Nikhil Gopal
BIOC 218 Final Project
December 3, 2010

Comparison of Short Read De Novo Alignment Algorithms

Abstract

The objective of this paper is to survey the algorithms used for de novo alignment of short read data. Since the quality of the sequence bases which are aligned is important, this paper starts by comparing conventional sequencing methods and next-generation sequencing platforms. Next-generation sequencing poses new challenges to the bioinformatics community. A description of several de novo alignment algorithms is provided, after which there is a discussion about their differences in approach and whether or not the programs provide solutions to mitigate the disadvantages of using next-generation sequencing technology. From here, this paper describes a suggested implementation of a de novo alignment algorithm building upon the successful principles of the short-read de novo aligners surveyed.

Conventional Sequencing

There are three main approaches to conventional sequencing. Some approaches are similar to each other and some are very different. However, all of these conventional sequencing methods share certain properties with their output.

Hierarchical Sequencing [1]

The first sequencing method is called "hierarchical sequencing," and it was the sequencing method of choice for the Human Genome Project. Hierarchical sequencing involves cutting genomic DNA into ~150Mb pieces and inserting them into BAC vectors. These BAC vectors are then transformed into E. Coli, replicated, and stored. The BAC inserts are then isolated, and each 150Mb fragment is mapped and ordered ("golden tiling path"). The "golden tiling path" is again randomly sheared into even smaller pieces. Each piece is cloned into a plasmid and sequenced on both strands. Contigs are now created with the sequence data and the genome is then assembled (with about 8x coverage).

Shotgun Sequencing [1]

The second sequencing method is called "shotgun sequencing." This method was used by Celera in their effort to sequence and assemble the human genome. This method is known to be great for small genomes (such as that of prokaryotes) which do not contain too many repetitive sequences. The shotgun sequencing approach basically cuts out the use of BACs and goes right into the step of shearing DNA into random fragments and cloning them into plasmids (for both strands). From here, contigs are assembled and aligned. Omitting the BAC step makes this method more prone to errors. Since the chromosomal location of each BAC is known, there are not as many truly random pieces to assemble with the hierarchical method. "For example, if a 500 kb portion of a chromosome is duplicated and each duplication is cut into 2kb fragments, then it would be difficult to determine where a particular 2 kb piece should be located in the finished sequence since it occurs twice. You might think, 'who cares since they're duplicates?' But duplications seldom retain their original sequences; they tend to drift over time. So a small region may be retained while other parts may mutate. This might create overlapping sequences for small pieces that are located several hundred kb apart on the chromosome" [1]. It seems like another drawback from a straight shotgun sequencing approach is obtaining false positive alignments by concatenating two portions of the genome hundreds of base pairs apart.

Sanger Sequencing [6]

Sanger sequencing is another sequencing method. Sanger uses a special type of nucleotide in addition to normal nucleotides (ddNTP). These nucleotides have a hydrogen group where the hydroxyl group should be on the 3' end. This prevents phosphodiester bonds from forming and terminates the DNA chain. First, the DNA is denatured into two separate strands with heat and a primer is then annealed to one of the template strands. These primers can be specially constructed to bind to special parts of the template strand, giving the ability to sequence a region of interest. The primer or the nucleotide is fluorescently labeled so that they can be identified on a gel. The solution is divided into 4 tubes, A, C, T, and G, and each tube is filled with all four DNA nucleotides along with its corresponding ddNTP. Since the ddNTPs are randomly integrated, the fragments are all different sizes. However, all of the fragments have the same starting position. After this process, the DNA is denatured and run on a gel. Sanger sequencing produces reads up to 1000 bases long and usually have about 10x coverage.

Synopsis of Conventional Sequencing Methods

In a nutshell, the hierarchical approach to sequencing is more time consuming and expensive compared to the shotgun approach. However, the hierarchical approach method has its advantages in that the sequence data is less prone to producing inaccurate assemblies. Sanger sequencing does not produce reads as long as that of hierarchical or shotgun sequencing. Hierarchical and shotgun sequencing have about 8x coverage and Sanger sequencing has about 10x coverage—when it comes to coverage, they do not differ by much.

Next-generation Sequencing Methods

In modern times, many other methods have been developed to generate sequence data. In many ways, they are more advanced than the classical sequencing methods mentioned above. Instead of outlining how each platform works step-by-step as for conventional sequencing, only essential properties of these platforms will be covered. All of the below methods are considered next-generation sequencing platforms.

Illumina's HiSeq 2000 [7]

Illumina's HiSeq 2000 machine can produce 100 bp read lengths with 30x coverage and a relatively low error rate. Illumina technology is a form of flourescently labeled sequence technology. On the low end, it can generate up to 35 GB of data with reads of 35 bases in length. On the high end, it can generate up to 200 GB of data with reads of 100 bases in length. The HiSeq provides about 30x coverage. The HiSeq is also capable of multiplexing samples. This expands the type of experiments that can be done with the instrument and potentially adds more complexities to the sequence data.

Roche's 454 [8] [20]

Roche's 454 machine has longer read lengths than other methods. This mitigates the difficulty of mapping repetitive regions. However, these longer reads do not come without high error rates (especially in homo-polymer repeats). 454 has been used to do de novo alignment of bacterial and insect genomes. The 454 instrument is a parallelized version of pyro-sequencing technology. The instrument generates about .5 GB of data with read lengths of about 400 bases.

Life Technology's SOLiD 4 [9]

Life Technology's SOLiD can produce between 35 and 50 base pairs for each read. SOLiD has two-base encoding which provides a form of error correction. SOLiD sequences by a process of ligation. The instrument generates about 35 GB of data on the low end with reads of about 35 bases in length. On the high end, it can generate about 100GB of data with reads of up to 50 bases in length. SOLiD provides about 15x coverage. SOLiD can also multiplex data (up to 1,536 per run).

PacBio RS [19]

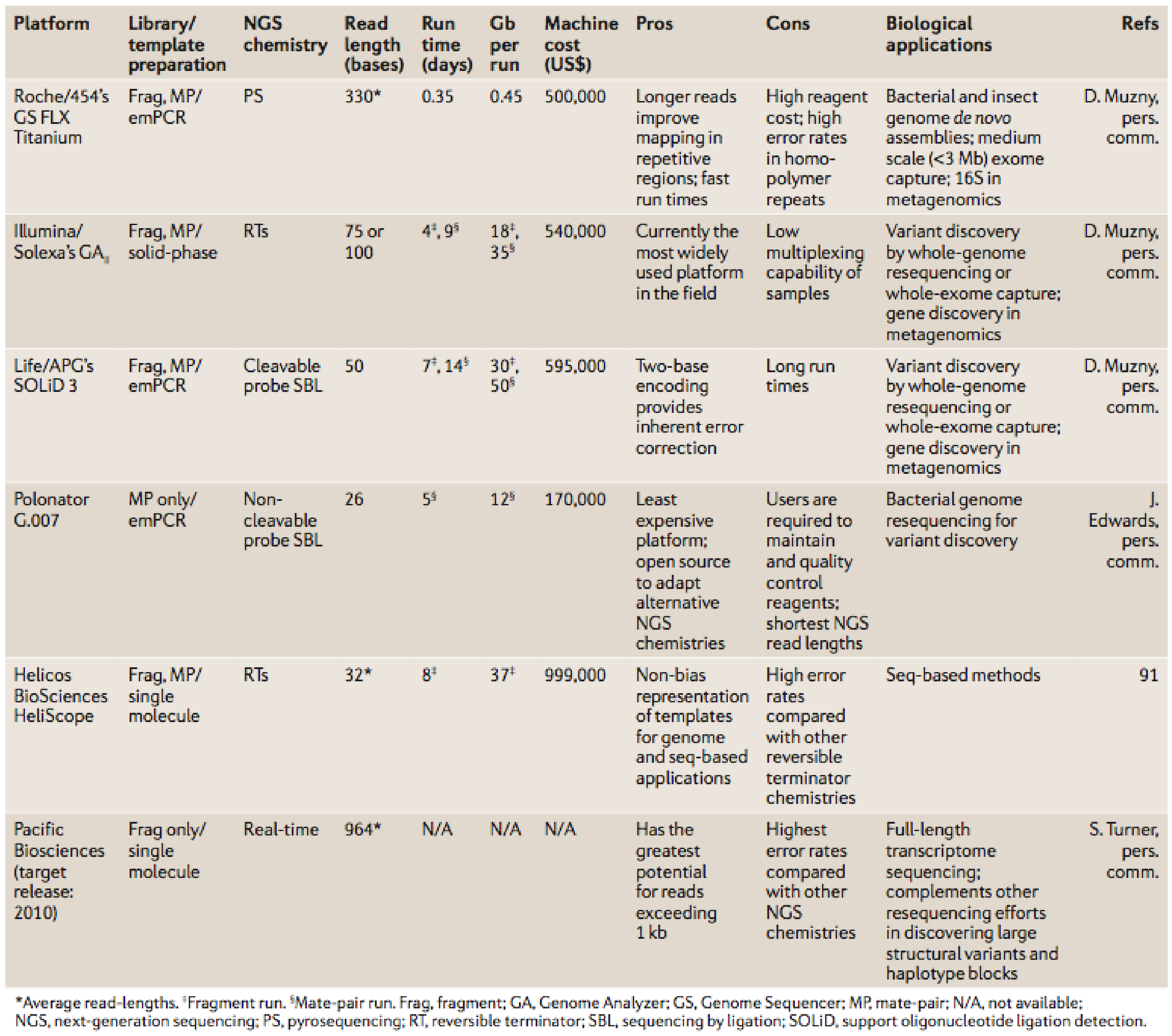
Pacific Biosystems is a long awaited technology. PacBio's machines have the ability to go out to 1000 base pairs. However, it is important to note that this technology has the highest error rate of all of the methods [20]. This technology seems to be comparable to Sanger sequencing in terms of read length. Although little information is currently available about the specifics of this instrument, this technology will likely play a large role in the future of next-gen sequencing.

Summary Table of Next-gen Sequencing Data

Platform	Mate-Pair	Throughput (GB)	Read Lengths (bp)	Run-Time	% Data >Q30	Coverage
Illumina HiSeq 2000	no	25-35 GB	1x35	1.5 Days	>90%	30x
	yes	150-200 GB	2x100	8 Days	>80%	30x
Life SOLiD 4	no	25-35 GB	1x35	3.5-4.5 Days	80%	15x
	yes	80-100 GB	2x50	12-16 Days	80%	15x
Roche 454	N/A	~0.5 GB	400	~1 Day	40%-60%	N/A
PacBio RS	N/A	N/A	~1000	N/A	N/A	N/A

*Please note that this table contains data about the low-end and high-end configurations for the Illumina and Life Technologies platforms. This is why there are two rows of data for these and not the others [7] [8] [9] [19] [20].

Dr. Michael L. Metzker has published an informative summary table about some very similar next-gen sequencing platforms in his paper, "Sequencing technologies — the next generation," which was published in *Nature*. He provides some of the same metrics provided in the table above about these similar instruments and also notes the advantages and disadvantages of each platform [20].



Platform	Library/ template preparation	NGS chemistry	Read length (bases)	Run time (days)	Gb per run	Machine cost (US$)	Pros	Cons	Biological applications	Refs
Roche/454's GS FLX Titanium	Frag, MP/ emPCR	PS	330*	0.35	0.45	500,000	Longer reads improve mapping in repetitive regions; fast run times	High reagent cost; high error rates in homo-polymer repeats	Bacterial and insect genome *de novo* assemblies; medium scale (<3 Mb) exome capture; 16S in metagenomics	D. Muzny, pers. comm.
Illumina/ Solexa's GA$_{II}$	Frag, MP/ solid-phase	RTs	75 or 100	$4^{\ddagger}$, $9^{\S}$	$18^{\ddagger}$, $35^{\S}$	540,000	Currently the most widely used platform in the field	Low multiplexing capability of samples	Variant discovery by whole-genome resequencing or whole-exome capture; gene discovery in metagenomics	D. Muzny, pers. comm.
Life/APG's SOLiD 3	Frag, MP/ emPCR	Cleavable probe SBL	50	$7^{\ddagger}$, $14^{\S}$	$30^{\ddagger}$, $50^{\S}$	595,000	Two-base encoding provides inherent error correction	Long run times	Variant discovery by whole-genome resequencing or whole-exome capture; gene discovery in metagenomics	D. Muzny, pers. comm.
Polonator G.007	MP only/ emPCR	Non-cleavable probe SBL	26	$5^{\S}$	$12^{\S}$	170,000	Least expensive platform; open source to adapt alternative NGS chemistries	Users are required to maintain and quality control reagents; shortest NGS read lengths	Bacterial genome resequencing for variant discovery	J. Edwards, pers. comm.
Helicos BioSciences HeliScope	Frag, MP/ single molecule	RTs	32*	$8^{\ddagger}$	$37^{\ddagger}$	999,000	Non-bias representation of templates for genome and seq-based applications	High error rates compared with other reversible terminator chemistries	Seq-based methods	91
Pacific Biosciences (target release: 2010)	Frag only/ single molecule	Real-time	964*	N/A	N/A	N/A	Has the greatest potential for reads exceeding 1 kb	Highest error rates compared with other NGS chemistries	Full-length transcriptome sequencing; complements other resequencing efforts in discovering large structural variants and haplotype blocks	S. Turner, pers. comm.

*Average read-lengths. ‡Fragment run. §Mate-pair run. Frag, fragment; GA, Genome Analyzer; GS, Genome Sequencer; MP, mate-pair; N/A, not available; NGS, next-generation sequencing; PS, pyrosequencing; RT, reversible terminator; SBL, sequencing by ligation; SOLiD, support oligonucleotide ligation detection.

Synopsis of Next-gen Sequencing Methods

When it comes to the most accurate instrument, it looks like SOLiD wins the contest with 99.49% accuracy[1] [9]. The Pacific Biosystems machine blows everything else out of the water when it comes to read length—PacBio RS reads are 4 to 10 times longer than that of other instruments. However, it has the lowest accuracy rates [20]. The Hiseq provides the most coverage out of the instruments, and it also has the highest throughput [9][7]. The HiSeq 2000 and SOLiD both have the ability to mate-pair which is a big advantage. Dr. Micheal Brudno from the University of Toronto has stated, "For handling alignment gaps the use of mate pairs is critical: if the alignment gap is supported by mate-pair data where one of the pair-ends does not map, it is possible to do de novo assembly on just the small set of reads to recover the polymorphic region." [10] The ability to generate mate-pair reads already gives the

1 Please refer to the Applied Biosystems website. There is a specification sheet available for download that contains this data.

HiSeq 2000 and SOLiD an advantage over the other platforms for de novo sequencing. The HiSeq 2000 provides the highest number of quality reads. This is an important factor to consider as having more high quality data in a short-read dataset makes alignment substantially easier [10]. Unfortunately, although 454 generates longer read lengths, it comparatively does not come up to par in terms of throughput or bases passed filter. Based on the information above, the HiSeq 2000 or SOLiD are the best choices for generating short-read data. However, 454 is usually the sequencing method of choice for researchers interested in de novo alignment (traditionally chosen for de novo alignments due to the longer read lengths). In the future, the PacBio RS is going to be a formidable competitor for the 454 machine.

In many ways, next-generation sequencing is a blessing. It drives down the cost per base of sequencing and generates magnitudes of more data than conventional sequencing would. However, this technology does pose new hurdles for the scientific community. These machines have the capability to generate such a high magnitude of data that it is difficult to handle all of it! Since all of these technologies produce short reads, it is difficult to deal with the reads that align to repetitive regions of a genome. Another pain point of next-generation sequencing is that these platforms are more prone to error than conventional sequencing.

In the end, no matter what sequencing method one chooses to use, conventional or next-generation, the output is sequence data. The overwhelming majority of de novo aligners have been written for data output by the Illumina, Sanger, and 454 platform. [22].

Overview of De Novo Alignment Algorithms for Conventional Sequencing Data

The de facto standard approach for de novo alignment has changed over the years. Traditionally, there were genome assemblers created to assemble Sanger sequencing data and data from Celera. At the heart of these assembly techniques, the core algorithm is as simple as overlapping similar reads into progressively longer sequences until the entire genome is assembled. This is called the layout-overlap-consensus method. It may be interesting to examine two famous genome assemblers named Phrap and TIGR to illustrate the underlying simplicity of these techniques (although they may seem quite involved on the surface).

Phrap [2] [3]

Phrap is a very popular assembly program. It applies the simple core algorithm explained above, but it also performs a few additional steps that make its assemblies great. Dr. Eric Roberts, a computer science professor at Stanford University has summarized the steps in the Phrap algorithm:

1. Trim any homopolymer runs at the end of reads and construct read complements.
2. Find pairs of reads with matching words and eliminate any exact duplicate reads. Run swat comparisons of pairs of reads which have matching words and compute a swat score.
3. Find probable vector matches and mark them so that they are not used in the assembly.
4. “Find near duplicate reads.”
5. “Find reads with self-matches.”
6. “Find matching read pairs that don't have confident matching segments.”
7. Use pairwise matches to identify confirmed parts of reads and use these to compute revised quality values.
8. Compute LLR scores for each match (this is based on the qualities of the sequence bases).
9. “Find the best alignment for each matching pair of reads that have more than one significant

alignment in a given region" (essentially the highest LLR-scores among several overlapping reads).
10. "Identify probable chimeric and deletion reads."
11. Construct contig layouts using consistent pairwise matches in order of decreasing score.
12. "Construct contig sequence as a mosiac of the highest quality parts of the reads."
13. Align reads to contig and tabulate inconsistencies and adjust the LLR-scores of the contig sequence.

TIGR [4]

TIGR compares fragments based on oligonucleotide content before attempting assembly. This eliminates the need for a more careful comparison between the majority of fragment pairs, which in turn reduces the amount of time a computer takes to assemble the data. The algorithm can be condensed into about seven key steps:

1. "Perform pairwise fragment comparisons for the entire dataset to generate a list of potential fragment overlaps."
2. "Use the distribution of the number of potential overlaps for each fragment to label fragments as a repeat or a nonrepeat sequence."
3. Start with a nonrepeat sequence as the initial assembly seed or a repeat sequence if no nonrepeat sequences are left ("if no more sequence fragments are available then the program quits").
4. Use potential overlap list to attempt merges between the assembly at hand and nonrepeat fragments.
5. "When no potential overlaps with nonrepeat sequence fragments remain for the current assembly, increase the stringency of the match criteria and enforce clone length constraints when attempting to merge with repeat fragments."
6. "If due to a merge with a repeat sequence, a nonrepeat sequence is added to the potential overlap list." At this point it would loop back to step 4.
7. "When there are no sequence fragments left on the current potential overlap list, output information about the current assembly and return to step 3." This process is essentially rinsed and repeated until a full assembly is output.

Synopsis of De Novo Alignment Algorithms for Conventional Sequencing Data

Phrap makes it a point to trim reads before assembly where as TIGR does an initial pass to justify not doing a very careful assembly (and optimize for speed). Phrap also actively looks for and removes duplicate reads as well as reads that look like they may have vector sequence. Also, unlike TIGR, Phrap directly takes read quality into account. TIGR and Phrap both revolve around a pairwise alignment method.

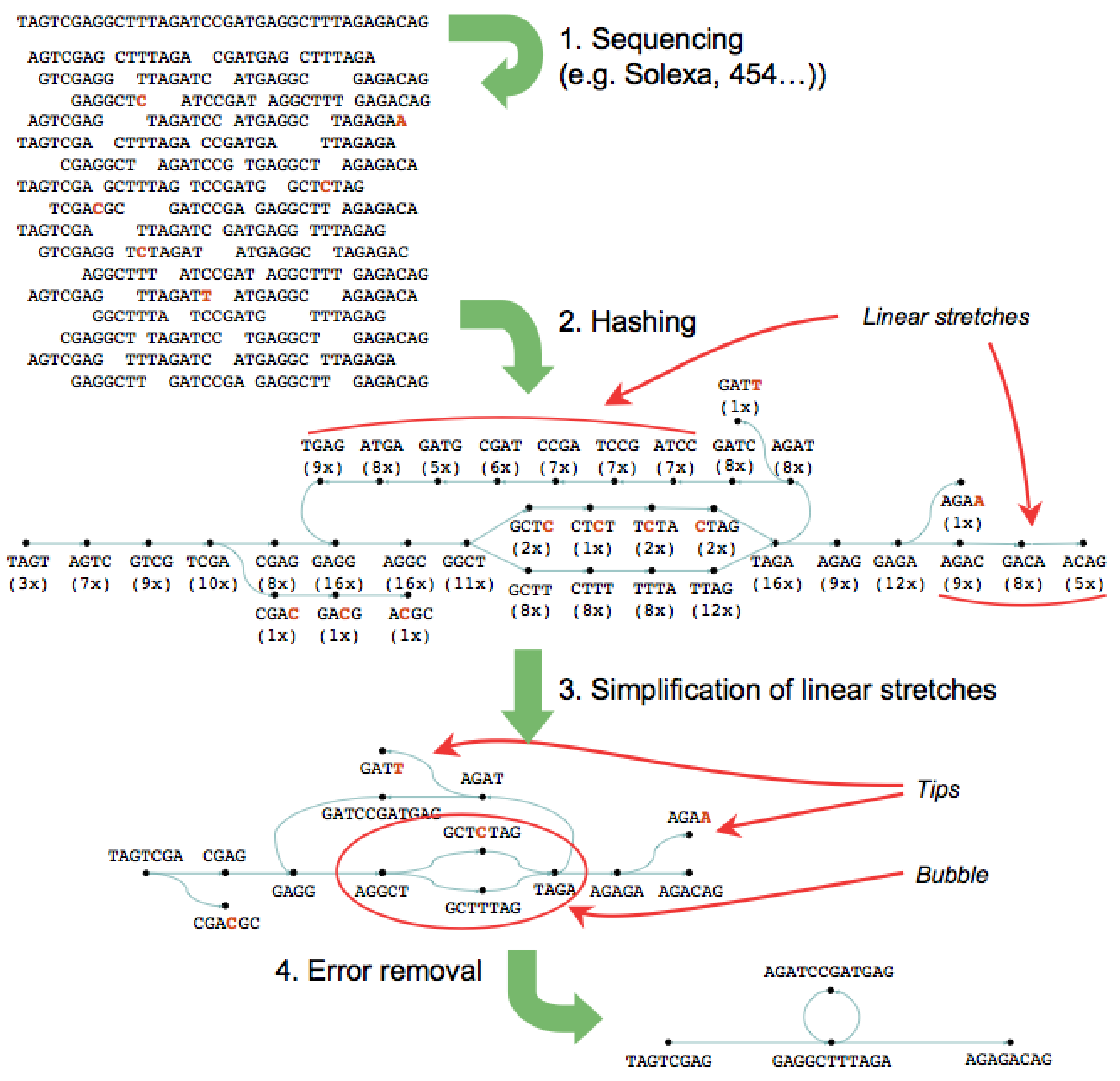
Overview of De Novo Alignment Algorithms for Next-gen Sequencing Data

Even though the layout-overlap-consensus approach of assembling a genome is conceptually simple, it can actually be a difficult task to complete with next-generation sequencing data. One of the main issues is that these older assemblers are difficult to scale to large sequence datasets [12]. Short-read sequence datasets need deeper coverage to be considered equivalent to traditional Sanger reads. The volume of these short read datasets make using the layout-overlap-consensus method very computationally expensive. As this is the case, some prefer a different approach which uses De Bruijn graphs. Many of today's most frequently used aligners use the De Bruijn method as it has been proven to be faster and more efficient for de novo assembly of short-reads. [11]

Velvet [11]

Velvet is a very popular de novo aligner for short-read sequence data. It is generally used to de novo align genomes of all sizes, but it is able to handle larger genomes especially well. “Velvet has already proved successful in removing errors and isolating long unique regions from experimental datasets: a human BAC and Streptococcus Suis” [11]. The algorithm consists of two main steps:

1. First, take the sequence data and create a De Bruijn graph using k-mers. The k-mer variable is important as it determines a balance between sensitivity and specificity (the lower the k-mer size compared to the read, the more “connective” the De Bruijn graph is). The De Bruijn graph is then hashed according to the word length.
2. After the graph has been created and the reads hashed, errors are removed. The two types of errors that are removed are “tips” and “bubbles.” Tips are low quality read ends which do not overlap with other reads and bubbles constitute errors in the middle of a long read or two erroneous read ends overlapping. Low coverage nodes which have not been associated with any contigs are removed. However, dubious overlaps (a result of bubble errors) are conserved and remapped. This alternative data structure is important as it helps to quickly eliminate errors without sacrificing low coverage regions—this helps maintain the integrity of the De Bruijn graph. Velvet is ideal for reads between 25 and 50 base pairs.


TAGTCGAGGCTTTAGATCCGATGAGGCTTTAGAGACAG
AGTCGAG CTTTAGA CGATGAG CTTTAGA
GTCGAGG TTAGATC ATGAGGC GAGACAG
GAGGCTC ATCCGAT AGGCTTT GAGACAG
AGTCGAG TAGATCC ATGAGGC TAGAGAA
TAGTCGA CTTTAGA CCGATGA TTAGAGA
CGAGGCT AGATCCG TGAGGCT AGAGACA
TAGTCGA GCTTTAG TCCGATG GCTCTAG
TCGACGC GATCCGA GAGGCTT AGAGACA
TAGTCGA TTAGATC GATGAGG TTTAGAG
GTCGAGG TCTAGAT ATGAGGC TAGAGAC
AGGCTTT ATCCGAT AGGCTTT GAGACAG
AGTCGAG TTAGATT ATGAGGC AGAGACA
GGCTTTA TCCGATG TTTAGAG
CGAGGCT TAGATCC TGAGGCT GAGACAG
AGTCGAG TTTAGATC ATGAGGC TTAGAGA
GAGGCTT GATCCGA GAGGCTT GAGACAG
1. Sequencing
(e.g. Solexa, 454…))
2. Hashing
Linear stretches
GATT (1x)
TGAG (9x) ATGA (8x) GATG (5x) CGAT (6x) CCGA (7x) TCCG (7x) ATCC (7x) GATC (8x) AGAT (8x)
GCTC (2x) CTCT (1x) TCTA (2x) CTAG (2x)
AGAA (1x)
TAGT (3x) AGTC (7x) GTCG (9x) TCGA (10x) CGAG (8x) GAGG (16x) AGGC (16x) GGCT (11x)
TAGA (16x) AGAG (9x) GAGA (12x) AGAC (9x) GACA (8x) ACAG (5x)
GCTT (8x) CTTT (8x) TTTA (8x) TTAG (12x)
CGAC (1x) GACG (1x) ACGC (1x)
3. Simplification of linear stretches
GATT
AGAT
GATCCGATGAG
Tips
GCTCTAG
AGAA
TAGTCGA CGAG
Bubble
GAGG AGGCT TAGA AGAGA AGACAG
GCTTTAG
CGACGC
4. Error removal
AGATCCGATGAG
TAGTCGAG GAGGCTTTAGA AGAGACAG

Euler [13]

Euler-SR is another popular short-read de novo aligner. Like Velvet, Euler-SR also uses De Bruijn graphs. Euler-SR is especially designed to handle mate-pair reads and error-prone reads. The Euler-SR algorithm can be summarized in three steps:

1. Detect accurate read prefixes and correct errors within them using frequent k-mers. Since efficiency of an alignment algorithm deteriorates as the frequency of errors increases, an effort is made to rid of as many as possible. The goal is to attain a set of near-perfect reads.
2. Construct a repeat graph on error corrected prefixes using k-mers. "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." This means that if the repeat graph of a

genome is known, it is possible to correct errors in a read by mapping the read to a path in the repeat graph and substituting the read by the path. The De Bruijn graphs tend to get three types of errors:

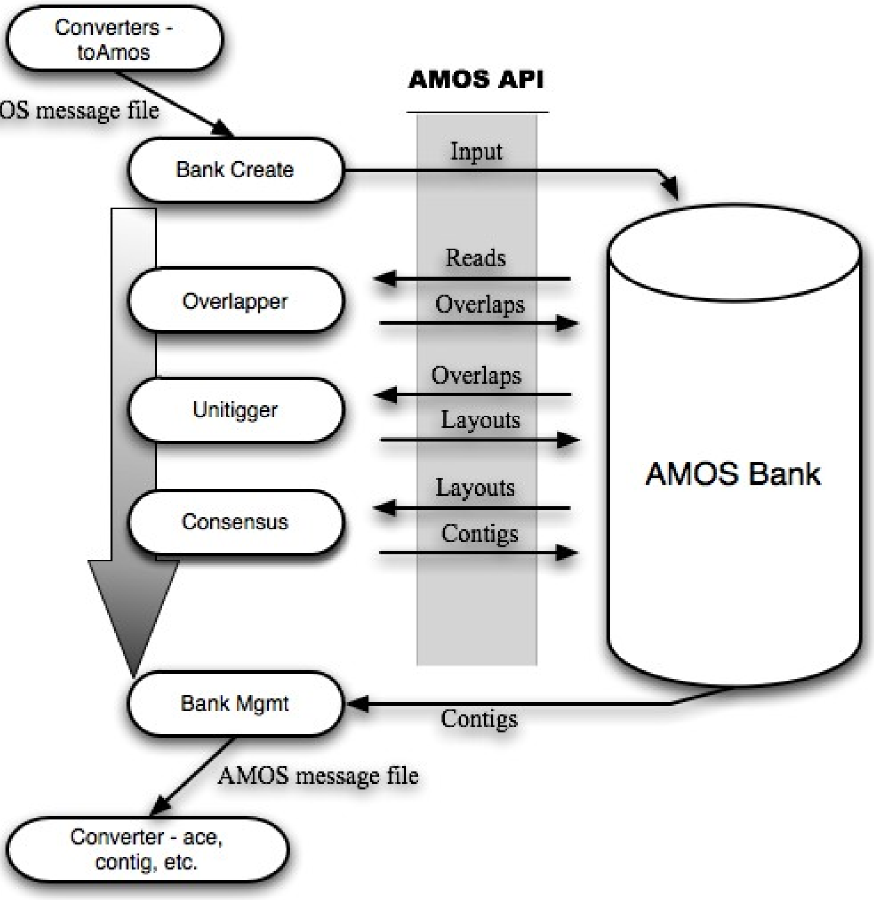
1. bulges – reads with an error or a SNP in the middle.
2. sinks – reads with errors at the end of the read.
3. chimeric reads – reads that transform the sequence in one end of the read to that of a distant part of the genome which creates a connection to an unrelated contig. All three of these error categories are removed.

3. Simplify the repeat graph after transforming mate-pairs into mate-reads. A set of mate-pairs are usually in the order of read1+GAP+read2. Euler takes this and uses the repeat graph to transform it to read1+SEQUENCE+read2.

Minimus [14]

There is a de novo aligner named Minimus. It is part of the AMOS ("A Modular, Open Source whole genome assembler") package [16]. It is better suited for smaller datasets although it can be scaled to work with large datasets as well. It is very stringent, so larger datasets tend to end up being highly fragmented. "...The reason for this fragmentation is the higher density of repeats...Eukaryotic genomes often contain high-copy repeats that disrupt the assembly process, even within the range of a BAC insert" [14]. It is generally recommended that execution be followed by other processing steps like scaffolding [17]. Minimus output generally has a low error rate, but the main selling point for Minimus is that it does not mis-assemble repetitive regions of the genome [15]. If one is using Minimus for a small dataset, it can be faster and more flexible than many of the existing tools, unless there are a high number of repetitive elements. "We compared Minimus to phrap on two median-sized assembly tasks, BAC clones and bacterial genomes, and found that Minimus is able to perform such assemblies more efficiently and more accurately than phrap, at the cost of producing smaller contigs" [14]. Interestingly, Minimus uses the traditional layout-overlap-consensus approach. However, it still uses a graphing technique to help assemble reads. The algorithm is as follows:

1. Shotgun reads are loaded into the AMOS databank (using the API they provide).
2. A hash-overlap program calculates the pairwise alignments between the input reads.
3. After the overlaps are determined, a graph is constructed by another program called the Unitigger. "The overlap graph contains a node for each shotgun read, and an edge connects two nodes if the corresponding reads overlap."
4. "The unitigger then uses several reduction steps to simplify this graph, and generate a set of unitigs"
5. Removal of substrings (reads that are completely contained inside another are removed).
6. The next step is called Transitive reduction. "For any set of three reads (A, B, and C), if the overlap between A and C can be inferred from the overlaps between reads A and B, and B and C, this overlap (i.e. the edge corresponding to this overlap) is removed from the graph."
7. Unique-join collapsing. "Every simple path in the graph (paths that contain no branches, i.e. all the nodes have in- and out-degrees equal to 1) are collapsed into a single vertex. Each such vertex represents an individual unitig."
8. Next comes the consensus stage. A full multiple alignment is constructed from the unitig reads using the approximate placement of the reads inferred from the overlap information as a guide.



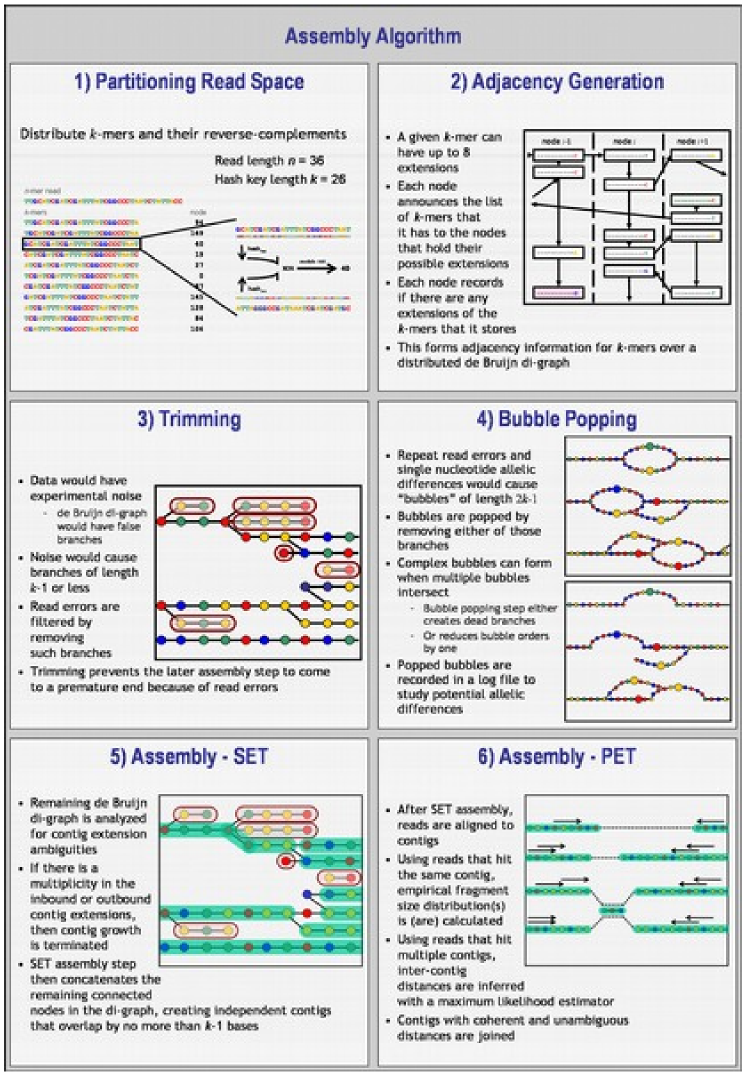
<u>ABySS</u> [5]

Another de novo aligner is called ABySS. ABySS is designed to de novo align genomes about 100Mb in size and is specifically intended for mate-pair experiments. The algorithm is as follows (summarized from a visual on the website) [5]:

1. Partition read space. "Distribute K-mers and their reverse-complements."
2. "Adjacency Generation." Each k-mer can have up to eight extensions and each node announces the list of k-mers that it has to the nodes that hold their possible extensions. Each node records if they are any extensions of the k-mers that it stores. This forms adjacency information for k-mers over a distributed De Bruijn graph.
3. "Trimming." Data sometimes has experimental noise which would show up as false branches on the De Bruijn graph. This noise would cause branches of length k-1 or less. Read errors can be filtered by removing these sort of branches. Trimming prevents the later assembly step to come to a premature stop due to read errors.
4. "Bubble Popping." As mentioned with the other methods that use de brujin graphs, read errors and SNPs cause bubbles of length 2k-1. ABySS pops these bubbles by removing either of the two branches. If a complex bubble forms, such as when multiple bubbles intersect, the bubble popping step either creates dead branches or reduces the bubbles orders by one. ABySS records these popped bubbles in a separate outfile for later examination.
5. "Assembly – SET." The De Bruijn graph is analyzed for contig extension ambiguities. "If there is a multiplicity in the inbound and outbound contig extensions, then contig growth is terminated." The SET assembly step then concatenates the remaining connected nodes in the De

Bruijn graph, creating independent contigs that overlap by no more than k-1 bases.

6. “Assembly – PET.” “After SET assembly, the reads are aligned to contigs. Using reads that hit the same contig, empirical fragment size distributions are calculated. Using reads that hit multiple contigs, inter-contig distances are inferred with a maximum likelihood estimator. Contigs with coherent and unambiguous distances are then joined.”



Assembly Algorithm

1) Partitioning Read Space

Distribute k-mers and their reverse-complements

2) Adjacency Generation

- A given k-mer can have up to 8 extensions
- Each node announces the list of k-mers that it has to the nodes that hold their possible extensions
- Each node records if there are any extensions of the k-mers that it stores
- This forms adjacency information for k-mers over a distributed de Bruijn di-graph

3) Trimming

- Data would have experimental noise
  - de Bruijn di-graph would have false branches
- Noise would cause branches of length k-1 or less
- Read errors are filtered by removing such branches
- Trimming prevents the later assembly step to come to a premature end because of read errors

4) Bubble Popping

- Repeat read errors and single nucleotide allelic differences would cause “bubbles” of length 2k-1
- Bubbles are popped by removing either of those branches
- Complex bubbles can form when multiple bubbles intersect
  - Bubble popping step either creates dead branches
  - Or reduces bubble orders by one
- Popped bubbles are recorded in a log file to study potential allelic differences

5) Assembly - SET

- Remaining de Bruijn di-graph is analyzed for contig extension ambiguities
- If there is a multiplicity in the inbound or outbound contig extensions, then contig growth is terminated
- SET assembly step then concatenates the remaining connected nodes in the di-graph, creating independent contigs that overlap by no more than k-1 bases

6) Assembly - PET

- After SET assembly, reads are aligned to contigs
- Using reads that hit the same contig, empirical fragment size distribution(s) is (are) calculated
- Using reads that hit multiple contigs, inter-contig distances are inferred with a maximum likelihood estimator
- Contigs with coherent and unambiguous distances are joined

<u>Synopsis of De Novo Alignment Algorithms for Next-gen Sequencing Data</u>

The majority of these methods all use De Bruijn graphs to connect reads. Although conceptually more complex than the layout-consensus-overlap approach, this is actually a very reasonable way of going about connecting reads since it is computationally inexpensive and errors can be easily identified and removed. ABySS is merciless in the way it eliminates bubble errors compared to Euler and Velvet as it just seems to randomly pick one to rid of. Although Velvet and ABySS can both handle mate-pair data, it seems like Euler is very much specifically designed to handle mate-pair reads. Velvet is special in that it hashes its De Bruijn graph according to word length; this may be computationally expensive but is still less computationally expensive than doing a pairwise alignment [11]. Euler and ABySS both make an effort to treat the read data before analyzing it. The Velvet algorithm does not mention anything about trimming or error correcting reads (it lets the error removal step of the De Bruijn graph clean out the bad data), and Minimus makes it a point to keep read trimming separate from the alignment—the idea is that the definition of a "bad" read depends on a number of factors and to be truly accurate read corrections must be done independently of the alignment program [14].

Discussion

Dr. Andreas Sundquist, a bioinformatics expert from Stanford University has stated, "Which alignment algorithm you should use depends on many factors: 1) the sequencing technology, 2) read length, 3) number of reads and available compute resources, 4) sensitivity/scoring requirements. Unfortunately, there is not an aligner available that would do well in every situation." [10]. Dr. Sundquist has also said, "Similar to the choice of alignment algorithm, in choosing an assembly algorithm we must consider the underlying sequencing technology, its read lengths, whether we have paired reads, the overall sequencing protocol, and the size of the genome being assembled." [10] This is a fantastic survey of the proper considerations to make when working with genomic data. To add to this comment, the size of the read lengths and the size of the genome are the most important considerations in a de novo alignment algorithm next to the sequencing platform that was used. In general, it looks as if the conventional approach is better for longer reads and the De Bruijn approach is better suited for short-read data. Although there are not any official rules, it is clear that certain de novo alignment programs are usually used to align genomes of a certain size. It seems like Minimus is better suited for smaller genomes without too many repetitive regions, Velvet and Euler are generally used for genomes of all sizes (although Euler needs to have mate-pair data and Velvet handles larger genomes especially well), and ABySS is best suited for genomes of about 100Mb in size.

The layout-overlap-consensus approach is great for reads created via a conventional sequencing method. This greedy algorithmic approach these de novo aligners take is justified because the sequence data per read is more accurate and the overlaps between reads is much larger than that of short-reads. Since only local information is considered, repetitive elements can confuse the aligner and cause mis-assemblies [18] . If configured properly, this approach can work well for short-read data as well (as shown by Minimus) [14]. However, like the rest of the programs that use the layout-overlap-consensus approach, Minimus cannot handle repetitive elements as elegantly as the programs that use the De Bruijn approach can. This is a major drawback as it is one of the chief problems posed by short-read data. In a sense, Minimus is a step behind Phrap and TIGR as it completely relies on the user to feed it "good" data where as Phrap and TIGR make an effort to eliminate "bad" reads. However, Minimus does deal with the problem of having large datasets by organizing the data into a data bank using the AMOS api.

The De Bruijn approach has its own merits in that it can be significantly cheaper (computationally) than pairwise alignments, and it is easy to identify and eliminate what could be "bad" reads. This method is specifically designed for short-reads but can also be applied to longer reads. De Bruijn graphs work great with single-read datasets, but it is clear that they can perform much better with mate-paired datasets due to the additional information the aligners can extract and use (insert sizes). The De Bruijn approach takes care of the three major short read sequencing problems mentioned earlier—it is orders of magnitude faster than comparing every read to every other read, solving the problem of having very large datasets. Pevzner et al. are in favor of abandoning the overlap-layout-consensus approach and have stated that, "EULER, in contrast to the CELERA assembler, does not mask such repeats but uses them instead as a powerful fragment assembly tool." [23]. Euler has showed us that there is an elegant way to handle repeats, solving the repetitive element problem. Euler has also shown us that it is possible to correct sequencing errors before alignment (although not all of the next-generation de novo aligners do this), solving the problem of filtering out or correcting poor reads.

Taking all of this into consideration, it seems De Bruijn graphs tackle all of the problems posed by next-generation sequencing data. The only problem it may not truly tackle is that of data quality. However, this is ultimately determined by the sequencing platform chosen and the treatment of the data thereafter.

After examining the properties of some sequencing platforms and of some de novo aligners, it does not seem like there is as big of an advantage to having longer reads as there once was. Although longer, more accurate reads make de novo alignment a much easier task no matter which sequencing method is chosen, there seem to be some very respectable short read assemblers available which provide solutions to the problems posed by next-gen sequencing. Longer reads (from next-generation sequencers) tend to have higher error rates, and this can make assembly more difficult without proper coverage. Plus, if a high-throughput dataset is mate-paired, it can overcome the shortcoming of lacking longer reads. Chaisson et al. ask, "Does read length really matter?" The finding of that study (which used mate-paired data) is that "...the assembly hardly improves after the read length exceeds 35 nt" [13]. Based on this conclusion, the HiSeq 2000 and SOLiD should be the preferred sequencing method for de novo alignments due to the high-magnitude of passed filter data and high coverage. What was once posed a challenge for the bioinformatics community (large short-read datasets) can now be viewed as an advantage.

A Suggested Approach for a Custom Short-read De Novo Aligner

As Dr. Sunquist mentioned, there is not an aligner that can do well in every situation. This being the case, bioinformatics scientists should first focus on a target genome size and then focus on writing algorithms optimizing on speed and efficiency rather than accuracy (which will come naturally through inputting good data). Dr. Sundquist has also said, "The additional longrange information for a de novo assembler is invaluable in avoiding misassemblies in repetitive regions. Algorithms that use mate-pairs will need to know the insert sizes of the read pairs, and oftentimes it's beneficial to have several different insert sizes: smaller ones to help assemble local regions, and larger ones to help disambiguate large, repetitive structures" [10]. Based on the invaluable advantages paired-end data provides, especially for de novo alignment, I would strongly suggest the input datasets be mate-paired although it would not be a requirement.

Michael Schatz et al. say, "Assemblers primarily focus on correcting errors, reconstructing unambiguous regions, and resolving short repeats. These assemblers have successfully assembled small genomes from short reads, but have had limited success scaling to larger mammalian-sized genomes, in part, because they require constructing and manipulating graphs far larger than can fit into memory" [12]. The primary cause of memory inefficiency in De Bruijn graphs is due to erroneous read data (discrepancy between bases will cause the graph to add additional branches). Looking into the future, these next-generation sequencing companies are probably going to work on producing more data with better quality as well as making the instruments more accurate. Although read quality is an important factor in de novo alignment algorithm efficiency and speed, this really depends on the sequencing method that was used.

After choosing a suitable sequencing platform, the sequence data to be aligned needs to be of as high quality as possible. It is crucial to keep the error rate as low as possible for all reads, but definitely under a 1% error rate as this seems to be the threshold where having data of any worse quality has little affect on making the alignment significantly more difficult [10]. Zhao et al. have developed a method called EDAR (An Efficient Error Detection and Removal Algorithm for Next Generation Sequencing Data) which sifts through the read data for sequencing errors and leaves only true errors [21]. The data pushed through the EDAR algorithm are significantly better—a 1% error rate can be turned into a 0.38% error rate [21]. It has been shown that this alone improves the performance of short-read de novo aligners [21]. In the past, de novo aligners have always had to trade off accuracy for speed [10]. With this approach, the trade-off between speed and accuracy may not be as large as it once was. It may be possible to align this more accurate data in less time.

Since certain de novo aligners tend to be used for certain size genomes, it would be appropriate to define what size genomes this de novo aligner is meant to handle. Aiming for the smaller genomes such as those of microbials will work in harmony with the ultimate goal of speed and efficiency. Since the De Bruijn graph will contain a magnitude of reads, it is appropriate to optimize for specificity rather than sensitivity (it is better to avoid false positive alignments rather than false negative alignments). Keeping these philosophical tenets in mind, the next step would be to create a De Bruijn graph using k-mers. The De Bruijn graph should be hashed using the k-mers. "The costliest part of constructing a de Bruijn graph consists in hashing all the reads, according to a given word length. This operation offers nonetheless an advantageous time complexity compared to a general pairwise alignment of all the sequences, especially given the high coverage depths encountered" [11]. The maximum k-mer size should be the length of the read, and the minimum k-mer size should be as shown in the equation below:

(length of the read) - floor_function[.1*(length of the read)]
*Assuming 15x-30x coverage

This formula has been set with the hope that the k-mers will be liberal enough to allow a few mismatches but strict enough to prevent the De Bruijn graph from becoming too "connective." The target read size is between 35 and 100 bases, and these numbers are the driving factor in setting the above equation. Using this narrow range of k-mers will further drive down the time complexity required to complete the graphing and hashing operations.

Due to the initial error corrections, there should theoretically be very few "sink" errors and only a few "bulge" errors. This means "chimeric" errors are the only graph errors that pose a threat to an otherwise optimized algorithm. If a reference genome is available, it might be beneficial to use it to quickly perform a local alignment on a potential chimeric error. By performing a somewhat liberal local alignment on the chimeric read and its flanking sequences (k-bases out on either side), its location can be inferred relative to the rest of the assembly (verify that the chimeric error is really a chimeric error) while at the same time accounting for SNPs in the genome. If a reference genome is not available for this comparative genomics sub-step, then this step would be replaced by an initiative to eliminate all potential chimeric errors.

The next big hurdle is dealing with the repetitive sequences. Minimus does not mis-assemble repetitive regions of the genome due to the high stringency of its algorithm—something to be strived for. Rather than including repetitive sequences with the main De Bruijn graph, making additional mini De Bruijn graphs composed of solely repetitive reads will help parallelize and streamline the graph. Relative to the main De Bruijn graph, these should be much smaller in size; and when a read must be connected to a repetitive sequence, the main De Bruijn graph will connect to a node that represents another De Bruijn graph for that repetitive element. This will preserve the topology of the graph while making the data even easier to handle computationally and ultimately concatenate.

Overall, this approach builds on the strengths of the other de novo alignment programs while avoiding their weaknesses (brought about by analyzing datasets too large or too small) by explicitly defining the characteristics of the data the program is designed for. If applicable, there is not any reason why one should not try to leverage known reference genome information. Compared to de novo sequencing, it is an order of magnitude faster to re-sequence a genome because it does not have to compare every read with every other read and find the shortest common subsequence [22]. Even during a de novo alignment, a reference genome can be an invaluable source of validation or perhaps even serve as a guide. This medley of carefully chosen optimizations results in a powerful, streamlined approach for short-read de novo alignment.

Literature

[1] Sequencing Whole Genomes, Hierarchical vs Shotgun Sequencing [Internet]; [cited 2010 Nov 27]. Available from: http://www.bio.davidson.edu/courses/genomics/method/shotgun.html

[2] Green, Phil. 1999. Documentation for PHRAP and CROSS_MATCH [Internet]; [cited 2010 Nov 26]. Available from: http://www.phrap.org/phredphrap/phrap.html

[3] Roberts, Eric. The PHRAP Algorithm [Internet]; [cited 2010 Dec 1]. Available from: http://www-cs-faculty.stanford.edu/~eroberts/courses/soco/projects/2000-01/computers-and-the-hgp/phrap.html

[4] Sutton et al. 1995. TIGR Assembler: A New Tool for Assembling Large Shotgun Sequencing Projects. Genome Science and Technology. 1(1): 9-19.

[5] BC Cancer Agency. ABySS, Assembly By Short Sequences – a de novo, parallel, paired-end sequence assembler [Internet]; [cited 2010 Nov 28]. Available from: http://www.bcgsc.ca/platform/bioinfo/software/abyss

[6] Obenrader, Sarah. The Sanger Method [Internet]; [cited 2010 Dec 2]. Available from: http://www.bio.davidson.edu/Courses/Molbio/MolStudents/spring2003/Obenrader/sanger_method_page.htm

[7] Illumina, Inc. HiSeq 2000 System [Internet]; [cited 2010 Nov 28]. Available from: http://www.illumina.com/systems/hiseq_2000.ilmn

[8] Roche. 454 System [Internet]; [cited 2010 Nov 28]. Available from: http://454.com/products-solutions/system-features.asp

[9] Life Technologies. SOLiD 4 System [Internet]; [cited 2010 Nov 28]. Available from: https://products.appliedbiosystems.com/ab/en/US/adirect/ab?cmd=catNavigate2&catID=607061&tab=DetailInfo

[10] Genome Web [Internet]. 2009. Assembly and Alignment Algorithms for Next-Gen Sequence Data Technical Guide; [cited 2010 Dec 1]. Available from: http://www.genomeweb.com/node/919228

[11] Zerbino D, Birney E. (Poster) Velvet: de novo assembly using very short reads. EMBL-EBI [Internet]; [cited 2010 Nov 28]. Available from: http://www.ebi.ac.uk/~zerbino/velvet/velvet_poster.pdf

[12] Schatz et al. 2009. Contrial: Assembly of Large Genomes using Cloud Computing. SourceForge, Contrail Wiki [Internet]; [cited 2010 Nov 25]. Available from: http://sourceforge.net/apps/mediawiki/contrail-bio/index.php?title=Contrail

[13] Chaisson et al. 2008. De novo fragment assembly with short mate-paired reads: Does the read length matter? Cold Spring Harbor Laboratory Press [Internet]. Available from: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2652199/?tool=pubmed

[14] Sommer et al. 2007. Minimus: a fast, lightweight genome assembler. BMC Bioinformatics [Internet]. [cited 2010 Nov 24]; 8:(64). Available from: http://www.biomedcentral.com/1471-2105/8/64

[15] Short Read Alignment and Assembly. Center for Bioinformatics and Computational Biology [Internet]; [cited 2010 Dec 1]. Available from: http://www.cbcb.umd.edu/research/SR-assembly.shtml

[16] AMOS. Sourceforge, AMOS Wiki [Internet]; [cited 2010 Nov 27]. Available from: http://sourceforge.net/apps/mediawiki/amos/index.php?title=AMOS

[17] Minimus. SourceForge, Minimus Wiki [Internet]; [cited 2010 Nov 27]. Available from: http://sourceforge.net/apps/mediawiki/amos/index.php?title=Minimus

[18] Genome Sequence Assembly Primer. Center for Bioinformatics and Computational Biology [Internet]; [cited 2010 Dec 1]. Available from: http://www.cbcb.umd.edu/research/assembly_primer.shtml

[19] Pacific Biosystems. PacBio RS System [Internet]; [cited 2010 Nov 28]. Available from: http://www.pacificbiosciences.com/products

[20] Metzker, Michael L. 2010. Sequencing technologies — the next generation. Nature Reviews [Internet]; [cited 2010 Nov 20]. Available from: http://www.nature.com/nrg/journal/v11/n1/abs/nrg2626.html

[21] Zhao et al. 2010. EDAR: An Efficient Error Detection and Removal Algorithm for Next Generation Sequencing Data. Journal of Computational Biology. 17(11). 1549-1560.

[22] Sequence Assembly. Wikipedia [Internet]; [cited 2010 Dec 3]. Available from: http://en.wikipedia.org/wiki/Sequence_assembly

[23] Pevzer et al. 2001. An Eulerian path approach to DNA fragment assembly. PNAS. 98(17): 9748-9753