A Parameterization Study of Short Read Assembly Using the Velvet Assembler

Alex Christopher Elliot, A. Louise Perkins, and Sumanth Yenduri

University of Southern Mississippi,
730 East Beach Blvd, Long Beach, MS 39560

Abstract. In this study, we examine approaches to the problem of assembling large, contiguous sections of genetic code from short reads generated from laboratory techniques. We explore the Eulerian Path approach in detail, utilizing a de Bruijn Graph, and demonstrate current software technologies and algorithms using a sample genome. We investigate the input parameters of Velvet and discuss their implications.

Keywords: Velvet, De Bruijn Graphs, Genetic Assembly, Euler Path.

1 Introduction

Since the discovery of the DNA double helix in 1953 [1], science has sought to fully understand the information contained within it [2]. In a macro view, understanding an organism's genome can help reveal its phylogeny and origins, while the micro view can uncover information about disease susceptibility and cure. Small sections of an individual organism's genetic fingerprint that indicate the presence or absence of a particular trait are called genetic biomarkers. These biomarkers can be used to, for example, determine relation between organisms, gauge exposure to a particular genetic toxicant, predict inherited disease, or determine an optimal treatment approach. In order to understand genetic information, one must find a way to read the information contained within DNA or RNA. Genetic sequencing techniques were first developed in the early 1970's [3]. These complex methods, including the wandering-spot technique were very labor intensive.

Fredrick Sanger [4] and Gilbert [3] independently published research in 1977 that greatly simplified the sequencing process. The Sanger method is a chain terminating technique that uses of dideoxynucleotide triphosphates (ddNTPs) to selectively terminate long strands of genetic material [5]. In this method, single stranded, denatured DNA source material is cloned and separated into four separate solutions containing one of ddATP, ddTTP, ddCTP, or ddGTP each. The dideoxynucleotides terminate the multiple copies of the DNA strand at each location of the target base, resulting in strands that begin at the origin and have length of the base location index. The output of the four dideoxynucleotide solutions is then separated by gel electrophoresis or flourescent absporption if dyes were used. The result is an index location of each base in the source DNA to a one-base resolution. Sanger sequencing

S.C. Satapathy et al. (Eds.): Proceedings of the InConINDIA 2012, AISC 132, pp. 179–186.
springerlink.com

generates long reads of about one thousand bases, but requires weeks to months of costly laboratory time [6]. This technique is susceptible to cloning error [7], as parts of the cloning vector may enter the resulting sequence.

An alternative to Sanger sequencing, pyrosequencing was developed by Nyrén and Ronaghi at the Royal Institute of Technology in Stockholm in 1998 [8]. This method involves iterative addition of bases in an enzymatic solution of Sulfurylase, Luciferase and Apryase. As each base bonds to the source material, a measurable amount of light is released per base. Repeat bases yield proportionately more light. After each base is introduced and bound, an enzyme is added to remove all unused bases before the next base is added.

Pyrosequencing results in short length reads with an upper limit of approximately 500 bases, however commercial implementations are constantly increasing the maximum read length. Pyrosequencing is also less expensive to perform than traditional techniques, with companies such as 454 Life Sciences producing all-in-one units [9]. This technique can produce approximately 25Mbp/4hr [10]. As this technique does not require traditional cloning, it is not susceptible to vector cloning error. It is, however, potentially less accurate in homopolar regions with a long series of repeating bases. Pyrosequencing techniques normally result in many copies of overlapping short reads. After the laboratory work is complete, the reads must then be assembled into a representation of the source sequence. Although various solutions exist for this problem, all require some amount of a priori assumption and reliance on yet to be fully verified metrics. The challenge, algorithmically, is to determine how each of the reads fits into the larger sequence. Information to support the selection amongst candidate solutions can come from existing, reference genomes, statistical models, or sheer read coverage. Once sequenced, data can be added to large, publically accessible genome databases such as NCBI [11] or GenomeNet [12]. The NCBI Basic Local Alignment Search Tool (BLAST) can be used to find regions of local similarity between sequences. The program [13] compares nucleotide or protein sequences to sequence databases and calculates the statistical significance of matches. BLAST can be used to infer functional and evolutionary relationships between sequences as well as help identify members of gene families.

2 The EULERIAN Approach

In this section, we describe the application of the EULERIAN path to short read assembly and its differences as compared to earlier methods. We discuss one available implementation – Velvet, and provide insight into its algorithm. Older approaches to the problem of read assembly were designed around the assembly of few, long reads. Many available programs utilized the “overlap-layout-consensus” paradigm which tests each possible read pair combination to determine the best matches. Each read is represented as a node, and each detected overlap is drawn as an arc between the overlapping nodes. Once matches are scored, the assembly is generated based on overlap scoring. Unfortunately, determining the layout leads to the NP-complete Hamiltonian Path Problem [14]. The difficulty of the Hamiltonian Path Problem is exacerbated when attempting to operate on an increased number of reads.

Pevzner proposed an alternative solution to the read assembly problem for sequencing by hybridization [15]. By making use of the de Bruijn Graph, he reduced read assembly to a solvable Eulerian Path Problem. Further work by Idury and Waterman [16] applied the Eulerian path to short fragment assembly by treating short fragment assembly as Sequencing by Hybridization problem. Pevzner, Tang and Waterman refined their Eulerian graph techniques in 2001 to include methods of error correction and repeat handling in data [15]. A de -Bruin graph is a directed, n-dimensional graph of m symbols that represents overlaps between sequences of symbols. In graph theory, an n-dimensional de Bruijn graph of m symbols is a directed graph representing overlaps between sequences of symbols. It has mn vertices, consisting of all possible length-n sequences of the given symbols (the same symbol may appear multiple times in a sequence). If one of the vertices can be expressed by shifting all symbols by one place to the left and adding a new symbol at the end of another vertex, then the latter has a directed edge to the former vertex. Although de Bruijn graphs are named after Nicolaas Govert de Bruijn, they were discovered independently both by de Bruijn (1946) and I. J. Good (1946). Much earlier, Flye Sainte-Marie (1894) implicitly used their properties.

Zerbino and Birney released a set of algorithms called "Velvet" [17] to manipulate de Bruijn graphs for genomic sequence assembly. In their implementation of the graph, a k-mer is defined as a substring of length k, extracted from a read. Each node contains a series of overlapping k-mers, with each overlap having length k-1 bases. Each node is attached to another, "mirror" node which contains the reverse series of k-mers. These mirror nodes take into account the complementary nature of genetic material. Nodes whose last k-mer overlaps with the first k-mer of another node are connected by a directed arc. (Fig 2.1) The assembled contiguous sequence or "contig" is represented by a traversal from the first k-mer of the first node through connected arcs to each other node.

Once the input reads have been hashed into k-mers and assembled into nodes and arcs, the resulting graph must be simplified and cleared of errors. Velvet simplifies the graph by combining adjacent nodes with only one incoming and outgoing arc. This reduces the node count to only nodes with multiple arcs. Error correction is performed by eliminating "tips" and "bulges." A "tip" is defined as a chain of nodes connected at only one end, and Velvet removes tips that do not meet minimum length and coverage requirements. A "bulge" is a redundant path that starts and ends at the same nodes as other paths with similar sequences. Velvet again employs a length threshold and simple sequence identity to condense or merge a bubble. Velvet is thus composed of four stages: hashing the reads into k-mers, constructing the de Bruijn graph, correcting errors, and resolving repeats. The first stage, graph construction, is memory intensive. The time complexity of error correction depends mainly on number of nodes in the graph, which is a result of read coverage, error rate, and number of repeats in the source material. The graph search used during error detection and correction employs the Dijkstra algorithm which has a time complexity of O(NlogN) when implimented with a Fibonacci heap [18]. Repeat resolution also depends on the number of nodes present in the graph and the average length of those nodes.

3 Methods

To illustrate the operation of Velvet, we chose a specific, active coding gene of Escherichia coli str. K-12 substr. MG1655. This gene, NP_415534, codes for the enzyme proline dehydrogenase/pyrroline-5-carboxylate dehydrogenase which functions as a fused DNA-binding transcriptional regulator [12]. The E. coli genome has been extensively studied and fully sequenced [19] allowing for comparison of our assembly results with established sequence data. The NP_415534 gene sequence was obtained from GenomeNet [12] in its full form as an ASCII formatted fasta file [11]. This reference gene contains a total of 3963 ordered nucleotides.

From the reference file, we used the read simulation function of MetaSim [20] to output two sets of simulated reads. The first set represents an "exact" or reference set in which, 5000 reads were taken directly from the source gene without introduced error. The output reads have a normal distribution across the source gene and an average read length of 997.87 base pairs. To illustrate real world data, we also generated a set of reads modeling the read output of the LifeSciences 454 sequencer [9]. These 5000 simulated reads contained 29890 insertions and 7321 deletions. Each insertion is the addition of an extra base not present in the original material. Each deletion is the removal of one base from the original material. Locations of these induced errors are based on characteristics of pyrosequencing such difficulties accurately reading homopolar regions. Average Read Length was 258.21 base pairs. Each of the simulated read sets were run through the Velvet Assembler using varying values of k-mer length (k), expected coverage (exp_cov) and coverage cutoff (cv_cut). Automation of parameter variation and report generation was assisted by the standardized velvet assembly report script project [21]. Expected coverage is the expected frequency of repeats of each source base. This is a function of the source material and the depth at which the sequencing was performed.

Table 1 shows the parameter permutations used and their results for the simulated 454 reads. "kmer" is the selection of k or kmer length. "cvCut" is coverage cutoff, a threshold used to determine if a node in the constructed de Bruijn graph should be included as part of the final assembly. "exp," expected coverage, is the expected frequency of repeats of each source base. "ctgs" is the number of contigs. "asmLg," "mean," and "max" refer to the total length, mean, and maximum length of all assembled contigs respectively. "N50" refers to the length of the shortest contig in an assembly such that the sum of contigs of equal length or longer is at least 50% of the total length of all contigs. "1k" is the number of contigs over 1000 bases long. "tiles" is the number of reads that are used in an assembly. "rdPc" is percentage of input reads used in the assembly. Lower frequency nodes with coverage below the coverage cutoff value are suspected to be erroneous and are subsequently removed during graph error correction, especially during tip and bulge removal. This threshold specifies how many read k-mers must overlap for each contig kmer. The number of kmers per read is a function of read length L and k-mer length K (L-K+1) [21]. AMOS files of selected final assemblies were generated with velvet and opened for analysis with Hawkeye [22].

Table 1. Assembly Parameter Permutations

kmer	cvCut	exp	ctgs	asmLg	N50	mean	lk	max	tiles	rdPc
21	2	4	4253	114471	26	26	0	56	1394	27.88
21	3	6	2603	70464	26	27	0	44	2236	44.72
21	4	8	1484	40732	27	27	0	70	2638	52.76
21	6	12	107	3896	36	36	0	144	3375	67.50
21	10	20	60	4007	91	66	0	381	4997	99.94
21	12	24	56	4366	110	77	0	411	5000	100.00
23	2	4	4337	128149	29	29	0	60	1378	27.56
23	3	6	2638	78246	28	29	0	66	2914	58.28
23	4	8	692	20845	29	30	0	57	1438	28.76
23	6	12	212	7328	35	34	0	119	3065	61.30
23	10	20	65	4453	99	68	0	311	4999	99.98
23	12	24	46	4099	133	89	0	344	4998	99.96
27	2	4	3989	137723	33	34	0	82	1655	33.10
27	3	6	2418	83130	32	34	0	62	1588	31.76
27	4	8	1067	37296	34	34	0	81	2490	49.80
27	6	12	233	9410	41	40	0	130	3637	72.74
27	10	20	47	4250	136	90	0	559	4999	99.98
27	12	24	44	4331	145	98	0	594	5000	100.00
31	2	4	3804	150919	38	39	0	81	1118	22.36
31	3	6	1251	49497	37	39	0	73	596	11.92
31	4	8	383	15473	39	40	0	88	1053	21.06
31	6	12	134	7062	54	52	0	195	4228	84.56
31	10	20	22	3605	266	163	0	497	4975	99.50
31	12	24	22	3790	273	172	0	491	5000	100.00

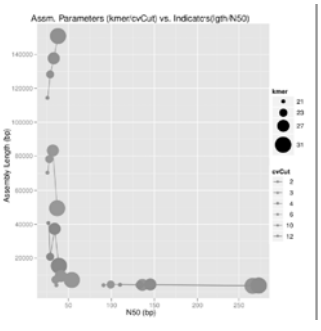

Fig. 1. Assembly Parameters (kmer/cvCut) vs. Indicators (lgth/N50). This scatterplot illustrates the effect of kmer length and coverage cutoff on N50 and assembly length. N50 refers to the length of the shortest contig in an assemblysuch that the sum of contigs of equal length or longer is at least 50% of the totallength of all contigs. Here we see a logarithmic distribution where higher cvCut values generate larger contigs.

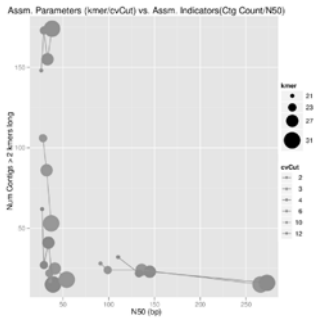

Fig. 2. Assembly Parameters (kmer/cvCut) vs. Assembly Indicators (Ctg Count/N50). This plot shows the influence of kmer length and cvCut on the number of contigs produced with greater than 2*k length. Again, we see a somewhat logarithmic function with higher cvCut and higher kmers producer longer and fewer isolated contigs.

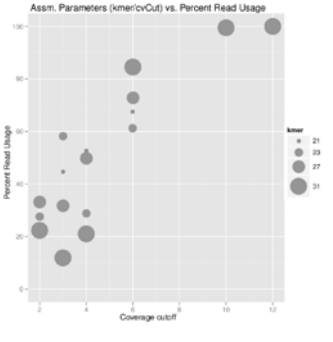

Fig. 3. Assembly Parameters (kmer/cvCut) vs. Percent Read Usage. This figure compares kmer and cvCut to the percent read usage. With sufficiently high k, read utilization increases with coverage cutoff.

The results of these assemblies were then compared back to the original, reference gene sequence using BLASTN [13]. BLASTN outputs a percent identity which shows the similarity to the reference sequence as well as a base by base alignment which shows direct matches, deletions, insertions and substitutions for each assembled node.

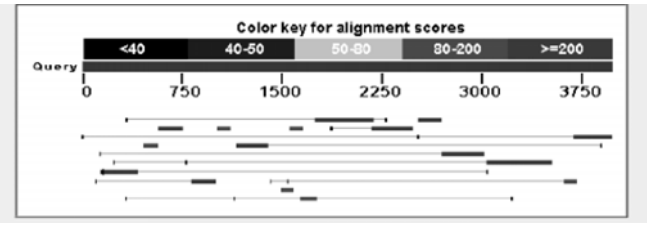

Fig. 4. BLAST Map for Simulated 454 Reads (k = 31, coverage cutoff = 12, expected coverage = 24). BLAST maps the 16 resultant contigs of the 454 simulated reads at k-mer length 31, expected coverage 24, and coverage cutoff 12. The level of similarity to the reference gene is shown by the color, with red being the best quality. Contigs appear to high quality and well distributed.

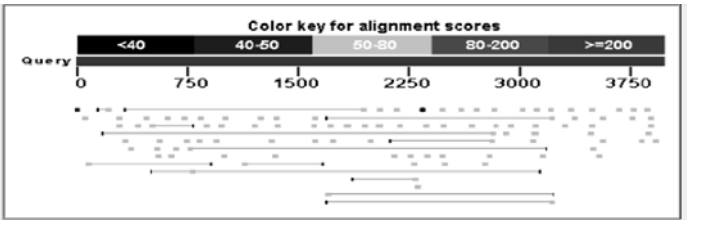

Fig. 5. BLAST Contig Scoring for Simulated 454 Reads (k = 21, coverage cutoff = 2, expected coverage = 4). This assembly resulted in 4253 nodes and n50 of 16. BLAST maps the resultant contigs above a scoring threshold. The level of similarity to the reference gene is shown by the color, with red being the best quality. Contigs appear to be very small with medium to poor quality, yet well distributed.

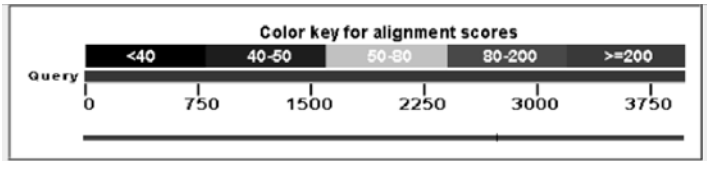

Fig. 6. BLAST Contig Scoring for Simulated "Exact" Reads (k = 21, coverage cutoff = 2, expected coverage = 4). This assembly maps the single resultant contig of the simulated "exact" reads at k-mer length 21, expected coverage 4, and coverage cutoff 2. The assembled contig achieved 100% reference gene coverage at 100% identity with a length of 3963 bases.

4 Conclusions

Read assembly remains an inexact science, relying heavily on statistical modeling and inference for error correction and graph simplification. Our synthetic assembly

experiment demonstrates how heavily parameter selection influences final assembly, thus consideration must be made when designing an experiment and performing the assembly. The value of k depends primarily on the nature of the source genome, particularly the length and abundance of repeats. With sufficiently high k, read utilization and resultant contig length increases with coverage cutoff, due to the removal of lower coverage nodes, however this elimination can lead to mis-assemblies. A delicate balance exists between easing coverage limits to increase final assembled contig length and a reduction in accuracy. Some experiments, such as preliminary genome sequencing may seek wider coverage and fewer but longer nodes at the expense of 100% accuracy of individual bases, whereas small target sequencing of short gene segments may obtain the higher accuracy required by increasing read coverage. As the algorithms continue to mature, research into the automated choice of parameters will assist scientists when faced with the challenge of read assembly. Obtaining and integrating the various scripts and applications was a chore, as each had its own set of dependencies and special setup instructions. Velvet assembly and the associated tools would benefit from a cloud implementation, similar to that of NCBI's BLAST to provide a full suite of assembly tools with minimal or no configuration. Further efforts to understand the parameterization of short read assembly using Velvet should expand both the source data and selected parameter value set, possibly to include eukaryotic data. A more detailed study of k-mer length selection could also include recursive scanning of a reference genome for maximum repeat length and a priori comparison to the genomes of similar organisms. Continued effort to understand and evaluate the decisions used when simplifying or error correcting the de Bruijn graph will lead to higher quality assemblies and serve to unify the field. This includes statistical decision making as well as reference to biological markers and archived genomic data.

References

[1] Watson, J.D., Francis, H.C.: A Structure for DNA. Nature 171, 737–738 (1953)
[2] Watson, J.D., Francis, H.C.: Genetical Implications of the structure of Deoxyribonucleic Acid. Nature 171, 964–967 (1953)
[3] Gilbert, W., Maxam, A.: The nucleotide seq. of the lac operator. Proc. Natl. Acad. Sci. U.S.A 12(70), 3581–3584 (1973)
[4] Sanger, F., Nicklen, S., Coulson, R.A.: DNA sequencing with chain-terminating inhibitors. Proc. Natl. Acad. Sci. U.S.A 12(74), 5463–5467 (1997)
[5] Tamarin, R.H.: Principles of Genetics, 4th edn. Wm. C. Brown Publishers (1993)
[6] Ewing, B., Phil, G.: Base-Calling of Automated Sequencer Traces UsingPhred. II. Error Probabilities. Genome Res., 186–194 (1998)
[7] Ewing, B., et al.: Base-Calling of Automated Sequencer Traces UsingPhred. I. Accuracy Assessment. Genome Res. 8, 175–185 (1998)
[8] Ronaghi, M., Uhlén, M., Nyrén, P.: A sequencing method based on real-time pyrophosphate. Science 363, 365 (1998)
[9] Margulies, M., et al.: Genome Sequencing in Open Microfabricated High Density Picoliter Reactors. Nature 437, 376–380 (2003)
[10] Altschul, S.F., et al.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acid Res. 25, 3389–3402 (1997)

[11] Linz, P.: An Intro. to Formal Lang. & Automata, 4th edn. Jones & Bartlett, Boston (2006)
[12] Pevzner, P.A.: 1-Tuple DNA sequencing: computer analysis. J. Biomol. Struct. Dyn. 7, 63–73 (1989)
[13] Ramana, M.I., Michael, W.S.: A New Algorithm for DNA Sequence Assembly. Journal of Computational Biology 2(2), 291–306 (1995)
[14] Zerbino, D.R., Ewan, B.: Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. Genome Research 18, 821–829 (2008)
[15] Gross, J.L., Yellen, J.: Handbook of graph theory. CRC Press, Boca Raton (2004); 69 DRAFT 04/02/2010 AE LLC
[16] Blattner, F.R.: The complete genome sequence of E. coli K-12. Sci., 1453–1462 (1997)
[17] Richter, D.C., et al.: MetaSim—A Sequencing Simulator for Genomics and Metagenomics. PLoS ONE 3(1), e3373 (2008)
[18] Schatz, M.C., et al.: Hawkeye: an interactive visual analytics tool for genome assemblies. Genome Biology 8, R34 (2008)
[19] NCBI. FASTA format description, http://www.ncbi.nlm.nih.gov/blast/fasta.shtml
[20] Kyoto University Bioinformatics Center. GenomeNet (March 22 (2010), http://www.genome.jp
[21] Leipzig: Standardized-velvet-assembly-report - Project Hosting on Google Code (March 22, 2010), http://code.google.com/p/standardized-velvet-assembly-report
[22] Roche Diagnostics Co. Products & Solutions - Syetem Benefits: 454 Life Sciences, a Roche Company (March 22, 2010), http://454.com/products-solutions/system-benefits.asp