Supplementary Materials

1 Second chance assembly

In low-coverage regions it may be necessary to use “Second Chance Assembly” to complement the standard EULER-USR assembly pipeline. The “Second Chance Assembly” attempts to assemble reads discarded during error correction along with 'hooks': edges from the assembly on fixed reads that they may be assembled with. The motivation is that fragmentation may be caused by a relatively small number of reads that are discarded during error correction, and if they overlap the ends of contigs they may be brought into the original assembly to bridge gaps.

In real sequencing projects, read coverage varies greatly and is typically dependent on sequence features. For example, a 45 nt inverted palindrome was correctly read only 3 times in the entire set of 50 nt long reads with 500X coverage (in the human BAC dataset). In some applications (e.g., single cell sequencing), the variation in coverage is even more extreme. The variations in read coverage affect the k-mer coverage $Coverage_k(i)$ defined as the number of reads covering the position i in the genome and fully containing the k-mer starting at this position. We define a *k-mer gap* as a longest sequence of consecutive genomic positions with $Coverage_k(i) < m$, where m is the multiplicity threshold. The fluctuations in read coverage create k-mer gaps and each such gap fragments the repeat graph, as shown in Figure 1. It is possible to reduce the fragmentation of the repeat graph using a lower k-mer multiplicity threshold during error correction, or further by simply constructing the de Bruijn graph on all available reads, as error correction filters out reads that cannot be fixed. On the dataset of 50 nt long reads (50X coverage) lowering the multiplicity threshold from 5 to 2, and further to 1, decreases the number of gaps in the genome coverage from 27 to 15, and to 9 correspondingly. Lowering the threshold increases the size of the de Bruijn graph, due to inclusion of a number of erroneous reads, resulting in possible miss-assemblies. Therefore, we choose the k-mer multiplicity threshold conservatively as a lower bound. The threshold of 5, used in error correcting the dataset of 50 nt long reads, filters 8.1% of original reads resulting in a high quality set of reads that covers the majority of the genome (99.6%). This allows us to drastically reduce memory constraints relative to no thereshold, and operate on a more simple graph; the graph on reads without a k-mer multiplicity threshold has 320,261 vertices and 338,836 edges versus 10,060 vertices and 11,464 edges in the de Bruijn graph on

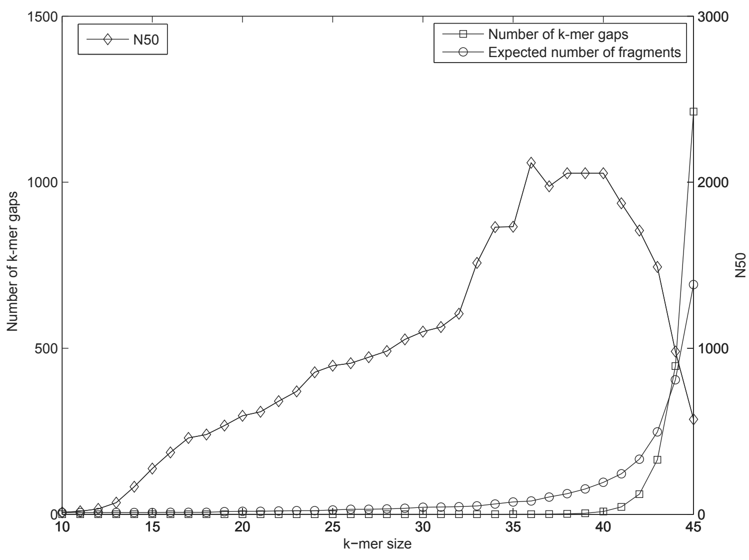

Figure 1: The choice of k-mer size affects the number of k-mer gaps (left Y-axis) in the the genome (human BAC). An increase in the k-mer size increases both the number of k-mer gaps and the number of connected components in OPTIMAL-ASSEMBLY(k) (negative effect). On the other hand, it increases the N50 size of OPTIMAL-ASSEMBLY(k) (positive effect). The N50 values are presented on the right Y-axis. For the Human BAC the optimal k-mer size is under 40 bases.

error corrected data. Table 1 shows the number of gaps in the genome coverage formed by reads on different stages of our algorithm.

We examined the tradeoff between the k-mer size and repeat graph fragmentation by constructing the de Bruijn graph on k-mers with k ranging from 10 to 45 nt using perfect reads. The results are shown in Figure 1. The number of gaps grows similarly to that expected by the Poisson distribution of fragments if coverage is uniform, however it grows slightly more quickly indicating sample bias at the resolution of less than the size of a read.

2 EULER-USR runtime

The EULER-USR is given as a set of t reads of length n each of the overall length $N = t \cdot n$. Error correcting reads and the construction of the de Bruijn/repeat graph takes $O(N \cdot \log N)$ time [1]. Threading requires a depth first search starting at the edge on which the read begins. While the total number of edges searched per read may be high, it is usually small for typical repeat graphs with small vertex degrees (like most repeat graphs). Therefore, the overall running

Read length	Coverage threshold	Original reads	SA corrected reads	Threaded and Second Chance reads
BAC35	5	27	69	41
BAC50	5	27	59	56
simBAC100	5	14	17	14

Table 1: The number of k-mer gaps at different stages of the EULER-USR assembly ($k = 20$). The increase in the number of k-mer gaps after the error correction is mainly caused by discarding reads.

time is typically $O(N \cdot \log N)$.

The Velvet assembly ran considerably faster than EULER-USR: 7 minutes compared to 48 minutes for ECOLI dataset (on a desktop PC). Since the data required roughly a week to generate, we do not anticipate assembly time to be a bottleneck in bacterial sequencing projects.

3 Detailed description of how mate-paths are used to resolve repeats

Once mate-pairs have been transformed into mate-reads, we use the resulting *mate-paths* to resolve the repeats and simplify the repeat graph (compare with the Eulerian Superpath Problem discussed in [2]). Mate-paths that begin and end on edges with sequences unique to the genome may be used to resolve repeats. To detect if an edge is (likely) unique, first consider a repeat graph $G = (V, E)$ constructed on a genome. We will use paths in the repeat graph defined by reads (read-paths) and mate-pairs (mate-paths) to label edges as unique or not (read-paths and mate-paths correspond to superpaths in [2]). The motivation for determining which edges are unique are from solutions to the *Chinese Postman Path* (CPP) discussed in [1]. A CPP P^{CP} is a path of minimal length that visits every edge at least once. Edges that are visited multiple times on the CPP are not unique. The path P^{CP} may be replaced by an Eulerian path if edges are replaced by multiedges with a count equal to the number of times they are traversed in P^{CP} [2]. Let $\mathcal{P} = (P_1, \dots P_n)$ be a set of arbitrary subpaths from P^{CP}, but for the sake of simplicity, assume that no path in $\mathcal{P}$ is fully contained by any other path in $\mathcal{P}$. If an edge e is represented by a sequence that is unique in the genome, then only one path in $\mathcal{P}$ may begin with e, and only one path from $\mathcal{P}$ may end with e. When more than one path from $\mathcal{P}$ begins with (ends with) e, there is more than one sequence in the genome that begins with (ends with) e, therefore e is not unique. The converse is not necessarily true; it is possible that only one path from $\mathcal{P}$ starts and ends at e because $\mathcal{P}$ is a set of arbitrary subpaths from P^{CP}. We denote an edge as unique if the above condition holds given a set of paths $\mathcal{P}$.

In assembly projects, the P^{CP} is not known and the goal is to detect it using mate-pair information. Mate-paths that start and end on unique edges must be part of P^{CP}. To detect unique edges using reads, the set of sub-paths from the repeat graph are defined by read-paths $\mathcal{P}^R$ and mate-paths $\mathcal{P}^M$. A path that begins and ends on unique edges is a *resolving path.* To find resolving paths, let $\mathcal{P} = \mathcal{P}^R \cup \mathcal{P}^M$. Again, for simplicity, assume no path in $\mathcal{P}$ is fully contained by any other path in $\mathcal{P}$ (if not, $\mathcal{P}$ is further processed to remove such paths). For each edge e, $\mathit{start}(e)$ is the the number of paths starting at e, and $\mathit{end}(e)$ is the number of paths ending at e. When a path P from $\mathcal{P}$ begins at an edge e_s with $\mathit{start}(e_s) = 1$ and ends on an edge e_e with $\mathit{end}(e_e) = 1$, P is a resolving path, and may be used to link e_s to e_e in the assembly.

The repeat graph is transformed using resolving paths to route paths out of repeat tangles (similar to the *equivalent transformations* in [2]). Given a resolving path P begining on an edge (s_{start}, d_{start}), and ending on edge (s_{end}, d_{end}). The edge (s_{start}, d_{start}) is replaced by an edge (s_{start}, d_{end}), and edges (s_{start}, d_{start}) and (s_{end}, d_{end}) are removed from the graph. The sequence of (s_{start}, d_{end}) is set to the sequence of the path (e_{start}, P, e_{end}), and all reads from mate-pairs that supported P are mapped to the new edge. Each edge contains a list of reads that map to it in the assembly. Reads that supported path (e_{start}, P, e_{end}) are removed from edges along P, and re-mapped to the new edge (s_{start}, d_{end}). When no reads map to an edge, the edge is removed from the graph.

There are two difficulties when labeling edges as unique: an edge corresponding to a duplicated sequence is marked as unique (false positive), and an otherwise unique edge may be considered duplicated (false negative). The false positive labels result from missing paths due to a lack of coverage, i.e. no end of a mate-pair is sampled from a duplicated sequence. When coverage is high, we found that the rate of misclassifications is very low when considering edges of a minimal length 200 for *E. coli.* The second misclassification arises when paths are added erroneously to the graph due to reads that contain sequencing errors. Although error correction is performed both prior to assembly, and in graph correction operations, errors typically remain in repeat regions. We found that the set of paths $\mathcal{P}$ contains very few erroneous edges, although only using $\mathcal{P}$ results in more false positive edges. As a result, we resolve tangles first using $\mathcal{P}^R \cup \mathcal{P}^M$, then only $\mathcal{P}^M$.

4 Analysis of simBAC100 assembly

We analyzed the simBAC100 dataset to evaluate how threading improves the assembly quality 2. The Spectral Alignment error correction routine trims reads to on average 46.6 bases. However, after threading, the average read length was recovered to 94.5 bases. The longer read length allowed us to perform repeat resolution with a larger k-mer size (50), resulting in a further 18% improvement in N50 assembly as compared to simBAC35. There is a large disparity between the assembly on perfect reads of length 100, and the assembly on threaded reads, even though the average threaded read length is over 90 nt. Most of the repeats in the genome are due to repetitive elements longer than 100 bases.

When the repeats diverge from the consensus sequence they may be resolved by perfect reads. However, we thread reads through the repeat graph, a simplified representation of the genome that merges repeats into a consensus. When the repeat consensus is longer than the average read length, small mutations that differentiate repeat copies are lost, and therefore are not resolved by reads only covering a portion of the repeat.

No. reads	Total	Reads spanning one edge	Reads spanning two edges	Reads spanning > 2 edges	Average read length (after threading)
correct/correct	2819	2523	115	181	97
correct/incorrect	5	2	3	0	100
incorrect/corrected	323589	272290	16480	34819	96
incorrect/incorrect	4116	717	888	2511	91

Table 2: The results of read threading on simBAC100 dataset. Reads are classified into four categories: correct/correct (if threading does not change a correct read), correct/incorrect (if threading turns a correct read into incorrect), incorrect/correct (if threading turns an incorrect read into correct), and incorrect/incorrect (if threading turns an incorrect read into an incorrect read). The table classifies reads in each of these four categories depending on how many edges in the repeat graph they span.

5 Detailed comparison of EULER-USR and Velvet assemblies

Figure 2 illustrates the results of both Velvet and EULER-USR by mapping contigs to the *E.coli* genome.

References

[1] Chaisson MJ, and Pevzner PA. Short Read Fragment Assembly of Bacterial Genomes. *Genome Research*, 18:324–330, 2008.

[2] Pevzner PA, Tang H. Fragment assembly with double-barreled data. *Bioinformatics*, S225–233, 2001.

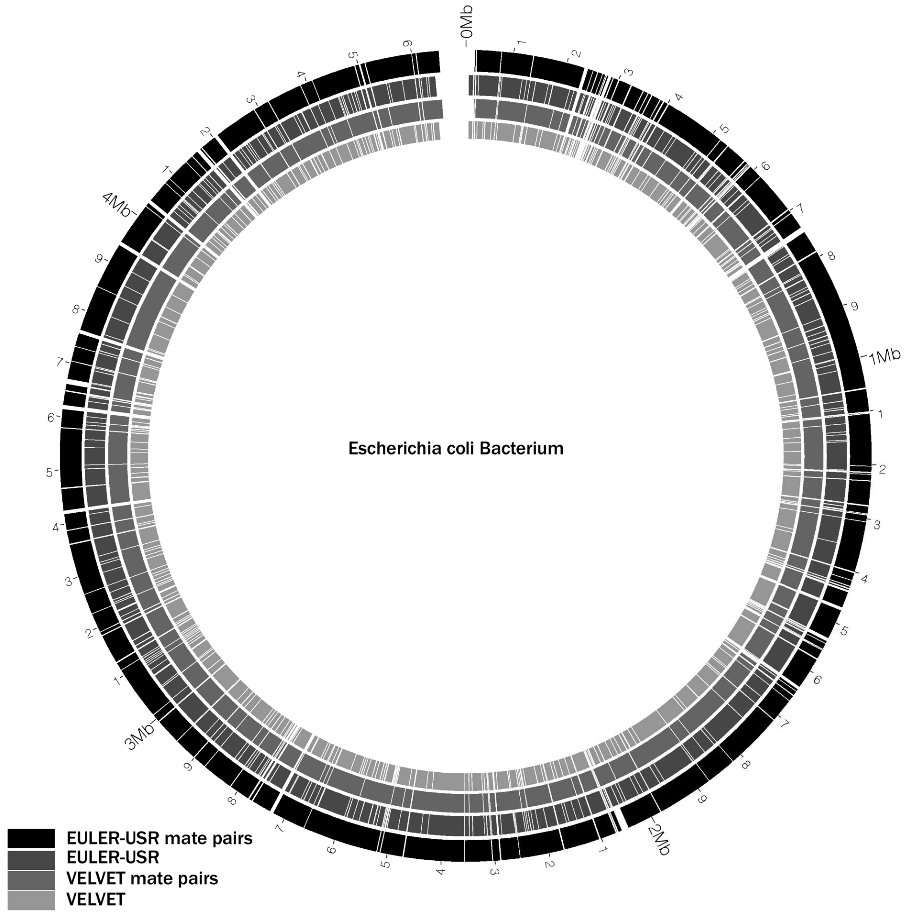

Figure 2: Comparison of contigs generated by EULER-USR and Velvet. The gap at the top shows where the reference genome was linearized.