



## De novo fragment assembly with short mate-paired reads: Does the read length matter?

Mark JP Chaisson, Dumitru Brinza and Pavel A Pevzner

*Genome Res.* published online December 3, 2008

Access the most recent version at doi:[10.1101/gr.079053.108](https://doi.org/10.1101/gr.079053.108)

---

<b>Supplemental Material</b>	<a href="http://genome.cshlp.org/content/suppl/2009/01/13/gr.079053.108.DC1.html">http://genome.cshlp.org/content/suppl/2009/01/13/gr.079053.108.DC1.html</a>
<b>References</b>	Article cited in: <a href="http://genome.cshlp.org/content/early/2008/12/03/gr.079053.108#related-urls">http://genome.cshlp.org/content/early/2008/12/03/gr.079053.108#related-urls</a>
<b>P&lt;P</b>	Published online December 3, 2008 in advance of the print journal.
<b>Accepted Preprint</b>	Peer-reviewed and accepted for publication but not copyedited or typeset; preprint is likely to differ from the final, published version.
<b>Email alerting service</b>	Receive free email alerts when new articles cite this article - sign up in the box at the top right corner of the article or <a href="#">click here</a>

---

---

Advance online articles have been peer reviewed and accepted for publication but have not yet appeared in the paper journal (edited, typeset versions may be posted when available prior to final publication). Advance online articles are citable and establish publication priority; they are indexed by PubMed from initial publication. Citations to Advance online articles must include the digital object identifier (DOIs) and date of initial publication.

---

To subscribe to *Genome Research* go to:  
<http://genome.cshlp.org/subscriptions>

---

# *De novo* Fragment Assembly with Short Mate-Paired Reads: Does the Read Length Matter?

Mark J. Chaisson <sup>\*</sup>    Dumitru Brinza <sup>†</sup>    Pavel A. Pevzner <sup>† ‡</sup>

October 22, 2008

## Abstract

Increasing read length is currently viewed as the crucial condition for fragment assembly with next-generation sequencing technologies. However, introducing mate-paired reads (separated by a gap of length *GapLength*) opens a possibility to transform short mate-pairs into long *mate-reads* of length  $\approx$  *GapLength* and thus raises the question whether the read length (as opposed to *GapLength*) even matters. We describe a new tool, EULER-USR, for assembling mate-paired short reads and use it to analyze the question whether the read length matters. We further complement the ongoing *experimental* efforts to maximize read length by a new *computational* approach for increasing the effective read length. While the common practice is to trim the error-prone tails of the reads, we present an approach that substitutes trimming with error correction using repeat graphs. An important and counterintuitive implication of this result is that one may extend sequencing reactions that degrade with length “past their prime” to where the error rate grows above what is normally acceptable for fragment assembly.

## 1 INTRODUCTION

The field of high-throughput sequencing has grown recently in both applications and computational support. This is enabled by the many platforms that exist for high-throughput sequencing, including those produced by 454 Life Sciences [18], Illumina 1G Genome Analysis System ([www.illumina.com](http://www.illumina.com)), Applied Biosystems SOLiD Sequencing ([www.appliedbiosystems.com](http://www.appliedbiosystems.com)), and Helicos GSS Sequencing ([www.helicosbio.com](http://www.helicosbio.com)). Although the 454 sequencing platform is now producing reads that are of similar length to Sanger reads, the underlying paradigm is that a higher throughput may be achieved at the sacrifice of read length. The technologies with the highest throughput currently available produce short, 20-40 base reads, distinguished as *ultrashort* reads.

Many recent successful applications of ultrashort reads have used the reference genomes for whole genome re-sequencing [12], chromatin remodeling mapping [27] and whole-genome methylation [1] profiling. An analysis in [31] showed that the number of reads uniquely mapped to the human genome grows with the increase in reads length but reaches a plateau after the first  $\approx$  40 nucleotides. This implies that there is little incentive to increase the read length in re-sequencing applications since it provides little return on investment. While generating 40 nt reads is usually sufficient for

---

<sup>\*</sup>Bioinformatics Program, University of California San Diego, La Jolla, CA 92093, USA. MC 0412

<sup>†</sup>Department of Computer Science and Engineering, University of California San Diego, La Jolla, CA 92093, USA.

<sup>‡</sup>To whom correspondence should be addressed.

resequencing, *de novo* fragment assembly may require longer reads. The Illumina platform can easily generate longer reads, but the error rates deteriorate after the first 35 nt making the ends of reads not suitable for fragment assembly. This again provides little incentive to generate longer reads. By extending the effective length of reads, our EULER-USR assembler generates more reads that span the repeats and thus improves the assemblies.

Most *de novo* assemblers for Sanger reads follow the “overlap-layout-consensus” paradigm that is optimized for such reads [13, 15] and does not scale well for short read assembly. In contrast, most approaches to short read assembly including EULER-SR [3], Velvet [32], and ALLPATHS [2] use an alternative Eulerian approach that model the assembly problem as a search for an Eulerian path in a de Bruijn graph [23].

The advantage of the Eulerian approach is that it generates a theoretically optimal assembly of reads of length  $k$  (in high-coverage projects) by essentially mimicking fragment assembly as Sequencing by Hybridization (SBH) problem on a virtual DNA array with all  $k$ -mers [25]. Idury and Waterman, 1995 [14] demonstrated how the Eulerian approach for SBH can be applied to fragment assembly of Sanger reads. However, the Eulerian approach works best for error-free reads and quickly deteriorates as soon as the reads have even small number of basecalling errors. To alleviate this limitation, two different *error-correction* approaches are used: error correction in reads prior to assembly [23, 2], and *post-hoc* graph corrections that remove spurious edges from the assembled de Bruijn or A-Bruijn graphs [22, 32].

Correcting errors in reads prior to assembly was shown to be useful for both Sanger and 454 Life Sciences reads [23, 3] (error rates reduced forty-fold from 1.2% to 0.03% for Sanger reads). However, this approach essentially transforms already rather accurate reads ( $\approx 1\%$  error rate) into nearly error-free reads. The efficiency of the existing error correction approaches quickly deteriorates with even a small increase in the error rates of original reads (e.g., from 1% to 3%). For example, an optimized error correction tool from Tammi et al., 2003 [28] was able to reduce the error rates from 3.4% to 0.3% on simulated data, only a tenfold decrease. While it looks like a reasonably low error rate, it makes it nearly impossible to apply the Eulerian approach that does not tolerate even such seemingly small error rates. This makes the full-length Illumina reads “ineligible” for Eulerian assembly (Figure 1 illustrates that the error rate is  $\approx 20\%$  at the ends of reads) and sets an accuracy bottleneck for developing new sequencing technologies.

In this paper we focus on the Illumina technology and describe how to increase the “usable” length of error-prone Illumina reads while keeping them nearly error free. Although this paper limits the benchmarking of EULER-USR to the Illumina technology, our algorithm is applicable to any technology characterized by high error rates, such as the Helicos platform. While the average error rate in Illumina reads is under 2% for the first  $\approx 30$  nucleotides, it quickly increases in the tails of the reads reaching  $\approx 20\%$  at position 50. No existing short read assembly tool can efficiently deal with such high error rates and the conventional wisdom is that the read tails become unusable when the error rate exceeds 3%. Below we introduce an alternative error correction approach that uses the de Bruijn graph constructed on the accurate read prefixes in order to correct the error-prone read suffixes by fitting them into the de Bruijn graph. Since EULER-USR can assemble error-prone reads, we hope that it can catalyze developments of sequencing platforms aimed at generating longer but less accurate sequencing reads.

Reads may be combined with mate-paired information to further improve the quality of assemblies and the next generation sequencing companies are actively pursuing both increasing the read length and effectively generating mate-pairs. For example, Illumina recently increased the effective read length from 35 to 50 nucleotides and announced the plans to provide capabilities for

75-100 nt reads in 2009. On the other hand, Illumina and various sequencing centers are exploring applications of jump and PET libraries [21] for generating mate-pairs in the context of short read technologies. While increasing the read length is a high priority for most next-generation sequencing companies, there exists an opinion that the read length almost does not matter if one uses mate-paired reads. Indeed, Pevzner and Tang, 2001 [24] demonstrated that most *mate-pairs*: “read<sub>start</sub> - GAP of length  $d$  - read<sub>end</sub>” can be transformed into *mate-reads* “read<sub>start</sub> - SEQUENCE of length  $d$  - read<sub>end</sub>” by filling in the gap of length  $d$  with the nucleotide sequence representing an appropriate path in the de Bruijn graph. As a result, one can generate contiguous long reads of length  $2 \cdot l + d$  (*span* of mate-pairs) from short mate-paired reads of length  $l$  making the read length almost irrelevant (typically,  $d \gg l$ ).

We show how EULER-USR utilizes mate-pairs to significantly improve assembly, and further use it to answer the question whether read length matters. We demonstrate that the answer to this question is closely linked with the *efficiency* of transforming mate-pairs into mate-reads (the percentage of mate-pairs successfully transformed into mate-reads). When the read length exceeds a certain threshold, the *read length barrier*, the efficiency reaches nearly 100%, so that the read length indeed does not matter. For example, for the *E. coli* genome, the read length barrier is  $\approx 35$  nt.<sup>1</sup> This is good news for technologies with reads already longer than 35 nt (e.g., Illumina reads) but bad news for technologies with shorter reads (e.g., Helicos and SOLiD reads). However, while the current parameters of Illumina reads may be already sufficient for reliable assembly of some bacterial genomes, they are not sufficient for slightly larger genomes like *Saccharomyces cerevisiae* with higher read length barrier. This observation reveals a synergy between EULER-USR error-correction approach to increasing the read length and EULER-USR approach to transforming mate-pairs into mate-reads. Indeed, while the length of the “usable” portions of Illumina reads is currently below the read length barrier for yeast (Figure 1), EULER-USR error correction allows one to increase the effective read-length beyond the read length barrier. Therefore, while the mate-paired information represents by far the most important factor for improving the assembly quality, the read length also provides a valuable contribution to the assembly.

The EULER-USR software for assembling mate-paired short reads is publicly available at <http://euler-assembler.ucsd.edu>.

## 2 Methods

We have developed the EULER-USR algorithm for assembling mate-paired and error-prone ultrashort reads. In addition to the previous Eulerian approaches [23] that correct reads based on  $k$ -mer multiplicities, EULER-USR corrects reads based on how they map to *repeat graphs* [23, 3], a generalization of the de Bruijn graphs.

The de Bruijn graph of a genome is constructed on the set of all  $k$ -mers in the genome. Vertices  $u$  and  $v$  are connected by a directed edge  $(u, v)$  if there is a  $(k + 1)$ -mer in the genome that has the  $k$ -mer  $u$  as a prefix and the  $k$ -mer  $v$  as a suffix. A small example of a de Bruijn graph is shown in Figures 2a,b. As a result, every substring of length  $> k$  in the genome maps to a unique path in its de Bruijn graph. Similar to the de Bruijn graph of a genome (Figure 2c), one can construct the de Bruijn graph of reads (Figure 2d) on the set of all  $k$ -mers present in reads. The

<sup>1</sup>We emphasize that the read length barrier depends on the genome, the span of mate-pairs, coverage, error-rates in reads, variability in gap length, etc. The read length barrier  $\approx 35nt$  for *E.coli* was computed under the assumption that the span is  $300 \pm 30$  nt.

de Bruijn graphs of real genomes are very complex, a reflection of a large number of repeats with slightly varying repeat copies. A repeat graph of a genome (or reads) is a “simplified” version of the de Bruijn graph with small bulges and whirls removed [22, 20] (Figures 2e,f). The key observation in the Eulerian assembly is that the repeat graph of a genome can be approximated by the repeat graph of reads and thus may be constructed from reads alone, i.e., without knowing the genome [22] (compare Figures 2e and 2f).

If the repeat graph of the genome is known, one may correct errors in a read by simply mapping this read to a path in the repeat graph and substituting the read by the path.<sup>2</sup> While this procedure would result in a nearly error-free set of reads, it is not clear how to construct a repeat graph from inaccurate reads and how to further map such reads to the repeat graph. In our approach, we construct the repeat graph from (accurate) read prefixes and then map entire reads (with inaccurate suffixes) to this graph by *threading*.

The EULER-USR algorithm consists of the following 3 steps that can be further supplemented by the threading procedure that we will describe later (section 2.4):

- Detecting accurate read prefixes and correcting errors within them using frequent  $k$ -mers. This operation generates the set of extremely accurate (nearly error-free) read prefixes.
- Constructing the repeat graph on error corrected prefixes using  $k$ -mers.
- Simplifying the repeat graph after transforming mate-pairs into mate-reads.

## 2.1 Detecting and error-correcting accurate read-prefixes

Although the quality of Illumina reads deteriorates with length, the prefixes are quite accurate (less than 2% error rate). While many reads can be turned into error-free reads by our error correction algorithm, errors will remain in low-quality reads even after error correction. In this case, it is important to detect the longest read prefix that may be error-corrected and discard the reads that cannot be corrected.

We correct reads using a modified version of the *Spectral Alignment (SA)* algorithm described in [23, 4]. The  $SA(Spectrum, read)$  method corrects *read* given a set of  $k$ -mers *Spectrum*. Given a set of reads  $\mathcal{R}$  and a frequency threshold  $m$ , we define a spectrum  $Spectrum = Spectrum_k(\mathcal{R}, m)$  as the set of all  $k$ -mers that appear at least  $m$  times in reads from  $\mathcal{R}$ . The set of such *solid*  $k$ -mers approximates the set of all  $k$ -mers in the genome. We define a read as *error-free* if all its  $k$ -mers are solid, otherwise, we attempt to make a read error-free by mutating a few nucleotides in the read [23, 3]. We only consider substitution mutations since there are few insertions/deletions in Illumina reads.

We use a greedy heuristic to find the minimum number of mutations to make every  $k$ -mer in a read solid. It records the number of  $k$ -mers that are made solid for all 3 possible mutations at every position in the read. The mutation that makes the highest number of  $k$ -mers in the read solid is applied if it “solidifies” more than  $t$   $k$ -mers, where  $t$  is an internal threshold [3]. This heuristic is iteratively applied either until all  $k$ -mers in the read become solid, or until there are no mutations that can solidify more than  $t$   $k$ -mers. We output the prefix of the read that is solid, or discard the read if none of it is solid. This partitions all reads that are not discarded into an (accurate) *prefix* and (less accurate) *suffix* that will be corrected at a later stage. The set of reads is divided into

<sup>2</sup>This procedure may also result in *error corruption* [23].

two partitions: *fixed* and *unfixable*. The fixed partition has reads that have a solid prefix, and the unfixable partition contains reads with no solid  $k$ -mers. A similar method is used in [2].

In order to choose the multiplicity threshold  $m$ , we assume that  $k$ -mers are generated from a mixture of two models: one erroneous, and the other correct (Figure 3). If we assume positional independence of errors in reads, the multiplicity of erroneous  $k$ -mers follows a Poisson distribution, and the multiplicity of correct  $k$ -mers follows a Poisson with a large mean that may be approximated by a Gaussian. We choose  $m$  by fitting a Poisson and Gaussian mixture model to the distribution of  $k$ -mer multiplicity and finding the first local minimum of this distribution (Figure 3).

## 2.2 Constructing the repeat graph on error-corrected read prefixes

We construct the de Bruijn graph on the error-corrected read prefixes and further transform it into the repeat graph. Reads that contain errors or SNPs in the middle create short undirected cycles in the de Bruijn graph called bulges, and reads that contain errors in the end create erroneous sources or sinks. Finally, some read errors form chimeric reads by transforming the sequence in one end of the read to that of a distant part of the genome, creating an edge that erroneously connects two unrelated contigs. Transformation of the de Bruijn graph into the repeat graph amounts to removing bulges, erroneous sources/sink edges, and chimeric reads as described in [3].

In high-coverage projects, fixed reads provide sufficient coverage across the genome to create a repeat graph that encodes the entire genome. However, many sequencing projects still contain low-coverage regions (often due to sampling bias). Since the error correction step relies upon redundancy of  $k$ -mers in reads, the reads in regions of low coverage may be discarded at this step, causing fragmentation of the assembled contigs. We found that the unfixable partition often contains many reads from the low-coverage regions, and their exclusion from the assembly causes fragmentation of the repeat graph. In such cases it may be necessary to use what we call “Second Chance Assembly” – an assembly of all reads discarded during error correction (see Supplementary Material)

## 2.3 Simplifying the repeat graph by transforming mate-pairs into mate-reads

Since the initial proposal to use mate-paired reads for shotgun sequencing [30], the mate-paired reads have been considered essential to *de novo* sequencing. Although in the past the short read mate-paired data have not been available, various next-generation sequencing vendors have recently released modules for high-throughput production of mate-pairs. EULER-USR utilizes the mate-paired reads by modifying the EULER-DB approach [23] for incorporating mate-pairs into the Eulerian assembly framework.

When mate-pairs are available, the input to the fragment assembly is a set of mate-pairs: “read<sub>start</sub> - GAP of length  $d$  - read<sub>end</sub>”. The key idea of EULER-DB [24] is using the repeat graph to transform the mate-pairs “read<sub>start</sub> - GAP of length  $d$  - read<sub>end</sub>” into *mate-reads* “read<sub>start</sub> - SEQUENCE of length  $d$  - read<sub>end</sub>” by filling in the gap of length  $d$  with an appropriate path in the repeat graph. As a result, EULER-DB has an ability to generate contiguous long reads of length  $2 \cdot l + d$  from short mate-paired reads of length  $l$ . These long mate-reads are subjected to the traditional Eulerian assembly afterwards. Each mate-read corresponds to a *mate-path* between the mate-paired reads mapped to the repeat graph. In reality, the gap length is not fixed but varies from  $d - \delta$  to  $d + \delta$ .

While EULER-DB worked well for traditional Sanger sequencing [22], it needs to be modified for short reads. The key parameter of EULER-DB is the *efficiency*, the percentage of mate-pairs successfully transformed into mate-reads. This transformation is trivial if there is a single path of length  $\approx d$  between  $\text{read}_{start}$  and  $\text{read}_{end}$  in the repeat graph, as the path is simply used to fill the gap between the reads. This was indeed the case for the overwhelming majority of Sanger reads making EULER-DB rather efficient. However, the repeat graphs for short reads are often very complex and in some cases there are multiple paths of length  $\approx d$  between paired reads (Figure 4). Pevzner and Tang, 2001 [24] described this problem and proposed an iterative approach to solve it (see Figure 4a in [24]). EULER-SR [3], Velvet [32], and ALLPATHS [2] all implement the idea of filling the gap between mate-pairs using the repeat graph and apply it to simulated data (e.g., compare Figure 4b in [24] with Figure 5c in [2]). Below we apply EULER-DB to real Illumina mate-paired reads and show how to extend EULER-DB to analyze complex repeat graph characteristic for short read sequencing.

The “Breadcrumb” “All paths definition” procedures in Velvet and ALLPATHS are both aimed at filling up the gap between the mate-paired reads in the repeat graph. For most mate-pairs in *E.coli* there is only one path between mate-paired reads. In such cases the paths found by the “Breadcrumb” and “All paths definition” methods are equivalent to EULER-DB. However, the remaining mate-pairs are important for resolving complex tangles and Velvet and ALLPATHS describe different approaches to addressing this challenge. Below we describe how a simple extension of EULER-DB addresses this problem.

When there are multiple paths in the repeat graph between a mate-pair ( $\text{read}_{start}$ ,  $\text{read}_{end}$ ), we may choose a path with maximum *support* from mate-pairs. Figure 4 illustrates the situation when there are many ways to transform the mate-pair ( $\text{read}_{start}, \text{read}_{end}$ ) into a mate-read using one of the paths between them. Let edge  $e_{start}(e_{end})$  be the edge where  $\text{read}_{start}$  begins ( $\text{read}_{end}$  ends). A mate-pair supports a path  $P$  between  $e_{start}$  and  $e_{end}$  if one of the reads is in either  $e_{start}$  or  $e_{end}$ , and the other read is in an edge in  $P$ . The number of such mate-pairs for a path  $P$  is denoted  $support(e_{start}, e_{end}, P)$ . The path  $P$  with the highest value of  $support(e_{start}, e_{end}, P)$  is used to create the mate-path  $P^+ = (e_{start}, P, e_{end})$ . Note that due to gaps in coverage there may be edges in the path that are not supported. Furthermore, errors in reads may create reads that support edges not on the optimal path. Therefore, we use a path with maximal  $Support(P)$  among all paths between  $\text{read}_{start}$  and  $\text{read}_{end}$  unless  $Support(P)$  falls below  $MinSupport$  threshold. Further details of transforming mate-pairs and into mate-reads are given in the Supplementary Information.

## 2.4 Assembling error-prone reads (error-correction by threading)

Each read corresponds to a unique *read path* in the de Bruijn graph representing the sequence of the read. Since the repeat graph approximates the de Bruijn graph, a similar argument applies to the read-paths in the repeat graph. A read may be mapped to the de Bruijn graph by aligning it to a closest subpath in the graph. Since the de Bruijn graph is built on read prefixes, the path corresponding to every read prefix (*prefix-path*) is known thus facilitating the read mapping. We may find the path that the entire read maps to by searching all subpaths continuing from the known prefix-path, a process we refer to as *threading reads through a graph*.

In the case that threading is invoked, the EULER-USR( $k, m, l$ ) algorithm proceeds in five steps. The user has a choice of either specifying all three parameters:  $k, m$ , and  $l$ , or specifying a single parameter  $k$ . In the latter case, EULER-USR selects the suitable parameters  $m$  and  $l$  automatically.

- Detecting accurate read prefixes and correcting errors within them using frequent  $k$ -mers (multiplicity  $m$  and higher). This operation generates the set of extremely accurate (nearly error-free) read prefixes.
- Constructing the repeat graph on error corrected prefixes using  $k$ -mers. This operation generates the set of  $k$ -mer contigs.
- Threading entire reads through the repeat graph to extend the effective read length. This operation generates the set of accurate *threaded* reads.
- Constructing the repeat graph on threaded reads and generating  $l$ -mer contigs using  $l$ -mers ( $l > k$ ).
- Simplifying the repeat graph by transforming mate-pairs into mate-reads

Once the repeat graph has been constructed on the (accurate) read prefixes, we attempt to map every fixed read to the graph. However, while mapping of the (accurate) read prefixes is well defined, mapping of (inaccurate) read suffixes is ambiguous. EULER-USR utilizes the repeat graph to correct errors in read suffixes.<sup>3</sup>

Every read  $prefix * suffix$  not discarded by error correction is represented as a concatenation of its  $prefix$  and  $suffix$ . Since the genome is an Eulerian traversal of its repeat graph, all substrings of the genome map to paths in the repeat graph. While the accurate  $prefix$  may be uniquely mapped to a path  $path(prefix)$  in the repeat graph, it is not clear how to map the entire read  $prefix * suffix$  since  $suffix$  is inaccurate. We argue that to map  $prefix * suffix$ , one has to choose one of the extensions of  $path(prefix)$  among all paths of length  $n$  (read length) that begin with  $path(prefix)$ . We denote the set of such paths as  $\mathcal{P}$  and argue that a path in  $\mathcal{P}$  with the minimum edit distance to the read represents the “best” mapping of the read  $prefix * suffix$  to the repeat graph. While in many cases such a *thread-path*  $path(prefix * suffix)$  may be used to correct the read  $prefix * suffix$ , it has to be done with caution (see below).

If  $\mathcal{P}$  has a single edge (Figure 5a), we correct the read with the sequence on the edge. However, these reads may not be used to resolve repeats and thus are vestigial in terms of improving the assembly. If  $\mathcal{P}$  has multiple paths (Figure 5b), we rank paths  $P_1, P_2, \dots$  in  $\mathcal{P}$  in the increasing order of their Hamming distances  $dist(P_1), dist(P_2), \dots$  to the read  $prefix * suffix$ . While it is tempting to choose the “optimal” path  $P_1$  for correcting errors in the read  $prefix * suffix$  it has to be done with caution. The problem is that the sequencing errors in the inaccurate  $suffix$  may transform it into an alternative string that maps to a “wrong” path in the repeat graph. We therefore check that (i) the optimal path  $P_1$  is sufficiently similar to  $prefix * suffix$ , and (ii) the second best path  $P_2$  is sufficiently dissimilar from  $prefix * suffix$ . To check these conditions, we use a parameter  $f$ , the expected error rate in read suffixes, and classify a path  $P$  as similar to the read if  $dist(P) \leq f \cdot |suffix|$ , and dissimilar otherwise. If both conditions (i) and (ii) are satisfied we use  $P_1$  for correcting the read  $prefix * suffix$ , otherwise we iteratively trim before the position of the last difference, and re-thread the read until both these conditions are satisfied.

To obtain the final assembly that takes advantage of the longer threaded reads, we build the repeat graph on  $l$ -mers for  $l > k$  (where  $k$  was used to build the original repeat graph) so that repeats of length  $l$  and shorter are resolved. The tradeoff between the  $k$ -mer size and repeat resolution is that edges from the repeat graph  $G$  of the genome (constructed on  $k$ -mers) will be split in the repeat

<sup>3</sup>Error-corrected read suffixes only contribute to enlarging the assembled contigs and do not contribute to base-calling.

graph of reads if there is a gap longer than  $n - k$  between read start positions, where  $n$  is the read length. When  $l > k$  the repeat graph of reads constructed on  $l$ -mers will be more fragmented than the repeat of reads constructed on  $k$ -mers. We prevent fragmentation of the repeat graph of reads constructed on  $l$ -mers by creating artificial reads  $x \circ y$  for every pair of adjacent edges (contigs)  $x$  and  $y$  in the repeat graph of reads constructed on  $k$ -mers ( $k$ -mer contigs). One can either set the maximal  $l = n - 2$  or empirically chose  $l$  to minimize fragmentation of the graph.

## 3 Results

### 3.1 Datasets

We use the following datasets to benchmark EULER-USR and compare it with Velvet [32], the most accurate among the recently published short read assemblers.

- ECOLI. A set of 29.8 million paired Illumina reads from the  $\approx 4.6$  Mb *E.coli* genome (227X coverage). For benchmarking we randomly selected 10 million reads from this dataset. Mapping of reads to *E.coli* genome revealed that 87% are error-free. The separation between mate-pairs is  $145 \pm 25$  bp.
- BAC50 and BAC35. A set of Illumina reads from an  $\approx 170Kb$  human BAC generated at the Joseph Ecker lab at the Salk Institute.<sup>4</sup> This BAC was sequenced over several runs allowing us to generate an error profile that was not biased to a single run. A total of 2 million 35-50 base reads were generated for this BAC resulting in 500X coverage allowing us to choose an appropriate subset for a typical coverage benchmark. We randomly selected reads resulting in 50X coverage by 50 nt long reads. Mapping these reads from the BAC to the reference sequence revealed that 20% are error-free and 12% have only 1 error. Furthermore, 84% are error free in the first 30 bases.

While EULER-USR is designed to work with longer reads, the Velvet assembler is optimized for 35 base long reads and the performance deteriorates for longer reads. In order to make a fair comparison with Velvet we created a 35 nt read dataset BAC35 by truncating 50 nt reads from BAC50 to 35 nucleotides resulting in 35X coverage of the BAC.

- simBAC100 and simECOLI100. A set of simulated 100 base reads from the human BAC. This set was generated to check whether EULER-USR can support extending sequencing reactions well beyond their prime. We simulated reads by mapping all 2 million reads from the dataset generated at Salk Institute to the BAC, extending them up to 100 bases, and simulating random errors. We further selected a set of resulting 100 base reads so that the coverage was 200X (to ensure that results were not biased by gaps in coverage). Errors in the resulting 100 base long reads were simulated using a 1% error rate for the first 35 bases, and 20% error rate for the remaining 65 bases. This error profile leads to a challenging assembly problem without attempting to model reads characteristics for a particular technology. Our method for correcting errors using a repeat graph requires that the entire genome is covered by the high quality prefixes of reads. The simBAC100 dataset includes as many reads as the BAC50 dataset to ensure that the entire genome is represented in the repeat graph constructed on 35 bases read prefixes.

---

<sup>4</sup>This BAC has a repeat content representative of the rest of the human genome.

The simECOLI100 is the set of simulated 100 base reads from *E. coli* (100X coverage). Errors were added to the reads according to the same 1%-20% error profile used for simBAC100.

- simBAC35. A set of 35 base read prefixes from simBAC100. Because the simBAC100 dataset uses simulated reads, it is not directly comparable to the assemblies on real reads. Instead, to perform proper comparison for the effect of threading reads, we compare the assembly on 100 base (inaccurate) simulated reads to the assembly on the 35 base (accurate) simulated reads.

### 3.2 Benchmarking

We compared five recently published *de novo* short read assemblers aimed at short reads: SSAKE [29], SHARCGS [6], VCAKE [16], Edena [11], and Velvet [32]. Of all assemblers, Velvet performed the best in terms of N50 contig size.

SSAKE is designed for assembling error-free reads, and SHARCGS is designed for reads with error-rate below 0.05%. Running these two methods on Illumina reads resulted in filtering of all reads on a preprocessing step or (if preprocessing is turned off) in an inferior assembly quality. VCAKE is an improvement of SSAKE and is able to assemble reads with higher error rate. For the BAC35 dataset, VCAKE, Velvet, Edena, and EULER-USR produced similar assemblies. However, the quality of assemblies generated by VCAKE or Edena deteriorate with increasing the read length (BAC50 dataset) producing more fragmented assemblies than Velvet. Furthermore, the assemblies for the ECOLI dataset by both Velvet and EULER-USR resulted in doubling the N50 contig size as compared to VCAKE or Edena. We therefore decided to limit the detailed benchmarking to Velvet only.

A goal in the Eulerian approach is to construct the repeat graph  $G_k(Reads)$  on the set of *Reads* that best approximates the “ideal” repeat graph  $G_k(Genome)$  of the *Genome* [3] (denoted as REPEAT-GRAPH( $k$ ) in the follow-up Tables and Figures), as every Eulerian path in the repeat graph corresponds to a possible solution of the fragment assembly problem. However, due to fragmentation, the REPEAT-GRAPH statistics are misleading because the longer the contigs are, the more likely they are to be fragmented by low coverage regions (note that fragmentation of long contigs leads to a quick deterioration of the N50 size). Therefore, uneven distribution of reads over the genome may turn approximating  $G_k(Genome)$  into an unattainable goal. To setup a more realistic goal, we transform the set *Reads* into the error-free set *PerfectReads* (by substituting every read with the sequence of the genome it maps to). In the absence of mate-pairs, the repeat graph  $G_k(PerfectReads)$  constructed on this set of reads represents the best assembly our assembler aims for while assembling the real reads (referred to as OPTIMAL-ASSEMBLY in the follow-up Tables and Figures).

### 3.3 How are assemblies improved by mate-paired reads?

To benchmark EULER-USR and VELVET on the ECOLI dataset, we first evaluated assembly with unpaired reads in order to later gauge the effect of mate-pairs. Both the EULER-USR and Velvet ( $k = 27$ ) assemblies were close to the theoretically optimal assembly (Table 1 and Figure 6) with similar N50 sizes (20K for EULER-USR and 16K for Velvet) and no miss-assemblies. The contigs longer than 500 bases in EULER-USR assembly (those likely to be non-repetitive) contain 6 mismatches, 3 insertions, and 1 deletion (30 mismatches, 3 insertions, and 3 deletions in the Velvet

assembly).<sup>5</sup>

Table 1 and Figure 6 compare EULER-USR and Velvet and illustrate that mate-pairs significantly improve the assemblies. The N50 length for *E.coli* assembly increases from 16K to 45K for Velvet and from 19K to 62K for EULER-USR. EULER-USR generated 127 contigs longer than 1000 bp that is comparable to the typical number of contigs resulting from pre-finished Sanger assembly of bacterial genomes (see [22] and <http://nbc.scd.edu/euler/benchmarking/bact.html>). Only two (relatively short) contigs produced by EULER-USR were misassembled. An alignment of the assemblies to the *E. coli* genome is shown in the Supplementary Materials.

Assembly	N50	Length (# contigs) >20000 nt	Length (# contigs) >5000 nt	Length (#contigs) >1000 nt
REPEAT-GRAPH(30)	22,173	2,432,772 (69)	4,232,578 (237)	4,484,685 (331)
EULER-USR unpaired	20,096	2,233,252 (68)	4,212,353 (249)	4,490,810 (355)
Velvet unpaired	16,424	1,953,255 (59)	4,068,326 (262)	4,484,065 (416)
EULER-USR mate-pairs	62,015	4,207,753 (72)	4,481,764 (96)	4,524,074 (113)
Velvet mate pairs	45,427	3,800,552 (79)	4,419,542 (131)	4,507,932 (167)

Table 1: A comparison of assemblies of *E. coli* 35 base Illumina reads, unpaired and mate-paired. *N50*: the size of the contig such that 50% of the assembly is contained in contigs of size N50 or greater. *Length (# contigs) (>20000 nt)*, total length of all contigs longer than 20000 and total number of such contigs. We found that Velvet produces optimal results when run as Velvet(27,5).

### 3.4 How are assemblies improved by read-threading?

When estimating input parameters for EULER-USR, our mixture model suggested the multiplicity threshold  $m = 5$  for BAC50 dataset with  $k$ -mer size 20. The accuracy of error correction was evaluated by mapping error-corrected reads to the BAC (Table 2). From the original sets of reads, 91.3% of the BAC35, 88.6% of the BAC50 datasets, and 98.0% of the simBAC100 dataset were retained after error-correction based on Spectral Alignment. During graph correction, the removal of certain edges from the graph truncates the reads that map to these edges, further shortening the average read length. While most reads in this dataset were trimmed only by a few nucleotides, others become rather short and need to be extended with threading as described above.

Table 3 presents the statistics of the N50 contig size as well as the cumulative contig size for various datasets (reported for contigs longer than 1000, 500, and 100 bases). Since the N50 statistic is limited we show the differences in assemblies by plotting the cumulative length of contigs ordered by size (Figure 7). Figure 7 shows how longer threaded reads improve the assembly quality for both real and simulated reads.<sup>6</sup> Figure 7 illustrates that while VELVET and EULER-USR show similar results for the BAC35 dataset (no read threading), the EULER-USR assembly improves for BAC50 and simBAC100 datasets (due to its ability to utilize the error-prone reads) while VELVET assembly hardly changes. Indeed, even with a modest increases in read length from 35 nt to 50 nt, read threading increases N50 contig size by 13% and the total length of long contigs (longer

<sup>5</sup>This analysis underestimates the basecalling errors by limiting them to long contigs and thus avoiding the most difficult repeated regions. Nevertheless, the basecalling accuracy appears to be comparable or even better than the accuracy of high-coverage Sanger sequencing.

<sup>6</sup>In some cases the statistics for EULER-USR is slightly better than for OPTIMAL-ASSEMBLY due to subtle differences in contig reporting.

Dataset	Original reads		SA corrected reads			Threaded reads after graph correction	
	Length	Error rate(%)	Average length	Error rate(%)	Retained reads (%)	Average length	Error rate(%)
BAC35	35	0.92	34.9	0.01	91.3	34.9	0.004
BAC50	50	4.36	46.7	0.04	88.6	49.3	0.049
simBAC100	100	13.3	46.6	0.07	98.0	94.5	0.050
simECOLI100	100	12.6	50.5	0.003	99.6	98.8	0.017

Table 2: Error rate (per read base) and average length of reads on different stages of the EULER-USR threading algorithm. The error rate is computed by mapping reads to the genome. We compute the length and error rate for the original reads, reads corrected by Spectral Alignment that are retained after graph correction, and finally after threading. The increased error rate after threading is due to threading reads through their consensus sequence in the repeat graph (rather than de Bruijn graph).

than 1000 nt) by 22%. Figure 8 shows how assembly improves with increase in read coverage when assembling the *E. coli* genome and leads to a conclusion that the coverage increase beyond 55X results only in modest increase in assembly quality.

When the BAC50 dataset is assembled without threading reads, the N50 contig size is 1752 with a net assembly size of 171301. Read-threading only improves the quality of assembly by correcting reads that pass through 3 or more edges (most reads map to one or two edges in the repeat graph). Table 4 shows the number of reads that are correctly fixed with threading and how many edges they are threaded through for BAC50 dataset (see Supplementary Material for analysis of simBAC100 dataset). Although a very small fraction of reads are threaded through more than 3 edges, they improve the quality of assembly.<sup>7</sup>

Since bacterial genomes have a compact gene structure, we analyzed how many genes are captured within contigs constructed by various programs. Even though the repeat graph of *E. coli* is fragmented into many contigs (over 500 for REPEAT-GRAPH(30)), many contigs are long, and contain multiple genes. We mapped the contigs of REPEAT-GRAPH, EULER-USR, and Velvet assemblies to the genome, and counted the number of genes that contained entirely within a contig. Table 5 illustrates that contigs produced by Velvet and EULER-USR capture a large number of bacterial genes thus enabling various applications. For example, one can perform MS/MS proteomics analysis of bacterial genomes with Illumina contigs almost as efficiently as with completed genomes [9].

The *E. coli* genome contains relatively few repetitive elements compared to the human genome, and so the longer reads should be able to resolve more repeats in *E. coli* than in the human BAC. In the simECOLI100 dataset, the usable read length was increased from an average of 50.5 nt to 98.8 nt (Table 2) by threading. When we compare the assembly on the simulated read prefixes to the threaded reads, the N50 contig size more than doubles to  $\approx 45K$ . This indicates that the announced increase in Illumina read length to 100 nt (planned for 2009) will lead to significant improvement in assembly in the case of unpaired reads. While it is an important improvement in applications like single cell sequencing (where the mate-paired protocols are not available yet), it remains unclear whether the read length (as opposed to span) matters in case of *mate-paired*

<sup>7</sup>For BAC50 dataset, EULER-USR has 20 mismatches and 2 insertions, a higher error rate as compared to ECOLI dataset.

Assembly method/dataset	N50	Length (# contigs) >1000	Length (# contigs) >500	Length (#contigs) >100
<b>BAC35</b>				
REPEAT-GRAPH(25)	1869	97946 (34)	126774 (74)	153378 (194)
OPTIMAL-ASSEMBLY(25)	1609	92758 (39)	119773 (78)	153348 (225)
EULER-USR(20,2,25)	1786	89905 (39)	118576 (80)	147227 (210)
Velvet(21,5)	1428	87551 (43)	113522 (80)	138554 (173)
<b>BAC50</b>				
REPEAT-GRAPH(40)	4168	143224 (39)	160679 (64)	175246 (128)
OPTIMAL-ASSEMBLY(40)	2023	129392 (55)	152551 (87)	176491 (193)
EULER-USR(20,5,40)	2022	119489 (45)	148319 (86)	171082 (164)
Velvet(31,5)	1381	84292 (39)	112886 (78)	139176 (169)
<b>simBAC35</b>				
REPEAT-GRAPH(25)	1869	97946 (34)	126774 (74)	153378 (194)
OPTIMAL-ASSEMBLY(25)	1847	95180 (35)	159359 (85)	175305 (242)
EULER-USR(20,5,25)	1818	98187 (39)	129671 (85)	162109 (242)
Velvet(21,5)	1844	97174 (36)	122816 (73)	144578 (153)
<b>simBAC100</b>				
REPEAT-GRAPH(50)	7163	167112 (31)	172990 (39)	175444 (53)
OPTIMAL-ASSEMBLY(50)	3971	162129 (47)	169794 (58)	176908 (94)
EULER-USR(20,5,50)	2639	140718 (47)	163244 (80)	175900 (135)
Velvet(31,5)	695	36796 (22)	77557 (80)	120147 (241)
<b>simECOLI100</b>				
REPEAT-GRAPH(50)	59656	4519592 (140)	4528404 (152)	4583803 (533)
EULER-USR(20,10,50)	44710	4519083 (182)	4531837 (200)	4562551 (366)

Table 3: Assembly statistics of various datasets of Illumina reads. *N50*: the size of the contig such that 50% of the assembly is contained in contigs of size *N50* or greater. *Length (>1000)*, *Length (>500)*, and *Length (>100)*: the total length of all contigs longer than 1000, 500 nt and 100 nt, respectively. For Velvet(*k* – *mer size*, *coverage*) we found that the coverage cutoff *t*=5 maximizes the assembly quality. The effect of threading reads on assembly quality may be seen by comparing simBAC35 and simBAC100. The rows describing REPEAT-GRAPH(50) and OPTIMAL-ASSEMBLY(50) are identical since the reads cover the entire BAC in simBAC100 dataset. In all tests there was a single misassembly (in simECOLI100 dataset).

No. reads	Total	Reads spanning one edge	Reads spanning two edges	Reads spanning > 2 edges	Average read length (after threading)
correct/correct	100942	95781	2161	2550	50
correct/incorrect	207	174	27	6	50
incorrect/correct	55515	52408	1464	1643	42
incorrect/incorrect	347	254	33	60	45

Table 4: The results of read threading for BAC50 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.

Method	REPEAT-GRAPH	EULER-USR	Velvet
# complete genes	3937 (95.2%)	3956 (95.7%)	3912 (94.6%)

Table 5: Mapping of the 4136 genes from *E.coli* into theoretical repeat graph and repeat graph constructed by EULER-USR and Velvet for the ECOLI dataset (see Table 1 for a description of the parameters). We show the number of genes that are contained entirely within the constructed contigs.

reads (i.e., would increasing the length of mate-paired reads from 35 nt to 100 nt lead to significant improvements in assembly if the span remains fixed?)

### 3.5 Does the read length matter?

The availability of two methods to resolve repeats (mate-pairs and threading) brings the question whether they may be used in conjunction to further improve assemblies. To test this, we simulated mate-pairs with span  $300 \pm 30$  nt in the genomes of *E. coli* and *S. cerevisiae*. The read length  $r$  was fixed in each simulated dataset, with a minimum read length of 25 nt and maximum length 100 nt. The goal was to evaluate whether the quality of assembly (e.g.,  $N50(r)$ , N50 length for reads of length  $r$ ) with longer reads improves as compared to the quality of assembly with shorter reads (e.g., whether 100 nt mate-paired reads result in a better assembly than 35 nt reads). We define the *read length barrier* as the read length after which the quality of assembly does not significantly improve (e.g., N50 does not increase by more than 5%).

The error-free mate-paired reads were simulated starting at every genomic position and each simulated dataset with reads of length  $r$  was assembled with EULER-USR ( $k$ -mer size 24). To evaluate the quality of assemblies, we used N50 contig size and *efficiency* (the percentage of mate-pairs transformed into mate-reads). The efficiency should be analyzed with caution because only mate-pairs spanning multiple edges in the repeat graph contribute to improving the assembly (similar effect is illustrated in Table 4 for single reads). In the *E. coli* assembly of Illumina reads, this was only 3.9% of all mate-pairs. As a result, for the simulated *E. coli* assembly, the efficiency is rather high for all read lengths (varying from 97.8% for  $r = 25$  to 99.0% for  $r = 55$  and to 99.2% for  $r = 100$ ). However, even a small increase in efficiency translates into significant increase in N50 statistics (varying from  $\approx 40$ K for  $r = 25$  to  $\approx 60$ K for  $r = 55$ ). Therefore, small increases

in efficiency may reflect very significant increase in the number of “useful” mate-pairs, i.e., mate-pairs that improve the assembly. To better gauge the contribution of such “useful” mate-pairs, we introduce the *relative efficiency*, the percentage of *useful* mate-pairs transformed into mate-reads (we call a mate-pair useful if its reads reside on different edges). For the yeast dataset, the relative efficiency varies from 61% for  $r = 25$  to 80% for  $r = 55$  (the maximum value among all read lengths is 81%). These results indicate that for the yeast dataset efficiency hardly change after the read length exceeds  $\approx 60$  nt.

Our attempt to answer the question “Does the read length matter?” is limited in many aspects (e.g., reads were simulated error-free, and coverage was perfectly uniform) and it only answers the question whether the read length matters for EULER-USR assemblies (rather than for a theoretically optimal assembly with mate-pairs).<sup>8</sup> However, it reveals that for the *E.coli*, the assembly hardly improves after the read length exceeds 35 nt (efficiency=98.7%, N50 contig size  $\approx 60$ Kb). The assembly deteriorates when the read length decreases from 35 to 25 indicating that the read length barrier for *E.coli* (with the chosen simulation parameters) is  $\approx 35$  nt.<sup>9</sup>

For the *S. cerevisiae* genome, the assembly quality only slightly improves after the read length exceeds 60 nt (N50 is  $\approx 70$ K for  $r = 60$  but drops to  $\approx 62$ K at  $r = 45$  and to  $\approx 41$ K at  $r = 25$ ). It indicates that the read length barrier for *S. cerevisiae* (with chosen simulation parameters) is  $\approx 60$  nt.

## 4 Discussion

The recent addition of mate-paired reads to the arsenal of short read technologies opened a possibility of assembling complex genomes for a fraction of the cost of the traditional Sanger sequencing. We demonstrated that the Eulerian approach is well suited for assembling mate-paired short reads by transforming mate-pairs into mate-reads using repeat graph. We further complemented the approach from [24] by selecting the most “supported” mate-reads to resolve some difficult cases when a mate-pair may be transformed into multiple mate-reads.

In addition to incorporating mate-pairs into fragment assembly, we also show that the conventional wisdom of “read trimming” may be substituted by threading to correct error-prone read tails. We demonstrate that if a sequencing technique “suffers” from quality degradation along the length of a read, it may still be used effectively in *de novo* assembly. Despite the fact that short read assemblies are rather fragmented, we demonstrate that most bacterial genes map to single contigs thus enabling gene discovery and annotation of bacterial genomes.

The Eulerian approach models the error-prone suffixes of the reads as short edges to vertices of out-degree zero. All recently developed short read assemblers remove such edges from the graph (e.g., via the *erosion* procedure in [22]) thus essentially discarding information contained in the error-prone read suffixes. Therefore, even if the reads are not explicitly trimmed, they are implicitly trimmed after the de Bruijn graph is constructed (e.g., using the “clipping” procedure in Velvet [32] or “removal of hanging ends” procedure in ALLPATHS [2]). EULER-USR differs from these approaches by utilizing information in the error-prone read prefixes.

<sup>8</sup>The theoretically optimal algorithms for assembling mate-paired reads remain unknown even for error-free reads and fixed distance between between mate-pairs [19].

<sup>9</sup>We found that assemblies of mate-pairs with average span  $d \pm \sigma$  may be sensitive to the parameter  $\sigma$  even for the same  $d$ . For example, simulated assemblies with error-free reads may have lower quality than the real assemblies with the same  $d$  but different  $\sigma$ .

Our study on the use of mate-paired reads in conjunction with read threading revealed that there exist some synergy between these two approaches when the read length remains below the read length barrier. While mate-pairs represent the major factor in improving the assembly quality, read threading contributes to further improvements in assembly. The next challenge for short-read technologies is to assemble larger and more complex genomes. The ability to exploit any information possible to resolve repeats will become important when assemblers move to mammalian genomes.

## 5 Acknowledgments

This research is supported NIH grant 1R21HG004130-01 as well by NSF grant EIA-0303622. We thank Dirk Evers, Klaus Maisinger, and Jacques Retief for insightful discussions about the Illumina technology and to Xiaohua Huang and Eric Roller for many discussions on emerging next-generation sequencing technologies. We are grateful to Ronan O'Malley and Joseph Ecker for providing us with the BAC reads.

## References

- [1] Barski A, Cuddapah S, Cui K, Roh TY, Schones DE, Wei G, Chepelev I, and Zhao K. High-resolution profiling of histone methylations in the human genome. *Cell*, 129:823–837, 2007.
- [2] Butler J, MacCallum I, Kleber M, Shlyakhter IA, Belmonte MK, Lander ES, Nusbaum C, Jaffe DB. ALLPATHS: de novo assembly of whole-genome shotgun microreads *Genome Res*, 18:810–820, 2008.
- [3] Chaisson MJ, and Pevzner PA. Short Read Fragment Assembly of Bacterial Genomes. *Genome Research*, 18:324–330, 2008.
- [4] Chaisson M, Tang H, and Pevzner PA. Fragment assembly with short reads. *Bioinformatics*, 20:2067–2074, 2004.
- [5] Chiu KP, Wong CH, Chen Q, Ariyaratne P, Ooi HS, Wei CL, Sung WKK, and Ruan Y. PET-Tool: a software suite for comprehensive processing and managing of Paired-End diTag (PET) sequence data. *BMC Bioinformatics*, 7:390, 2006
- [6] Dohm JC, Lottaz C, Borodina T, and Himmelbauer H. SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. *Genome Research*, 17:1697–706, 2007.
- [7] Edwards A, Caskey T. Closure Strategies for Random DNasequencing. *Methods*, 3:41–47
- [8] Feuk L, Carson AR, and Scherer SW. Structural variation in the human genome. *Nature Reviews Genetics*, 7:85–97, 2006.
- [9] Gupta N, Tanner S, Jaitly N, Adkins JN, Lipton M, Edwards R, Romine M, Osterman A, Bafna V, Smith RD, Pevzner PA, Whole proteome analysis of post-translational modifications: applications of mass-spectrometry for proteogenomic annotation. *Genome Res*, 17:1362–77, 2007.
- [10] Harris TD, Buzby PR, Babcock H, Beer E, Bowers J, Braslavsky I, Causey M, Xie Z., and others. Single-molecule DNA sequencing of a viral genome. *Science*, 2008 320 106–109.

- [11] Hernandez D, Franois P, Farinelli L, Osteras M, and Schrenzel J. De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. *Genome Res*, 18:802–809, 2008.
- [12] Hillier LW, Marth GT, Quinlan AR, Dooling D, Fewell G, et al.. Whole-genome sequencing and variant discovery in *C. elegans*. *Nat Methods*, 5:183–188, 2008.
- [13] Huang X, Wang J, Aluru S, Yang S, and Hillier L. PCAP: A whole genome assembly program. *Genome Research*, 13:2164–2170, 2003.
- [14] Idury RM, Waterman MS. A New Algorithm for DNA Sequence Assembly. *J of Comp Biology*, 2:291–306, 1995.
- [15] Jaffe DB, Butler J, Gnerre S, Mauceli E, Lindblad-Toh K, Mesirov JP, Zody MC, and Lander ES. Whole-genome sequence assembly for mammalian genomes: Arachne 2. *Genome Research*, 13:91–96, 2003.
- [16] Jeck WR, Reinhardt JA, Baltrus DA, Hickenbotham MT, Magrini V, Mardis ER, Dangl JL, and Jones CD. Extending assembly of short DNA sequences to handle error. *Bioinformatics*, 23:2942–2944, 2007.
- [17] Kim S, Tang H, and Mardis E. Genome sequencing technology and algorithms. *Artech House Publishers*, 2007.
- [18] Margulies M, Egholm, M. Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437:326–327, 2005.
- [19] Medvedev P, Georgiou K, Myers EW, Brudno M. Computability of Models for Sequence Assembly. *WABI* 289–301, 2007.
- [20] Myers EW. The fragment assembly string graph. *Bioinformatics*, Suppl 2:79–85, 2005.
- [21] Ng P, Tan JJ, Ooi HS, Lee YL, Chiu KP, Fullwood MJ, Srinivasan KG, Perbost C, Du L, Sung WK, Wei CL, Ruan Y. Multiplex sequencing of paired-end ditags (MS-PET): a strategy for the ultra-high-throughput analysis of transcriptomes and genomes. *Nucleic Acids Research*. 34:e84, 2006.
- [22] Pevzner PA, Tang H, and Tesler G. De novo repeat classification and fragment assembly. *Genome Research*, 14:1786–1796, 2004.
- [23] Pevzner PA, Tang H, and Waterman MS. An Eulerian path approach to DNA fragment assembly. *PNAS*, 98:9748–9753, 2001.
- [24] Pevzner PA, Tang H. Fragment assembly with double-barreled data. *Bioinformatics*, S225–233, 2001.
- [25] Pevzner PA. 1-Tuple DNA sequencing: computer analysis. *J Biomol Struct Dyn*, 7:63–73, 1989.
- [26] Pop M, and Salberg SL. Bioinformatics challenges of new sequencing technology. *Trends in Genetics*. 24:133–141, 2008.

- [27] Schones DE, Zhao K. Genome-wide approaches to studying chromatin modifications. *Nat Rev Genet*, 9:179–191, 2008.
- [28] Tammi MT, Arner E, Kindlund E, and Andersson B. Correcting errors in shotgun sequences. *Nucleic Acids Research*, 31:4663–4672, 2003.
- [29] Warren RL, Sutton GG, Jones SJM, and Holt RA. Assembling millions of short DNA sequences using SSAKE. *Bioinformatics*, 23:500–501, 2007.
- [30] Weber JL, Myers EU. Human whole genome shotgun sequencing. *Genome Research*, 7:401–409, 1997.
- [31] Whiteford N, Haslam N, Weber G, Prugel-Bennett A, Essex JW, Roach PL, Bradley M, and Neylon C. An analysis of the feasibility of short read sequencing. *Nucleic Acids Research*, 33:e171, 2005.
- [32] Zerbino DR, Birney E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res*, 18:821–829, 2008.

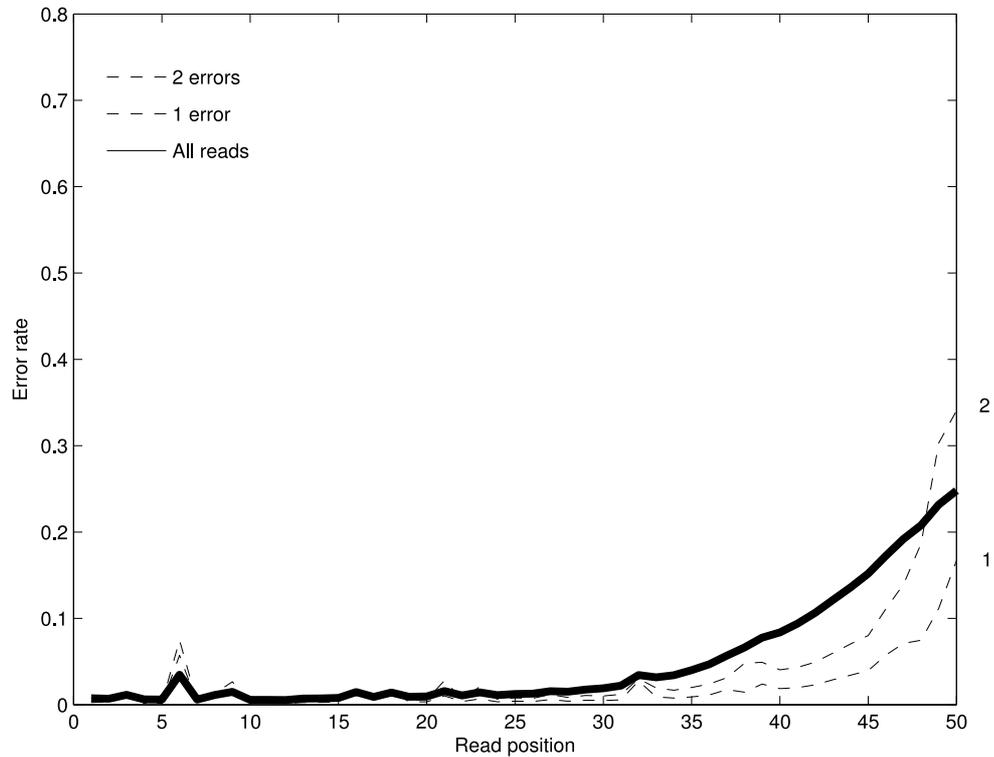


Figure 1: The positional profile of basecalling errors for Illumina reads for 2 million 50 nt long reads from a human BAC. The error rate across reads is shown (solid) along with the error rate for reads with a fixed number of errors. The erroneous nucleotides in each read are detected by mapping the read to the reference genome. The high error rate in position 6 is due to the bias in our particular data set rather than a systematic problem with the Illumina technology.

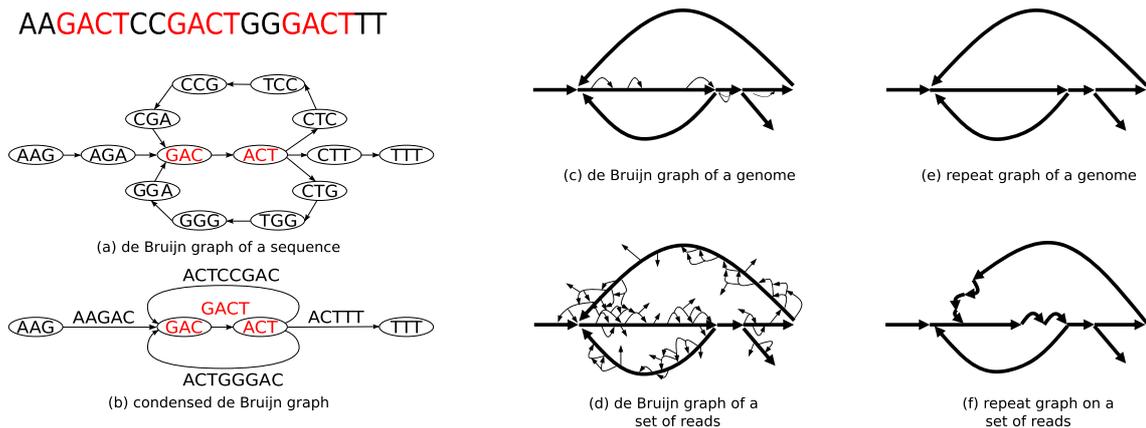


Figure 2: From de Bruijn graphs to repeat graphs. The de Bruijn graph of a sequence contains a vertex for every  $k$ -mer in the sequence, and an edge  $(u, v)$  for every pair of consecutive (overlapping)  $k$ -mers in the sequence (a). The condensed de Bruijn graph replaces all paths containing non-branching vertices by a single edge labeled by the sequence that generated the path (b). When the condensed de Bruijn graph is constructed on a genome, it contains some small bulges and whirls representing repeats with slightly varying repeat copies (c). In the repeat graph the bulges and whirls are removed (e). The de Bruijn graph of reads contains additional spurious bulges and whirls caused by sequencing errors in reads (d). The goal of the Eulerian assembly is to construct the repeat graph of reads (f) that approximates the repeat graph of the genome. Different papers use different terminology, e.g., the edges of these graphs are referred to as “blocks” in [32] and “unipaths” in [2].

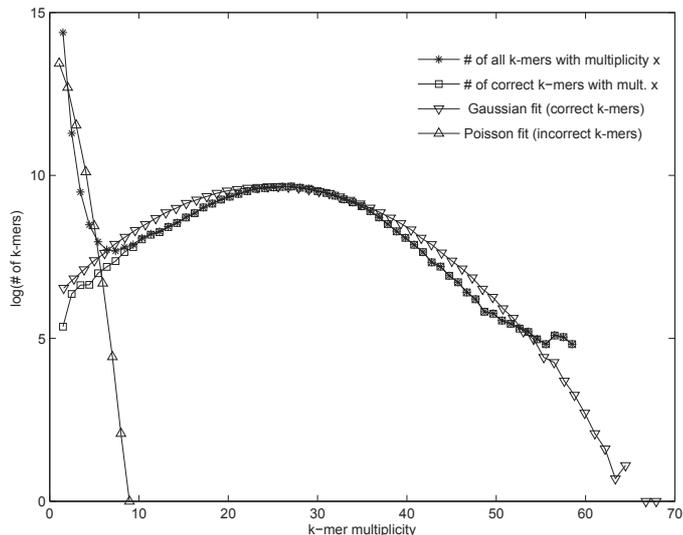


Figure 3: Choosing the multiplicity threshold for error correction. All  $k$ -mers appearing in the reads are classified as *correct* if they appear in the genome and *incorrect* otherwise. For a multiplicity  $x$ , let  $correct(x)/incorrect(x)$  be the number of correct/incorrect  $k$ -mers with multiplicity  $x$  (the plots are shown for 50 base long Illumina reads from a human BAC and  $k = 20$ ). As expected, most high-multiplicity  $k$ -mers are correct and most low-multiplicity  $k$ -mers are incorrect. A Poisson/Gaussian mixture model was fit to the distribution of all  $k$ -mer multiplicities in order to model the process of generating incorrect (Poisson) and correct  $k$ -mers (Gaussian). To show the fit of the model, the  $k$ -mer multiplicities were generated according to the estimated parameters  $\lambda=0.95$ ,  $\mu=25$ , and  $\sigma=9.38$ , with a mixing parameter  $\omega = 0.95$ . One may find the multiplicity  $m$  with good separation between correct and incorrect  $k$ -mers by estimating the first local minimum from the distribution of  $k$ -mer counts, or the minimum of the sum of the probabilities of the mixture model, a more smooth distribution. For multiplicity threshold  $m=5$ , only 0.6% of correct 20-mers have multiplicity  $<5$  and only 0.3% of incorrect 20-mers have multiplicity  $\geq 5$ .

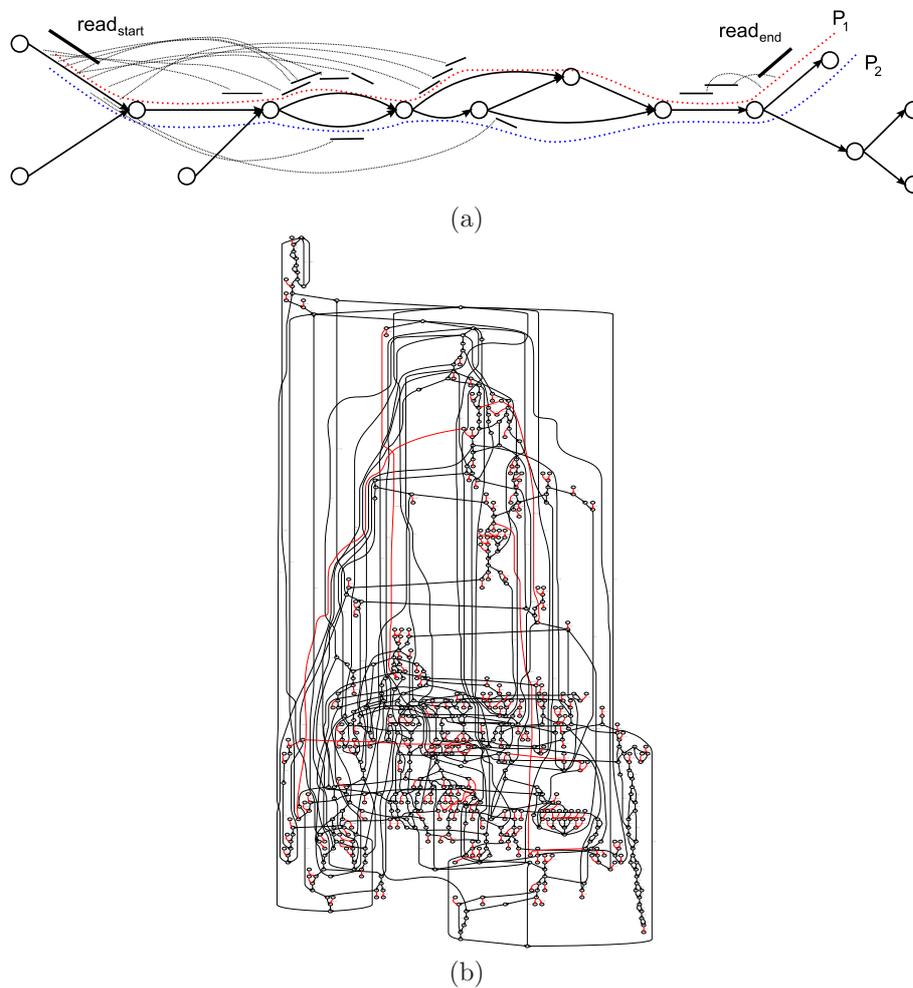


Figure 4: (a) A fragment of a made-up repeat graph formed by three divergent copies of a repeat. There are many possible paths from  $read_{start}$  to  $read_{end}$ . To transform the mate-pair “ $read_{start}$  - GAP of length  $d$  -  $read_{end}$ ” into a mate-read “ $read_{start}$  - SEQUENCE of length  $d$  -  $read_{end}$ ”, we compute the support for every path between  $read_{start}$  and  $read_{end}$  and select a path with maximum support. In this example, the “red” path  $P_1$  has greater support than the “blue” path  $P_2$ . (b) A fragment of the real repeat graph of *E. coli* (constructed from ECOLI dataset) illustrating that transformation of mate-pairs into mate-reads may fail in some cases. Red edges represent unique (typically long) contigs while black edges represent repeats.

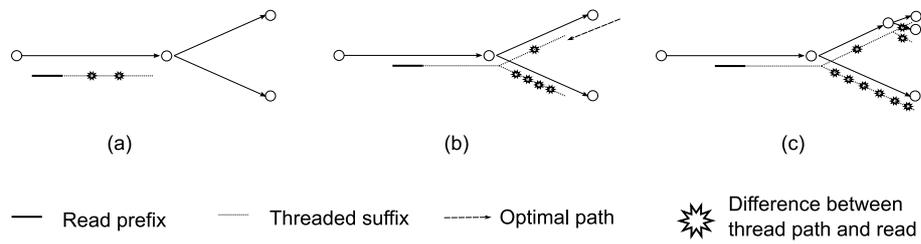


Figure 5: Mapping reads to the paths in the repeat graph. In (a) a read maps to a single edge. In (b) a read maps to two paths, and the closest one is chosen. In (c) a read may be mapped to two similar paths implying that trimming is required.

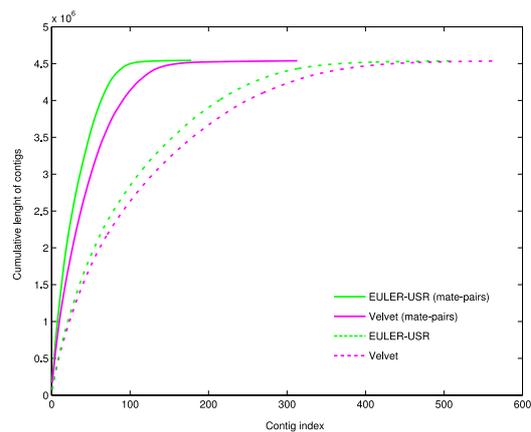


Figure 6: Comparison of EULER-USR and Velvet using both paired reads, and the same reads with mate-pair information removed (ECOLI dataset). The contigs are ordered in the decreasing order of sizes and the cumulative size of  $x$  longest contigs is shown on the y-axis.

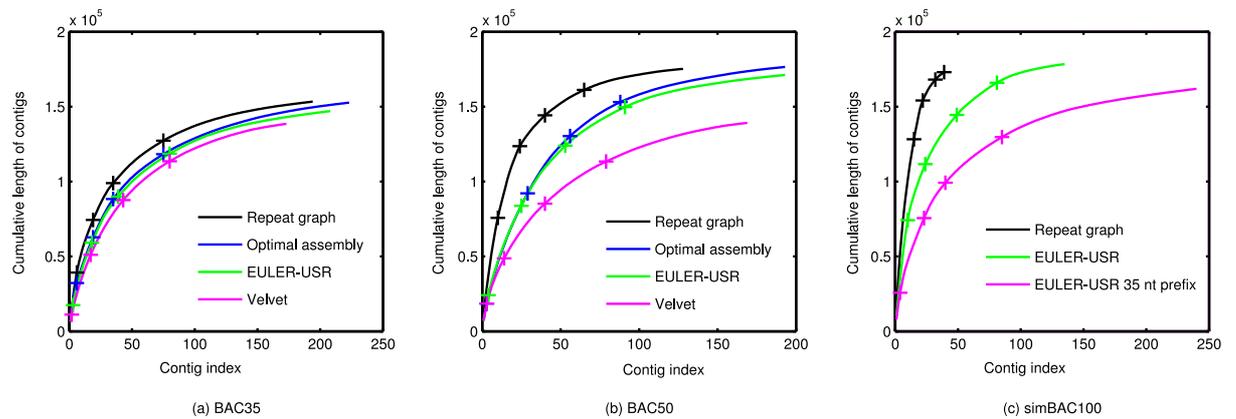


Figure 7: Comparison of EULER-USR (threading) and Velvet. In each plot, the contigs are ordered in the decreasing order of sizes and the cumulative size of  $x$  longest contigs is shown on the y-axis (only contigs longer than 100 bases are shown). See Table 3 for the choice of parameters of all programs in these plots. For all assemblies of the BAC, the locations of the contigs closest to lengths 5000, 2000, 1000, and 500 bases are shown with a "+" mark.

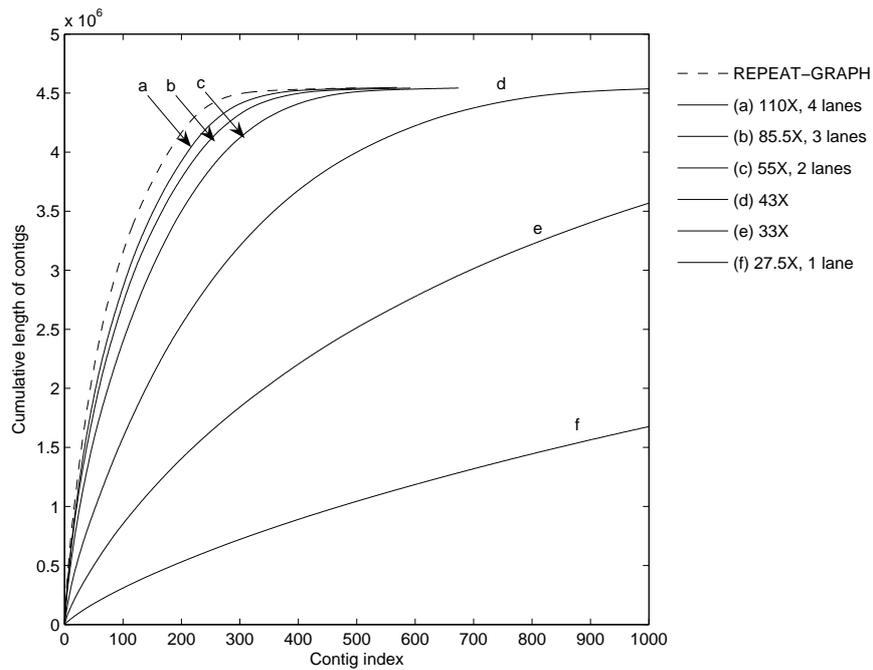


Figure 8: Statistics of assembly for various read coverages (*E. coli* genome). The cumulative length of contigs in order of decreasing length is shown for the 1000 longest contigs. The cumulative length of contigs of the repeat graph on the genome is shown as a dashed line.