**M221 Project 1**

**Harold Z. Wang**

**Introduction**

The lectures have covered extensively general concept of de Brujin graph-based genome assembly. Nevertheless, few words have been spared on the limitations of this approach in the face of non-ideal conditions, specifically short read lengths and repeats. Specific toy examples of problems caused by these two conditions have been shown, but important quantitative questions remain as to what degree the assembly approach can tolerate them. This project seeks to partially address these questions via simulations.

**Methods**

**Simulations**

All coding work is done in Python. The genome itself is simulated as a 1000 length random string of ACGT. The reads are simply the k-mer composition of the simulated genome.

We start the de Bruijn graph assembly by constructing edges from each read via its k-1 mers. We identify the starting and ending node by identifying nodes where the in degree does not match its out degree. To generate repeats, we generate the fixed length sequences, then randomly insert them into the genome for a fixed number of times at random positions. Note that a small possibility exists that a repeat sequence might be inserted into an existing repeat sequence, however we decided not to exclude this behavior, since feel that our repeat behavior mirrors those of mobile element insertions in real life.

**De Bruijn graph-based assembly via random walk**

To start our de Brujin graph algorithm, we first add one edge from the end node to the starting node to make a plus one graph, which consists of one big Eulerian cycle. Then, we proceed with the algorithm to find a Eulerian cycle. We implement the random walk as follows: at each branching node where there are multiple outgoing edges, our algorithm randomly chooses one with equal probability. At the end of its run, we delete the artificially inserted edge, and output the sequence corresponding to the resulting Eulerian path. We then compare the assembled sequence with the original sequence and check if they match each other exactly. Note that if for some reason the de Bruijn graph starts out as a Eulerian cycle, then we pick a random start point and do not add the artificial edge. In this case, although we might in the end recover a shifted Eulerian path that correspond to the original sequence, due to our inability to identify the start and end node, the chance of us reconstructing the original sequence is very low.

**BEST algorithm implementation**

The BEST algorithm was implemented onto the adjacency matrix derived from the plus one graph. We multiply the matrix by minus one, then replace the diagonals with the in-degrees. The BEST theorem states that the number of Eulerian cycles is the product of the i-th cofactor and the n minus one factorials for all diagonal elements. To handle cases where the product of the factorials becomes too big, as in the case of short read lengths, we implemented a cutoff. When the product of the determinant and the largest factorial exceeds 10,000, our implementation will return 10,000 as the result.

**Evaluations**

The failure mode of the de Bruijn graph-based assembly stems from multiple unique Eulerian paths generating multiple unique sequences. Given the random nature of our algorithm, we use a sampling approach to assess performance. For each parameter set, we repeat our entire analysis 50 times starting with the genome sequence generation. At each iteration we apply the de Bruijn graph-based assembly algorithm and the BEST theorem to the same sequence. This is to preserve comparability between the two approaches, as there is no randomness in the number of Eulerian paths in a given graph.

For de Bruijn graph-based assembly, we use the average percentage success rate of the algorithm to return the original sequence (defined as ) as an indicator of performance. For BEST theorem, we use the average of the inverse of the number of Eulerian cycles as indicator (defined as ). The intuition is that this number would be the success rate of a random walk algorithm in recovering the correct Eulerian cycle. We expect that would be consistently lower than , due to the fact that multiple Eulerian cycles/paths can still yield the same sequence.

**Results**

**Varying read length with no repeats**

We see from figure 1a that the behavior of the graph-based assembly displays a sigmoid threshold behavior. For read length lower than 8 letters, the chance of a successful assembly is essentially zero, while for read length greater than 11 the chance became one.

**Varying read length with repeats**

From figure 1b, we see the behavior of the assembler follows the same general behavior with repeats added, although the threshold has shifted considerably from around 10 to around 24. We note that 24 is just above the length of the repeat sequence. This suggests that the critical criteria of whether a sequence with repeats can be assembled is whether reads are long enough to span repeats. The mechanism might be that the longer the read, the more reads can span a single repeat sequence and the less the repeat interferes with assembly.

**Varying repeat length with read length fixed**

We also approach the relationship between read length and repeat length by varying repeat length, as in figure 1c. A similar behavior is observed, with the success rate of the assembler at one for repeat length smaller than 15, and going rapidly to zero success rate if repeat length is greater than 18. Based on results here and in the previous section, we can say in general the read length needs to be around 5 letters longer than repeat length for successful assembly.

**Varying repeat frequency**

A markedly different behavior seems to govern the relationship between repeat frequency and assembly success rate, as unlike the previous three results, figure 1d displays a roughly linear relationship. Significant noise is also present, suggesting a large amount of instability as repeat frequency becomes large.

**Number of Euclidean paths**

In figure 2, we overlay our results from the BEST theorem implementation with our results from the de Bruijn graph assembly. If every Euclidean path results in a unique sequence, then we would expect the two results to overlap each other perfectly. We observe that the behavior of the two results closely follow one another, however the percentage success rate of the assembly algorithm stayed consistently higher than the result from the BEST theorem for every condition, which aligns with our expectations that the number of Euclidean paths are slightly higher than the number of unique output sequences.

**Discussion**

We have successfully implemented a de Bruijn graph-based assembly algorithm and assessed its behavior in the presence of repeats in genome sequence. We recognize that we are still dealing with more or less an idealized situation, but several interesting observations could be made.

We recognize that for our genome sequence length of one thousand we have a minimum read length of about 11 for assembly to be generally successful. An interesting question could be asked here: what is the general relationship between genome length and read length, and how might that relationship change with different numbers of letters available?

We observe a direct relationship between read length and repeat length. We note that in the two cases we observe, the read length needs to be around 5 letters longer than repeat length for the assembly to be reliable. We hypothesize that this is related to the number of reads that span the entire repeat, as well as the genome sequences that can be unambiguously assigned to a particular repeat, which may affect the capability of the assembler to assign unique positions for repeat sequences.

Interpreting our results for varying repeat frequency is more complicated. We propose two potential explanations. First, the success rate might be linearly related to the total length of the repeats, or equivalently, the ratio of the repeat length vs genome sequence length. Secondly, it may be related to fragmentation of the genome sequence. Further work would be needed to determine the merits of each hypothesis.

Figure 1 de Bruijn graph simulation results. The success percentage is found by dividing the number of trials where the original sequence was recovered over 50 (the number of trials).

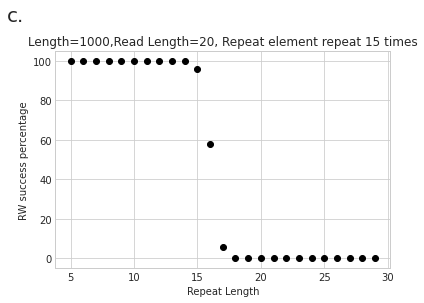
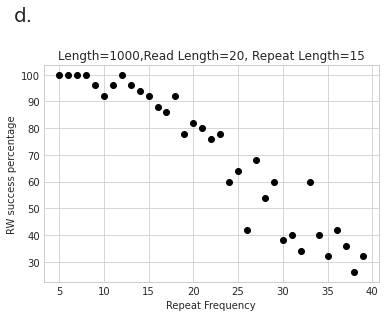
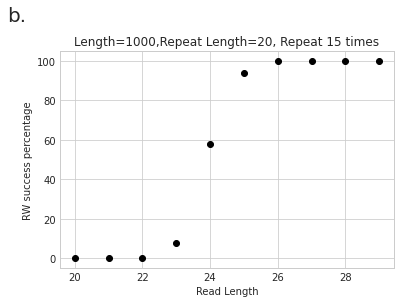
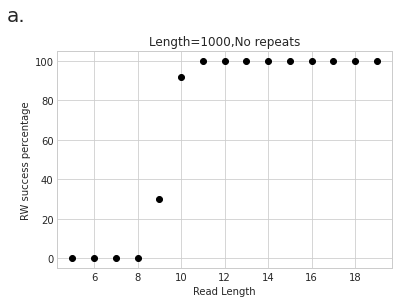


Figure 2 BEST theorem results (red circles) overlaid on de Bruijn graph simulation results. Each data point corresponds to the average of the inverse of the number of Euclidean paths over 50 trials, times 100. The 50 trials are the same as those in figure 1.

