Universiteit Utrecht

Faculteit Bètawetenschappen

Determining plasmids from short read sequences

Bachelor Thesis

Yaïr Hein
5720176

Mathematics (TWIN)

Supervisors:

Martin Bootsma Supervisor
University of Utrecht

Anita Schürch Supervisor
University of Utrecht

Sergio Arredondo Alonso Supervisor
University of Utrecht

09-06-18

Abstract

In this thesis we present a novel method to determine the DNA-sequence of plasmids from a De Bruijn Graph. The method uses coverage data as well as estimates, based on the program mlplasmids, how likely a certain contig is plasmidal or chromosomal. The algorithm first thins the Bruijn Graph by removing all edges which are unlikely to be plasmidal. Next, all simple circular paths are determined using Johnson's algorithm. Finally a Markov Chain Monte Carlo method is used to determine the circular paths which fit the observed coverage data well. The method discussed in this thesis could identify some small plasmids in a few real De Bruijn graphs of *Enterococcus faecium* and we have several suggestions to modify the method to find larger plasmids as well.

Contents

1 Introduction

Plasmids are circular pieces of DNA that reside in bacteria in addition to their chromosomal DNA. They can confer bacteria with properties such as antibiotic resistance. What sets plasmids apart from the chromosomes is that they can be transmitted horizontally between bacterial host cells, thus allowing certain genes to spread more easily even among different bacterial species[1]. Genomes may contain any non-negative number of plasmids.

To fully understand outbreaks, plasmid reconstruction is required. Comparing plasmid genomes within an outbreak with one another helps to map the transmissions between persons and trace back infections to a common source [2]. For these types of comparisons it is best if the plasmid genomes are as complete as can be.

The DNA double helix is made up of two base strands. Both strands are made up of the base molecules adenine, cytosine, guanine and thymine denoted by A, C, G and T respectively. Each of these bases on one strand is connected to one base on the opposite stand. The only allowed base pairs are A-T, T-A, C-G and G-C, hence the sequence on either one of the strings characterizes the complete DNA and all of its base pairs. We can therefore denote a DNA strand as a string of the bases A, C, G and T; for example ATCGATTGAGCTCTAGCG. Complete genomes of chromosomes and plasmids are always circular loops of such strings. Substrings consisting of k base pairs taken from such loops are called k-mers.

As it stands, there are two commonly used methods to study DNA, long-read sequencing and short-read sequencing. Long-read sequencing is able to capture large DNA sequences at once, large enough to capture the overall structure, but capital costs restrict the usage [3]. Moreover, long-read sequencing is prone to errors. They have a higher error rate affecting the DNA sequence than short read sequencing techniques. On the other hand, the read length obtained from short-read technologies only allows to represent a genome as approximately hundred substrings called contigs, in which it is challenging to infer plasmid- or chromosome-origin.

Short-read length ranges from 50 to 250 base pairs. These reads are split into smaller substrings, called k-mers of more constant lengths. These k-mers are assembled into sets of contigs by looking at content overlaps of the k-mers. Contigs are the strings that can unambiguously be retrieved from overlaps within this k-mer data. Once all contigs have been determined, you are left with a graph that captures in which ways these contigs can be connected to one another called a De Bruijn Graph. A more mathematical description of De Bruijn Graphs is given in Appendix A. Currently the most general way to solve these De Bruijn Graphs, is to use a mapping algorithm which tries to find solutions by aligning sequences against databases of known genomes[4]. Some problems with this method are that it has trouble capturing structural information (such as the genome sequences on a large scale) [1] and that the method is based on known genomes. If a certain plasmid has not been found before then this method will not find it. Other existing methods to recover plasmids from De Bruijn Graphs include programs such as Recycler [5] and PlasmidSPAdes [6], each of which have their own shortcomings[7]. Recycler for instance was tested with fabricated data and it could only reliably identify small plasmids in real graphs.

This thesis takes a look at the possibilities for automating this second step, reconstructing plasmids from De Bruijn Graphs, using overlooked but useful data, the coverage of the contigs, as well as the probabilities of contigs being chromosomal or plasmidal based on the program mlplasmids [8], which is based on a machine-learning algorithm that considers the k-mer frequences of contigs. The focus of this assembly is more about finding which contigs appear together in the same plasmids rather than providing the complete plasmids.

The data used in this thesis is from the short-read sequencing of the genomes of E8867, E0139, E1334 and E7237, which are four different *Enterococcus faecium* strains. Each of these genomes has been sequenced to completion using long-read sequencing. This allowed us to define a set of complete plasmid sequences to assess the correctness of the outcome given by our proposed algorithm. All data and a python script of the algorithm discussed later in the thesis are available at https://github.com/yaires/plasmids.

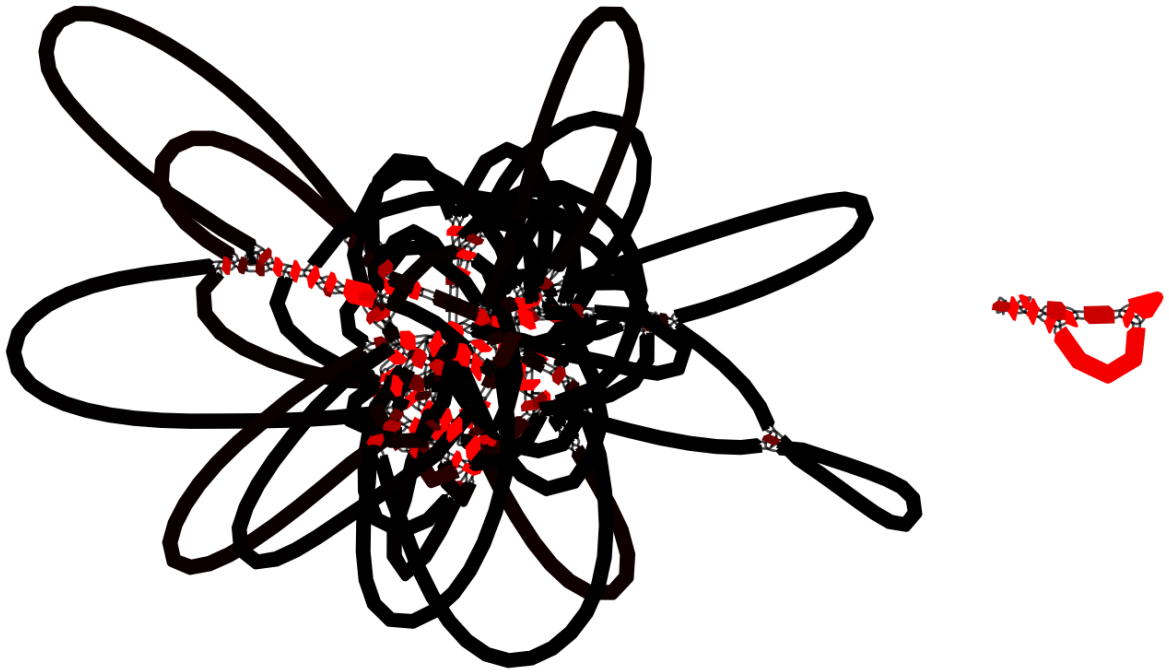

Figure 1: A visualisation of the De Bruijn Graph of E8867 generated by Bandage. Every thick line corresponds to a contig. Brighter shades indicate a higher coverage.

2 Theory

2.1 What are contigs?

The plasmids we want to examine can be expressed as circular strings of nucleotide bases, ranging from 1000 to 300000 base pairs. Consider for example the following string GTTTCCACTTTGGAGCAC. This consists of a set of 4-mers, GTTT, TTTC, TTCC, TTCA, CCAC, CACT, ACTT, CTTT, TTTG, TTGG, TGGA, GGAG, GAGC, AGCA, GCAC, CACG, ACGT, CGTT, which are obtained from the short-read data. Our goal is to reconstruct the genome from these k-mers. A quick glance reveals that GGAG is the only read ending with GAG and GAGC is the only read beginning with GAG, hence GGAGC must appear in the original genome. Continuing this reasoning one can deduce that TTTGGAGCAC, TTTCCAC and CACTTT must appear. These unambiguous pieces are called contigs. Based on these contigs, one may still end up with different solutions like TTTCACTTTCACTTTCACTTTGGAGCAC since the total short read data is not unambiguous. One can visualize the possible solutions using a De Bruijn Graph. Here the contigs are thought of as strings where the endpoints are connected to other contigs that have overlaps in their endpoints, e.g., the end of CACTTT gets connected to beginning of TTTGGAGCAC. An example of a De Bruijn Graph is shown in Figure 1. In practice the first step of reducing short reads to a De Bruijn Graph is done using an algorithm such as SPAdes or Velvet [4]. A more mathematical description of a De Bruijn Graph and contigs is given in Appendix A.

A genome always contains any non-negative number plasmids and one larger chromosome, each of which can be expressed as a circular string of bases. These are not separated from one another before taking short reads, hence a De Bruijn Graph taken from a short-read sequence of this genome will put all contigs from the chromosome and all plasmids in this genome in a single graph. Ambiguities in this graph therefore occur not only when there is a repeated section within the chromosome or any of these plasmids, but also when a

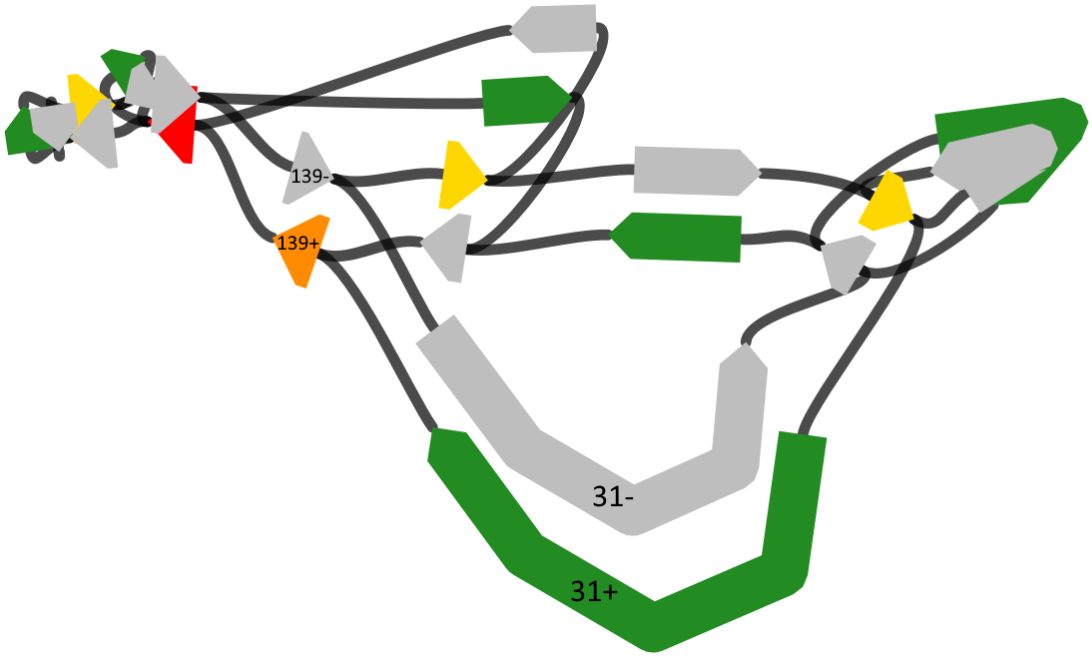

Figure 2: A section of the De Bruijn Graph of E8867 which shows that the complementary contigs 31+ and 31- are both connected to 139+. Each grey contig is a complement of some coloured contig of similar size.

section is repeated among any of the plasmids and the chromosome. Even though we are only interested in the structures of the plasmids, De Bruijn Graphs will still contain lots of contigs which appear only in the chromosomal DNA string.

DNA always consists of pairs of two complementary strings, so each contig always has its complementary version present as well. In the format of the De Bruijn Graphs, complementary contigs have complementary k-mer contents, so all bases A replaced by T and bases C replaced by G and vice-versa. We characterize each pair of complementary string by a 'positive' and 'negative' contig, for example 45+ and 45-. There is no rule to tell which complement should be 'positive' and which should be 'negative'. The signs are initially assigned randomly and they are only used to tell the complements apart. For any two connected contigs, their complements must be connected too. Contig complements are also read in opposite directions. Consider a complementary pair of contigs named 45+ and 45-. If the end of 45+ is connected to 3+, then the beginning of 3- is also connected to 45- in the De Bruijn Graph.

Next we wish to convert this De Bruijn Graph to a conventional directed graph, in order to use existing path-finding algorithms. Since complements do not provide any additional information, one may be inclined to ignore contig complements when converting a De Bruijn Graph into a conventional directed graph. This goes wrong when a particular contig can indirectly be connected to both complements of some contig. In this case the De Bruijn Graph cannot be separated anymore into two complementary subgraphs. An example of interfering complements is given in Figure 2 which is actually taken from one of the graphs we are using, E8867. We, therefore, use all information given in the De Bruijn Graph, turning all complementary contig pairs into node pairs and all connections into directed edges, where the direction points in the direction the contigs are read in. If the end of 45+ is connected to the beginning of 3+ then this edge points from node 45+ to node 3+. For any circular path in this graph we now expect to be able to find its complementary circular path as well. This is good since physically, the chromosomes and plasmids consist of two complementary

circular strings too. On the other hand, chromosomes and plasmids have no predetermined direction, while our graphs do. This however does not cause any problems since contig directionality was properly assigned during the creation of the De Bruijn Graph.

2.2 Contig properties

The length of a contig generally ranges from 100 to 300000 base pairs. Short read sequencing keeps track of how many times each read occurs. Due to the nature of the method, these numbers are not completely accurate. The number of times a base-pair or k-mer is present in the sequencing reads is called coverage. Here, we considered k-mer coverage reported by SPAdes. Since longer contigs have a larger k-mer content to average over, their coverage should be more accurate than that of smaller contigs, hence length will be used as a weight for the coverage accuracy. The data we will be using has the coverage numbers per contig normalized such that they correspond to the actual number of times a contig is thought to appear in the genome. The fact that each plasmid may have multiple copies of itself within a genome while there is one chromosome will be reflected by the coverage numbers. If a contig has a high coverage, then it is either present in at least one of the plasmids or it has to be a repeated element of the chromosome.

Another property we consider is the probability of whether a contig is plasmidal or chromosomal determined by the program mlplasmids [7]. The program uses a machine-learning algorithm to estimate how likely each contig is to be part of a plasmid or the chromosome based on its k-mer contents. It has been calibrated on known genomes of *E. faecium* that had been mapped using long-read sequencing. Note that this program does not consider coverage, so all of the plasmid probabilities are independent of coverage. Our algorithm will be based on the following properties:

- connections
- L_i length
- p_i plasmid prediction probability
- x_i measured coverage

We have De Bruijn Graphs, contig property data and lists of the actual plasmids present for *E. faecium*. genomes E0193, E1334, E7237 and E8867. The lists containing the actual plasmid solutions were constructed with the help of long-read sequencing data. Unfortunately, due to technical limitations, not all of the plasmid solutions supplied for the graphs of E0193 and E1334 are circular paths. We will still use the data from these graphs for data analysis. A look at the data of the contigs (see Figure 3) reveals the following.

There are no short contigs for which mlplasmids can predict whether the contig is chromosomal of plasmidal. This is to be expected since the predictions of mlplasmids are based on k-mer data and the shorter the contig, the less k-mers are available. Every contig with a length below approximately $400 \approx 10^{2.6}$ base pairs consistently has a plasmid probability very close to 0.5 meaning that mlplasmids does not provide us any information on whether the contig is chromosomal or plasmidal.

We expected to see higher uncertainties in coverage for contigs with lower lengths. The fact that the coverage numbers of contigs with higher lengths tend to lie closer to integer values supports this. Physically, the amounts of times contigs appear are integer values. Deviations from integer values are caused by small errors in the short-read data.

In general, the plasmid probabilities do not reliably identify the contigs which are not plasmidal. mlplasmids does however reliably identify all contigs appearing in plasmids solutions with a coverage of about 1 as having a chance of being plasmidal of above 0.5, with the exception of very short contigs. The reason some contigs appearing in plasmid solutions with higher coverage end up as chromosomal according to the program can be explained by contigs appearing in both the chromosome and a plasmid at least once.

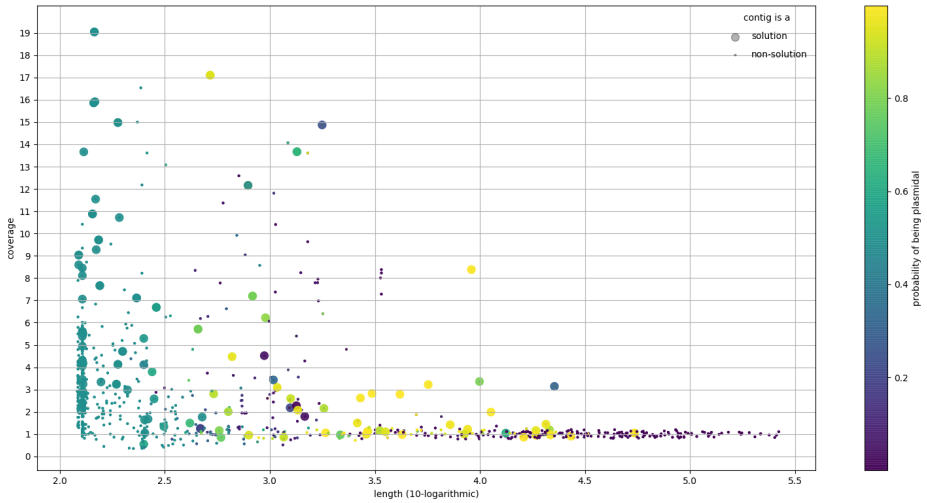

Figure 3: A scatter plot containing the data of the plasmid probabilities, length and coverage of all contigs from the graphs E0193, E1334, E7237 and E8867. All contigs which appear in the actual plasmids (as determined by data from long-reads) appear as large dots.

Due to the relatively small sample size used here and the unknown behaviour of the distributions of most properties, it is unfortunately not feasible to do a proper data analysis of the contig properties. We will therefore assume the above observations to be true in general and base our algorithm on them.

3 Approach

3.1 Reducing the dataset

Our goal for each De Bruijn Graph is to find the set of circular paths of contigs which most likely correspond to the actual plasmids. Because of the large size of the graphs it is infeasible to generate all circular paths and then analyse those. The optimal algorithm for finding cycles in a graph is Johnson's algorithm which has a computation time of $O((\eta+1)(m+n))$ with n the number of nodes, m the number of edges and η the number of loops [9]. It turns out that the number of loops increases a lot with general graph size. There are a number of ways we can simplify this graph before running any path-finding algorithm, since the majority of contigs in a graph are exclusively chromosomal and we are not interested in finding the chromosome.

The graphs we are using have around 400 nodes if you include different complements of the contigs and 600 edges. The first simplification is to remove all contigs with a length of below 800 and patch up the connections. All nodes which were connected to one of these short contigs are reconnected to all of the nodes this short contig was connected to. This step should not remove too much actual information since in the previous paragraph we have seen that short contigs tend to have high uncertainties in their coverage values and plasmid prediction probabilities. This action decreases the total amount of nodes while increasing the number of connections per node. The total amount of different paths between any pair of large contigs decreases, hence the overall complexity is less in the reduced graph.

The assumption was that from the remaining contigs, only contigs expected to appear once, say with a coverage of less than 1.5 and with odds in favour of being chromosomal could reliably be identified as contigs that could not appear in any of the plasmid solutions. Instead of removing all of these contigs, we remove edges based on this assumption where their plasmid probabilities are defined to be the mean of the probabilities

of the contigs they connect. This lets the selection be more accurate, since contigs likely to be chromosomal may now still go through as long as they are connected to contigs likely to be plasmidal.

So in this step the plasmid probabilities of all contigs with a coverage higher than 1.5 are set to 1. An overview of all edges and their plasmid probability means is given in Figure 4. One can see a clear divide between edges in the bottom half where no edges occur in any of the plasmid solutions and the edges in the upper half, a mixture of edges which occur in solutions and edges which do not. By throwing away all edges with a plasmid probability lower than 0.6, the complexity is reduced by a lot while we do not lose any of the actual solutions. There is unfortunately no guarantee that this approach works for all De Bruijn Graphs.

A thing to note here is that circular paths found in these simplified graphs do not directly correspond to the actual solutions due to the removal of short contigs from the intial graph. Only sets of larger contigs can be found. From now on, whenever we compare loops to the given plasmid solutions, we consider these solutions with all short contigs left out.

3.2 Finding circular paths

We now use Johnson's algorithm to find all simple circular paths in the remaining graphs. A simple circular path is a loop which does not contain any node more than once. The plasmid solution of E8867, namely '47-,80+,31+,52-,61-,63+,55-,80+' contains a particular node more than once and would therefore not get picked up directly by the Johnson algorithm. It would however still find circular paths '47-,80+' and '31+,52-,61-,63+,55-,80+'. One just needs to keep in mind that the larger path consisting of them is also a possibility if these two then end up as likely plasmids by the eventual algorithm. This list of simple loops may still contain loops which have both complements of the same contig, since an unaltered Johnson algorithm considers those different nodes. We delete such loops. Although there is no clear evidence that such loops do not exist, our plasmid solutions do not contain any.

It is a problem that for two out of four graphs (E0139 and E1334) the supplied solutions are not circular. For now we are assuming this is caused by errors within the De Bruijn Graphs and disregard these graphs in our testing. A workaround would require a different approach that does not rely on finding circular paths.

3.3 Fitting the circular paths

At this point we have a finite set of loops which should contain the actual plasmids. The next step is to figure out which of these loops are most likely to be the actual plasmids by looking at coverage.

Ideally, the De Bruijn Graph with its coverage numbers is expected to be a projection of the actual chromosome and plasmids. For example, consider a genome consisting of one chromosome and one plasmid which has two copies present. Assuming there are no repeated sections, this leads to a graph where all contigs exclusive to the chromosome have a coverage of one, all contigs exclusive to the plasmid have a coverage of two and all contigs present in both chromosome and plasmid have a coverage of three. Reconstructing the plasmid and the chromosome from this graph would not be too hard.

In practice the actual coverage numbers have some deviations. We will assume the observed coverage numbers follow some normal distribution centered around the integer values corresponding to the actual amount of times their contigs appear in the genome. We will denote the actual amount of times a certain contig i is present in the genome by an unknown integer parameter μ_i and we denote the vector $(\mu_1, \ldots, \mu_n)$ by $\boldsymbol{\mu}$. Call the standard deviation of this particular contig σ_i, which will be some function of the other attributes of contig i. The measured coverage numbers are now represented by a random variable $X_i \sim N(\mu_i, \sigma_i)$. Say there are a total of n contigs. Given a particular set of coverage value measurements $\boldsymbol{x} = (x_1, ..., x_n)$, the likelihood is

$$L(\boldsymbol{\mu}, \boldsymbol{\sigma}) = \prod_{i=1}^{n} \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{1}{2} \left(\frac{x_i - \mu_i}{\sigma_i} \right)^2 \right). \tag{1}$$

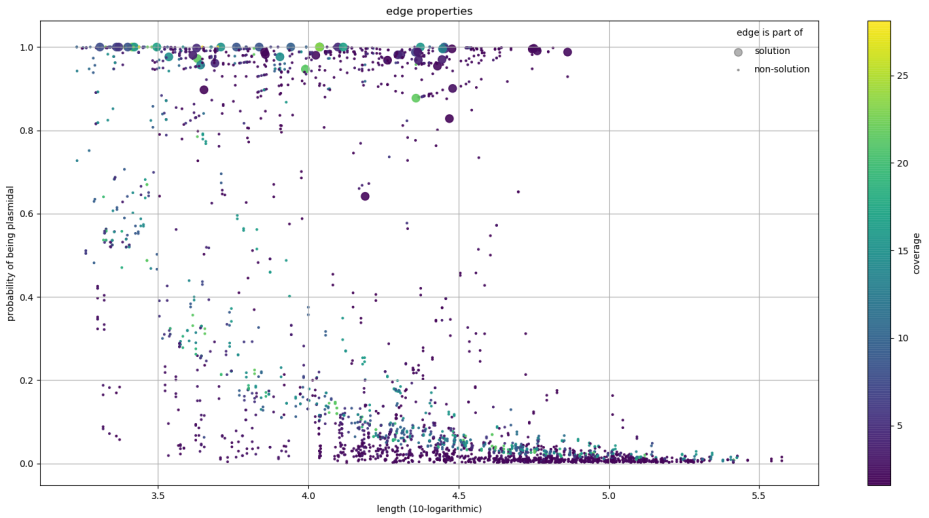

Figure 4: An overview of all edges after removing contigs with a length of below 800. The plasmid probabilities on the y-axis are calculated as the mean of the contigs they connect where contigs with a coverage of above 1.5 had their probabilities set to 1. The coverage and length are calculated as the sum of contigs a given edge connects.

the values of μ_i that are most likely to produce $x_1, .., x_n$ maximize this function. We will consider all σ_i to be constants which we define later. By taking the logarithm of this function, one can easily find that maximizing this for $\boldsymbol{\mu}$ is equivalent to minimizing

$$\sum_{i=1}^{n} \left(\frac{x_i - \mu_i}{\sigma_i} \right)^2 . \tag{2}$$

In other words, finding a weighted least squares. We are however not free to just choose and fit all values of $\boldsymbol{\mu}$. Say the total list of loops being examined has m loops in it, loops denoted by $\boldsymbol{y}_1, .., \boldsymbol{y}_m$. Each loop $\boldsymbol{y}_j$ can be considered a vector of length n where each entry i corresponds to the amount of times contig i appears in this loop, for example $\boldsymbol{y}_j = (0, 0, 1, 1, .., 0)$. Because all loops are simple loops, no loop can contain any contig more than once, so the entries in $\boldsymbol{y}_j$ only take on 0 and 1. The values we are now going to change are the amounts $r_1, ..., r_m$ of loops we think the actual graph consists of. All values of r_j are non-negative integers. $\boldsymbol{\mu}$ then relates as follows to the values of r_j:

$$\boldsymbol{\mu} = \sum_{j=1}^{m} r_j \boldsymbol{y}_j . \tag{3}$$

In other words, an estimate of μ_i corresponds to the r_j summed over all loops contig i is present in.

This does however raise a problem if there are several different combinations of loops that cover the same 'optimal' values of $\boldsymbol{\mu}$. Consider the following example. Suppose we had a list of the loops '1,2' ,'3,4' and '1,2,3,4' and an optimal fit when $\mu_i = 1$ in all i. One would find that one loop '1,2,3,4' would have resulted in the graphs just as likely as the combination of '1,2' and '3,4', even though the single longer loop seems more likely now considering that it was less likely to have appeared in the De Bruijn Graph by chance than the two smaller loops. We therefore add a slight bias for longer loops, so fewer loops in total, to the likelihood in the form of a prior. Let k be the amount of different loops a proposed cover has. In other words, the amount of $j = 1, .., m$ for which $r_j \geq 1$. The prior we use is a geometric distribution

$$P(k) = (1 - p)^{k-1} p. \tag{4}$$

Where the expectation value of this distribution $1/p$ should equal the amount of different loops one expects to find. The amount of different replication genes in a genome could for instance be used to base this value on. We are unfortunately not basing this value on the replication genes now, because we do not have the necessary data. We will test the algorithm for different values of p later. The unnormalized posterior probability is now given by

$$p(\boldsymbol{x}|\boldsymbol{r}) \propto (1-p)^{k-1} p \prod_{i=1}^{n} \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x_i - \mu_i}{\sigma_i}\right)^2\right). \tag{5}$$

The proposed function for the uncertainty of contig i is given by

$$\sigma_i = \frac{C}{p_{plasmid,i}} \sqrt{\frac{x_i}{L_i}}. \tag{6}$$

Here x_i is the measured coverage of this contig, L_i is its length in base pairs, $p_{plasmid,i}$ is the probability of this contig being plasmidal and C is some unknown constant. As mentioned before, the coverage of a contig is an average of coverages of k-mers they are derived from. The average amount of k-mers a contig consists of is proportional to the contig's length in base pairs, hence the standard deviation of the coverage average is inversely proportional to the root of its length. A similair thing applies for the coverage itself. If multiple copies exist of the same contig, the contig coverage can be seen as a sum of the coverages of the copies, hence the standard deviation is proportional to the root of the coverage itself.
All contigs classified as part of the chromosome with high coverage were assumed to be present in both plasmids and the chromosome. Their presence in the chromosome may still cause an increase in coverage. We solve this by factoring in the plasmid probability, essentially lowering the weight of these contigs in the later calculations.

We use a Markov Chain Monte Carlo algorithm to find the distribution of the values $r_1, \ldots, r_m$ which fit the coverage data well. We start the chain at $r_i = 0$ in all i and calculate the posterior probability for this state. Each iteration, we first calculate the posterior probability for the current state $\boldsymbol{r}$ and then sligtly alter it as follows. Select three different loops, so uniformly at random choose three different values $j = 1, .., m$. For each of these loops we add one, subtract one or do nothing to the loop count r_j with equal probability. If the current count is zero, we add one with probability $1/3$ and do nothing with probability $2/3$, such that the probability to propose a state $\boldsymbol{r}$ given a state $\boldsymbol{r'}$ is the same as the probability to propose the state $\boldsymbol{r'}$ given the state $\boldsymbol{r}$.
If the posterior probability of these new values is higher than the previous posterior probability, then these new values are automatically accepted and the same step is repeated starting from these new values. If the new posterior does not exceed the old one then these new values are still accepted with a probability equal to the ratio of the posterior probabilities

$$\frac{p(\boldsymbol{x}|\boldsymbol{r^*})}{p(\boldsymbol{x}|\boldsymbol{r})} = (1-p)^{k^*-k} \exp\left(-\frac{1}{2}\sum_{i=1}^{n}\left(\frac{x_i - \mu_i^*}{\sigma_i}\right)^2 - \left(\frac{x_i - \mu_i}{\sigma_i}\right)^2\right). \tag{7}$$

Since all posterior probabilities are nonzero, one can easily deduce that using this process, each state can be accessed from any other state at some point with finite probability. We iterate this process a certain number of times and keep track of the values of $r_1, ..., r_m$. We find the average values of r_j and their deviations throughout the process. The loops with high average values of r_j should be similar to actual plasmids solutions. A brief proof of this property is given in Appendix B

4 Results and Discussion

4.1 Results

Graph E8867 has 400 nodes and 552 edges including double-counting for different complements. After removing contigs shorter than 800 base pairs, there are 162 nodes and 1348 edges. After removing all edges with an average plasmid probability of below 0.65, we are left with 31 nodes and 67 edges. The remaining nodes are all nodes which still have edges connected to them. Running Johnson's algorithm and then removing

graph		E8867	E0139	E1334	E7237
inintial	nodes	400	334	216	644
	edges	552	470	310	893
no short contigs	nodes	162	148	98	270
	edges	1349	322	228	1208
edges removed	nodes	31	40	43	112
	edges	67	84	103	584
distinct loops		6	24	97	>700

Table 1: The four graphs and their node and edge counts initially, after removing all contigs shorter than 800 base pairs and after removing all edges with an average plasmid probability of below 0.65. It also lists the amount of distinct loops each graph contains.

contig	47	80	31	52	61	63	55
x	3.36	6.22	3.14	3.22	2.82	2.62	2.78
σ/C	0.023	0.097	0.037	0.024	0.030	0.032	0.026

Table 2: The remaining contigs and their coverage, x. The uncertainty σ is based on the contig's other properties as in equation 6 and is used to weight the coverage during the iteration process.

loops containing the same or complementary contigs, results in 6 distinct loops. The numbers during this process for the other graphs are listed in Table 1.

The initial loop finding process applied to the graph E8867 left 6 loops, see Table 2. The solution of this graph is a single plasmid denoted by '63,61,52,55,80,47,80,31'. This is not a simple loop so we expect to find both '63,61,52,55,80,31' and '47,80' with similar values of coverage r. Note how all contigs have roughly the same coverage except contig 80 which has about twice that amount. This is exactly what one would expect considering 80 is the only contig which appears twice in the plasmid.
We ran the algorithm for different values of C and p using $8 * 10^6$ iterations each time while keeping track of the average coverage r and their standard deviations σ_r throughout the iterations. For $p = 0.001$ the effects of the prior are supposed to be negligible and for other values, the higher p, the stronger the algorithm's tendency to favour states whith less diversity in coexisting loops.
From the results in Table 3 we see that the loop '47,80' consistently ends up with large coverage. This isn't surprising, since it is the only loop which has 47 in it and contig 80 has a higher coverage than the rest. This loop does not interfere with the presence of other loops. The times this loop does start to get lower coverage averages is when C is high and p is high, which is when the effects of coverage are outdone by the algorithm's tendency to have fewer different loops present at a time.

An interesting phenomenon is happening with the other part of the solution, '63,61,52,55,80,31'. For small values of p, it consistently ends up with lower coverage than the other loops covering the same contigs. An explanation is that the algorithm favours the smaller loops simply because there are more possibilities to fit them in the graph. If there are more valid states where a certain loop has high r, the algorithm is more likely to try states where this loop has high r even when none of these states result in a particularly good fit. For large values of p the algorithm does however nicely approach the optimal solution. In theory $1/p$ should equal the expected number of plasmids which is two in this case according to the long-read solutions, so $p = 0.5$ should be the most realistic choice here. For this value the options seem to be about even. The fact that the results are more in line with our expectations for higher values of p does not mean that this generally the case.

The long-read plasmid solutions supplied were not circular in graphs E0139 and E1334, so there has been some error made while either generating the De Bruijn Graphs or the long-read plasmid solutions. The loops deemed most likely to be solutions by our algorithm showed no resemblance to the supplied long-read solutions. There really is no telling if this is caused by errors in the data or the approach, so we are not going to spend much time analysing these graphs.

$C = 10$, $p =$	0.001		0.5		0.8		0.95	
loop	r	σ_r	r	σ_r	r	σ_r	r	σ_r
'63,61'	1.06	0.86	0.94	1.08	0.66	1.14	0.18	0.69
'80,31,52,55'	1.06	0.86	0.94	1.08	0.66	1.14	0.22	0.78
'63,31,80,55'	1.06	0.86	0.94	1.08	0.63	1.12	0.25	0.78
'52,61'	1.06	0.86	0.94	1.08	0.63	1.12	0.22	0.78
'63,61,52,55,80,31'	0.87	0.85	1.12	1.12	1.71	1.36	2.60	1.00
'47,80'	3.26	0.49	3.26	0.49	3.26	0.49	3.26	0.49

$C = 50$, $p =$	0.001		0.5		0.8		0.95	
loop	r	σ_r	r	σ_r	r	σ_r	r	σ_r
'63,61'	1.07	0.92	0.97	1.05	0.78	1.14	0.38	0.94
'80,31,52,55'	1.23	0.92	1.17	1.08	0.97	1.21	0.49	1.06
'63,31,80,55'	1.03	0.89	0.94	1.02	0.76	1.12	0.37	0.93
'52,61'	1.27	0.94	1.17	1.10	0.92	1.21	0.42	1.01
'63,61,52,55,80,31'	0.76	0.80	0.87	0.98	1.23	1.25	2.12	1.30
'47,80'	3.65	2.10	3.51	2.20	3.12	2.36	2.00	2.41

Table 3: An overview of the six remaining loops in the graph of E8867 and the average values of coverages r and their standard deviations σ_r during a Markov-Chain Monte Carlo process of $8*10^6$ iterations for $C = 10$, 50 and $p = 0.001$, 0.5, 0.8, 0.95

The remaining graph E7237 seems to be too large for the current approach and parameters. With an edge plasmid probability cut-off of 0.987 instead of 0.65, Johnson's algorithm found over $3*10^7$ loops in 10 minutes which reduced to 710 loops after removing complements and order. with a cut-off of 0.97 this already takes over an hour. It is safe to assume that with a cut-off of 0.65 like we used for the other graphs, this graph is not going to have all of its loops listed within a reasonable time by any computer. Fortunately the edges in the long read solutions for this graph all have have a high enough plasmid probability to make it with a cut-off of 0.99. With this cut-off we find 574 distinct loops.

The solutions of E7237 are '69,106' and '62,106,117,132,127,121'. A list of the most prevalent loops during a process of $2 * 10^6$ iterations with parameters $C = 30$ and $p = 0.5$ is given in Table 4. A list of some of the attributes of the contigs seen in this table is given in Table 5. The smaller loop '69,106' gets picked up easily, similar to how the smaller loop in E8867 stood out. What this loop has in common with the previous one is that there is one contig with high coverage which does not show up in any of the other simple loops. The loop consisting of '124' stands out because of its uncertainty as in equation 6 is given by $\sigma/C = 0.0081$ as seen in Table 5 making the weight of this contig in particular very low. This is caused by the fact that this is a contig with a very low plasmid probability that did not get removed due to it having a coverage higher than 1.5. Normally if a certain loop shared other contigs which were not as unrestrained, this would not be a problem. In this case there is little stopping '124' from attaining all kinds of values that do not correspond to its actual coverage. This reflected by the abnormally high deviation $\sigma_r = 9.82$ during the process.

The large solution '62,106,117,132,127,121' ends up with a coverage $r = 0.015$. It is not listed in table 4. It does seem to have decent number of contigs in common with the higher ranked loops. These loops also contain lots of other contigs not part of any of the solutions which still have high plasmid probabilities and coverages such as 57, 85, 122 and more. The existence of such contigs can be explained by the presence of a plasmid which was somehow not listed in the solutions obtained by the long-reads or errors in the probabilities by mlplasmids.

4.2 Discussion

We know that all connections need to be symmetric, if 45- had an edge pointing to 3+ then 3- should have an edge pointing to 45+. This behaviour should always be present during any phase of the process. The posterior plasmid probabilites provided by mlplasmids however, break this symmetry. In some rare cases

$C = 30$, $p = 0.5$		
loop	r	σ_r
'101,132,127'	0.10	0.33
'87,122,92,132,79'	0.11	0.33
'100,127,96'	0.12	0.34
'87,124,122,92,132,79'	0.14	0.39
'79,132,122,124,87'	0.14	0.37
'85,134,135,121'	0.16	0.38
'79,132,122,87'	0.17	0.38
'106,62,122,124,87,79,132,117'	0.18	0.38
'106,62,117'	0.22	0.41
'101'	0.23	0.45
'100,127,121,96'	0.26	0.48
'132,121,127'	0.51	0.75
'132,127'	0.62	0.89
'105,27,63,62,106,117,127,132,57'	0.94	0.22
'124'	8.48	9.82
'106,69'	8.80	0.77

Table 4: An overview of the loops of the graph of E7237 with an average coverage of $r > 0.10$ and their values of σ_r during a Markov-Chain Monte Carlo process of $2 * 10^6$ iterations with parameters $C = 30$ and $p = 0.5$

contig	x	σ/C	$p_{plasmid}$
106	14.87	103	0.9303
69	8.38	1034	0.9754
124	14.07	0.0081	0.0097
105	6.404	0.945	0.0582
27	1.01	31896	0.9978
63	1.18	8674	0.9887
62	1.99	5579	0.9914
117	2.07	601	0.9576
127	3.10	315	0.9492
132	4.52	1.07	0.0717
57	1.01	14317	0.9963
121	2.59	68.1	0.3752
100	4.80	363	0.8681
96	0.93	2756	0.9831
79	1.79	2484	0.8227
122	2.18	304	0.7330
85	1.20	3826	0.9547

Table 5: Some of the contigs in graph E7237 and their coverage, x. The uncertainty σ is based on the contig's other properties as in equation 6 and $p_{plasmid}$ is the original plasmid probability as given by mlplasmids.

the plasmid probabilities differ among complementary contigs. This results in certain edges remaining while their complements get removed during the step where edges are removed based on their plasmid probabilities. Johnson's circular path-finding algorithm now does not always end up finding a reversed loop for every loop it finds. Later on, complements are again merged to single contigs for the Markov-Chain Monte Carlo process. While this asymmetry does not break the process, it should theoretically not happen. The plasmid probabilities should be similar for complementary contigs, since there are no physical situations where one directed contig is taken from the plasmid while its conjugate is taken from the chromosome. The reason these differences do occur is that complementary string have different, albeit complementary k-mer compositions, which are the only things mlplasmids considers when assigning plasmid probabilities.

The data reduction methods based on the removal of short contigs and the subsequent removal of the new edges seems to be useful. There are still lots of improvements to be made here, such as doing a proper data analysis of the contig properties using a larger sample size. The reason we had so little data was due to the lack of public libraries of mapped genomes. A proper analysis would enable one to better define the standard deviation of the coverage, which is actually used in the algorithm. We also assumed the coverage to have a normal distribution centered around the real number of times a certain contig appeared. While to some extent, everything can be approximated by a normal distribution, there may be some other distribution better suited for this scenario.

A problem with listing the loops by their average coverages during the Markov-Chain Monte Carlo process is that these coverages do not directly correspond to how well a loop or a certain combination of loops fit the data. In general it seems that loops to which lots of other listed loops are similar to (meaning they have overlapping contigs) are at a big disadvantage in comparison to loops containing exclusive contigs, even though the existence of overlapping loops says little about how good of a fit a particular loop is. An alternative method that did not have this problem is described in Appendix C. It is not too much of a problem, since the aim is not necessarily to find the perfect combination of loops, but rather to find which contigs occur together in the same plasmids.

A Markov-Chain Monte Carlo process contains more information which could be useful than just the averages and the deviations of the variables. There may be some better way to extract an estimator of how good of a fit the loops are from this process. Furthermore, the constant C used as a factor in the weights of all contig coverages could have been implemented as a variable in the Markov-Chain Monte Carlo process. This would not only eliminate the need to arbitrarily pick some value for C, but its distribution during the process may also contain useful information about the accuracy of the data.
There turned out to be a large number of contigs classified as likely plasmids by mlplasmids which were not part of any of the supplied long-read plasmid solutions. In the graph E7237, the existence of such contigs caused the algorithm to find lots of loops that were not solutions or parts of solutions. A better understanding of where such contigs came from, would be necessary to make the algorithm work for larger graphs.

Data of which contigs contain replication genes could be useful, since we know each plasmid must always contain at least one of these. Even though some plasmids contain multiple replication gene contigs, and different plasmids in a genome may have the same replication genes, it would provide a decent estimate of how many different plasmids one can find. This can be used to choose the value of p in the geometric distibution we used as a prior. The fact that each valid plasmid must contain at least one replication gene contig, can itself also be used to trim the list of valid loops even further before running the Markov-Chain Monte Carlo algorithm. One problem with this approach is that when some plasmid contains a repeated contig, our algorithm intends to find the two loops this plasmid consists of. Since these are not both full plasmids themselves, one of these loops may not contain a replication gene contig. A workaround would require a different way of handling plasmids that are not simple loops.

Due to the removal of short contigs from the graph, the eventual outputs are only going to contain loops with all short contigs omitted. This makes the outputs incompatible with most other De Bruijn Graph handling programs such as Bandage, which require all contigs to be specified properly. Despite this, the information is still useful, since they still show which large contigs are likely to appear together. Path reconstruction can

be done by additional algorithm which finds the short contigs that best connect the long contigs per long contig solution.

At one point it was decided that there would be no loops containing the same different complements of the same contig. Apart from the fact that none of the plasmid solutions obtained from long reads have this, we cannot say whether such loops are non existent of not. The current setup of finding all combinations of contigs which form loops involves running Johnson's algorithm and then trimming the list of circular paths by removing those which have the same or complementary contigs in them in different orders. This process took some time for the largest graph. Recall that at one point a list of over $3 * 10^7$ loops was reduced to a list of 710 loops. This could be made a lot more efficient by checking for similar loops during Johnson's algorithm itself.

As mentioned earlier, for two out of the four De Bruijn Graphs we considered the supplied plasmid solutions were not even actual circular paths in the graphs. De Bruijn Graphs not containing the real plasmid as a circular path due to errors may be a common thing in practice, especially when dealing with plasmids consisting of large numbers of contigs. This severely hinders the general applicability of all methods considered here, since they rely so heavily on all solutions being fully circular paths.

Despite all the shortcomings of the current method, the fact that in some cases it was able to properly define some plasmids shows the potential of the presented approach. It is important to emphasize that all data used for testing was data taken from real graphs in contrast to other methods uniquely tested with artificial data which show poor performance when using graphs derived from real short-read sequencing data. With the right improvements, this method could be quite useful. Elements of this method could be used in combination with other existing methods to improve them.

A De Bruijn Graphs

A.1 The genome and short reads

In this section we will provide a more formal definition of De Bruijn Graphs and their properties from a mathematical perspective. A genome consists of one chromosome and any non-negative integer number of plasmids, each of which can be characterized by two complementary circular strings consisting of one of the four nucleotides A, C, T and G.

Definition A.1.1. All strings, both circular and non-circular will be denoted some vector $\mathbf{p} \in \{A, C, T, G\}^{L(\mathbf{p})}$ where $L(\mathbf{p})$ is the length of string $\mathbf{p}$.

The set of all strings is given by

$$\mathcal{S} := \bigcup_{n \geq 1} \{A, C, T, G\}^n \tag{8}$$

If a string $\mathbf{p} \in \mathcal{S}$ is in fact circular, we will use the notion that $p_{j+L(\mathbf{p})} = p_j$ for all j. Physically a circular string has no predetermined starting point and direction. So a string represented by $\mathbf{p} \in \mathcal{S}$ is physically identical to the same string with a different starting point $(p_{1+j}, p_{2+j} \ldots, p_{L(\mathbf{p})+j})$ and the same string in a different direction $p_{L(\mathbf{p})+j}, p_{L(\mathbf{p})-1+j}, \ldots, p_{1+j})$ for all $j \geq 0$.

Definition A.1.2. $\mathcal{G} \subseteq \mathcal{S}$ is the set of all representations of the chromosome and all plasmids in the genome, which are all circular strings.

For instance, if the genome consists of one chromosome which can be represented by the circular string 'ATCG', then $\mathcal{G}$ must contain 'ATCG', 'TCGA', 'CGAT' and 'GATC' as well as 'GCTA', 'CTAG', 'TAGC' and 'AGCT'. A genome cannot contain infinitely many plasmids, so $\mathcal{G}$ is a finite set.

Definition A.1.3. Let $\mathbf{p}$ be a finite or infinite (circular) string of nucleotides and $\mathbf{q}$ be a finite string of nucleotides. We say that $\mathbf{q}$ is a substring of $\mathbf{p}$, denoted by $\mathbf{q} \subseteq \mathbf{p}$, if there is an index j such that $(p_j, p_{j+1}, \ldots, p_{j+L(\mathbf{q})-1}) = (q_1, q_2, \ldots, q_{L(\mathbf{q})})$

Definition A.1.4. Let k be any natural number lower than the length $L(\mathbf{p})$ of the shortest string $\mathbf{p} \in \mathcal{G}$. A k-mer is a non-circular string of length k.

The set K of short-reads consists of all k-mers that are substrings of some chromosome or plasmid in the genome.

$$K := \{\mathbf{r} \in \mathcal{S} | L(\mathbf{r}) = k, \exists \mathbf{p} \in \mathcal{G}, \mathbf{r} \subseteq \mathbf{p}\} \tag{9}$$

Our goal is to reconstruct $\mathcal{G}$ from this set. Any candidate for $\mathcal{G}$ must have the property that all of its k-mers appear in K and that any k-mer in K can be retrieved from this $\mathcal{G}$. Unfortunately it is in most cases impossible to uniquely determine $\mathcal{G}$ from K. We can assume that none of the plasmids or chromosomes can be determined from this set with full certainty. If it was possible to unambiguously reconstruct some $\mathbf{p}$ from this set, we could remove it from the genome $\mathcal{G}$, then find K again and repeat this process until K does not contain any plasmids or chromosome with full certainty anymore.

In this definition it is assumed that every possible k-mer substring of the genome was present in the set of short-reads. This generally does not hold in practice since not all shot-reads have the same length. We can still assume every nucleotide in the genome to be covered by at least one of these short reads. It is now possible to choose some value of k such that the set of all k-mer substrings of the short reads correspond to all k-mers that are direct substrings of the genome.

A.2 Contigs and the De Bruijn Graph

The set Q of all strings of length k or more that one can reconstruct from the short read data is given by

$$Q := \{\mathbf{q} \in \mathcal{S} | L(\mathbf{q}) \geq k, (\mathbf{r} \subseteq \mathbf{q} \text{ and } L(\mathbf{r}) = k) \Rightarrow \mathbf{r} \in K\} \tag{10}$$

Lemma A.2.1. *If* $\mathbf{p} \in \mathcal{G}$ *then all substrings of* $\mathbf{p}$ *are elements of* Q

Proof. Let $\mathbf{p} \in \mathcal{G}$ and $\mathbf{q} \in S$ such that $\mathbf{q} \subseteq \mathbf{p}$. For any k-mer $\mathbf{r} \subseteq \mathbf{q}$ with $L(\mathbf{r}) = k$ we now have that $\mathbf{r} \subseteq \mathbf{p}$. This gives $\mathbf{r} \in K$ and thus by the definition of Q, we find that $\mathbf{q} \in Q$ □

Next we wish to single out all segments within the genome that can be determined uniquely and unambiguously from the short read data overlaps, meaning any string that has its contents uniquely characterized by the k-mer data K given any substring of length k of this string. Let U be the set containing such strings.

$$U := \{\mathbf{u} \in Q | \, (\mathbf{v} \in Q, \ L(\mathbf{v}) = L(\mathbf{u}) \text{ and } \exists j, \ (u_j, \ldots, u_{j+k-1}) = (v_j, \ldots, v_{j+k-1})) \Rightarrow \ \mathbf{u} = \mathbf{v}\} \tag{11}$$

Lemma A.2.2. *No string in* U *may contain the same k-mer twice or more at different indices.*

Proof. Suppose there exists some string $\mathbf{u} \in U$ such that $(u_j, \ldots, u_{j+k-1}) = (u_{j'}, \ldots, u_{j'+k-1})$ for some indices $j \neq j'$. Suppose $\mathbf{p}$ is some circular string such that $\mathbf{u} \subseteq \mathbf{p}$. We can choose the starting point of $\mathbf{p}$ such that $(p_1, \ldots, p_k) = (u_j, \ldots, u_{j+k-1})$. The properties of U give that $(p_1, \ldots, p_{j'-j+k-1}) = (u_j, \ldots, u_{j'+k-1})$ One can now easily prove that $(p_{n(j'-j)+1}, \ldots, p_{(n+1)(j'-j)}) = (u_j, \ldots, u_{j'-1})$ using induction over $n \geq 0$. This unambiguously characterizes an entire circular string, which contradicts the assumption that no plasmids or chromosomes could be determined from the k-mers with full certainty. This proves the claim. □

Corollary A.2.3. *There is a maximum length for all strings in* U*.*

Proof. Since every element of a string can only be one out of four different nucleotides, there are only 4^k different possible k-mers. A string with a length of $4^k + k$ or more would have to contain over 4^k different k-mers, which cannot be according to lemma A.2.2. This proves the claim □

Lemma A.2.4. *if* $\mathbf{u} \in U$*, then any substring of* $\mathbf{u}$ *longer than* k *is an element of* U*.*

Proof. Suppose $\mathbf{u} \in U$, $\mathbf{w} \subseteq \mathbf{u}$ and $L(\mathbf{w}) \geq k$. Let $\mathbf{v} \in Q$ be such that $L(\mathbf{v}) = L(\mathbf{w})$ and $(w_j, \ldots, w_{j+k-1}) = (v_j, \ldots, v_{j+k-1})$ for some index j. There must also exist some index j' for which $(u_{j'}, \ldots, u_{j'+k-1}) = (w_j, \ldots, w_{j+k-1})$. One can certainly find some extension $\mathbf{q} \in Q$ of $\mathbf{v}$ of the same size as $\mathbf{u}$ containing the k-mer at the same index. In other words, there exists $\mathbf{q} \in Q$ such that $\mathbf{v} \subseteq \mathbf{q}$ and $(u_{j'}, \ldots, u_{j'+k-1}) = (q_{j'}, \ldots, q_{j'+k-1})$. Since $\mathbf{u} \in U$, this gives $\mathbf{q} = \mathbf{u}$. We now have $\mathbf{w} \subseteq \mathbf{u}$, $\mathbf{v} \subseteq \mathbf{u}$ and $(w_j, \ldots, w_{j+k-1}) = (v_j, \ldots, v_{j+k-1})$, which gives $\mathbf{w} = \mathbf{v}$, since $\mathbf{u}$ may not contain the same k-mer twice starting at different indices according to theorem A.2.2. Segment $\mathbf{w}$ now meets the requirements to be in U. □

Definition A.2.5. Contigs are the largest strings that can be determined uniquely and unambiguously from the k-mer data. So every contig has the property that it contains all unambiguous strings it has a k-mer overlap with.

Let C be the set of all contigs.

$$C := \{\mathbf{a} \in U | \, (\mathbf{u} \in U \text{ and } \exists \mathbf{r} \in K, \ \mathbf{r} \subseteq \mathbf{a}, \ \mathbf{r} \subseteq \mathbf{u}) \Rightarrow \ \mathbf{u} \subseteq \mathbf{a}\} \tag{12}$$

Theorem A.2.6. *no two different contigs can have any k-mer overlaps*

Proof. Let $\mathbf{a}, \mathbf{b} \in C$ such that there exists some $\mathbf{r} \in K$ for which $\mathbf{r} \subseteq \mathbf{a}$ and $\mathbf{r} \subseteq \mathbf{b}$. The contig definition now gives that $\mathbf{a} \subseteq \mathbf{b}$ and $\mathbf{b} \subseteq \mathbf{a}$. This implies that $\mathbf{a} = \mathbf{b}$ which proves the statement by contraposition. □

Theorem A.2.7. *Any string in* U *is a substring of some contig.*

Proof. Let $\mathbf{u} \in U$ and assume $\mathbf{u} \notin C$. There now exists some $\mathbf{v} \in U$ which has a k-mer overlap with $\mathbf{u}$ for which $\mathbf{v} \not\subseteq \mathbf{u}$. Write $(v_{j'}, \ldots, v_{j'+k-1}) = (u_j, \ldots, u_{j+k-1})$. Assume that $j' \leq j$ and $L(\mathbf{v}) - j' \leq L(\mathbf{u}) - j$. Then we can take the segment $(u_{j-j'+1}, \ldots, u_{L(\mathbf{v})+j-j'}) \in U$ which has length $L(\mathbf{v})$ and shares the k-mer $u_{j'}, \ldots, u_{j+k-1})$ with $\mathbf{v}$. This gives that $\mathbf{v} = (u_{j-j'+1}, \ldots, u_{L(\mathbf{v})+j-j'})$ which contradicts $\mathbf{v} \not\subseteq \mathbf{u}$. Therefore we must either have $j' > j$ or $L(\mathbf{v}) - j' > L(\mathbf{u}) - j$. Assume without loss of generality that $j' > j$. The combined string $\mathbf{w} = (v_0, \ldots, v_{j'-1}, u_j, \ldots, u_{j+k-1})$ is strictly longer than $\mathbf{u}$ and contains both $\mathbf{v}$ and $\mathbf{u}$ as

substrings.

Let $\mathbf{a} \in U$ be such that $L(\mathbf{a}) = L(\mathbf{w})$ and there is some index l for which $(a_l, \dots, a_{l+k-1}) = (w_l, \dots, w_{l+k-1})$. This k-mer is a substring of at least one of $\mathbf{u}$ and $\mathbf{v}$. Assume without loss of generality that $(a_l, \dots, a_{l+k-1}) \subseteq \mathbf{v}$. Since $\mathbf{v} \in U$ the first section of $\mathbf{a}$ must correspond to $\mathbf{v}$ as in $(a_1, \dots, a_{L(\mathbf{v})}) = (v_1, \dots, v_{L(\mathbf{v})}) = (w_1, \dots, w_{L(\mathbf{v})})$. Since the overlap between $\mathbf{v}$ and $\mathbf{u}$ is contained in this section as well, $\mathbf{a}$ now also has a k-mer overlap with $\mathbf{u}$, namely $(a_{j'}, \dots, a_{j'+k-1}) = (u_j, \dots, u_{j+k-1})$. This lets us characterize the rest, $(a_{j'}, \dots, a_{L(\mathbf{w})}) = (u_j, \dots, u_{L(\mathbf{u})}) = (w_{j'}, \dots, w_{L(\mathbf{w})})$, hence $\mathbf{a} = \mathbf{w}$. We conclude that $\mathbf{w} \in U$.

Let $\mathbf{u} \in U$ and assume there is no $\mathbf{c} \in C$ such that $\mathbf{u} \subseteq \mathbf{c}$. Using the previous result, we can find some $\mathbf{v_1} \in U$ which is strictly longer than $\mathbf{u}$ for which $\mathbf{u} \subseteq \mathbf{v_1}$. There can also be no $\mathbf{c} \in C$ such that $\mathbf{v_1} \subseteq \mathbf{c}$. One can inductively find an infinite sequence $(\mathbf{u}, \mathbf{v_1}, \mathbf{v_2}, \dots)$ in U of strictly increasing length. This contradicts the assertion from corollary A.2.3 that there is a maximum length for all strings in U, thus for every $\mathbf{u} \in U$ there must be some $\mathbf{c} \in C$ such that $\mathbf{u} \subseteq \mathbf{c}$ □

Corollary A.2.8. *The contigs C partition the set of k-mers K, by grouping k-mers together if they are substrings of the same contig.*

Proof. Note that $K \subseteq Q$. According to theorem A.2.7 every k-mer is contained in at least one contig and according to theorem A.2.6 every k-mer is contained in at most one contig, so every k-mer is contained in exactly one contig. This allows us to partition K by which contigs they are contained in.

□

Definition A.2.9. For each pair of contigs $\mathbf{a}$ and $\mathbf{b}$ we say that $\mathbf{a}$ is connected to $\mathbf{b}$ if there is a string $\mathbf{q} \in Q$ such that $\mathbf{a} \subseteq \mathbf{q}$, $\mathbf{b} \subseteq \mathbf{q}$ and $\mathbf{a}$ comes before $\mathbf{b}$ and there exists no other contig $\mathbf{c} \in C$, $\mathbf{a}, \mathbf{b} \neq \mathbf{c}$ such that $\mathbf{c} \subseteq \mathbf{q}$.

Definition A.2.10. By regarding all contigs as nodes and all connections as directed edges, one obtains a directed graph, called a De Bruijn Graph.

We will now prove that each chromosome or plasmid corresponds to a unique circular path in this graph, such that the contents of the contigs on the path appear in the chromosome or plasmid in order.

Consider some genome $\mathcal{G}$ and some plasmid or chromosome with a given direction $\mathbf{p} \in \mathcal{G}$.

We can uniquely split up the plasmid or chromosome $\mathbf{p}$ into a sequence of contigs $(\mathbf{c_1}, \dots, \mathbf{c_t})$ as follows. Start with the contig $\mathbf{c_1}$ that contains $(p_1, \dots, p_k)$. Here we are using corollary **??**. For each $j = 2, \dots, L(\mathbf{p})$, if the k-mer $(p_j, \dots, p_{j+k-1})$ is contained in a different contig than the previous k-mer $(p_{j-1}, \dots, p_{j+k-2})$ add the contig containing $(p_j, \dots, p_{j+k-1})$ to the sequence. If a k-mer, or some larger section is repeated without any k-mers belonging to another contig inbetween, the contig containing this k-mer is repeated in the sequence the amount of times this k-mer is repeated.

Theorem A.2.11. *A sequence of contigs obtained using this method is a circular path in the De Bruijn Graph.*

Proof. Any two contigs in order $\mathbf{c_i}$ and $\mathbf{c_{i+1}}$ must contain k-mers $(p_{j_i}, \dots, p_{j_i+k-1})$ and $(p_{j_{i+1}}, \dots, p_{j_{i+1}+k-1})$ for some j_i and j_{i+1} respectively. Let $\mathbf{q} \subseteq \mathbf{p}$ be the shortest substring containing $\mathbf{c_i}$, $\mathbf{c_{i+1}}$ and $(p_{j_i}, \dots, p_{j_{i+1}+k-1})$. Lemma A.2.1 gives $\mathbf{q} \in Q$. Suppose a third contig $\mathbf{a}$ had a k-mer overlap with this $\mathbf{q}$. Since both $\mathbf{c_i}$ and $\mathbf{c_{i+1}}$ have k-mer overlaps with $(p_{j_i}, \dots, p_{j_{i+1}+k-1})$ and $\mathbf{a}$ cannot have k-mer overlaps with $\mathbf{c_i}$ or $\mathbf{c_{i+1}}$, $\mathbf{a}$ must have a k-mer overlap with $(p_{j_i}, \dots, p_{j_{i+1}+k-1})$, which violates the definition of sequence $(\mathbf{c_1}, \dots, \mathbf{c_t})$. There can therefore be no third contig $\mathbf{a}$ which has a k-mer overlap with string $\mathbf{q} \in Q$, hence $\mathbf{c_i}$ and $\mathbf{c_{i+1}}$ are connected. Using the circularity of $\mathbf{p}$ one can also prove analogously that $\mathbf{c_t}$ and $\mathbf{c_1}$ are connected. We conclude that the sequence of contigs $(\mathbf{c_1}, \dots, \mathbf{c_t})$ is present in the De Bruijn Graph as a circular path. □

B Markov-Chain Monte Carlo

In our method the measured data was given by a vector $\mathbf{x}$ and each state corresponded to a vector $\mathbf{r}$, where each state had a posterior probability $p(\mathbf{x}|\mathbf{r})$. Consider a Markov-Chain of random variables Z_i which can return states of the form $\mathbf{r}$. Every iteration, given some state $\mathbf{r}$, the process would consider some slightly altered state $\mathbf{r}^*$ with such probability that the next iteration, state $\mathbf{r}$ would be considered with the same probability, we denote this by $P(\mathbf{r} \leftrightarrow \mathbf{r}^*) = P(\mathbf{r}^* \leftrightarrow \mathbf{r})$. Now we introduce the extra check where one accepts an alteration from $\mathbf{r}$ to $\mathbf{r}*$ with probability 1 if $p(\mathbf{x}|\mathbf{r}^*) > p(\mathbf{x}|\mathbf{r})$ and probability $p(\mathbf{x}|\mathbf{r}^*)/p(\mathbf{x}|\mathbf{r})$ if $p(\mathbf{x}|\mathbf{r}^*) < p(\mathbf{x}|\mathbf{r})$. Thus the Markov-Chain Z_i is characterized by

$$P_{\mathbf{r},\mathbf{r}^*} = P(Z_{i+1} = \mathbf{r}^*|Z_i = \mathbf{r}) = \begin{cases} P(\mathbf{r} \leftrightarrow \mathbf{r}^*), & \text{if } p(\mathbf{x}|\mathbf{r}^*) \geq p(\mathbf{x}|\mathbf{r}) \\ \frac{p(\mathbf{x}|\mathbf{r}^*)}{p(\mathbf{x}|\mathbf{r})} P(\mathbf{r} \leftrightarrow \mathbf{r}^*), & \text{otherwise} \end{cases}. \tag{13}$$

The process of altering states that we use has the property that each state can be accessed from any other state in a certain number of iterations with finite probability. In other words, Z_i is an irreducible Markov Chain. Note that while we do not know the exact posterior probabilities, we do have access to unnormalized posterior probabilities. This is not a problem, since the definition of our Markov-Chain only used ratios. We find that the probabilities $\pi_{\mathbf{r}} = p(\mathbf{x}|\mathbf{r})$ are normalized and satisfy the time-reversibility equations namely that for any two states $\mathbf{r}$ and $\mathbf{r}^*$

$$\pi_{\mathbf{r}} P_{\mathbf{r},\mathbf{r}^*} = \pi_{\mathbf{r}^*} P_{\mathbf{r}^*,\mathbf{r}} \tag{14}$$

Because without loss of generality, if $p(\mathbf{x}|\mathbf{r}^*) \geq p(\mathbf{x}|\mathbf{r})$, then

$$\pi_{\mathbf{r}} P_{\mathbf{r},\mathbf{r}^*} = p(\mathbf{x}|\mathbf{r}) P(\mathbf{r} \leftrightarrow \mathbf{r}^*) = p(\mathbf{x}|\mathbf{r}^*) \frac{p(\mathbf{x}|\mathbf{r})}{p(\mathbf{x}|\mathbf{r}*)} P(\mathbf{r}^* \leftrightarrow \mathbf{r}) = \pi_{\mathbf{r}^*} P_{\mathbf{r}^*,\mathbf{r}} \tag{15}$$

We now know that $\pi_{\mathbf{r}} = p(\mathbf{x}|\mathbf{r})$ must correspond to the limiting probabilities too. Given any initial state $\mathbf{r}_0$ the limit of random variables Z_i generated by our algorithm will follow the posterior distribution.

$$\lim_{i \to \infty} P(Z_i = \mathbf{r}|Z_0 = \mathbf{r}_0) = p(\mathbf{x}|\mathbf{r}) \tag{16}$$

Hence the values $\mathbf{r}$ attains during the process will be distributed approximately according to the posterior probabilities given enough iterations. In order to find which value of $\mathbf{r}$ maximizes the posterior probability we take the average over the values of $\mathbf{r}$ during the process. While such an average would indeed correspond to the maximum in some cases (for instance if the posterior probabilities were normally distributed) or be a decent approximation in others, there is no guarantee that such a mean would indeed be a good maximum for the posterior probability. As mentioned in the discussion, the distribution obtained from this process contains much more information than we are currently using and this should be considered in the future.

C Alternative methods

Instead of using a geometric prior, at one point we tried to limit the amount of different loops present at a time by simply testing different combinations of loops and listing the optimal likelihoods. First we get a list of all loops that can be plasmids using the conventional methods, so the loops listed in table 3 in the case of graph E8867. Then we list all different combinations of a number of these loops equal to the expected amount of plasmids, so two in this case. This list now consists of $\frac{6!}{4!2!}$ entries ('63,61' , '80,31,52,55'), ('63,61' , '52,61'), etc. For each of these entries we run the Markov-Chain Monte Carlo algorithm limited just these entries for a relatively short number of iterations and calculate the average likelihood. For graph E8867 this method found the pair of loops ('63,61,52,55,80,31', '47,80') to be the best fit by far, which is the true solution.

A big advantage of this method is that loops do not get into the way of each other during the process. The likelihood of the fit directly corresponds to how well a specific combination of loops fits with the data. This cannot be said for the method we ended up using, where the outputs are average coverages r of loops during the process. A disadvantage of the alternative method is that it only lists the optimal combinations of a specific amount of loops. The number of different combinations one can make also gets very high for larger lists of loops.

References

[1] A. Orlek, N. Stoesser, M. F. Anjum, M. Doumith, M. J. Ellington, T. Peto, D. Crook, N. Woodford, A. S. Walker, H. Phan, et al., *Plasmid classification in an era of whole-genome sequencing: Application in studies of antibiotic resistance epidemiology* (2017), retrieved from https://www.frontiersin.org.

[2] M. Achtman and Z. Zhou, *Distinct genealogies for plasmids and chromosome* (2014), retrieved from http://journals.plos.org.

[3] S. Koren and A. M. Phillipy, *One chromosome, one contig: complete microbial genomes from long-read sequencing and assembly.* (2015), retrieved from https://www.ncbi.nlm.nih.gov.

[4] P. E. C. Compeau, P. A. Pevzner, and G. Tesler, *How to apply de bruijn graphs to genome assembly.* (2011), retrieved from https://www.ncbi.nlm.nih.gov.

[5] R. Rozov, A. Kav, D. Bogumil, N. Shterzer, E. Halperin, I. Mizrahi, and R. Shamir, *Recycler: an algorithm for detecting plasmids from de novo assembly graphs* (2016), retrieved from https://www.ncbi.nlm.gov.

[6] D. Antipov, N. Hartwick, M. Shen, M. Raiko, A. Lapidus, and P. Pevzner, *plasmidspades: assembling plasmids from whole genome sequencing data.* (2016), retrieved from https://www.ncbi.nlm.gov.

[7] S. Arredondo-Alonso, R. J. Willems, W. van Schaik, and A. C. Schurch, *On the (im)possibility of reconstructing plasmids from wholegenome short-read sequencing data* (2017), retrieved from https://www.ncbi.nlm.nih.gov.

[8] P. Adams, H. Adamsson, and G. E. Macklemore, The name of the journal **4**, 201 (1993).

[9] E. Birmele, R. Ferreira, R. Grossi, A. Marino, N. Pisanti, R. R., and G. S., *Optimal listing of cycles and st-paths in undirected graphs* (2013), retrieved from https://pdfs.semanticscholar.org.