National University of Singapore

School of Computing

Department of Computer Science

Graduate Research Paper

De Novo Genome Assembly using Paired-End Short Reads

by

Pramila Nuwantha Ariyaratne

2009

Supervisor: Ken Sung

Index

1. Introduction

1.1 Motivation

Realization of the complete genome sequence is the first important step in analyzing a particular organism. Once the nucleotide sequence is known, various analyses can be performed to gain insight on the function of the organism. Specialized software can be used to predict genes of the organism. Combined with techniques such as SAGE and GIS-PET, we can uncover new transcripts or genes. Technologies such as ChIP-chip, ChIP-seq, or ChIP-PET can aid us discover new transcription factor binding sites (TFBS). Hence, knowing the complete genome sequence of an organism facilitates the understanding of the organism in multiple ways.

Despite this fact, de novo assembly of a complete genome is still far from straight forward. Initial bottlenecks were largely wet-lab bound. However, lately sequencing technology has made progress by leaps and bounds. The main challenge presently lies in computational processing of wet-lab data. Our objective is to present a set of innovative algorithms which can manipulate next generation wet lab sequencing data to assemble underlying genome sequence as complete as theoretically possible.

1.2 Sequencing background

A genome consists of one or many chromosomes. Each chromosome consists of two long complementary strings of DNA (DeoxyriboNucleic Acid) winded in a double helix structure (see Figure 1). The objective of genome sequencing is to determine the exact order in which DNA occurs in each chromosome. While this may sound straight forward in theory, the actual procedure is infinitely more complicated due to the fact that current technology limits the maximum ‘sequencable’ fragment length to ~1000 base pairs (bp) , where as a chromosome can span hundreds of millions of bp. Therefore the sequencing community has adapted ‘whole genome shotgun sequencing’ approach to decode large genomes.

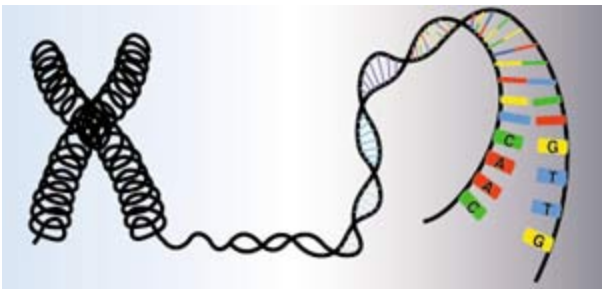

Figure 1: Chromosome structure [1]

The whole genome shotgun sequencing approach is as follows. Initially multiple copies of the target DNA sequence are sheared into small fragments. The length of the fragments is generally fixed to a particular desired size. Each fragment is then individually sequenced to obtain their DNA sequence in the form of A, C, G, T or N, referring to four DeoxyriboNucleic Acids and N for ambiguous basecalls. In some cases the fragment is sequenced from both ends to obtain both forward and reverse reads. The most challenging part of shotgun sequencing is 'arranging' these short fragments to obtain the original genome and our focus is concentrated on this aspect of the pipeline.

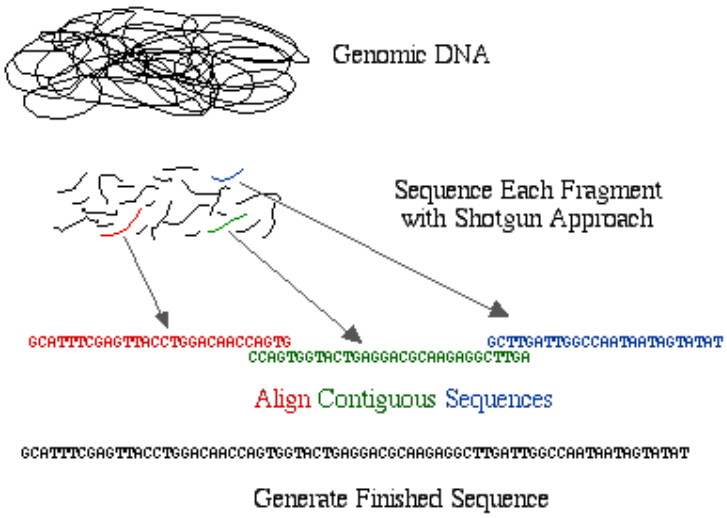

Figure 2: Whole genome shotgun sequencing overview [2]

The process of assembling genome sequences depends on the sequencing platforms and strategies. Until mid 2000 the only sequencing platform available was ABI Sanger/Capillary sequencing. It is capable of reading up to 600bp from each end of a DNA fragment. However actual number of fragments it can read within a specified time was low, leading to very low throughput. As this was the only sequencing platform

available for nearly a decade, most previous genome assembly software was optimized to use fragments of this size.

454 Life Sciences released GS20 sequencing platform which was capable of sequencing up to 400bp at much higher throughput. Assembling sequences generated by this platform was not much different from assembling capillary sequences. Therefore existing algorithms were adapted with slight modifications.

In 2006 Illumina Solexa 1G sequencing platform was introduced to the market. Initially, it was capable of sequencing 25bp tags at a throughput far exceeding both capillary and 454 sequencing at a much lower cost. The short fragment length impeded the de novo assembly of large mammalian genomes. However with its inherent capability to produce paired reads (figure 3), sequencing bacterial genomes was still a possibility. Previous generation of genome assembling software was not particularly geared for assembling such short reads and they were not explicitly designed to take advantage of paired reads. Therefore new approaches were needed to de novo assemble Solexa data. Several such algorithms have proposed and we will be looking into some of the widely used ones further on.

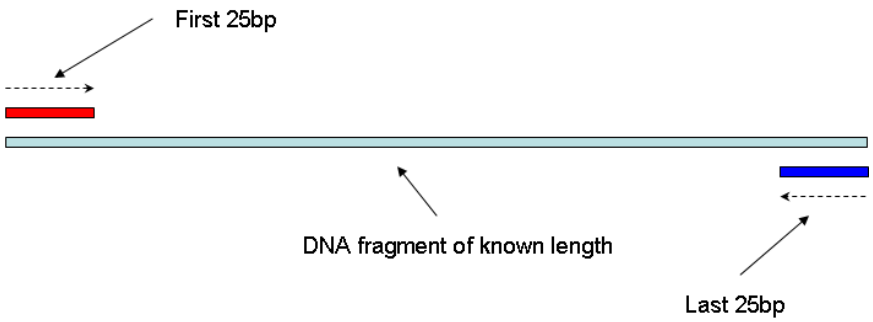

Figure 3: Paired sequencing. First and last sequence tags of a fragment are sequenced and stored together.

In 2007 ABI launched a competing sequencing technology 'ABI SOLiD' sequencing platform, which too is capable of producing a massive number of short paired reads. A comparison between these next generation sequencing technologies is given in figure 4.

	Roche 454 FLX	Illumina Solexa	ABI SOLiD
Sequencing approach	pyrophosphate release	bridge amplification	ligation
Read length	200-300bp	25-35bp	35bp
Read yield	~ 100Mb	~ 1Gb	~ 3Gb
Time per run	7.5 hours	3 to 5 days	Upto 8 days
Reagents cost	~ $5000 / run	~ $3000 / flowcell (1Gb)	~ $3000 / slide (1.5Gb)
Instrument cost	~ $500,000	~ $475,000	~ $525,000
Pros	More runs / year Long reads	Low cost / base Requires less starting material	Low cost / base Requires less starting material
Cons	High cost / base	Short read length Less runs / year	Short read length Less runs / year

Figure 4: Comparison of next generation sequencing platforms. Data obtained from [8]

1.3 Problem description & challenges

Before further analyzing the problem, we need to define the following.

Read length

- Length of each forward/reverse read generated by sequencing machine. Depending on the sequencing technology used, this may not be a constant value for a given library. But we will assume so for our purpose.

Insert size (fragment length)

- Distance between forward – reverse read in the genome

Coverage

- Approximate number of copies of original genome being sequenced. This is equal to *read length x 2 x no_of_reads / genome_length* for paired read libraries.

Contig

- An assembled sequence which we assume forms a contiguous region of the target genome.

Scaffold (Super contig)

- Series of contigs assumed to be in the same order as they are in target genome, possibly separated by a string of 'NNN'.

N50 contig size

- The size of the contig such that 50% of the total assembled length is contained in contigs of size longer than or equal to that size.

The ‘De novo sequence assembly using paired-end short reads’ problem can be succinctly stated as follows:

Given a set (sets) of paired reads where each forward and reverse read is separated by a known distance in the source genome, reconstruct the complete source genome.

However the actual assembly is complicated by the presence of errors and repeats. Errors in paired-end short reads are of mainly two forms.

Sequencing errors

- This may happen during sequencing phase when a particular base is misread as a different base. In some sequencing platforms, it is also possible to have additional or missing base pairs. But this scenario is rare in platforms such as Illumina Solexa 1G and ABI SOLiD, so we omit insertions / deletions of base pairs from our error analysis.

Ligation errors (chimera)

- Ligation errors occur during library preparation when ends of two different fragments are ligated and assumed to have originated from the same fragment.

Sequencing errors can be detected / corrected by having more than one read covering the same position. Most sequencing errors can be modeled as purely random. If we have sequenced to a depth of 5x, should one read at a particular position exhibit a sequencing error, we can still expect other 4 reads covering that position not to contain any errors and therefore correct the erroneous read (figure 5). However there are many regions in the genome which are nearly identical except for a difference in a single base pair. In such cases the assembler should be prudent enough to resolve the two regions separately, rather than collapse it into a single contig. (figure 6)

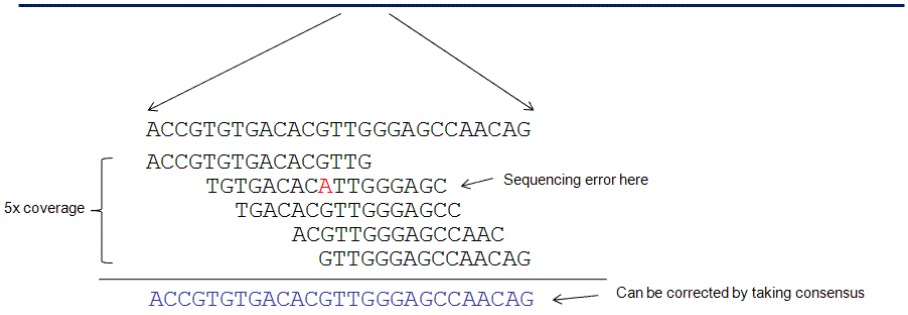

Figure 5: Correcting sequencing errors using excess coverage

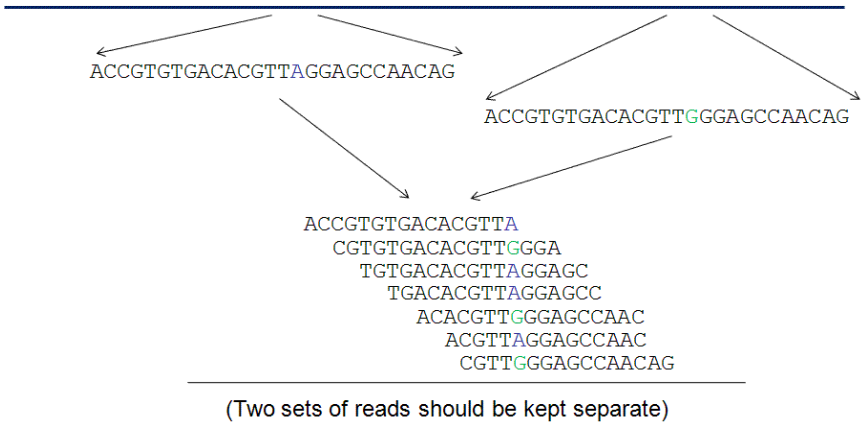

Figure 6: Not a sequencing error. Should not be 'corrected'

Assembling repeat regions of a genome is one of the most complicated aspects of de novo genome assembly. Repeats can be largely categorized into the following types.

Tandem repeats

- When a small block of sequences, usually a few bases to a few tens of bases, immediately repeat many times, it is called a tandem repeat (figure 7). The repeat segment maybe a few hundred base pairs long. In some cases there can be slight variations in between different blocks. Correctly assembling such regions using short reads can be extreme challenging, if not impossible.

ATATATATATATATATATATATATAT

CGTCGTCGTCGTCGTCGTCGTCGT

AGGTCGAGGTCGAGGTCGAGGTCG

Figure 7: Three different tandem repeat sequences

Large repeat regions

- Some sequences, possibly spanning more than a few thousand base pairs, tend to occur multiple times in different regions of the genome. However, there can be slight variations between each occurrence. (figure 6) In such cases assembler should be able to identify such regions and resolve the differences whenever possible. Note that in some cases it maybe theoretically impossible to resolve such differences and assembler should flag such base pairs as ambiguous.

In the next section we take an in-depth look at a few different de novo genome assembly approaches which are currently available.

2. Current approaches

2.1 Traditional approach

Before the arrival of high throughput short fragment sequencing, de novo genome assembly was a somewhat straight-forward problem. Sanger capillary sequences allowed reads lengths up to 600bp which resulted in a significant overlap between adjacent reads. Most assemblers of that era, such as ARACHNE [6], followed the Overlap-Layout-Consensus approach.

Each read was aligned against all other reads. The result is represented as an *Overlap Graph* where each read is a node and a directed edge exists between nodes A to B if and only if 3' of read A overlaps significantly with 5' of read B. A unique traversal between any two nodes in this graph will give rise to a potential contig. Therefore the problem was reduced to finding the set on most consistent traversals in the overlap graph.

Similar to next generation sequencing platforms, Sanger capillary sequencing too was subjected to various errors. Single base pair misreads could be easily corrected by taking consensus sequence across all other reads which have significant overlap with problematic basecall. Another source of errors in Sanger sequencing is *chimeric reads*, where due to the cloning procedure prior to sequencing two different regions of the genome can form a contiguous sequence. Similar errors are less likely in next generation sequencing due to shortness of reads and wet lab preparation procedures.

However the challenges faced in de novo assembly of next generation sequencing data vastly differ from the above, therefore a fundamentally different approach is needed. Compared to 600bp read length of Sanger sequencing, next generation sequencing machines are limited to 25-50bp length reads. This short coming can be somewhat overcome by relying on higher sequencing depth. Read coverage of 10x is sufficient for bacterial genome assembly with traditional Sanger reads, where as coverage up to 100x is not uncommon with next generation data. However expected overlap between two reads is still limited by read length, and in many cases overlap length is limited to around 20bp. Massive number of reads, coupled with such short overlap would result in a highly

convoluted overlap graph with millions of nodes and many non-specific edges in between them. In addition, many noise reduction steps in traditional approaches take advantage of very specific long overlaps between Sanger reads. In absence of such, new methods for noise reduction had to be developed.

There are a few approaches developed specifically to overcome these challenges presented by next generation sequencing data. Most of these algorithms are based on De Bruijn graph approach popularized by Pavel Pevzer et al [7] in 2001. Therefore our analysis of present approaches will be preceded by an overview of De Bruijn graph method.

2.2 De Bruijn graph overview

De Bruijn graph approach to de novo sequence assembly was presented as an alternative to traditional Overlap-Layout-Consensus approach. Although it was initially designed to be used with long Sanger reads, some of its properties are more suited for short read sequences. Therefore we have seen newer approaches designed to deal with short reads have increasingly adapted De Bruijn graph method. Therefore before analyzing specific algorithms, it is necessary to have a sound understanding on De Bruijn graph itself.

De Bruijn graph is a set of nodes and edges where each node represents a sequence of length *k*. A directed edge between two nodes *A* and *B* exists if and only if k-1 length suffix of *A* is equal to k-1 length prefix of *B*.

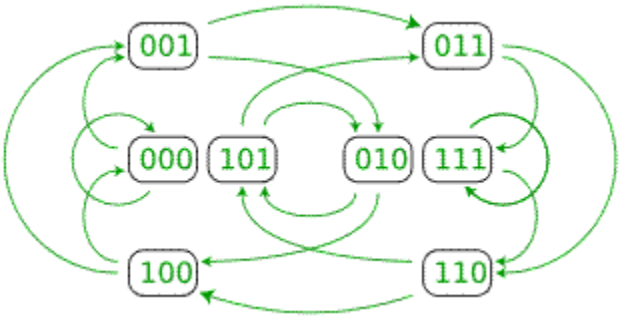

Figure 8: A simple de Bruijn graph [4]

Nodes comprise of length *k* subsequences of all input reads. The length *k* is a critical parameter of the assembly. A large *k* will result is a less convoluted and more linear

graph which would be straight forward to traverse, but may miss some overlaps. The smaller *k* will result in a highly connected graph, but will require some post processing to isolate the correct path.

The graph is further simplified by merging any two nodes with single outgoing/incoming edge; the sequence resulting from the traversing the merged path is represented in the resulting node. This process is illustrated in figure 9.

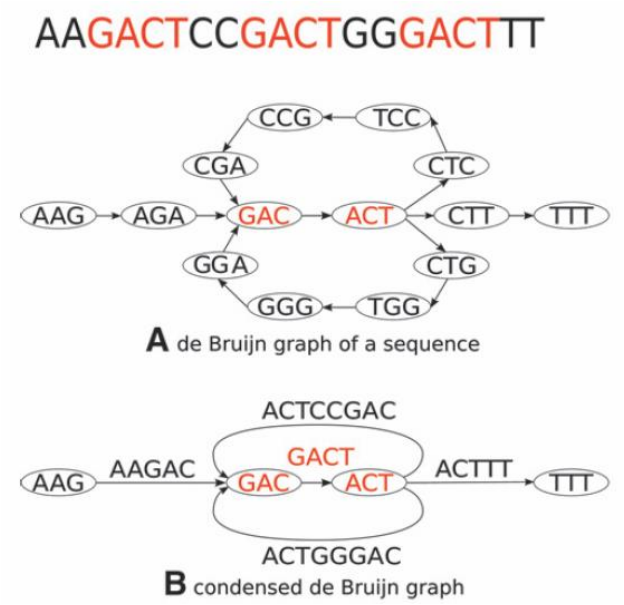

Figure 9: Simplifying the de Bruijn graph [5]

Any error that was present in paired reads is propagated to the de Bruijn graph. A sequencing error towards the end of a read will take form of a 'tip' in the de Bruijn graph. 'Tip' is a node which does not have any outgoing edges, as the erroneous base would likely to cause that none of the other reads would overlap with it. Sequencing error in middle of a short read will be manifested as a 'bubble' in the de Bruijn graph, as there is likely to be two different paths from one node to the other, one resulting in the correct sequence and the other the erroneous sequence.

Once the De Bruijn graph is constructed, assembling the target genome can be simplified into finding an euler traversal without any ambiguities. However this case is rather idealistic as presence of various forms of noise will complicate this process. Therefore

most De Bruijn graph based approaches concentrates mainly on noise reduction and that accounts for most differences between various related approaches.

2.3 SSAKE, VCAKE & SHARCGS

SSAKE (Short Sequence Assembly by progressive K-mer search and 3' read Extension), VCAKE (Verified Consensus Assembly by K-mer Extension) and SHARCGS (SHort-read Assembler based on Robust Contig extension for Genome Sequencing) are some of the very first de novo genome assembly software designed to work with short read sequencing. All three algorithms are base on the same principal. The assembly starts by selecting an unused read as the initial contig and then searching for other reads which overlaps with the 3' of the current contig. If an unambiguous consensus can be found the contig is extended and the process is repeated. Three different methods differ in way which the handle errors. However none of the above methods makes use of paired reads and therefore we focus our attention elsewhere.

2.4 VELVET

Velvet [3] is perhaps the most widely used short read genome assembler currently available. It is popular due to its simplistic approach, speedy execution and relatively accurate results.

Velvet is based on the de Bruijn graph approach and employs a few novel methods to deal with noise inherent with short read data. As mentioned above, sequencing errors can manifest in De Bruijn graph as either 'tips' or 'bubbles'. Velvet deals with 'tips' by truncating it if the following conditions are met. The length of the 'tip' sequence should be less than twice the length of the k-mer and the node at which the 'tip' connects to the rest of the graph should have at least one outgoing edge with larger count than the edge connecting the tip. Otherwise the 'tip' is assumed genuine and will result in a separate contig.

'Bubbles' occur in De Bruijn graph as a result of sequencing errors (in the middle of read) or SNPs. In case of a 'bubble' there would be two different paths between two nodes. One of the paths could be due to the erroneous read(s) and in such a case that path

should be merged with the one resulting in the correct sequence. Velvet introduced a novel method named 'Tour bus algorithm' which carries out detection and correction of such errors.

'Tour bus' algorithm selects an arbitrary node as the starting position and traverses the entire De Bruijn graph in breath-first fashion. However, in this case the 'distance' between two consecutive nodes is defined as length of sequence denoted by destination node divided by number of reads connecting the two nodes. Therefore the distance between two nodes which are connected by more reads is less and such node is traversed prior to traversing nodes connected by lesser number of reads. If 'Tour bus' algorithm reaches a node that have been traversed before via a different path, it will backtrack both paths until a common ancestor is found while extracting the underlying sequence of each path. The two sequences are aligned, and in case there is no significant difference between them, the two paths are merged while preserving the sequence of the path with lower distance (higher coverage).

Velvet uses an algorithm named 'Breadcrumbs' to make use of paired reads to span over repeat regions. Initially Velvet identifies all nodes which are longer than the maximum insert size of the paired reads. These are referred to as 'long nodes'. Then all paired reads are mapped to the graph and any non-unique mappings or mappings that spans larger than the insert size are ignored. Nodes which are connected to a 'long nodes' via at least 5 read paired are marked. Now there is a better chance of finding a unique path between two 'long nodes' by traversing only via the marked nodes. (figure 10) The results of paired reads and 'Breadcrumbs' algorithm are illustrated in figure 12.

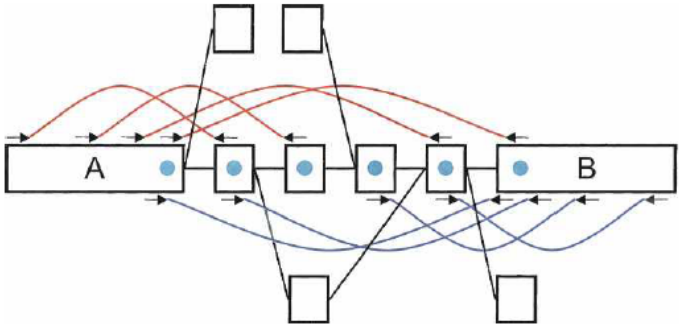

Figure 10: Illustration of 'Breadcrumbs' algorithm [3]. A unique path between 'long nodes' A and B can be found after marking nodes (blue dots) which are connected to them via paired reads.

According to results presented in [3] Velvet performed admirably well against SSAKE and VCAKE. It produces far less contigs of far greater size in much less time with only a small increase in memory usage. The statistics presented testify that error correction algorithms employed by Velvet are highly effective and reduced the number of nodes (potential contigs) considerably.

Assembler	No. of contigs	N50	Average error rate	Memory	Time	Seq. Cov.
Velvet 0.3	470	8661 bp	0.02%	2.0G	2 min 57 sec	97%
SSAKE 2.0	265	1727 bp	0.20%	1.7G	1 h 47 min	16%
VCAKE 1.0	7675	1137 bp	0.64%	1.8G	4 h 25 min	134%

Figure 11: Comparison of Velvet against other short read assemblers on *Streptococcus suis* Solexa experimental data [3]

However we have a few gripes against Velvet. Currently k-mer size in Velvet is limited to 31 (on 64 bit machines). This is possibly due to bitwise operations carried out in overlap detection. But as the read length on next generation sequencing machines continues to increase (currently ~100bp), allowing a higher k-mer size would substantially reduce the complexity of De Bruijn graph. In future Velvet may need to accommodate longer k-mers possibly at performance penalty.

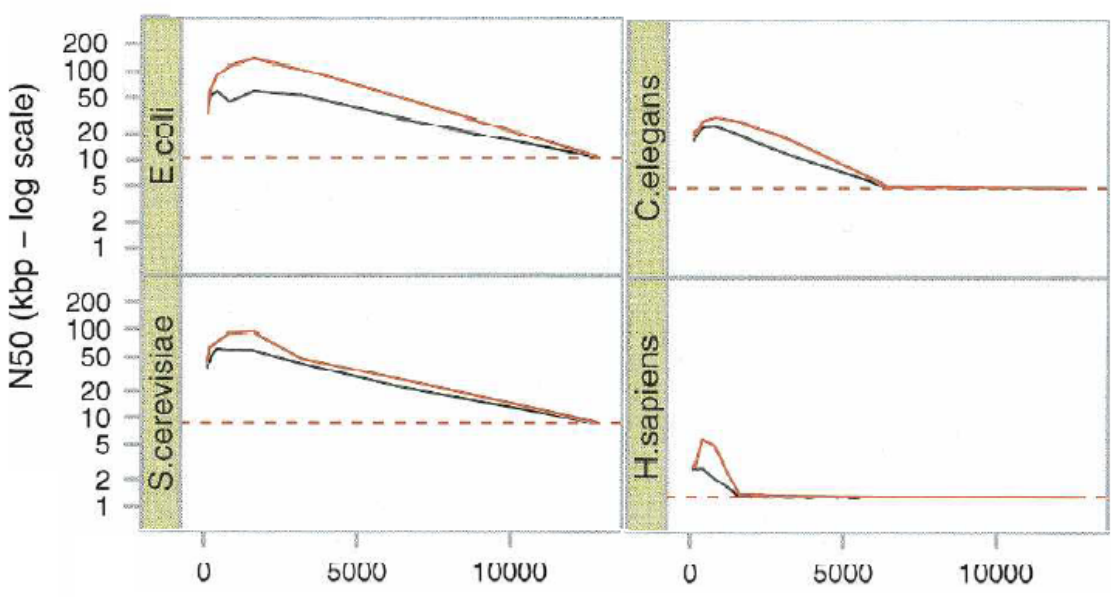

Figure 12: Results of 'Breadcrumbs' algorithm. Dotted line denotes results without 'Breadcrumbs'/paire reads. Solid lines represents 'Breadcrumbs' results. Contig lengths in black and supercontig lengths in red.

While ‘Tour bus’ algorithm appears to be very robust and effective it is likely to collapse large repeat regions with subtle differences in to a single node. We find such regions are common in various bacterial genomes and there exists small difference between each region that may hold key to functioning of those organisms. But with ‘Tour bus’ algorithm those repeat regions are likely to be merged in to a single consensus sequence, thus all such differences will be lost unless any post-processing is carried out to correct them. In presence of paired reads ‘Breadcrumbs’ algorithm maybe able to resolve such subtle differences if it is carried out before the ‘Tour bus’. But this is likely not the case as ‘Breadcrumbs’ algorithm appear to be more a post-processing step.

The use of paired reads itself seems an afterthought in Velvet. Rather than using paired information initially during construction of De Bruijn graph the program uses paired reads as a post processing step towards end of the execution. This results in Velvet having to deal with a complicated De Bruijn graph where as it could have been avoided.

Velvet also fails to use subtle paired read information such as average span or the standard deviation of insert size to its advantage. Furthermore it does not explicitly deal with tandem repeats which is possibly the biggest hurdle in short read assembly, yet frequented in all bacteria and eukaryotic genomes.

2.5 EULER-USR

Similar to Velvet, EULER-USR [5] (EULER-Ultra Short Reads) is a set of tools for de novo genome assembly using short reads based on De Bruijn graph approach. It is able to incorporate paired read information to further the assembly. As the actual assembly using De Bruijn graph is well studied topic, the paper focus more on various error detection and correction methods.

Incremental improvements to Solexa platform has resulted in longer (than 35bp) reads. However the quality of the sequencing suffers toward the end of each read. Error rate in Solexa reads remains less than 2% for first 30bp but rapidly increase to around 20% at 50^{th} bp. This makes the latter part of the sequence unreliable for de novo assembly

without any form of correction. EULER-USR presents a few novel methods to correct these errors and make the full length of the read available for assembly.

EULER-USR makes use of the fact that prefix of each read sequence is likely to be of better quality that the latter part. However prefix itself is prone to errors should therefore be assessed and corrected. Therefore the first step in the program is to obtain a set of error-free read prefixes.

Given the set of reads *R* and threshold *m*, a k-mer is said to be *solid* is the k-mer occurs at least *m* times in the set of reads. Given enough coverage, set of such solid k-mers approximate all k-mers in the original genome. A read is assumed to be error-free is all its k-mers are solid. In case a read contains one or more non solid k-mers the program allows a few base-pair mutations until all k-mers become solid and the read is error-free. A read that cannot be made error-free is discarded and the prefixes of the error-free reads are used for the next step.

In the following step EULER-USR uses prefixes of all error-free reads to build a De Bruijn graph. The graph is further processed to remove 'tips' and 'bubbles' to obtain a 'Repeat Graph'. (figure 13).

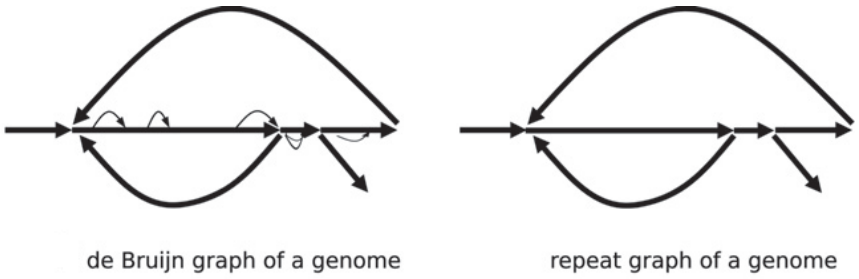

Figure 13: De Bruijn graph and repeat graph of same genome.

In order to correct more error prone read suffixes, EULER-USR uses the repeat graph in a technique which it refers to as 'read threading'. As the repeat graph is based on prefixes of error-free reads, prefix of any read is likely to be in the repeat graph. For each such read the program locates the prefix sequence in the repeat graph and then extends into the suffix of the read following the sub-path with minimum edit distance. If there are any differences it is regarded as an error in the read suffix and is corrected. Once this

procedure is complete the repeat graph is reconstructed using full-length reads instead of just the prefixes.

EULER-USR makes use of paired reads in a different way than that of Velvet. EULER-USR attempts to reconstruct the sequence in between two reads so that each paired read of insert size *d* will become a single read sequence of length *d + 2 x read length*. For each paired read it searches within repeat graph for path between two nodes where the two reads maps to. In case there is only a single such path with distances similar to insert size then that path is chosen. In case there are multiple paths it selects the path with highest support. Support for each path is defined as the number of paired reads where one end of the read maps to starting or ending node and the other end maps to a node in the concerned path. This is illustrated in figure 14.

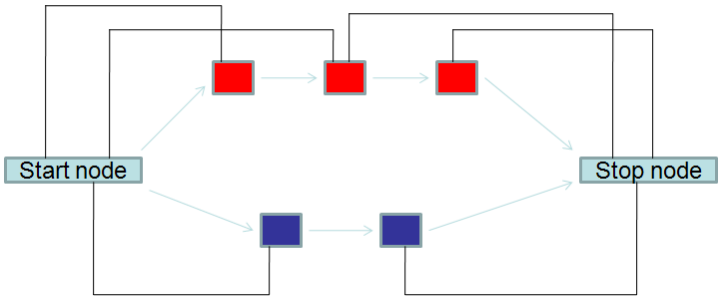

Figure 14: Definiton of support. Black lines denote paired read mappings. Read path has support of 4 and blue path has support of 2.

Once the correct path is determined, the sequence across that path is assembled and assumed to be the full sequence in between the paired read. Once the full sequence for all paired reads are found the repeat graph is updated again using that information. This would be the final repeat graph and can be traversed similar to De Bruijn graph to find the contigs.

For comparison purposes EULER-USR and Velvet was run on 35bp E. Coli dataset sequenced by Solexa platform. The results are summarized in figure 15. Based on N50 contig size EULER-USR seems to outperform Velvet both with and without paired reads. Results also highlight the advantage of having paired reads, as both programs will then result in higher N50 size with less number of contigs.

Assembly	N50	Length (# contigs) >20,000 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)

Figure 15: Comparison of EULER-USR and Velvet. Repeat graph denotes the theoretical maximum. [5]

An obvious reason for the improvement of results over Velvet is effectiveness of error correction algorithms employed by EULER-USR. Performance of two error correction algorithms is summarized in figure 16. 'SA corrected reads' in this instance refers to reads corrected by solid k-mer strategy. For instance in 50bp Solexa reads EULER-USR was able to reduce the error to a very acceptable 0.05% from an unusably high 4.36% without truncating the 3' end of the reads.

	Original reads		SA corrected reads			Threaded reads after graph correction	
Data set	Length	Error rate (%)	Average length	Error rate (%)	Retained reads (%)	Average length	Average 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

Figure 16: Results of EULER-USR error correction. Error rate for Human BAC reads were estimated by mapping to human genome. [5]

Although EULER-USR seems like another algorithm based on De Bruijn graph approach, the novelty of the method seems to lay in the error detection and correction. Judging by the presented results it is very effective. In spite of this we feel that overly aggressive error correction methods employed by EULER-USR may work to its disadvantage. The subtle base pair differences within large repeat regions are likely to be lost during this process, especially given that error correction procedures do not take paired read information in to account for localization.

Another complain which was also common to Velvet is the fact that use of paired reads in EULER-USR seems a post processing step. Again EULER-USR fails to account for tandem repeats; in fact the *support* function in paired read 'filling' procedure could favour paths containing tandem repeat regions as then higher number of 'support reads' will map to those paths.

2.6 ALLPATHS

ALLPATHS [11] is an algorithm developed by Broad Institute exclusively for de novo sequencing with paired reads. The algorithm is somewhat based on a graph based approach similar to Velvet and EULER-USR but differentiates itself in some crucial ways.

ALLPATHS initially carries out a sequence error correction step using the same solid k-mer method used in EULER-USR. Then the corrected sequences are used to build a 'unipath graph' which is in theory identical to the repeat graph structure in EULER-USR. Each linear segment in unipath graph is called a unipath and forms the basis for assembly algorithm.

One crucial way ALLPATHS differ from other methods is that it incorporates localization of reads with the help of paired reads. A unipath with 'normal' coverage (to exclude repeat regions) is chosen as a seed, and any other unipaths iteratively connected to the seed via paired reads are defined to be in the neighbourhood. The size of the neighbourhood is typically limited to 10kb distance for each direction. Any paired reads where one tag maps to any of the neighbourhood unipaths are also considered to be in the neighbourhood. From here on the assembly program works within this isolated neigbhourhood. This break down the complexity of the problem while also makes the program usable on clustered computer systems, effectively speeding up the execution.

The program dwell on finding all paths (hence the title) between two tags of each paired read (similar to EULER-USR) but proves that this is intractable in the presence of repeat regions as number of possible paths between each pair could run in to thousands. To conquer this problem ALLPATHS presents as ingenious solution.

It extends each tag of a read pair based on overlap with other tags. Once the tags are sufficiently long it is able to find other extended paired reads which would overlap with both tags. Then the two paired reads can be combined together to obtain a longer reads with a reduced variance. Consider the example below.

$$B.C \quad —(40 \pm 5) \rightarrow \quad W.X.X.Y$$

represents an extended paired read, where left tag is made of two segments B and C and right segment with four segments. The distance between the two ends is 40 +/- 5 k-mers. This paired read can be merged with the following,

$$A.B \quad —(50 \pm 5) \rightarrow \quad W.X$$

And obtain

$$A.B.C —(40 \pm 5/\sqrt{2}) \rightarrow W.X.X.Y$$

Now not only that read lengths are longer, the variance of distance in between are decreased. This process can be carried out iteratively until the full sequence between the paired read is obtained. Gaps between unipaths in each neighbourhood can be filled will such reads. Once neighbourhood sequence is complete they can be used to assemble the whole genome.

ALLPATHS was run on various sets of simulated data by the authors and the results are summarized below.

	Inputs			Outputs				
Species	Ploidy	Genome size (kb)	Reference N50 (kb)	Component N50 (kb)	Edge N50 (kb)	Ambiguities per megabase	Coverage (%)	Coverage by perfect edges ≥10 kb (%)
C. jejuni	1	1800	1800	1800	1800	0.0	100.0	100.0
E. coli	1	4600	4600	4600	4600	0.0	100.0	100.0
B. thailandensis	1	6700	3800	1800	890	2.7	99.8	99.5
E. gossypii	1	8700	1500	1500	890	2.6	100.0	99.9
S. cerevisiae	1	12,000	920	810	290	28.7	98.7	94.9
S. pombe	1	13,000	4500	1400	500	19.1	98.8	97.5
P. stipitis	1	15,000	1800	900	700	8.6	97.9	96.3
C. neoformans	1	19,000	1400	810	770	4.5	96.4	93.4
Y. lipolytica	1	21,000	3600	2200	290	6.2	99.1	98.6
Neurospora crassa	1	39,000	660	640	90	17.4	97.0	92.5
H. sapiens region	2	10,000	10,000	490	2	68.2	97.3	0.2

Figure 17: Summary of ALLPATHS results [11]

ALLPATHS results are very impressive. Not only that it is able to produce assemblies with large N50 sizes, the number of errors is very few. Our analysis showed at ALLPATHS was even able to correctly assemble complex tandem repeat regions, which was thought to be beyond realm of short read sequencing.

However there are many shortcomings in ALLPATHS approach. One of the assumed data set for above simulation is a set of short reads of span 500bp +/- 5bp at 40x

coverage. In practice getting such high throughput with such a sharp peak is rather unrealistic. Other disadvantage with ALLPATHS is that it performance. Some assemblies required up to 64GB memory in a multi-node cluster environment running for 1.5 days, and does not compare favourably with program such as Velvet which are extremely fast and efficient.

In conclusion ALLPATHS bring forward some interesting ideas to field of paired read assembly, but however unrealistic demands on both wet lab and computer resources has made this approach lacking.

[1] Sequencing the Genome, Genome News Network
http://www.genomenewsnetwork.org/articles/06_00/sequence_primer.shtml

[2] Sequencing strategies for whole genomes,
http://www.bio.davidson.edu/courses/GENOMICS/method/shotgun.html

[3] Daniel Zerbino and Ewan Birney. Velvet: Algorithms for De Novo Short Read Assembly Using De Bruijn Graphs. Genome Res. 18: 821-829. 2008

[4] De Bruijn Graphs – Wikipedia, http://en.wikipedia.org/wiki/De_Bruijn_graph

[5] Mark J. Chaisson, Dumitru Brinza and Pavel A. Pevzner. De novo fragment assembly with short mate-paired reads: Does read length matter? Genome Res. 19:336-346. 2009

[6] Serafim Batzoglou, David B. Jaffe, Ken Stanley, et al. ARACHNE: A whole genome shotgun assembler. Genome Res. 2002 12: 177-189

[7] Pavel A. Pevzner, Haixu Tang, and Michael S. Waterman. An Eulerian path approach to DNA fragment assembly. Proc Natl Acad Sci U S A. 2001 Aug 14; 98(17):9748-53

[8] http://genomics.ucr.edu/about/reports/SequencerComparison1207_Table.pdf

[9] Mark J. Chaisson, Haixu Tang and Pavel A. Pevzner. Fragment assembly with short reads. Bioinformatics 20:2067-2074, 2004.

[10] Pavel A. Pevzner and Haixu Tang. Fragment assembly with double barreled data. Bioinformatics, S225-233, 2001.

[11] Jonathan Butler, Iain MacCallum, Michael Kleber, Ilya A. Shlyakhter, Matthew K. Belmonte, Eric S. Lander, Chad Nusbaum, and David B. Jaffe. ALLPATHS: De novo assembly of whole-genome shotgun microreads. Genome Res. 18: 810-820. 2008