exactly one outgoing edge. Following the outgoing edges until we find a vertex with more than one incoming edge or zero or more than one outgoing edge constructs a path spelling a unitig.

3.2 Guided Assembly

Reference-guided genome assembly can be done by aligning the read set to a reference and partitioning the reads based on the alignments into smaller, similar sets of reads [19]. The small sets are then assembled into contigs and later the set of contigs are assembled into super-contigs.

For linkage map-guided assembly, we partition the reads based on the linkage map. A linkage map is a set of markers M which are assigned to a set of bins B. Each marker $m \in M$ belongs to a single bin $b \in B$. The bins are further assigned to chromosomes and within each chromosome the order of the bins is given.

We assume now that each read has been assigned a set of bins based on the linkage map. Now this coarse-grained ordering tells if a subset of reads clearly belong before or after another subset of reads. We encode this ordering into the overlap graph by assigning each vertex a set of colours representing the bins, resulting in a *coloured overlap graph.*

A coloured overlap graph is thus a directed graph $G = (V, E)$ accompanied by a colouring $c : V \to \mathcal{P}(\mathbb{N})$, where $\mathcal{P}(\mathbb{N})$ is the power set of natural numbers. In the coloured overlap graph we define the *colour consistent indegree* $\deg_c^-$ of a vertex v_i to be the number of in-neighbours of v_i that have at least one colour that is the same as or adjacent to a colour of v_i, i.e.

$$\deg_c^- v_i = |\{v_j | (v_j, v_i) \in E \text{ and } \exists c_j \in c(v_j), c_i \in c(v_i) \text{ s.t. } |c_j - c_i| \leq 1\}|.$$

Similarily we define the *colour consistent outdegree* $\deg_c^+$ as

$$\deg_c^+ v_i = |\{v_j | (v_i, v_j) \in E \text{ and } \exists c_j \in c(v_j), c_i \in c(v_i) \text{ s.t. } |c_j - c_i| \leq 1\}|.$$

The problem of guided assembly can then be modelled as finding a *rainbow path* in the coloured overlap graph. A rainbow path is a path such that no two vertices have the same colour [6]. We use a modified variant of rainbow paths; we allow paths to reuse a colour in consecutive vertices and we require the colours of a path to be consecutive and increasing. I.e. colour i must be followed by colour $i+1$ on the path. A more formal definition of this assembly problem can be stated as:

▶ **Definition 2.** *Guided unitig assembly problem.* Given a directed graph $G = (V, E)$, and a colouring $c : V \to \mathcal{P}(\mathbb{N})$, find all maximal paths $P = v_1 v_2 \cdots v_n$ such that

$$\forall v_i \in \{v_1, \ldots, v_{n-1}\} \deg_c^+ v_i = \deg_c^- v_{i+1} = 1$$

and

$$\forall c \in c(v_i), c \geq \max\left(c(v_{i-1})\right).$$

We note that the above definition only recovers forward strand paths in the coloured overlap graph. However due to the structure of the graph, for each reverse strand path there also exists a corresponding forward strand path in the graph.

Rather than modifying the layout step of the OLC-assembly pipeline, we implement a graph cleaning step, which attempts to remove edges that make unitigs not be rainbow paths.

4 Methods

4.1 Overview of Our Method

Constructing a linkage map [17] involves generating a draft assembly and localising the markers, which are typically single nucleotide variations, on this draft assembly. The markers are then further placed into bins based on the hereditary patterns seen in a cross-bred population of individuals. The bins are assigned to chromosomes and the order of the bins within each chromosome is known. Thus the markers give a partial order of the genome.

The input to our method is the draft assembly on which the linkage map was built, the linkage map, and long reads. We first use Minimap2 [13] to map the long reads on the draft assembly to assign colours (i.e. bin numbers) to the long reads. Miniasm [12] is then used to build the overlap graph of the long reads and missing colours are propagated in the overlap graph. The overlap graph is then cleaned based on the colour assignments of the reads. Finally the unitigs are read from the cleaned overlap graph. Figure 1 shows an overview of our method. The colouring, propagation, and cleaning steps are discussed in more details in the following subsections.

4.2 Colouring

To colour the underlying overlap graph, we map all the reads against the draft genome using Minimap2 [13] and store the longest mappings for each read. We extend each of the longest mappings into naive linear "alignments" by stretching the start and end positions to cover the entire read. As the insertion errors are the dominant error type [20], we limit the number of characters by which we extend the mappings (by default we use a limit of 250 bp).

All the extended mappings are then stored in a simple query structure, where each chromosome in the reference genome is split into equal length blocks. For each marker in the linkage map, we query the index for reads that overlap with the marker and assign every overlapping read the colour the marker in the linkage map belongs to.

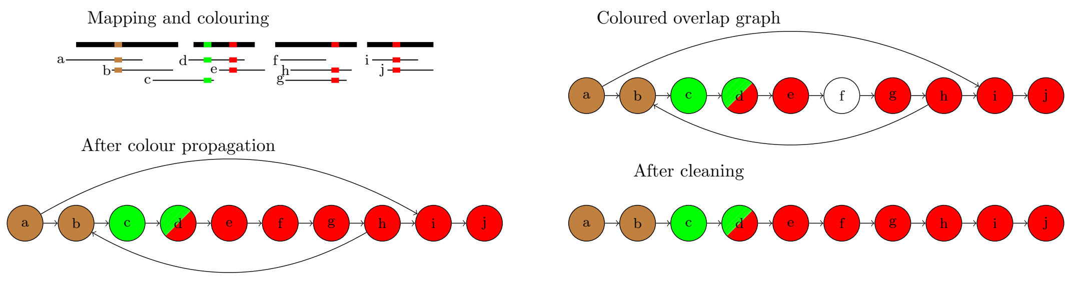

Figure 1 Overview of our method. First the reads are mapped to the draft assembly and assigned colours (top left). Each colour represents one bin in the linkage map. In this example we have three bins (brown, green and red) and the ordering of the bins is brown < green < red. All bins belong to the same chromosome. Miniasm is then used to construct the overlap graph which is augmented with the colours. Next vertex f is coloured through the colour propagation process. Finally we remove the edges (a, i) and (h, b) because they have inconsistent colourings.

As there can still be reads with no mapping to the reference, we attempt to colour them using the overlap graph. For each uncoloured vertex u we find the set of coloured vertices $V_c(u)$ that are reachable from u by paths that traverse only uncoloured vertices. The uncoloured vertex u is then given all the colours that the vertices in $V_c(u)$ have, i.e. $c(u) = \bigcup_{v \in V_c(u)} c(v)$. The set $V_c(u)$ can be found by a breadth first search on the graph starting at vertex u.

If a coloured vertex is too far from the uncoloured vertex, we get an over-approximation of the colour for the vertex. To reduce this effect, we apply a limit to the depth of the search (by default 10). We can also get conflicting colours from the propagation. If there are missing colours between the propagated colours, we are skipping some part of the genome entirely. As we cannot use the colouring to usefully clean the graph, we simply remove the read from the graph entirely.

4.3 Graph Cleaning

To make sure any unitig path is a valid rainbow path, we remove any edges between vertices with inconsistent colourings from the overlap graph. If the vertices share a colour, the colourings are always consistent as they can be merged without affecting the connectivity of the colours in the graph. Edges between vertices with different colours are inconsistent if there are no adjacent colours between the vertices. Such an edge could never be part of a rainbow path because the path would not have consecutive colours.

Therefore we define an edge (v_i, v_j) to be *inconsistently coloured* if there are no colours $c_i \in c(v_i)$ and $c_j \in c(v_j)$ such that the colours c_i and c_j would be the same or consecutive, i.e. $|c_i - c_j| \leq 1$. Our graph cleaning step remove

5 Results

We implemented our method in a tool called Kermit. Kermit uses Miniasm [12] to find the overlaps between the reads and to perform the layout step to find unitigs. In between these two steps Kermit colours the vertices of the overlap graph and cleans the graph by removing inconsistently coloured edges as explained in the previous section. We compared Kermit to the Miniasm pipeline [12]. Both the overlap and the mapping steps were done using Minimap2 v2.9 [13], an improved implementation of Minimap. All experiments were run on 32 GB RAM machines equipped with 8 cores.

Table 1 Summary of read data used in the experiments.

Data set	Reads	Total bases	Coverage	Accession
S. cerevisiae (simulated)	36,639	486,048,334	39.98	simulated
C. elegans (simulated)	296,230	3,930,689,562	39.99	simulated
S. cerevisiae	52,208	690,899,144	56.83	PacBio DevNet[1)]
C. elegans	740,776	8,118,404,281	82.59	PacBio DevNet[2)]
H. erato	10,818,653	27,094,241,328	60.89	SRR3476970 SRR4039325
B. pendula	1,898,360	19,032,363,776	49.71	ERR2003767 ERR2003768

1) `https://github.com/PacificBiosciences/DevNet/wiki/Saccharomyces-cerevisiae-W303-Assembly-Contigs`
2) `https://github.com/PacificBiosciences/DevNet/wiki/C.-elegans-data-set`

Table 2 Summary of linkage maps used in the experiments.

Data set	Markers	Marker density	Bins	Bin density	Reference
S. cerevisiae (simulated)	100,009	0.008	19,283	0.002	simulated
C. elegans (simulated)	750,004	0.008	162,601	0.002	simulated
H. erato	2,781,314	0.007	145,863	0.002	Van Belleghem et al. [4] Generated by LepMap3 [17]
B. pendula	2,979,993	0.007	925,123	0.002	Salojärvi et al. [18]

5.1 Data

We performed experiments both on simulated and real data. All data used in the experiments is summarized in Tables 1 and 2.

First we performed experiments using both simulated reads and a simulated linkage map for *S. cerevisiae* and *C. elegans*. We simulated reads using SimLoRD [20] to a coverage of 40x, sampling read lengths from a real PacBio data set with shortest ($<$ 10,000bp) reads filtered out. To generate a simulated linkage map, markers were randomly placed on the reference and binned such that the bins are separated by at least 200bp. The number of markers was chosen to give a marker density similar to real linkage maps. The distance for binning is arbitrary; as real linkage maps are based on fragmented and misassembled draft genomes, the bins are rarely contiguous. The chosen values give similar densities of bins as the real linkage maps as shown in Table 2. These experiments on simulated read and linkage map data allow us to evaluate the colouring and cleaning steps because the origin of each read is known.

To understand how our method performs on real data, we first ran experiments using real PacBio reads for *S. cerevisiae* and *C. elegans* but still using the simulated linkage map as no real linkage maps exist for these species. Good reference genomes are available for these genomes so we could evaluate also the correctness of the resulting assemblies on these data sets. Finally to show that our method works on real linkage maps, we ran experiments on real PacBio reads and real linkage maps for *H. erato* and *B. pendula*. The genomes for these species are in draft stage so we could not reliably measure the correctness of these assemblies.

Table 3 Number of reads fully inside and outside their acceptable colour ranges.

	Reads inside	Reads outside
S. cerevisiae	36,562 (99.79%)	31 (0.08%)
C. elegans	296,134 (99.97%)	3 (0.001%)

Table 4 Number of edges supported by the positions the reads were simulated from. Graphs marked cleaned are also using the graph cleaning steps that are already implemented in Miniasm.

	Graph	True edges	False edges
S. cerevisiae	Miniasm	76,538 (99.39%)	466 (0.60%)
	Kermit	76,518 (99.93%)	52 (0.07%)
	Miniasm cleaned	7,146 (99.92%)	6 (0.08%)
	Kermit cleaned	7,114 (100.0%)	0 (0.0%)
C. elegans	Miniasm	668,012 (99.80%)	1,306 (0.19%)
	Kermit	667,970 (99.99%)	58 (0.003%)
	Miniasm cleaned	60,416 (99.95%)	28 (0.05%)
	Kermit cleaned	60,356 (99.997%)	2 (0.003%)

5.2 Colouring

We first evaluated the results of the colouring and propagation steps on the simulated reads and linkage maps. Because the reads are simulated, we know for each read the position where it derives from. Therefore we can also deduce the correct colours for each read as follows. For every read there are two markers in the linkage map that form the upper and lower limit of acceptable colours the read can get. All the colours should be between the colour of last marker before the read and the first marker after the read.

We are mostly interested in the set of reads that are fully within the correct range, i.e. no colour given to the read is incorrect, and the set of reads that are fully outside the correct range.

As can be expected, Table 3 shows that a vast majority of the reads are completely inside their colour ranges. The small amount of reads outside the range are reads that have been mapped to an entirely wrong position of the reference.

5.3 Cleaning

To evaluate the effectiveness of removing colour crossing edges, we find the simulated positions of the two reads corresponding to each edge in the graph and check whether they overlap in the reference genome. We consider those overlapping reads to be true in the sense that using that single edge in a contig would spell the correct sequence.

Miniasm already implements a cleaning step which is solely based on the read data. We counted the number of true and false edges both with and without this cleaning step to understand how each cleaning step affects the overlap graph.

Table 4 shows how Kermit is able to improve the percentages of true edges by removing false edges in the graph. Though the amounts of false edges removed is relatively small, any single wrong edge used in the assembly can cause a misassembly or break a unitig.

Table 5 Assembly statistics for simulated *S. cerevisiae* and *C. elegans* reads and simulated linkage maps.

	S. cerevisiae		*C. elegans*	
Assembly	Miniasm	Kermit	Miniasm	Kermit
# of contigs	26	22	291	261
Total length	11,831,837	11,728,421	102,040,817	101,632,493
	(97.32%)	(96.47%)	(103.81%)	(103.40%)
N50	605,399	640,779	2,337,914	2,293,633
NGA50	565,122	585,849	2,070,983	2,337,914
Misassemblies	2	1	7	7

Table 6 Assembly statistics for real *S. cerevisiae* and *C. elegans* reads and simulated linkage maps.

	S. cerevisiae		*C. elegans*	
Assembly	Miniasm	Kermit	Miniasm	Kermit
# of contigs	31	29	146	141
Total length	12,118,143	11,997,376	106,229,975	103,811,685
	(99.68%)	(98.69%)	(108.08%)	(105.62%)
N50	732,688	763,111	2,851,252	2,851,288
NGA50	345,801	376,187	252,615	254,858
Misassemblies	61	58	1,870	1,768

5.4 Assembly

Lastly, we compare the actual assemblies produced by our tool to those produced by Miniasm. To get a better picture of the assemblies, we also applied a consensus tool, Racon v1.2.0 [16], on the *S. cerevisiae* and *C. elegans* assemblies. On the *H. erato* and *B. pendula* assemblies Racon was very slow so we did not run it on those assemblies. The assembly statistics were generated with QUAST v4.6.1 [9].

Table 5 shows how the assembly statistics are improved using Kermit on the simulated *S. cerevisiae* and *C. elegans* reads and simulated linkage maps. The number of contigs is reduced and the NGA50 statistic is increased indicating a more contiguous assembly. Also the number of misassemblies is either reduced or stays the same.

Next we evaluated Kermit on the real read data and simulated linkage map of *S. cerevisiae* and *C. elegans*. Table 6 shows that also in this case the number of contigs and the number of misassemblies is reduced, whereas the NGA50 statistic is increased.

Finally we ran experiments on real read data and real linkage maps (*H. erato* and *B. pendula*). For these species only draft assemblies are available and thus we could not validate the produced assemblies and compute the number of misassemblies or the NGA50 statistics. Table 7 shows that for both data sets the number of contigs is reduced and N50 is increased indicating a more contiguous assembly. We also note that both Miniasm and Kermit struggle on the *H. erato* data. We believe this is largely due to the fact that we needed to pool PacBio reads from two experiments using two different individuals to have enough reads for assembly. This introduces heterogeneity to the data and makes assembly challenging.

Table 7 Assembly statistics for real *H. erato* and *B. pendula* reads and real linkage maps.

	H. erato		*B. pendula*	
Assembly	Miniasm	Kermit	Miniasm	Kermit
# of contigs	7,444	6,091	2,201	1,587
Total length	327,725,353	280,881,758	473,300,369	425,356,395
	(86.24%)	(73.92%)	(107.57%)	(96.67%)
N50	58,892	60,356	435,830	539,400

Table 8 Wall clock times for all steps taken by the tools. The consensus phase was very slow on the big genomes of *H. erato* and *B. pendula* so it was not run on those data sets.

	Tool	Overlap	Map	Colour	Layout	Consensus	Total
S. cerevisiae	Miniasm	52s	-	-	6s	4min 52s	5min 50s
(simulated)	Kermit	52s	6s	0s	6s	3min 29s	4min 38s
C. elegans	Miniasm	9min 55s	-	-	1min 58s	28min 19s	40min 17s
(simulated)	Kermit	9min 55s	2min 17s	5s	55s	28min 57s	42min 9s
S. cerevisiae	Miniasm	1min 9	-	-	8s	2min 45s	4min 2s
	Kermit	1min 09s	10s	1s	8s	2min 44s	4min 12s
C. elegans	Miniasm	16min 54s	-	-	4min	53min 28s	1h 14min
	Kermit	16min 54s	4min 8s	11s	2min 29s	50min 29s	1h 14min
H. erato	Miniasm	8h 40min	-	-	1h 30min	-	10h 10min
	Kermit	8h 40min	9min	2min	2h 42min	-	11h 32min
B. pendula	Miniasm	3h 24min	-	-	1h 32min	-	4h 57min
	Kermit	3h 24min	9min	17s	1h 37min	-	5h 11min

5.5 Performance

We recorded the wall clock time for all experiments. Table 8 shows that Kermit needs at most 5% more time than Miniasm on all data sets except *H. erato* on which it needs 13% more time. In some cases the total running time is even reduced. Additionally, we see that the consensus step using Racon easily dominates the pipeline in terms of time to complete.

6 Conclusions

We defined guided assembly with linkage maps by extending the unitig assembly model. Our method, Kermit, is implemented as a graph cleaning step and the contigs are generated with a simple unitig algorithm. As such the graph cleaning step could be used as a preprocessing step of a more complicated traversal algorithm for retrieving the contigs. Colouring the reads also leads naturally into non-overlapping bins of reads, that can be assembled independently. This allows massive parallelism in the assembly and could make more sophisticated assembly algorithms practical.

When defined as an independent graph cleaning step, our method guiding the assembly could be applied not only to other OLC-assemblers, but also to DBG-assemblers. In this case, the colours would be assigned to reads and the k-mers would get all colours present in reads where they derive from.

Our experiments show that Kermit successfully removes false edges from the overlap graph. Furthermore we showed that with only a modest increase in runtime Kermit improves the contiguity and correctness of assembly as compared to the Miniasm pipeline.

References

1 V. Ahola, R. Lehtonen, P. Somervuo, et al. The Glanville fritillary genome retains an ancient karyotype and reveals selective chromosomal fusions in Lepidoptera. *Nature Communications*, 5:4737, 2014.

2 B. Alipanahi, L. Salmela, S.J. Puglisi, M. Muggli, and C. Boucher. Disentangled long-read de Bruijn graphs via optical maps. In R. Schwartz and K. Reinert, editors, *WABI 2017*, volume 88 of *LIPIcs*, pages 1:1–1:14, Dagstuhl, Germany, 2017.

3 A. Bankevich, S. Nurk, D. Antipov, et al. SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *J Comput Biol.*, 19(5):455–477, 2012.

4 S.M. Van Belleghem, P. Rastas, A. Papanicolalaou, et al. Complex modular architecture around a simple toolkit of wing pattern genes. *Nature Ecology & Evolution*, 1:0052, 2017.

5 J. Catchen. Chromonomer. http://catchenlab.life.illinois.edu/chromonomer/, 2015. Accessed: 2018-04-27.

6 G. Chartrand, GL. Johns, KA. McKeon, and P. Zhang. Rainbow connection in graphs. *Mathematica Bohemica*, 133(1):85–98, 2008.

7 C.-S. Chin, P. Peluso, F.J. Sedlazeck, et al. Phased diploid genome assembly with single-molecule real-time sequencing. *Nature Methods*, 13:1050–1054, 2016.

8 J.L. Fierst. Using linkage maps to correct and scaffold de novo genome assemblies: methods, challenges, and computational tools. *Frontiers in Genetics*, 6:220, 2015.

9 A. Gurevich, V. Saveliev, N. Vyahhi N, and G. Tesler. QUAST: quality assessment tool for genome assemblies. *Bioinformatics*, 29(8):1072–1075, 2013.

10 M. Kolmogorov, J. Yuan, Y. Lin, and P. Pevzner. Assembly of long error-prone reads using repeat graphs. In *Proc. RECOMB 2018*, pages 261–263, 2018.

11 S. Koren, B.P. Walenz, K. Berlin, J.R. Miller, N.H. Bergman, and A.M. Phillippy. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome Res.*, 27:722–736, 2017.

12 H. Li. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*, 32(14):2103–2110, 2016.

13 H. Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 2018. (To appear).

14 H.C. Lin, S. Goldstein, L. Mendelowitz, S. Zhou, J. Wetzel, D.C. Schwartz, and M. Pop. AGORA: assembly guided by optical restriction alignment. *BMC Bioinformatics*, 13:189, 2012.

15 T. Paterson and A. Law. ArkMAP: integrating genomic maps across species and data sources. *BMC Bioinformatics*, 14:246, 2013.

16 R. Vaser R, I. Sovic, N. Nagarajan, and M. Sikic. Fast and accurate de novo genome assembly from long uncorrected reads. *Genome research*, 27:737–746, 2017.

17 P. Rastas. Lep-MAP3: robust linkage mapping even for low-coverage whole genome sequencing data. *Bioinformatics*, 33(23):3726–3732, 2017.

18 J. Salojärvi, O.P. Smolander, K. Nieminen, et al. Genome sequencing and population genomic analyses provide insights into the adaptive landscape of silver birch. *Nature Genetics*, 49:904–912, 2017.

19 K. Schneeberger, S. Ossowski, F. Ott, et al. Reference-guided assembly of four diverse Arabidopsis thaliana genomes. *PNAS*, 108(25):10249–10254, 2011.

20 B.K. Stöcker, J. Köster, and S. Rahmann. SimLoRD: Simulation of long read data. *Bioinformatics*, 32(17):2704–2706, 2016.