Near-optimal Assembly for Shotgun Sequencing with Noisy Reads

Ka Kit Lam
David Tse
Asif Khalak

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2014-10
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-10.html

January 30, 2014

Acknowledgement

Asif Khalak, David Tse

Near-optimal Assembly for Shotgun Sequencing with Noisy Reads

Ka-Kit Lam*, Asif Khalakˆ, David Tse*

*Department of EECS, UC Berkeley; ˆPacific Biosciences

Abstract. Recent work [1] identified the fundamental limits on the information requirements in terms of read length and coverage depth required for successful *de novo* genome reconstruction from shotgun sequencing data, based on the idealistic assumption of no errors in the reads (noiseless reads). In this work, we show that even when there is noise in the reads, one can successfully reconstruct with information requirements close to the noiseless fundamental limit. A new assembler, X-phased Multibridging, is designed based on a probabilistic model of the genome. It is shown through analysis to perform well on the model, and through simulations to perform well on real genomes.

Keywords De novo sequence assembly, genome finsihing, methods for emerging sequencing technologies
Contact author email kklam@eecs.berkeley.edu (Ka-Kit, Lam)

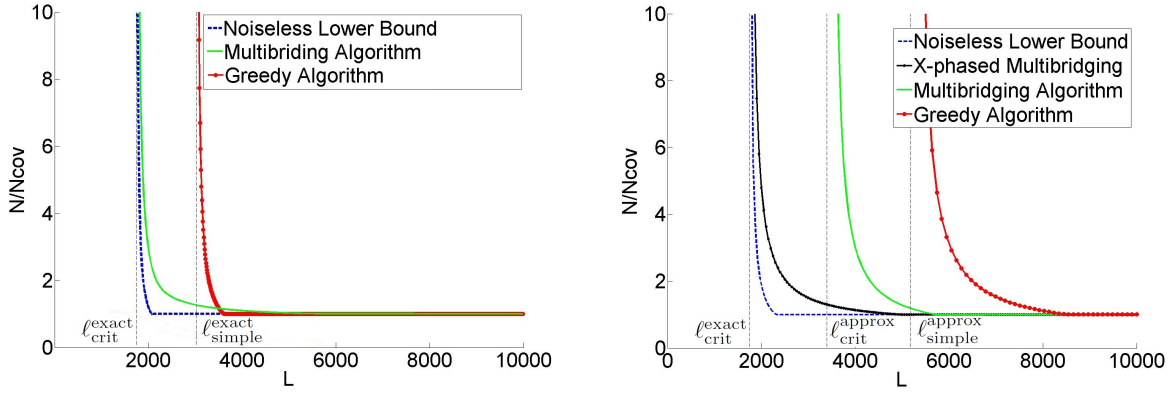
1 Introduction

1.1 Problem statement

Optimality in the acquisition and processing of DNA sequence data represents a serious technology challenge from various perspectives including sample preparation, instrumentation and algorithm development. Despite scientific achievements such as the sequencing of the human genome and ambitious plans for the future [20, 17], there is no single, overarching framework to identify the fundamental limits in terms of information requirements required for successful output of the genome from the sequence data.

Information theory has been successful in providing the foundation for such a framework in digital communication [18], and we believe that it can also provide insights into understanding the essential aspects of DNA sequencing. A first step in this direction has been taken in the recent work [1], where the fundamental limits on the minimum read length and coverage depth required for successful assembly are identified in terms of the statistics of various repeat patterns in the genome. Successful assembly is defined as the reconstruction of the underlying genome, i.e. genome finishing [16]. The genome finishing problem is particularly attractive for analysis because it is clearly and unambigiously defined and is arguably the ultimate goal in assembly. There is also a scientific need for finished genomes [12][11]. Until recently, automated genome finishing was beyond reach [5] in all but the simplest of genomes. New advances using ultra-long read single-molecule sequencing, however, have reported successful automated finishing [3, 8]. Even in the case where finished assembly is not possible, the results in [1] provide insights on optimal use of read information since the heart of the problem lies in how one can optimally use the read information to resolve repeats.

Figure 1a gives an example result for the repeat statistics of *E. coli* K12. The x-axis of the plot is the read length and the y-axis is the coverage depth normalized by the Lander-Waterman depth (number of reads needed to cover the genome [10]). The lower bound identifies the necessary read length and coverage depth required for *any* assembly algorithm to be successful with these repeat statistics. An assembly algorithm called Multibridging was presented, whose read length and coverage depth requirements are very close to the lower bound, thus tightly characterizing the fundamental information requirements. The result shows a critical phenomenon at a certain read length $L = \ell_{\text{crit}}^{\text{exact}}$: below this critical read length, reconstruction is impossible no matter how high the coverage depth; slightly above this read length, reconstruction is possible with Lander-Waterman coverage depth. This critical read length is given by $\ell_{\text{crit}}^{\text{exact}} = \max\{\ell_{\text{interleaved}}^{\text{exact}}, \ell_{\text{triple}}^{\text{exact}}\}$, where $\ell_{\text{interleaved}}^{\text{exact}}$ is the length of the longest exact interleaved repeat and $\ell_{\text{triple}}^{\text{exact}}$ is the length of the longest exact triple repeat in the genome, and has its roots in earlier work by Ukkonen on Sequencing-by-Hybridization [21]. The framework also allows the analysis of specific algorithms and the comparison with the fundamental limit; the plot shows for example the performance of the greedy algorithm and we see that its information requirement is far from the fundamental limit.



(a) *Information requirement for noiseless reads* (b) *Information requirement for noisy reads*

Fig. 1: Information requirement to reconstruct *E. coli* K12. $\ell_{\text{crit}}^{\text{exact}} = 1744, \ell_{\text{crit}}^{\text{approx}} = 3393$

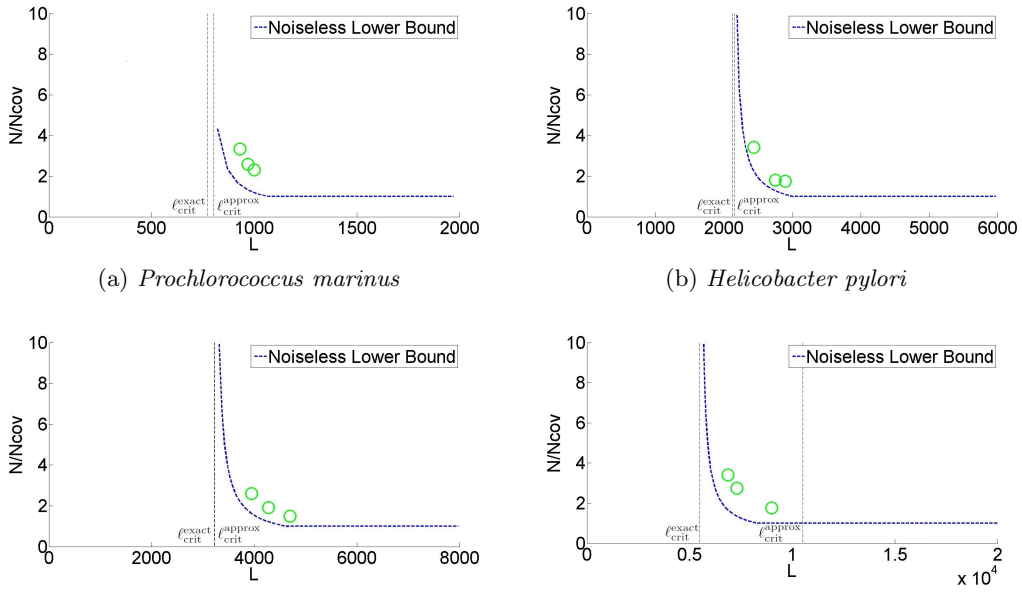
A key simplifying assumption in [1] is that there are no errors in the reads (noiseless reads). However reads are noisy in all present-day sequencing technologies, ranging from primarily substitution errors in

Illumina® platforms, to primarily insertion-deletion errors in Ion Torrent® and PacBio® platforms. The following question is the focus of the current paper: in the presence of read noise, can we still successfully assemble with a read length and coverage depth close to the minimum in the noiseless case? A recent work [6] with an existing assembler suggests that the information requirement for genome finishing substantially exceeds the noiseless limit. However, it is not obvious whether the limitations lie in the fundamental effect of read noise or in the sub-optimality of the algorithms in the assembly pipeline.

1.2 Results

The difficulty of the assembly problem depends crucially on the genome repeat statistics. Our approach to answering the question of the fundamental effect of read noise is based on design and analysis using a parametric probabilistic model of the genome that matches the key features of the repeat statistics we observe in genomes. In particular, it models the presence of long approximate repeats. Figure 1b shows a plot of the predicted information requirement for reliable reconstruction by various algorithms under a substitution error rate of 1%. The plot is based on analytical formulas derived under our genome model with parameters set to match the statistics of *E. coli* K12. We will show that it is possible in many cases to develop algorithms that approach the noiseless lower bound even when the reads are noisy. Specifically, the X-phased Multibridging algorithm has close to the same critical read length $L = \ell_{\text{crit}}^{\text{exact}}$ as in the noiseless case and only slightly greater coverage depth requirement for read lengths greater than the critical read length.

We then proceed to build a prototype assembler based on the analytical insights and we perform experiments on real genomes. As shown in Figure 2, we test the prototype assembler by using it to assemble noisy reads sampled from 4 different genomes. At coverage and read length indicated by a green circle, we perform assembly and succeed in reconstruction. We note that the information requirement is close to the noiseless lower bound. Moreover, the algorithm (X-phased Multibridging) is computationally efficient with the the most computational expensive step being the computation of overlap of reads/K-mers, which is also an unavoidable procedure in most assembly algorithms. The main conclusion we draw from this work is



(a) *Prochlorococcus marinus* (b) *Helicobacter pylori*

(c) *Methanococcus maripaludis* (d) *Mycoplasma agalactiae*

Fig. 2: Simulation results on a prototype assembler (substitution noise of rate 1.5 %)

that, with an appropriately designed assembly algorithm, the information requirement for genome assembly is surprisingly insensitive to read noise. The basic reason is that the redundancy required by the Lander-Waterman coverage constraint can be used to denoise the data. This is consistent with the asymptotic result

obtained in [13] and the practical approach taken in [3]. However, the result in [13] is based on a very simplistic i.i.d. random genome model, while the model and genomes considered in the present paper both have long approximate repeats. A natural extension of the Multibridging algorithm in [1] to handle noisy reads will allow the resolution of these long approximate repeats if the reads are long enough to bridge them, thus allowing reconstruction provided that the read length is greater $L = \ell_{\text{crit}}^{\text{approx}} = \max\{\ell_{\text{interleaved}}^{\text{approx}}, \ell_{\text{triple}}^{\text{approx}}\}$, where $\ell_{\text{interleaved}}^{\text{approx}}$ is the length of the longest *approximate* interleaved repeat and $\ell_{\text{triple}}^{\text{approx}}$ is the length of the longest approximate triple repeat in the genome. This condition is shown as a vertical asymptote of the "Multibridging" curve in Figure 1b. By exploiting the redundancy in the read coverage to resolve read errors, the X-phased Multibridging algorithm can phase the SNP's across the approximate repeat copies using only reads that bridge the exact repeats. Hence, it can achieve reconstruction with a read length close to $L = \ell_{\text{crit}}^{\text{exact}}$, same as the noiseless limit.

1.3 Related work

All assemblers must somehow address the problem of resolving noise in the reads in the genome reconstruction – however, the traditional approaches to measuring assembly performance makes quantitative comparisons challenging for unfinished genomes [14]. In most cases, the heart of the assembly problem lies in processing of the assembly graph, as in [22, 4, 19]. A common strategy for dealing with ambiguity from the reads lies in filtering the massively parallel sequencing data using the graph structure prior to traversing possible assembly solutions. In the present work, however, we are focused on the often overlooked goal of optimal data efficiency. Thus, to the extent possible we distinguish between the read error and the inherent ambiguity associated with the shotgun sampling process. The proposed X-phased Multibridging assembler thus adds information to the assembly graph from analyzing the underlying reads to resolve assembly graph ambiguities in a novel way.

1.4 Paper outline

The path towards developing that algorithm is described in the paper as follows. In section 3, the repeat characteristics of certain genomes are reviewed and we develop a parametric model to captures the long tail of the repeat statistics. Section 4 contains the core analytical and algorithmic development towards finding approaches with minimal information requirements – close to the noiseless lower bound. In section 5, simulation-based experiments on real and synthetic genomes are used to characterize the performance of a prototype assembler for genome finishing. Finally, section 6 describes an extension to the algorithm that addresses the problem of indel noise. We find similar qualitative behavior between the cases of indel noise and substitution noise, but with increased computational burden in our prototype assembler.

2 Shotgun sequencing model and problem formulation

2.1 Sequencing model

Let $\boldsymbol{x_G}$ be a length G target genome being sequenced with each base in the alphabet set $\Sigma = \{A, C, G, T\}$. In the shotgun sequencing process, the sequencing instrument samples N length L reads $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_N$, sampled uniformly and independently from $\boldsymbol{x_G}$. This unbiased sampling assumption is made for simplicity and is also supported by the characteristics of single-molecule (e.g. PacBio®) data. Each read is a noisy version of the corresponding length L subsequence on the genome. The noise may be base insertions, substitutions or deletions. Our analysis and simulations focus on substitution noise first. In Section 6, indel noise are addressed. In the substitution noise model, let p be the probability that a base is substituted by another base, with probability $p/3$ to be any other base. The errors are assumed to be independent across bases and across reads.

2.2 Formulation

Successful reconstruction by an algorithm is defined by the requirement that, with probability at least $1 - \epsilon$, the reconstruction $\hat{x}_G$ is within edit distance δ from the target genome $\boldsymbol{x}_G$. This formulation implies automated genome finishing, because the output of the algorithm is one single contig. The fundamental limit for the assembly problem is the set of (N, L) for which successful reconstruction is possible by *some* algorithm. If $\hat{x}_G$ is directly spelled out from a correct placement of the reads, the edit distance between $\hat{x}_G$ and $\boldsymbol{x}_G$ is of the order of pG, since the error rate is p. Hence, for concreteness, we fix $\delta = 2pG$. The quality of the assembly can be further improved if we follow the assembly algorithm with a consensus stage in which we correct each base, e.g. with majority voting. But the consensus stage is not the focus in this paper.

3 Genome statistics and modeling

3.1 Statistics for long approximate repeats in genome

Genome complexity lies in the repeat structure of DNA, and the success of a *de novo* assembly algorithm depends on whether one can resolve repeats. An exact repeat is defined to be two copies of a substring appearing at different position of the DNA genome. In [1], it was observed that the read length and coverage depth required for successful assembly using noiseless reads for quite many genomes is governed by the long *exact* repeats. It is because there are only a few long repeats within the genomes and those long repeats are the bottle neck for genome finishing. The distribution of approximate repeats, which becomes important in the case of noisy reads, is similarly long-tailed for many genomes in the GOLD database – as observed by [7] who studied the distribution of approximate repeats with 95% homology.

While exact repeats may be defined as the region terminated on each end by a single differing base, the structure of approximate repeats is more complicated. Therefore, it is instructive to look at the empirical data to see if it is possible to obtain a simple model that one can use to investigate the effect of approximate repeats in a systematical way.

Typical approximate repeats have two characteristics:

1. the regions surrounding the two copies of an approximate repeat are statistically independent. We term this surrounding region the *random flanking region*.
2. There are relatively few differences (SNPs) within the approximate repeat region.

A typical approximate repeat is shown in Fig 3.

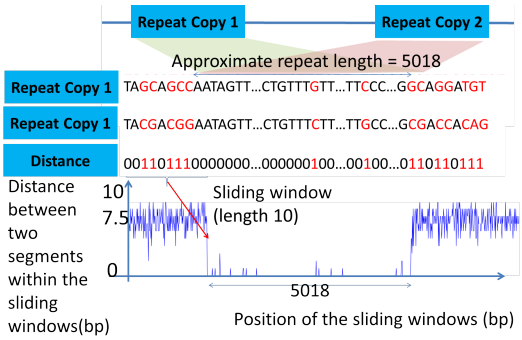


Fig. 3: Approximate repeat pattern

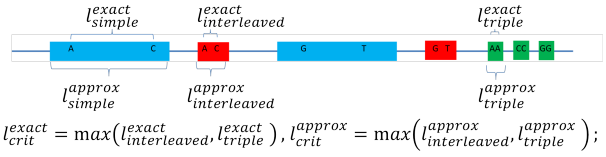


Fig. 4: Parametric model for the genome

In Figure 3, we analyze the neighborhood of an approximate long repeat from the *E. coli* K12 genome. Figure 3 shows the result of computing the Hamming distance between two copies of the approximate repeat within sliding windows of 10 bases. The x-axis of Figure 3 is the position of the window. Within the exact repeat region the distance, as shown on the y-axis, is 0 because the windowed substring are identical. In the genomes studied, the average distance (Hamming) in the flanking regions around the approximate repeat is approximately $7.5 = 10X\frac{3}{4}$, which suggests that these flanking regions are essentially statistically independent.Additional statistical analysis of the homology pattern in the neighborhood of the approximate repeat is detailed in the Appendix.

3.2 Parametric probabilistic model for genome

Consider a target genome as a random vector $\boldsymbol{x_G}$ of length G. It may be modeled by the following parametric model (Figure 4). First, we generate a random vector of length G, composed of uniformly and independently picked bases from the alphabet set $\Sigma = \{A, C, G, T\}$. This forms the random "background" of the genome. To model the longest approximate simple repeat, we randomly choose a segment of the genome of length $\ell^{\text{approx}}_{\text{simple}}$. and duplicate it at another random location in the genome to form a repeat. For the longest approximate

interleaved repeat, we randomly choose a segment of length $\ell^{\text{approx}}_{\text{interleaved}}$ (where $\ell^{\text{approx}}_{\text{interleaved}} < \ell^{\text{approx}}_{\text{simple}}$) and duplicate it elsewhere, which is randomly positioned to interleave with the longest approximate simple repeat. Finally, the longest approximate triple repeat is formed by randomly choosing three starting locations and generating a triple repeat of length $\ell^{\text{approx}}_{\text{triple}}$ (where $\ell^{\text{approx}}_{\text{triple}} < \ell^{\text{approx}}_{\text{simple}}$).

To model the SNPs within the approximate repeats, we first fix the parameters to be the length of longest exact duplicating segment($\ell^{\text{exact}}_{\text{simple}}$, $\ell^{\text{exact}}_{\text{interleaved}}$, $\ell^{\text{exact}}_{\text{triple}}$) within the approximate repeats and the number of SNPs($n_{simple}, n_{interleaved}$ and n_{triple}) within the approximate repeats. The location of the SNPs are then randomly chosen to fit these parameters. Without loss of generality, assume that $\ell^{\text{exact}}_{\text{simple}} > \max(\ell^{\text{exact}}_{\text{interleaved}}, \ell^{\text{exact}}_{\text{triple}})$. We note that $\ell^{\text{exact}}_{\text{simple}}$, $\ell^{\text{exact}}_{\text{interleaved}}$, $\ell^{\text{exact}}_{\text{triple}}$ corresponds to the length of the longest exact simple, interleaved, triple repeat respectively. For simplicity of notation, we define $\ell^{\text{approx}}_{\text{crit}} = \max(\ell^{\text{approx}}_{\text{interleaved}}, \ell^{\text{approx}}_{\text{triple}})$ and $\ell^{\text{exact}}_{\text{crit}} = \max(\ell^{\text{exact}}_{\text{interleaved}}, \ell^{\text{exact}}_{\text{triple}})$. Note that the Ukkonen's condition on read length is that $L > \ell^{\text{exact}}_{\text{crit}}$.

4 Algorithm design and analysis

4.1 An overview of algorithm design

In Fig 1a, we plot the curves of (N, L) requirement for two algorithms to successfully assemble noiseless reads. We briefly survey the two algorithms in [1]. Greedy Algorithm merges reads greedily based on the overlap score. The critical read length requirement is around $\ell^{\text{exact}}_{\text{simple}}$. Multibridging Algorithm utilizes a K-mer De Bruijn graph to capture the structure of the genome. It also utilizes long reads(or long K-mers of reads) to resolve long repeats. The corresponding information requirement is very close to the noiseless lower bound, with the critical read length requirement around $\ell^{\text{exact}}_{\text{crit}}$.

In the following sections, we design algorithms to assemble *noisy* reads . Fig 1b shows how we progressively reduce the information requirement(specifically on read length). We generalize Greedy Algorithm by defining a repeat-aware overlap rule for noisy reads. The critical read length requirement is around $\ell^{\text{approx}}_{\text{simple}}$(Sec 4.2). We generalize Multibridging Algorithm by extending DeBruijn graph to a noisy setting and performing graph surgery to remove faulty edges due to noise. The critical read length requirement is around $\ell^{\text{approx}}_{\text{crit}}$(Sec 4.3). Finally, we use X-phased Multibridging to phase long approximate repeats. X-phased Multibridging utilizes the redundancy induced by coverage through multiple sequence alignment and maximum likelihood estimation. We note that the information requirement of X-phased Multibridging is near the noiseless lower bound, with critical read length requirement around $\ell^{\text{exact}}_{\text{crit}}$ (Sec 4.4).

4.2 Overlap criterion for reads

Let x and y be two length-l i.i.d. randomly chosen segments (i.e. chosen from the random background.) A naive method to determine whether x and y are extracted from the same genomic location or not is by considering the Hamming distance(i.e. $d(x, y)$) between them . If $d(x, y) \leq \alpha \cdot l$, then they are , otherwise, they are not. This naive method is a reliable enough metric to differentiate every pairs of length-l segments that are extracted from the background genome given a long enough length $l > l_{iid}(p, \epsilon, G)$ and an appropriately chosen threshold $\alpha = \alpha(p, \epsilon, G)$.

By bounding both the false positive and false negative probability by $\epsilon/3$, one can find $l_{iid}(p, \epsilon, G)$ and $\alpha(p, \epsilon, G)$ to be the (L, α) solution of

$$\begin{cases} G^{2} \cdot & \exp(-L \cdot D(\alpha || \frac{3}{4})) = \frac{\epsilon}{3} \\ G \cdot & \exp(-L \cdot D(\alpha || 2p - \frac{4}{3}p^{2})) = \frac{\epsilon}{3} \end{cases}$$

where $D(a||b) = a \log \frac{a}{b} + (1-a) \log \frac{1-a}{1-b}$ is the Kullback-Leibler divergence.

However, when the segments are extracted from location covering the long repeats of the genome, this naive rule of determining overlap is not enough. Let us look at the example in Fig .5.

As shown in Fig. 5, the overall Hamming distance between two segments is not a reliable enough metric for determining overlap. It is because the long repeat leads to a small overall distance between the segments that are extracted from different copies of the repeat(e.g Segment 1 and Segment 3 in Fig. 5). The naive method will either result in a high false positive rate or a high false negative rate unless the length l is increased significantly. To properly handle such scenario, we can define a repeat-aware overlap rule (we call it RA-rule)

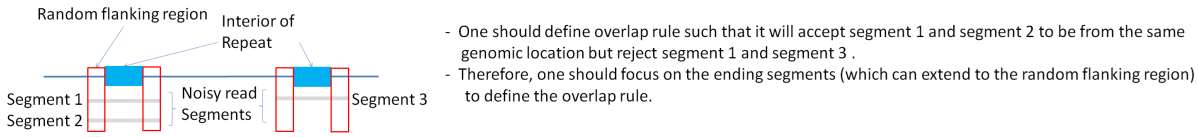


Fig. 5: Intuition about why we define the overlap rule to be RA-overlap rule

– (x, y) of length l are detected to be extracted from the same genomic location if and only if the distance between whole segments is $< \alpha \cdot l$ and both of its ending segments(of length l_{iid}) also have distance $< \alpha \cdot l_{iid}$.

(x, y) is said to be of overlap W under the RA-rule if length-W suffix of x and length-W prefix of y are detected to be extracted from the same genomic location under the RA-rule.

If we use Greedy Algorithm (Alg 1) to merge reads greedily with this overlap rule (RA-rule), the information requirement is shown in Prop 1 with the plot in Fig 1b. The key requirement for Greedy Algorithm to succeed is that every repeats (both the repeat interior and short extension to the random flanking region) need to be spanned by some reads. If we use the RA-rule to detect overlap of noisy reads, the short extension only need to be of length $\geq l_{iid}$ for reliable reconstruction.

Since, l_{iid} is of order of tens but $\ell_{\text{simple}}^{\text{approx}}$ is of order of thousands, the effect of noise on critical read length requirement is 2 order of magnitudes less than the requirement posed by the long repeats. This gives some intuition about why noise does not pay such a big role in terms of information requirement and why we can reduce the critical read length requirement close to $\ell_{\text{simple}}^{\text{approx}}$. The detail proof of Prop 1 is given in Appendix.

Algorithm 1 Greedy Algorithm

Initialize contigs to be reads
for $W = L$ *to* l_{iid} **do**
  if *any two contigs x,y are of overlap* W *under RA-rule* **then**
    merge x, y into one contig.
  end
end

Proposition 1. *With* $l_{iid} = l_{iid}(p, \frac{\epsilon}{3}, G)$, *if*

$$L > \ell_{\text{simple}}^{\text{approx}} + 2l_{iid}, \quad N > \max\left(\frac{G}{L - \ell_{\text{simple}}^{\text{approx}} - 2l_{iid}} \ln \frac{3}{\epsilon}, \frac{G}{(L - 2l_{iid})} \ln \frac{N}{\epsilon/3}\right)$$

then, Greedy Algorithm(Alg 1) is $\epsilon-$*feasible.*

4.3 Multibridging Algorithm

The critical read length requirement of Greedy Algorithm is bottle necked by $\ell_{\text{simple}}^{\text{approx}}$ because it requires all repeats to be spanned by reads. However, [15] uses DeBruijn graph to utilize the global structure of the genome to resolve repeats even when the reads cannot span all the repeats. By utilizing DeBruijn graph in the noisy reads setting, and by making use of the fact that a correctly assembled genome should correspond to a traversal of all the edges exactly once, we can resolve the longest approximate simple repeat with read length $\ell_{\text{crit}}^{\text{approx}} + 2 \cdot l_{iid} < L < \ell_{\text{simple}}^{\text{approx}}$. The algorithm is summarized in Alg 2 with the performance guarantee given in Prop 2. The plot is in Fig 1b. We note that Alg 2 can be seen as a noisy reads generalization of multibridging algorithm for perfect noiseless reads in [1].

Description and its performance

Proposition 2. *With* $l_{iid} = l_{iid}(p, \frac{\epsilon}{3}, G)$, if

$$L > \ell_{\text{crit}}^{\text{approx}} + 2l_{iid}, \quad N > \max\left(\frac{G \ln \frac{3}{\epsilon}}{L - \ell_{\text{crit}}^{\text{approx}} - 2l_{iid}}, \frac{G}{(L - 2l_{iid})} \ln \frac{N}{\epsilon/3}\right.$$

then, Multibridging Algorithm(Alg 2) is $\epsilon-$*feasible.*

Algorithm 2 Multibridging Algorithm

1. Choose K to be $\ell_{\text{crit}}^{\text{approx}} + 2l_{iid}$ and extract K-mers from reads.
2. Cluster K-mers based on the RA-rule.
3. Form uncondensed De Bruijn graph $G_{De-Bruijn} = (V, E)$ with the following rule:
   - a) K-mers clusters as node set V.
   - b) $(u, v) = e \in E$ if and only if there exists K-mers $u_1 \in u$ and $v_1 \in v$ such that u_1,$v_1$are consecutive K-mers in some reads.
4. Join the disconnected components of $G_{De-Bruijn}$ together by the following rule:

for $W = K - 1$ *to* l_{iid} **do**
  for *each node* x *which has either no predecessors / successors in* $G_{De-Bruijn}$ **do**
    a) Find the predecessor/successor y for x from all possible K-mers clusters such that overlap length(using any representative K-mers in that cluster) between x and y is W under RA-rule.
    b) Add dummy nodes in the De Bruijn graph to link x with y and update the graph to $G_{De-Bruijn}$
  end
end

5. Condense the graph $G_{De-Bruijn}$ to form G_{string} with the following rule:
   - a) Initialize G_{string} to be $G_{De-Bruijn}$ with node labels of each node being its cluster group index.
   - b) **while** $\exists$*successive nodes* $u \to v$ *such that* $out - degree(u) = 1$*and* $in - degree(v) = 1$ **do**
     bi) Merge u and v to form a new node w
     bii) Update the node label of w to be the concatenation of node labels of u and v
     end

6. Clear Branches of G_{string}:

for *each node* u *in the condensed graph* G_{string} **do**
  if $out - degree(u) > 1$ *and that all the successive paths are of the same length(measured by the number of node labels) and then joining back to node* v *and the path length* $< l_{iid}$ **then**
    we merge the paths into a single path from u to v .
  end
end

7. Condense graph G_{string}
8. Find the genome :
   - a) Find an Euler Cycle/Path in $G_{string}$and output the concatenation of the node labels to form a string $\boldsymbol{x}_{labels}$.
   - b) Using $\boldsymbol{x}_{labels}$ and look up the associated K-mers to form the final recovered genome $\hat{x}_G$.

Detail Proof is given in the Appendix. Here, we provide a high level highlight of why Multibridging algorithm can achieve what is claimed.

[Step1] We set a large K value to make sure the K-mers overlapping the longest approximate interleaved repeat and longest approximate triple repeat can be separated as distinct clusters.

[Step2] Clustering is done using the RA-rule because of the existence of long repeats.

[Step3] K-mer cluster corresponds to an equivalence class and this generalizes the notion of a single K-mer in noiseless case.

[Step4] Because of large K, the graph can be disconnected due to insufficient coverage. In order to reduce the coverage constraint, we connect the clusters greedily.

[Step5, 7] These two steps basically simplify the graph.

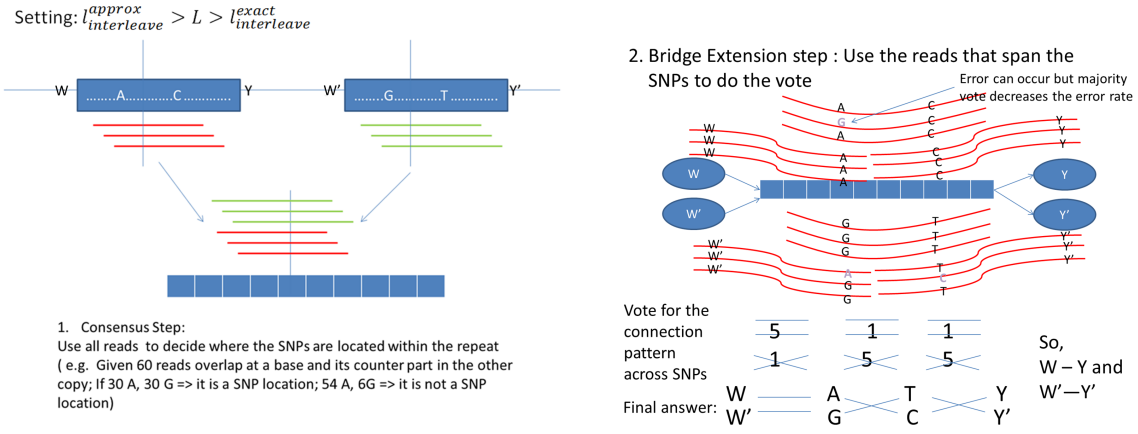
[Step6] Due to some residuals wrong merges in clustering near the boundary of the longest approximate simple repeat, we fix it by branch clearing.

[Step8] Since correct genome correspond to an Euler path in the condensed graph, we tranverse the graph to get that back.

Computational complexity improvement For the above algorithms, the most computational expensive step is clustering of K-mers/reads. This involves pairwise comparison of K-mers/reads and roughly run in $\tilde{\Theta}(N^2)$ if done in the naive way. In noiseless setting, one can easily do a lexicographical sort to identify identical K-mers and run in $\tilde{\Theta}(N)$. However, in the noisy setting, directly and only doing sorting once is not possible to cluster most of the K-mers. Here one can utilize the fact that the read length/K is long. One can use a short segment of the K-mers to do sorting and then cluster some K-mers together based on the sorting results. We then use another disjoint segment of the K-mers to perform the same operations. This is repeated for different disjoint small segments of the K-mers. For each round of sorting, we can cluster together a number of K-mers efficiently in $\tilde{\Theta}(N)$. The accuracy can be boosted up by combining the sorting

results from different rounds using disjoint set data structure that support union and find operations. We note that this computational trick is of the same spirit as locality sensitivity hashing.

4.4 X-phased Multibridging



(a) Consensus Step

(b) Bridge Extension Step

Fig. 6: Illustration of how to use SNPs to extend reads across repeats

An illustration of X-phased Multibridging Let us start by looking at the example in Fig 6. Here we have an approximate interleaved repeat having 2 SNPs in the middle but the read length $\ell^{\text{exact}}_{\text{interleaved}} < L$ and $L < \ell^{\text{approx}}_{\text{interleaved}}$. If we apply Multibridging to it, we will have two X-nodes(node such that in-degree and out-degree are both > 1) . One X-node corresponds to the longest approximate simple repeat and the other corresponds to the longest approximate interleaved repeat . The algorithm ends up finding two distinct Euler cycle and we do not know which one corresponds to the real genome. In order to resolve such situation, we need to use the SNPs within the longest approximate interleaved repeat to resolve that repeat(i.e. decide how W, W' and Y, Y' are linked in Fig 6).

First, we use all the reads from both copies to find out where the SNPs are located. Specifically, let D be the set of reads around the repeat region, and the ground truth genome segments being $\hat{s}_1$and $\hat{s}_2$. Without loss of generality, $\hat{s}_1(0) = W$ and $\hat{s}_2(0) = W'$. For each base $1 \le i \le \ell^{\text{approx}}_{\text{interleaved}}$, we do the following consensus, which can be easily implemented by counting the frequency of occurence of each alphabet overlapping at location i .

$$\max_{S \subset \{A,C,G,T\}^2, |S| \le 2} \mathcal{P}(\{\hat{s}_1(i), \hat{s}_2(i)\} = S \mid D) \tag{1}$$

Second, we use those reads that span the SNPs or random flanking region, to help us decide how to extend across the repeat. We let σ be the possible configuration at location SNP_1 to $SNP_{n_{interleaved}}$ and flanking region (e.g. $\sigma = (AAAY, CCCY')$) .We do the following maximum likelihood to determine the final sequence and extend across the approximate repeat, with $\hat{\sigma}$ being the estimator. Note that size of the feasible set is $2^{n_{interleaved}+1}$.

$$\max_{\sigma} \mathcal{P}(\hat{\sigma} = \sigma \mid D, \{\hat{s}_1(i), \hat{s}_2(i)\}_{i=1}^{\ell^{\text{approx}}_{\text{interleaved}}}) \tag{2}$$

In practice, for computational reason, that maximum likelihood can be approximated accurately even if it is replaced by a simple counting as illustrated in Fig 6 which we call count-to-extend algorithm(countAlg). It uses the raw reads to establish majority vote on how one should extend to the next SNPs using only the reads that span the SNPs. Summary of the algorithm is shown in Alg 3.

Performance Before getting into the detail analysis, let us have an intuitive understanding about why X-phased Multibridging can resolve repeats accurately. Since coverage is normally $> 20X$, we note that we have high precision in finding SNP location because we use all the reads overlapping at each base. As for the

Algorithm 3 X-phased Multibridging

1. Perform Step 1 to Step 7 of MultiBridging as in Alg 2
2. For every X-node $x \in G_{string}$
   - a) Align all the reads to the repeat region x
   - b) Consensus to find location of SNPs by solving Eq (1)
   - c) If possible, resolve repeat by either countAlg or by solving Eq (2) to phase the approximate repeat
3. Perform Step 8 of MultiBridging as in Alg 2

bridge extension step, with the following Lemma 1, we note that the error probability is also decently small if we have several such spanning reads and moderate read error probability even merely with the countAlg.

Lemma 1. *Let P_k ˜ Binomial(k,q) , which model k reads spanning 2 SNPs and with probability q of making an error in deciding the linkage of SNPs. Then, the failure probability of majority vote , with $k = 2m + 1$*

$$P_{2m+1}^{fail} = \mathcal{P}(P_{2m+1} \leq m) \leq_{2m+1} C_m \cdot p^{m+1}$$

We now analyze the error probability of Step 2 in Alg. 3(i.e. the repeat phasing step). Let $\mathcal{E}$ be the error event. ϵ_1 be the error probability for the consensus step. ϵ_2 be the error probability for the bridge extension step given k reads spanning each consecutive SNPs within the approximate repeat. And δ_{cov} be the probability for having k reads spanning each consecutive SNPs within the approximate repeat.

$$\mathcal{P}(\mathcal{E}) \leq \epsilon_1 + \epsilon_2 + \delta_{cov} \tag{3}$$

By calibrating bounds for ϵ_1 and ϵ_2, we can have a numerical sense about the accuracy of this algorithm. For Table 1, we have a length G= 5M size genome, having an approximate repeat length of 5000bp and with 2 SNPs within the repeat, partitioning the repeat into three equally spaced segments. In Table 1a, we simulated 100 rounds per data row and take the average error as the probability of having any base being incorrectly identified as SNPs/incorrectly not identified as SNPs in the majority vote , and we use $\epsilon_1$to denote that. We have used an estimated SNP rate of 0.01 as the input for the consensus algorithm to use as the prior distribution guess. In Table 1b, we have computed the upper bound for the error probability provided by Lemma 1. Since we have 3 segments within the approximate repeat, we use the following bound to evaluate $\epsilon_2 \leq= 3 \cdot (2p)^{k+1} \cdot_{2k+1} C_k$ with k being the number of reads spanning the any two SNPs. As we can see from Table 1, when $p = 0.01$, having as low as 20X coverage and $k = 3$ bridging reads, we can already have small enough error of $< 1\%$. So, we use $k = 3$ and use it to find the corresponding coverage requirement to have small δ_{cov} . This coverage constraint for X-phased Multibridging is also the bottleneck constraint we plotted in Fig 1b as shown before. The feasibility curve for X-phased Multibridging shown in Fig 1b is the condition to have 3 bridging reads spanning the longest exact interleaved repeat.

p	l^{approx}	Coverage (NL/G)	ϵ_1
0.01	5000	20	0.00
0.01	5000	40	0.00
0.01	5000	60	0.00
0.1	5000	20	0.16
0.1	5000	40	0.00
0.1	5000	60	0.00

(a) Calibration for ϵ_1

p	Number of bridging reads k	Upper bound for ϵ_2
0.01	1	0.060
0.01	3	0.0036
0.01	5	0.00024
0.1	11	0.089
0.1	21	0.022
0.1	31	0.0059

(b) Calibration for ϵ_2

Table 1: Calibration of error probability made by X-phased Multibridging

5 Simulation of the prototype assembler

With ideas presented before, we implement a prototype assembler that can automate genome finishing for reads corrupted by substitution noise. In particular, we test it on a simple synthetic case with substitution noise corrupted reads sampled from randomly generated genome with randomly thrown long repeats. This serves as a proof-of-concept that we can perform genome finishing with critical read length requirement close

to $\ell_{\text{crit}}^{\text{exact}}$. The plot is shown in Fig 7. We also apply this prototype assembler to sequence several genomes to finishing quality, with substitution noise corrupted reads sampled from the genome ground truth downloaded from NCBI. The assembly results are shown in Table 2 and the dot plot of the recovered genome against the genome ground truth is shown in Appendix. The detail design of the prototype assembler is presented in the Appendix and source code and data set can be found in [9].

Index	Species	G	p	$\frac{NL}{G}$	L	$\ell_{\text{simple}}^{\text{approx}}$	$\ell_{\text{crit}}^{\text{approx}}$	$\ell_{\text{crit}}^{\text{exact}}$	% match	Ncontig	$\frac{N}{N_{noiseless}}$	$\frac{L}{\ell_{\text{crit}}^{\text{exact}}}$
1	a	1440371	1.5%	37.36 X	930	1817	803	770	100.00	1	1.57	1.21
2	a	1440371	1.5%	33.14 X	970	1817	803	770	99.95	1	1.67	1.26
3	a	1440371	1.5%	29.60 X	1000	1817	803	770	99.99	1	1.66	1.30
4	b	1589953	1.5%	40.82 X	2440	4183	2155	2122	100.00	1	1.30	1.15
5	b	1589953	1.5%	21.31 X	2752	4183	2155	2122	99.99	1	1.19	1.30
6	b	1589953	1.5%	20.66 X	2900	4183	2155	2122	99.99	1	1.35	1.37
7	c	1772693	1.5%	30.03 X	3950	5018	3234	3218	99.96	1	1.36	1.23
8	c	1772693	1.5%	21.96 X	4279	5018	3234	3218	99.97	1	1.33	1.33
9	c	1772693	1.5%	17.03 X	4700	5018	3234	3218	100.00	1	1.31	1.46
10	d	1006701	1.5%	35.23 X	6867	15836	10518	5494	99.05	1	1.72	1.25
11	d	1006701	1.5%	19.88 X	7500	15836	10518	5494	97.86	1	1.30	1.37
12	d	1006701	1.5%	17.69 X	9000	15836	10518	5494	98.10	1	1.68	1.64

Table 2: Assembly of several genomes ((a) *Prochlorococcus marinus* (b) *Helicobacter pylori* (c) *Methanococcus maripaludis* (d) *Mycoplasma agalactiae*) with $\ell_{\text{crit}}^{\text{exact}} = \max(\ell_{\text{interleaved}}^{\text{exact}}, \ell_{\text{triple}}^{\text{exact}})$, $\ell_{\text{crit}}^{\text{approx}} = \max(\ell_{\text{interleaved}}^{\text{approx}}, \ell_{\text{triple}}^{\text{approx}})$ and $N_{noiseless}$ is the lower bound on number of reads in the noiseless case for $1 - \epsilon = 95\%$ confidence recovery

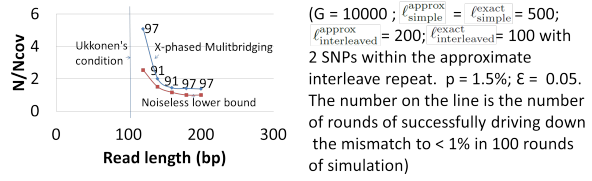
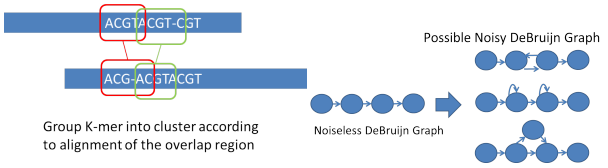


Fig. 7: Simulation of the assembly on randomly generated genome



(a) Form K-mer Clusters (b) Abnormality in (indel) DeBruijn Graph

Fig. 8: Treatment of read corrupted by indel

6 Extension to handle insertion/deletion noise

We also implement an extension of the prototype assembler to handle indel noise. We remark that one non-trivial generalization is the way that we form the noisy De-Bruijn graph for K-mer clusters. In particular, we first compute the pairwise overlap alignment among reads, then we use the overlap alignment to group K-mers into clusters. Subsequently, we link successive cluster of K-mers together as we do in Alg 2. An illustration is shown in Fig 8a. However, due to the noise being indel in nature, the edges in the noisy De Bruijn graph may point in the wrong direction as shown in Fig 8b. In order to handle this, we traverse the graph and remove such abnormality when they are detected.

We further test it on synthetic reads sampled from real genomes and synthetic genomes. Simulation results are summarized in Table 3 where p_i, p_d are insertion probability and deletion probability and rate is the number of successful reconstruction(i.e.simulation rounds that show mismatch $< 5\%$) divided by total number of simulation rounds.

Type	G	p_i	p_d	$\frac{NL}{G}$	L	$\ell_{\text{simple}}^{\text{approx}}$	$\ell_{\text{crit}}^{\text{approx}}$	$\ell_{\text{crit}}^{\text{exact}}$	$\frac{N}{N_{noiseless}}$	$\frac{L}{\ell_{\text{crit}}^{\text{exact}}}$	Rate
Synthetic	50000	1.5%	1.5%	23.0 X	200	500	200	100	2.25	2	28/30
Synthetic	50000	1.5%	1.5%	24.1 X	180	500	200	100	2.33	1.8	27/30
a	1440371	1.5%	1.5%	28.53 X	1000	1817	803	770	1.60	1.30	1/1
b	1589953	1.5%	1.5%	20.66 X	2900	4183	2155	2122	1.35	1.37	1/1

Table 3: Simulation Results for indel error treatment(Synthetic: randomly generated to fit the repeat statistics ; (a) : *Prochlorococcus marinus* ; (b): *Helicobacter pylori*)

References

1. Guy Bresler, Ma'ayan Bresler, and David Tse. Optimal assembly for high throughput shotgun sequencing. *BMC Bioinformatics*, 2013.
2. Zhirong Bao and Sean R Eddy. Automated de novo identification of repeat sequence families in sequenced genomes. *Genome Research*, 12(8):1269–1276, 2002.
3. Chen-Shan Chin, David H Alexander, Patrick Marks, Aaron A Klammer, James Drake, Cheryl Heiner, Alicia Clum, Alex Copeland, John Huddleston, Evan E Eichler, et al. Nonhybrid, finished microbial genome assemblies from long-read smrt sequencing data. *Nature methods*, 2013.
4. Sante Gnerre, Iain MacCallum, Dariusz Przybylski, Filipe J Ribeiro, Joshua N Burton, Bruce J Walker, Ted Sharpe, Giles Hall, Terrance P Shea, Sean Sykes, et al. High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proceedings of the National Academy of Sciences*, 108(4):1513–1518, 2011.
5. David Gordon, Chris Abajian, and Phil Green. Consed: a graphical tool for sequence finishing. *Genome research*, 8(3):195–202, 1998.
6. Asif Khalak, Ka Kit Lam, Greg Concepcion, and David Tse. Conditions on finishable read sets for automated de novo genome sequencing. *Sequencing, Finishing and Analysis in the Future*, May, 2013.
7. Sergey Koren, Gregory P Harhay, Timothy PL Smith, James L Bono, Dayna M Harhay, D Scott Mcvey, Diana Radune, Nicholas H Bergman, and Adam M Phillippy. Reducing assembly complexity of microbial genomes with single-molecule sequencing. *arXiv preprint arXiv:1304.3752*, 2013.
8. Sergey Koren, Michael C Schatz, Brian P Walenz, Jeffrey Martin, Jason T Howard, Ganeshkumar Ganapathy, Zhong Wang, David A Rasko, W Richard McCombie, Erich D Jarvis, et al. Hybrid error correction and de novo assembly of single-molecule sequencing reads. *Nature biotechnology*, 30(7):693–700, 2012.
9. Ka-Kit Lam, Asif Khalak, and David Tse. www.eecs.berkeley.edu/ ˜kakitone.
10. Eric S Lander and Michael S Waterman. Genomic mapping by fingerprinting random clones: a mathematical analysis. *Genomics*, 2(3):231–239, 1988.
11. Elaine Mardis, John McPherson, Robert Martienssen, Richard K Wilson, and W Richard McCombie. What is finished, and why does it matter. *Genome research*, 12(5):669–671, 2002.
12. Duccio Medini, Davide Serruto, Julian Parkhill, David A Relman, Claudio Donati, Richard Moxon, Stanley Falkow, and Rino Rappuoli. Microbiology in the post-genomic era. *Nature Reviews Microbiology*, 6(6):419–430, 2008.
13. Abolfazl Motahari, Kannan Ramchandran, David Tse, and Nan Ma. Optimal dna shotgun sequencing: Noisy reads are as good as noiseless reads. *Proceedings of the 2013 IEEE International Symposium on Information Theory, Istanbul, Turkey, July 7-12, 2013*, 2013.
14. Giuseppe Narzisi and Bud Mishra. Comparing de novo genome assembly: The long and short of it. *PLoS ONE*, 6(4):e19175, 04 2011.
15. 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.
16. Mihai Pop. Genome assembly reborn: recent computational challenges. *Briefings in bioinformatics*, 10(4):354–366, 2009.
17. DNA SEQUENCING. A plan to capture human diversity in 1000 genomes. *Science*, 21:1842, 2007.
18. Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948.
19. Jared T Simpson and Richard Durbin. Efficient de novo assembly of large genomes using compressed data structures. *Genome Research*, 22(3):549–556, 2012.
20. Peter J Turnbaugh, Ruth E Ley, Micah Hamady, Claire M Fraser-Liggett, Rob Knight, and Jeffrey I Gordon. The human microbiome project. *Nature*, 449(7164):804–810, 2007.
21. Esko Ukkonen. Approximate string-matching with q-grams and maximal matches. *Theoretical computer science*, 92(1):191–211, 1992.
22. Daniel R Zerbino and Ewan Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, 18(5):821–829, 2008.

A Appendix : Proof on Performance Guarantee

Here we use the short hand of l_{repeat} , $l_{interleaved}$ and l_{triple} to represent the corresponding approximate repeat length of longest simple, interleaved, triple repeat respectively.

A.1 Greedy Algorithm

Let us define a $\theta-$ neighborhood of the repeat specified by $x[a:b]$ and $x[c:d]$ to be the genomic location of repeat which are $x[a-\theta:b+\theta]$ and $x[c-\theta:d+\theta]$.

We say a repeat is $\theta-$bridged if there exists a read that cover the θ-neighborhood of at least one copy of the repeat. For simplicity of arguments, we assume $l_{repeat} >> \max(l_{intereleave}, l_{triple})$.

Lemma 2. *We first note the following sufficient conditions for Noisy Greedy to succeed.*

1. Merging at stages from L to $l_{iid}(p,\epsilon,G)$ are merging successive reads
2. Every successive reads have overlap with length at least $l_{iid}(p,\frac{\epsilon}{3},G)$

Theorem 1. *Under the generative model on genome, with* $l_{iid} = l_{iid}(p,\frac{\epsilon}{3},G)$, $\alpha = \alpha(p,\frac{\epsilon}{3},G)$ *,if*

$$L > l_{repeat} + 2\cdot l_{iid}$$
$$G > N > \max(\frac{G\ln\frac{3}{\epsilon}}{L - l_{repeat} - 2\cdot l_{iid}}, \frac{G\cdot\ln\frac{N}{\epsilon/3}}{L - l_{iid}})$$

, then $\mathcal{P}(\mathcal{S}^C) \le \epsilon$.

Proof. In order to prove that claim, let us break down into several subparts

Let E_1 be the event that condition 1 in Lemma (2) is not satisfied. E_2 be the event that condition 2 in Lemma (2) is not satisfied. E_3 be the event that the long/interleave/triple repeat is not $l_{iid}-$bridged.

Now we claim that with the chose (N, L) in the range,

1. $\mathcal{P}(E_1) \le \frac{\epsilon}{3}$
2. $\mathcal{P}(E_3) \le \frac{\epsilon}{3}$
3. $\mathcal{P}(E_2 \mid E_1^C \cap E_3^C) \le \frac{\epsilon}{3}$

We first see how we can use these to obtain the desired claim and proceed to prove each of the above sub-claims.

$$\begin{aligned}
\mathcal{P}(S^C) =& \mathcal{P}(E_1) + \mathcal{P}(E_2 \cap E_1^C) \\
=& \mathcal{P}(E_1) + \mathcal{P}(E_2 \cap E_1^C \cap E_3^C) + \mathcal{P}(E_2 \cap E_1^C \cap E_3) \\
\le& \mathcal{P}(E_1) + \mathcal{P}(E_2 \mid E_1^C \cap E_3^C) + \mathcal{P}(E_3) \\
\le& \frac{\epsilon}{3} + \frac{\epsilon}{3} + \frac{\epsilon}{3} \\
=& \epsilon
\end{aligned}$$

Now, we proceed to prove each of the sub-claims.

1. With $N > \frac{G\cdot\ln\frac{N}{\epsilon/3}}{L - l_{iid}}$, we have,

$$\begin{aligned}
\mathcal{P}(E_1) \le& N\exp(-\frac{N}{G}(L - l_{iid})) \\
\le& \frac{\epsilon}{3}
\end{aligned}$$

2. With $N > \frac{G\cdot\ln\frac{3}{\epsilon}}{L - l_{repeat} - 2\cdot l_{iid}}$we have,

$$\begin{aligned}\mathcal{P}(E_3) \leq & \exp(-\frac{N}{G}(L - 2l_{iid} - l_{repeat})) \\ \leq & \frac{\epsilon}{3}\end{aligned}$$

3. With the choice of $l_{iid} = l_{iid}(p, \frac{\epsilon}{3}, G)$, we have,

$$\begin{aligned}\mathcal{P}(E_2 \mid E_1^C \cap E_3^C) \leq & N^2 \cdot \exp(-l_{iid} D(\alpha||\frac{3}{4})) \\ & + 2N \exp(-l_{iid} D(\alpha||\eta)) \\ \leq & G^2 \cdot \exp(-l_{iid} D(\alpha||\frac{3}{4})) \\ & + 2G \exp(-l_{iid} D(\alpha||\eta)) \\ \leq & \frac{\epsilon}{3}\end{aligned}$$

Here we use the fact that there are indeed 4 types of overlap as in Fig 9. And given the bridging condition, we are only left with 2 types, namely, both ending segments outside/exactly one ending segment outside the longest repeat repeat region.

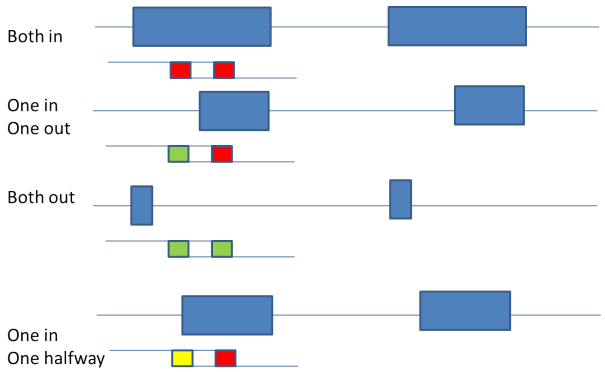


Fig. 9: Overlap Type

A.2 Simple De Bruijn Algorithm

Before continuing proving the performance of Multibridging Algorithm, it is instructive to analyze the following Simple De Bruijn Algorithm (Alg 4) because this is closely related to the Multibridging algorithm.

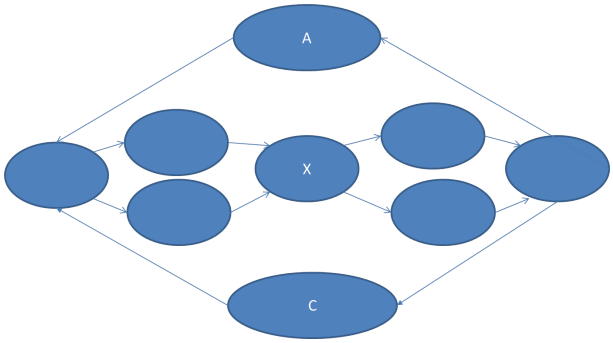
Algorithm 4 Noisy Simple DeBruijn

0. Choose K to be $\max(l_{interleave}, l_{triple})$
1. Extract Kmers from reads
2. Clusters Kmers
3. Form Kmer Graphs
4. Condense the graph
5. Clear Branches
6. Condense graph
7. Find Euler Cycle

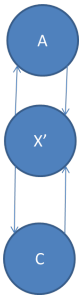
Here we first define several genomic region of interest which we will refer to in the proofs below.
a) S_0 = set of K-mers that are completely inside $l_{iid}-$ neighborhood of the longest repeat
b) S_1 = set of K-mers that are completely inside the longest repeat
c) $S_2 = S_0 \backslash S_1$

Lemma 3. *Here we provide several deterministic conditions that guarantee the success of the algorithm.*
1. Successive reads overlap with length at least K
2 K-mers are almost correctly clustered, that is,
a) K-mers from the same genomic location but not merged
b) x not in S_0 s.t. x get clustered with wrong K-mers
c) x in S_0 s.t.x get clustered with elements other than its own cluster/mirror cluster(mirror cluster is defined to be the cluster for the other copy of the repeat)
3) Repeat at both circle are at least $2 \cdot l_{iid}$ *separated(the interleaving segments between the repeat differ with at least* $2l_{iid}$*in length)*

Proof. We note that every length K segments $x \notin S_0$, they are represented as a distinct node in the K-mer graph because of the length K that we pick and the condition that successive reads overlap at least K bases. Moreover, for K-mers $x \in S_1$, they are condensed into the repeat as 'X' in Fig 10a. However, for the K-mers $x \in S_2$, they have chances not to merge properly, thus they form into the branches surrounding 'X' in Fig 10a. Because of condition 3, branch clearing will not eliminate the 'A' or 'C' in Fig 10a, further after condensing, we get the desired K-mer graph as in Fig 10b and this can be successfully read by a Eulerian Walk.



(a) Before branch clearing



(b) After branch clearing

Fig. 10: Branch clearing

Theorem 2. *If* $G > \frac{6}{\epsilon l_{iid}}, G \geq N \geq \frac{G \cdot \ln \frac{3N}{\epsilon}}{L - \max(l_{int}, l_{triple}) - 2l_{iid}}$ *, with* $l_{iid} = l_{iid}(p, \frac{\epsilon}{3}, G)$ $\alpha = \alpha(p, \frac{\epsilon}{3}, G)$*, then* $,\mathcal{P}(S^C) \leq \epsilon$

Proof. We first note that in order to obtain a bound on the error probability, we only need to separately bound the probability that each of the conditions in Lemma 3 fail,which are $\leq \frac{\epsilon}{3}$ each. Thus, combining, we get, $\mathcal{P}(S^C) \leq \epsilon$.

A.3 Multibridging Algorithm

An illustration of noisy multibridging algorithm is shown in Fig (11).

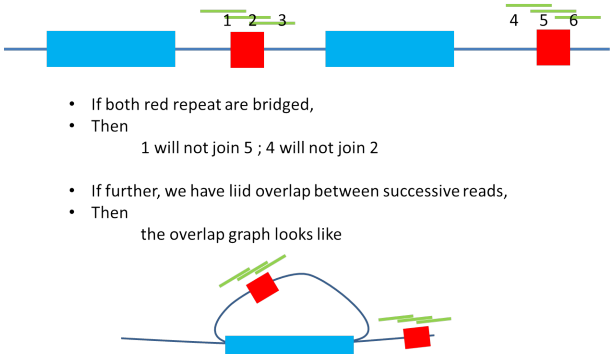


Fig. 11: An illustration of the Noisy Multi Bridging

Lemma 4. *Here are the deterministic conditions for the algorithm to succeed.*

1) Every successive reads overlap at least $l_{iid}(p, \frac{\epsilon}{3}, G)$

2) K-mers are almost correctly clustered, that is,

a) K-mers from the same genomic location but not merged

b) x not in S_0 s.t. x get clustered with wrong K-mers

c) x in S_0 s.t. x get clustered with elements other than its own cluster/mirror cluster

3) Repeat at both circle are at least $2 \cdot l_{iid}$ *separated*

4) When finding successors/predecessors, they are the real successors and predecessors

Proof. Along the same lines as the proof in Lemma (3), we only note that in this algorithm, we have an extra step of finding predecessor/successors. Moreover, the overlap here is significantly reduced to only l_{iid} instead of K in the Noisy Simple De Bruijn case.

Theorem 3. *With* $G > \frac{6}{\epsilon l_{iid}}$,$G \geq N \geq \max(\frac{G}{L-2l_{iid}} \ln \frac{N}{\epsilon/3}, \frac{G \ln \frac{3}{\epsilon}}{L-\max(l_{triple}, l_{interleave})-2l_{iid}})$,*with* $l_{iid} = l_{iid}(p, \frac{\epsilon}{3}, G)$ $\alpha = \alpha(p, \frac{\epsilon}{3}, G)$ *, then* $\mathcal{P}(S^C) \leq \epsilon$.

Proof. Here we note that with the given coverage, bridging conditions of the interleave repeat and the triple repeat are satisfied. And when this is true, then Condition 4 in Lemma 4 is true with high probability. Following the arguments in Theorem 2, we get desired.

B Appendix: Design and additional algorithmic components for the prototype assembler

B.1 Pipeline of the prototype assembler

The pipeline of the prototype assembler is shown in Fig 12. With a ground truth genome as input, the output is the performance of the whole pipeline by giving the mismatch rate.

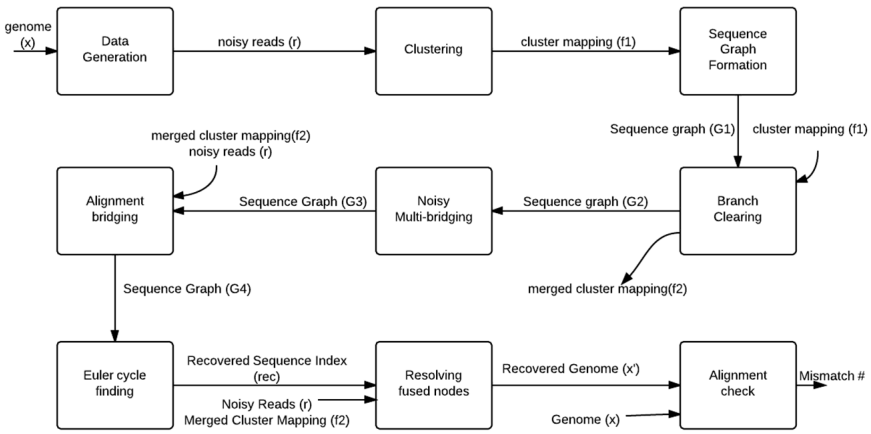


Fig. 12: Pipeline of the prototype assembler

Algorithm 5 Enhanced Multibridging Algorithm

Resolution of repeats:
0. Intially the weight of the edge are set to be 1.
1. While there is a X-node v :
a) For each edge (p_i,v) with weight $a_{p_i,v}$, create a new node $u_i = ^{p_i\rightarrow} v$ and an edge (p_i, u_i) with weight $1 + a_{p_iv}$. Similarly , for each edge (v,q_j), create a new node $w_j = v^{\rightarrow q_j}$ and an edge (w_j, q_j)
b) If v has a self-loop (v, v) with weight $a_{v,v}$, add an edge ($v^{\rightarrow v}$, $^{v\rightarrow} v$) with weight $a_{v,v} + 2$
c) Remove node v and all incident edges
d) For each pair of u_i, w_j adjacent in a read(extending to at least length of l_{iid} on both sides of the X-node), add edge (u_i, w_j). If exactly on each of the $u_i$and $w_j$nodes have no added edge, add the edge.
e) Condense the graph

B.2 A more robust branch clearing step

Since we employ a speed up step in the clustering and there may be K-mers that are not completely clustered correctly in the clustering step of Multibridging algorithm. Regarding that, we need to have a more robust branch clearing step. In particular, we first classfy nodes as "big" or "small" nodes based on the size of the nodes in the sequence graph. The key idea is to merge the small nodes together while keeping the big nodes unchanged. Starting from each big nodes, we tranverse the graph to detect all the small nodes that link the current big node to other big nodes. Then, we classify the small nodes into levels(depending on its distance from the current big node). After that, the small nodes in the same level are merged. Finally, we note that we keep the reachability among each big nodes.

B.3 Enhanced Multibridging that can resolve middle range repeats

We note that the ideas presented here can also be found in the prior work on the treatment of noiseles sreads. It is stated here for completeness. In the noisy setting, instead of considering the alphabet set to be $\Sigma = \{A, C, G, T\}$, one can consider the alphabet set as the cluster index of the K-mers.

C Appendix: Treatment of indel noise

C.1 Formation of K-mer DeBruijn graph for indel corrupted reads

In order to form K-mer DeBruijn graph for indel corrupted reads, we first need to have a clear notion of K-mers. We define K-mers to be the length K segments in the genome ground truth (as opposed to the usual

definition from the reads). Although we mostly work on the reads themselves, the definition of the Kmers are based on the ground truth. In order to successfully cluster K-mers , we need to do the following steps.

1. We first compute the pairwise alignement of the reads.

2. Based on the pairwise alignment, for each length K-segments, we know which should be aligned to which. We then group them together using the alignment result.

3. Finally, we end up with the length K segments from the reads clustering together, and now we use it as an operational way to identify the Kmers since each cluster will naturally correspond to a K-mers originated from the genome groundtruth(though there are a few discrepancy, mostly this is correct).

4. After we identify the K-mers clusters, we add an edge between them if there exists a read such that there are two consecutive Kmers originate from it.

Graph surgery to clear abnormality of the noisy DeBruijn graph Due to indel noise and runs of the same alphabet, the way that we form K-mers graph may need to abnormality of the graph. We thus perform a graph tranversal and identify the abnormality that are of short length(i.e. resulted from noise but not the genome structure). After that, we remove such abnormality. This step also involves transitive edge reduction and removal of small self loops.

C.2 X-phased step tailored for indel noise type

Generalization to handle Indel Error When dealing with indel noise, the neighborhood of reads can also affect consensus of the base. There we have to do sequence alignment in order to find the appropriate posterior probability in order to do a maximum likelihood estimate of whether a particular given genomic location is a SNP or not. In order to do that, we formulate the problem as a ML problem as follows.

$$\max_{T\in\Omega} \quad \Pi_{i\in S} P(R_i \mid T), \qquad P_{err} = P_{opt} \tag{4}$$

$$\max_{T\in\Omega'} \quad \Pi_{i\in S} P(R_i \mid T), \qquad P_{err} = P_{opt} + \delta_1 \tag{5}$$

$$\max_{T\in\Omega'} \quad \Pi_{i\in S'} P(R_i \mid T), \qquad P_{err} = P_{opt} + \delta_1 \tag{6}$$

$$\max_{T\in\Omega'} \quad \Pi_{(j,k)}\Pi_{i\in S'_{jk}} P(R_i \mid T_j^{j+k}), \quad P_{err} = P_{opt} + \delta_1 \tag{7}$$

$$\max_{T\in\Omega'} \quad \Pi_{(j,j+1)}\Pi_{i\in S'_{j,j+1}} P(R_i \mid T_j^{j+1}), \; P_{err} = P_{opt} + \delta_1 + \delta_2 \tag{8}$$

Here we also discuss about the places that we take approximation to enhance the computational efficiency in the steps of the previous reduction. From (1) to (2), we use some heuristics to find out the possible location of SNPs within the whole repeat in which disagreement is observed after several rounds of error correction. From (2) to (3), we remove all the reads that only span one single SNPs and it has no effect on the error of the detection problem that we are trying to solve. From (3) to (4), we further partition the reads into group in which S'_{jk} is the set of reads that only span the SNPs j to j+k. Doing this can decompose the ML problem into smaller subproblems with no effect on the accuracy. Finally, in practice, we take a first order approximation of (4) to (5) by only onsidering two SNPs for each subproblem.

As for each of the marginal probability distribution, the best way is to run Sum-Product algorithm to compute in a dynamic programming fashion similar to S-W alignment. But as pointed out in Quiver, this steps can be significanly speeded up using a Viterbi approximation and this is also what we implemented in the simulation code.

Simulation study We simulated on both synthetic and real data set with indel noise and on a double stranded DNA. In the simulation, we assume that the reads from the neighborhood of a repeat is given and our goal is to decide how to extend the reads to span the repeat copies into the flanking region correctly. The correctness is evaluated based on whether they can correctly extend the correct reads into the flanking region.

Repeat Type	C_s	L_s	C_l	L_l	p_{del}	p_{ins}	G	Homology	l^{approx}	l^{exact}	Success %
Randomly generated	50X	100	50X	240	10%	10%	10000	0.67%	300	150	99%
A repeat of Ecoli-K12	–	–	80X	3000	10%	10%	4646332	0.48%	5182	1507	89%
A repeat of Bacillus anthracis	–	–	80X	3500	10%	10%	5227293	0.23%	4778	2305	85%
A repeat of Meiothermus ruber	–	–	80X	750	10%	10%	3097457	1.40%	1217	257	94%

Table 4: Simulation results on long contig creator(C_s, L_s are coverage and readlength for short reads. C_l, L_l are coverage and readlength for long reads .p_{del}, p_{ins} are the probability of insertion and deletion. G is the length of the genome. Homology is the number of SNPs divided by the length of the approximate repeat. l^{approx}, l^{exact} are the length of the approximate and exact repeat being studied. Success % is the percentage of success in 100 rounds)

C.3 Edit distance metric calibration

We also do a study on whether we can use alignment score to differentiate segments from being extracted from the same genomic location or not. In Fig 13, the upper curve is the score for segment extracted from the same genomic location while the bottom curve is for completely iid randomly(irrelevant) generated segment. And we simulate it for 100 times at each length and the bar indicate 1 standard deviation from the mean.

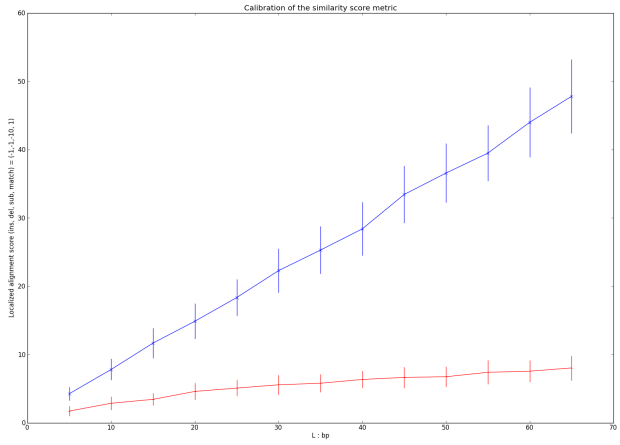


Fig. 13: A calibration for similarity score using global alignment computation.

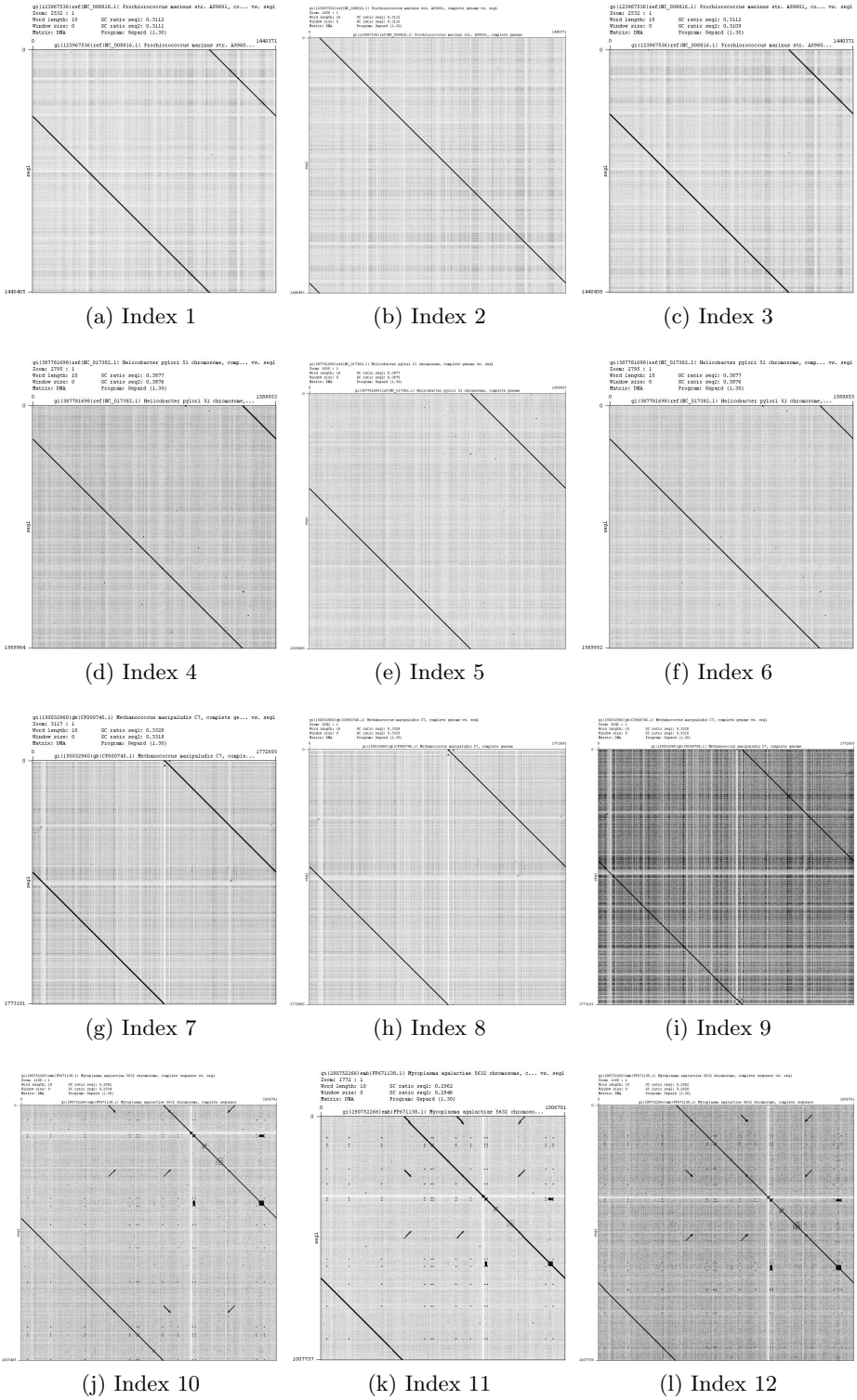
C.4 Tolerance in the Multibridging step

We note that due to the indel noise and the graph surgery that we perform, an X-node of the graph may be p times longer than the usual size of the approximate repeat, thus we should have a corresponding higher tolerance to use the reads to bridge across the repeats.

C.5 Computation speed up of alignment step

The key bottle neck in computation speed of the indel extension is on the pairwise alignment of the reads, which can be speeded up using the ideas in BLAST. We use sorting to identify exact matching fingerprint that identify the starting and ending location of the segment that need to be aligned with. After that, we do a local search instead of the whole dynamic programming search.

D Appendix of the dot plot of finished genomes



(a) Index 1 (b) Index 2 (c) Index 3

(d) Index 4 (e) Index 5 (f) Index 6

(g) Index 7 (h) Index 8 (i) Index 9

(j) Index 10 (k) Index 11 (l) Index 12

Fig. 14: Dot plot of recovered genomes against ground truth(according to index in Table 2)

E Appendix : Evidence behind model

E.1 Approximate Repeat

We let the underlying genome be $\boldsymbol{x}$ and use the short hand that $x[a:b]$ be the a^{th} to $(b-1)^{th}$ entries of $\boldsymbol{x}$.

Let $\boldsymbol{v}_1 = x[s_1 : s_1 + l]$ and $\boldsymbol{v}_2 = x[s_2 : s_2 + l]$ be two length l substrings of the genome with starting positions at s_1 and s_2 respectively. We call $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ be an approximate repeat of length l if

$d(x[s_1 - W : s_1], x[s_2 - W : s_2]) \geq 0.7W$

$d(x[s_1 + l : s_1 + l + W], x[s_2 : s_2 + l + W]) \geq 0.7W$

$d(x[s_1 - W + k : s_1 + k], x[s_2 - W + k : s_2 + k]) < 0.7W$ for all $0 < k < l$

To understand approximate repeat better, we plot the Hamming distance for consecutive disjoint window of length 10 as shown in Fig 15 .

E.2 Classification of approximate repeat

While repeats are studied in the literature[2], they are not investigated by looking at the ground truth. This is partially due to the insufficiency of data in the early days of genome assembly development. Therefore, based on the ground truth genome, we define several quantities that allow us to classify approximate repeat and understand the approximate repeat spectrum of genome. Here we define stretch and mutation rate. Stretch is defined to be the ratio of the length (l^*) of the longest exact repeat within an approximate repeat divided by the length (l_{approx}) of the approximate repeat. Mutation rate is defined to be number of mutation within approximate repeat divided by ($l_{approx} - l^*$). An illustration is shown in Fig 15.

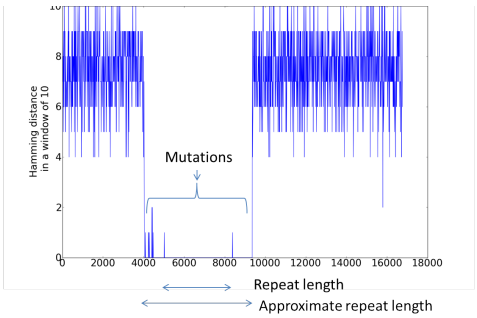


Fig. 15: Example of how to define stretch and mutation rate

Moreover, we do a scatter plot to classify the approximate repeats(approximate repeat having exact repeat length within top 20) and we have a plot of approximate repeat spectrum as in Fig 16.

From the plots in Fig 16, we classify approximate repeat as homologous repeat if the stretch is bigger than 1.25 and as non-homologous repeat if the stretch is less than 1.25.

For the scatter plot, every approximate repeat is a dot there with x coordinate and y coordinate being mutation rate and stretch respectively. And the color represents the length of that approximate repeat. For the approximate repeat spectrum plot, the red bar represent non-homogeneous repeat while the blue bar represent homologous repeat. The green dotted line indicates the length of the longest repeat.

We focus on genomes when the non-homologous repeat dominates, namely the longest interleave and the longest triple repeats are non-homologous because the stretch is relatively short which can be captured by our generative model. We do not distinguish between the length of approximate or exact repeat are considered to be the same and we do not distinguish between the two in the discussion because of the small stretch.

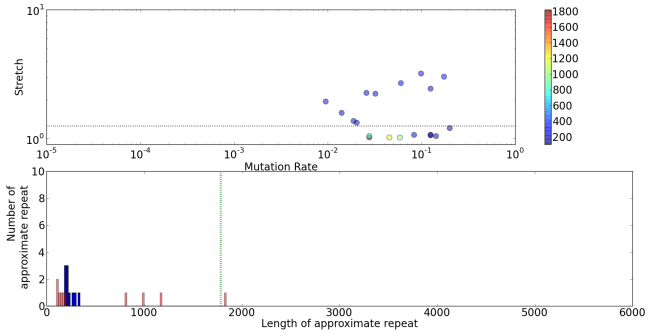


Fig. 16: Classification of approximate repeats and approximate repeat spectrum. The upper plot is scatter plot to classify approximate repeat. The lower plot is the approximate repeat spectrum

E.3 Stopping criterion for defining approximate repeat by MLE estimate

Parametric model Let L_k be the number of bases between the $(k-1)^{th}$ and the k^{th} SNPs starting from the right end-point of a repeat.

We consider the following probabilistic model for the L_k. $\{L_k\}_{k=1}^n$is taken as an independent sequence of geometrically distributed random variables with parameter $\Theta = \{p_1, p_2, r\}$ defined as follows.

$$L_k \sim \begin{cases} Geo(p_1) & \text{if } 1 \leq k \leq r \\ Geo(p_2) & \text{if } r < k \leq n \end{cases}$$

MLE estimate of parameters We now would like to estimate Θ given the observation of $\{\hat{L}_k\}_{k=1}^n$ by maximum likelihood estimation. Consider the log-likelihood function $L(\Theta) = \log \mathcal{P}(\{\hat{L_k}\}_{k=1}^n \mid \Theta)$.

$$\begin{aligned} L(\Theta) &= \log \mathcal{P}(\{\hat{L_k}\}_{k=1}^n \mid \Theta) && (9) \\ &= \log\{[\Pi_{k=1}^r (1-p_1)^{\hat{L_k}} p_1] \cdot [\Pi_{k=r+1}^n (1-p_2)^{\hat{L_k}} p_2]\} && (10) \\ &= r \log p_1 + [\sum_{k=1}^{r} \hat{L}_k] \cdot \log(1-p_1) + (n-r) \cdot \log p_2 + [\sum_{k=r+1}^{n} \hat{L}_k] \cdot \log(1-p_2) && (11) \end{aligned}$$

And we want to find $\hat{\Theta} = \arg\max_{\Theta} L(\Theta)$.

Observe that , if we fix $1 \leq r \leq n$, then the optimal $\hat{p}_1$ and $\hat{p}_2$ can be readily obtained by taking derivative on $L(\Theta)$ with respect to $p_1$and p_2 , specifically,

$$\hat{p_1} = \frac{1}{1 + \frac{\sum_{k=1}^{r} \hat{L}_k}{r}} \quad (12)$$

$$\hat{p}_2 = \frac{1}{1 + \frac{\sum_{k=r+1}^{n} \hat{L}_k}{n-r}} \quad (13)$$

$\hat{\Theta}$ can then be obtained by running over all integral $1 \leq r \leq n$ and use the corresponding optimal $\hat{p}_1$ and $\hat{p}_2$ to obtain the $L(\Theta)$,and finally we use the r that gives the highest value of $L(\Theta)$ as the MLE estimate given the observation.

Linear time algorithm to estimate the stopping criterion Moreover, this can be done by the following algorithm Algo 6, which run in linear time $\Theta(n)$ with respect to the number of observations n.

A sample plot is of who we can use the critierion to accurately define the ending of approximate repeat is shown in Fig 17.

Algorithm 6 Linear time algorithm to estimate the stopping criterion

1.
a)$C_0 \leftarrow 0$
b) for r = 1 to n
$C_r \leftarrow C_r + \hat{L}_r$
2.
a)$D_0 \leftarrow C_n$
b)for r = 1 to n
$D_r \leftarrow C_n - \hat{L}_r$
3. for r = 1 to n
$\hat{p}_1^{(r)} \leftarrow \frac{1}{1+\frac{C_r}{r}}$
$\hat{p}_2^{(r)} \leftarrow \frac{1}{1+\frac{D_r}{n-r}}$
$\Theta_r \leftarrow (r, \hat{p}_1^{(r)}, \hat{p}_2^{(r)})$
$X_r \leftarrow L(\Theta_r)$
4. find maximum among $\{X_r\}_{r=1}^n$, and the corresponding Θ_r is the MLE estimate.
5. (Differentiate between homologous and non-homologous repeat)
If the optimal $\hat{p}_1^{(r)}, \hat{p}_2^{(r)}$are too close (i.e. $\hat{p}_1^{(r)} > 0.2$), then claim $r = 1$; else, claim $\hat{r} = r$.

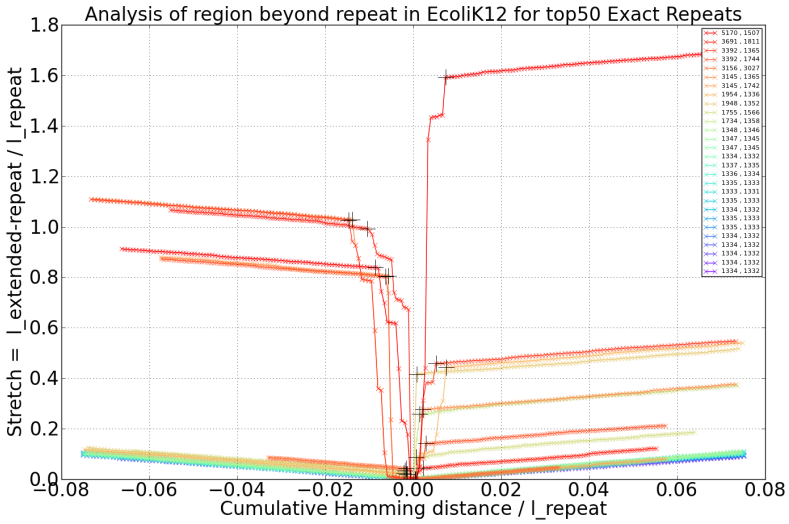


Fig. 17: An example plot that define the stopping point of approximate repeat by Algorithm 6