An Improved Maximum Likelihood Formulation for Accurate Genome Assembly

Aditya Varma, Abhiram Ranade and Srinivas Aluru
Dept. of Computer Science and Engineering
Indian Institute of Technology Bombay, Mumbai, India
{*ranade,aluru*}@*cse.iitb.ac.in*

***Abstract*—We present improvements to the recently proposed maximum likelihood method for genome assembly. We formulate the problem as one of direct convex optimization, and achieve the following improvements: Our method does not require identical read lengths and can deal with reads of varying lengths. We eliminate the requirement to a priori know a stringent estimate of the length of the genome or the need to use further expectation minimization to predict the most likely length. Instead, we explicitly incorporate the uncertainty in the length estimate by a range parameter without affecting the convexity required for the optimization. Results indicate that our method can generate accurate estimates of repeat counts and produces fewer and much longer contigs. These results mark a further advancement of maximum likelihood genome assembly and the potential of this approach in building future genome assemblers.**

***Keywords*-genome assembly; maximum likelihood; next-gen sequencing.**

I. Introduction

Assembly algorithms that reconstruct a genome from a large number of random shotgun DNA reads have been a staple of genome sequencing projects for over two decades. The importance of genome sequencing, and the many genome sequencing projects undertaken, have spurred considerable research in assembly algorithms. The dominant assembly paradigm is perhaps *overlap-layout-consensus*, in which overlaps between pairwise DNA reads are considered strong indicators of genomic co-location. Assembly is accomplished by processing such overlaps, often in a greedy heuristic manner based on a measure of the significance of such overlaps. Examples of assemblers that follow the overlap-layout-consensus paradigm include Arachne [1], Atlas [2], CAP3 [3], Celera Assembler [4], GigAssembler [5], PCAP [6], Phusion [7] and TIGR Assembler [8]. Because genomes have *repeats*, read overlaps may arise even in the absence of genomic co-location. The technique of paired-end shotgun sequencing, whereby reads are obtained from both ends of a DNA fragment whose length provides an approximate distance constraint can be used to resolve ambiguities and erroneous assemblies resulting from repeats.

A more elegant approach is taken in the class of *graph-based* assemblers, initially pioneered by Idury and Waterman [9] and subsequently extended by Pevzner into the Euler assembler [10]. In this formulation, reads are represented as paths in a *de Bruijn graph* of all k-mers from all reads. The goal is to find a superpath of these paths which corresponds to the reconstructed genome. In some sense, this method takes a global perspective of the assembly problem. However, it has not seen much use in large genome sequencing efforts primarily due to its memory-intensive nature. An effort to mitigate this is the *string graph formulation* due to Myers [11], which may be viewed as a space efficient version of the de Bruijn graph representation.

With the advent of next generation sequencing technology that permits massively parallel DNA sequencing while significantly reducing the length of individual reads (50-125 bp for Illumina Genome Analyzer vs. 750-1000 bp for Sanger), there is renewed interest in graph-based genome assembly methods. Short reads significantly worsen the reliability of overlaps as indicators of genomic co-location, affecting the quality of overlap-layout-consensus based assemblers. As a result, a number of de Bruijn graph based genome assembly programs have been developed to address short read assembly. While demonstration of many of these have been limited to bacterial genomes due to memory constraints [12]–[15], recent development of parallel methods is promising to scale to larger genomes [16]–[19].

In recent work, Medvedev and Brudno [20] proposed a maximum likelihood formulation of the genome assembly problem, where the goal is to infer a genome that is the most likely source of the given set of reads. They demonstrate that maximum likelihood formulation can achieve high quality reconstruction. While the sizes of the contigs they achieve are comparable with other assemblers for short reads, they fall short of what is achievable when assembling the longer Sanger reads [20]. As the experiments are conducted with simulated, error-free reads, it is likely the contig sizes will be further reduced in practice. Nevertheless, this is a significant result because the likelihood formulation resulted in accurate copy count prediction for repeats and resulted in accurate reconstruction of the genome. Improving assembly quality while also producing fewer and longer contigs is important from the perspective of significantly reducing finishing costs.

In this paper, we further advance maximum likelihood genome assembly by formulating the problem as one of direct convex optimization. Building upon the work of Myers [11], we formulate the maximum likelihood approach on the overlap-based string graph and demonstrate its applicability to reads of varying lengths, while the work [20] uses

essentially fixed size reads in analysis as well as experiments. In addition, our method requires knowledge of only approximate rather than the exact genome length as in [20]. Experimental results show that our method gives highly accurate genomic reconstruction and significantly longer contigs. For example, we were able to infer a 1 Mbp stretch of the *E. Coli* genome as 19 contigs for a 10x coverage of 500 bp reads, and as 17 contigs for a 50x coverage of 100 bp reads. While these results should be viewed in the context of simulated error-free read based experiments, they were achieved without taking advantage of mate pair constraints. We conclude that building of practical assemblers based on maximum likelihood principles is worthy of investigation.

II. Methods

We assume that the input to the assembly process is a set of reads $R = \{r_1, r_2, \ldots, r_n\}$, of possibly different lengths. They are assumed to come from the same strand of the DNA. While in general this cannot be assured, we ignore this issue for simplicity and note that it can be handled as in [11], [20].

¿From the reads, we construct the fragment assembly string graph essentially following Myers [11]. This graph is constructed so that each tour of it which visits every vertex at least once corresponds to a feasible assembly. A convex optimization is then formulated, which ideally would identify a single tour and hence a single assembly. However, in the presence of repeats longer than the reads lengths, we can only identify reasonably long contigs.

A. Fragment Assembly String Graph

Following Myers [11], we construct the string graph by creating the read overlap graph and effecting a series of transformations on the overlap graph. Suppose that τ is the chosen so that the probability of an overlap of length τ between random strings is exceedingly low. Then the (τ) overlap graph for a read set R is a directed, weighted (called length in what follows), labelled graph defined as follows.

Vertices: For each read r_i that is not contained in any other read[1] we construct one vertex. We will refer to this vertex as vertex i.

Edges: If there exits an overlap of length at least τ between a suffix of r_i and a prefix of r_j, we construct a directed edge from vertex i to vertex j, i.e. edge (i, j).

Labels and lengths: Suppose (i, j) is an edge. Let r_j be a concatenation of string s followed by a string l, where s specifies the prefix overlapping with r_i. Then the edge (i, j) is labelled l, and is assigned length $|l|$, i.e. the length of the string l.

It will be noted that the overlap graph almost has the property we require. Consider a tour which visits every vertex of the overlap graph at least once. The label of an edge (i, j) is simply the sequence that takes us from the end of read r_i to the end of read r_j. By concatenating the labels along the tour, we indeed get a string which contains all reads except the one corresponding to the starting vertex (tackled in Section II-D). The tour need not be simple, i.e. the tour may pass through certain vertices/edges more than once. Thus assemblies with repeats are also represented. However we can clean up the graph considerably by repeatedly performing the following two operations.

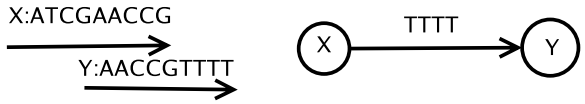

Figure 1. Representing read overlaps as edges in the string graph.

Transitive Reduction: Consider three vertices i, j and k. If there are edges (i, j), (i, k) and (j, k) in the graph, then edge (i, k) is removed. Note that the label of edge (i, k) is given by the concatenation of labels of (i, k) and (j, k). So from the perspective of a tour, the edge (i, k) is redundant since the same sequence can be spelt by taking the two edges (i, j) and (j, k). Transitive reduction helps because it typically reduces the number of edges in the overlap graph drastically. On general graphs, transitive reduction is an expensive operation. However, Myers [11] notes that an overlap graph can be transitively reduced in $O(|E|)$ time, aided by the labels. First we have the property that only an edge with a shorter label can lead to the removal of another edge. Second, the labels guide the search for which edge to reduce.

Collapse of simple paths: The transitively reduced overlap graph too may contain more vertices and edges than are necessary. Suppose we have a path in which all internal vertices have in-degree and out-degree of one each. We can collapse paths that contain such vertices into a composite edge whose label is the concatenation of the edges thus collapsed. E.g. A path $\{v_1, v_2, \ldots, v_m\}$ where in-degree(v_i) = out-degree(v_i) = 1 for $2 \leq i \leq m-1$ can be collapsed into a composite edge (v_1, v_m). While in general every edge need not be present in the assembly tour, edges that arise from collapsing paths must be present since otherwise the reconstruction will not contain the reads corresponding to the vertices in the collapsed paths. Thus we must distinguish such edges for future processing.

It should be clear that the operations described above leave invariant the tours possible in the graph, and hence the set of feasible assemblies. We next discuss how to narrow down the set of acceptible tours.

B. Maximum likelihood

The resulting graph after transitive reduction and collapse of simple paths is the (fragment assembly) String Graph. If the target genome was **covered**[2], then there exists some walk

[1]However, the contained vertices are not completely ignored; they will figure in the subsequent processing, see Section II-D

[2]If every subsequence of the genome of length $\tau + 1$ is in some read.

in the String graph which will recover the target genome. This walk must necessarily visit all vertices. It may or may not visit all edges and may visit some edges multiple times. The edges that are traversed more than once correspond to repeats. The key question to answer here is, how many times will each edge be traversed? Our approach to this problem builds up on the *maximum likelihood* approach of Medvedev and Brudno [20].

In principle, the idea is to simply consider all possible genomes which contain all the reads that have been observed, and pick the one which has the greatest probability of generating the observed read set. For this we need a model that specifies how a read set is generated from a candidate genome G. Let L_G denote the length of G and L_i the length of read r_i. In our model each read r_i is produced by running the following steps: (a) a length L_i is chosen from a length distribution which is fixed independent of the genome, (b) the starting position is chosen for the read uniformly from $1 \ldots L_G - L_i$. The sequence of bases from the chosen position of the chosen length then becomes the read r_i. Suppose a read r_i appears δ_i times in a candidate genome G. Then since the starting point is picked at random, the probability of r_i being generated is simply $P(L_i)\delta_i/(L_G - L_i)$, where $P(L_i)$ denotes the probability of selecting a length L_i for the read from the length distribution. The probability of generating the set R in the sequence $r_1, \ldots, r_n$ is simply $\prod_i P(L_i)\delta_i/(L_G - L_i)$. However, the same read set R can be generated through other permutations, e.g. if all reads are distinct, then the probability of observing R is $n!$ times the probability above. In general the exact probability is $\alpha \prod_i P(L_i)\delta_i/(L_G - L_i)$, where α depends upon the read set R but not on the genome G. Assuming $L_G \gg L_i$ we may write $L_G - L_i \approx L_G$, and observing that $\prod_i P(L_i)$ also does not depend upon G, we may write the probability P_G of generating R from G as:

$$P_G = \alpha' \prod_{i=1}^{n} \frac{\delta_i}{L_G}$$

where $\alpha' = \alpha \prod_i P(L_i)$ is a constant depending on R (and the length distribution) but not on G. Our goal is to find G that maximizes P_G, or equivalently $\prod_i \delta_i/L_G$. The maximization must satisfy certain constraints, which we describe next.

As noted, it must be possible to obtain G by a walk over string graph in which each vertex is visited at least once. Suppose that during this walk edge (i, j) is traversed x_{ij} times. Since i is visited on each traversal of (i, j) we must have $\delta_i = \sum_j x_{ij}$. Also suppose that l_{ij} is the length of edge (i, j) in the string graph. Each edge contributes $l_{ij}x_{ij}$ to the genome length and so we must have $L_G = \sum_{i,j} l_{ij}x_{ij}$. Thus using (V, E) to denote the string graph, and noting that l_{ij} are known, our problem is:

$$\text{maximize} \quad \prod_{i=1}^{n} \frac{\delta_i}{L_G} \tag{1}$$

$$\begin{aligned}
\text{such that} \qquad L_G &= \sum_{i,j} l_{ij}x_{ij} && (2)\\
\forall i \in V \ \delta_i &= \sum_j x_{ij} && (3)\\
\forall i \in V \ \sum_j x_{ij} &= \sum_j x_{ji} && (4)\\
\forall (i,j) \in E \ x_{ij} &\geq 0 && (5)
\end{aligned}$$

In this equation 4 simply asserts that the number of times the walk enters a node i is the same as the number of times it exits i.

The constraints enumerated above are linear but the objective function is not. In fact the objective function is not even concave, leaving open the possibility of multiple maxima and a difficult optimization problem.

C. An engineering approximation

The key observation is that it is often possible to get an estimate Q to the length L_G of the genome without doing an assembly. The estimate is not accurate, but has some error ϵ, (say 10 %). Thus we may write $L_G = Q(1+\epsilon)$. For small ϵ we can approximate $L_G \approx Qe^{\epsilon}$. Our objective then becomes $\prod_i \delta_i/L_G = Q^{-n}e^{-n\epsilon}\prod_i \delta_i$. Since Q is fixed it suffices to maximize $e^{-n\epsilon}\prod_i \delta_i$, or alternatively its logarithm:

$$\text{maximize} \sum_{i=1}^{n} \ln \delta_i - n\epsilon \tag{6}$$

$$\begin{aligned}
\text{such that} \qquad Q(1+\epsilon) &= \sum_{i,j} l_{ij}x_{ij} && (7)\\
\forall i \in V \ \delta_i &= \sum_j x_{ij} && (8)\\
\forall i \in V \ \sum_j x_{ij} &= \sum_j x_{ji} && (9)\\
\forall (i,j) \in E \ x_{ij} &\geq 0 && (10)\\
-0.1 \leq \ \epsilon &\leq 0.1 && (11)
\end{aligned}$$

Q is a constant, and the variables δ_i may be eliminated using equation 8. But now it is seen that the objective is concave in x_{ij} and linear in ϵ, and the constraints remain linear in the variables (x_{ij} and ϵ). So a unique maximum exists which can be found by convex optimization techniques.

D. Putting It All Together

We first discuss how contained reads and the reads eliminated during collapse of simple paths are handled. Consider a read r_a removed because it is contained in a read r_b which is not contained in any other read. Then r_a is included in the calculation in equation 6, with $\delta_a = \delta_b$. Contained reads do not affect any other equation. Likewise, every read r_c that is

contained in an edge (c, d) in the string graph is included, with $\delta_a = x_{cd}$. Since we increased the length of edge (c, d) to include the length of all the removed reads, these reads do not participate in equation 7.

Finally, to allow the reconstruction to consist of multiple contigs rather than just one, an additional vertex s is included. This has an edge to and from every uncollapsed vertex. These edges appear in the balance constraint (equation 9) for every other vertex u, but a balance constraint is not enforced for s. This vertex does not figure in the calculations in the remaining equations. If after solving we have $x_{su} > 0$, it indicates the start of a contig from u. Likewise $x_{vs} > 0$ indicates the end of a contig at v.

After these modifications, we formulate the (convex) optimization problem as discussed, and solve using the tool MOSEK (www.mosek.com). The solution gives us the values of the traversal count variables x_{ij}.

E. Annotated string graph

We first use the values x_{ij} generated to annotate the string graph. Interestingly it turns out that many of the traversal counts are zero. This happens if the genome contains repeats of length shorter than the read length but larger than τ, the minimum overlap length. We then remove edges whose traversal counts are zero. This can produce vertices whose outdegree is one – we collapse such edges into a single long path wherever possible. While collapsing these edges, their labels are concatenated to produce the label for the resultant edge. This new graph, together with the traversal counts (all of which are strictly greater than zero because we pruned the others), is called the *annotated string graph*.

F. What we report

The annotated string graph represents what our algorithm can say about the genome. Specifically, while we know that the genome must be a walk on this graph which passes through the edges as many times as specified, we do not know how precisely which of the outgoing edges to take next when it comes to a vertex. The algorithm can however firmly conclude that each edge of the graph must be present somewhere in the genome, and we report each edge as a *contig*. Since our experiments are with reads generated from known genomes, we check whether our reported contigs are present in the genome. Note that with mate pair information that is commonly available in genome sequencing projects, our assemblies can be further improved.

III. Results and Discussion

We conducted experiments on the following genomic segments: 1) Genome sequence of *E. Coli* 0157:H7 of length 1,000,080 bp (NT_034398.4), and 2) Human *Y* chromosome genomic contig of length 581,282 bp. The former is chosen because the *E. Coli* genome is widely used as a benchmark in many genome sequencing studies. We used the latter to conduct experiments on a genomic segment with higher repeat content. Each data set we tested consists of a read length and desired coverage. To test various read lengths, we experimented with 500 bp, 250 bp, 100 bp, and 50 bp reads. The coverage varied from 10x for 500 bp reads to 50x for the shorter reads. The former is seen as reasonable coverage for Sanger reads and the latter is on the low end of expected coverage with next-gen short read sequencing. The number of reads is computed based on the coverage and target genome length. Each read is sampled from a random location, uniformly selected from the set of positions in the genomic segment. The reads are taken to be error-free and of the specified length. The latter is for convenience, and is not a constraint of the algorithm. Read lengths typically vary within a band surrounding the specified length, and in any case are not identical due to trimming based on quality scores etc. We do not expect this to have a noticeable impact on the results. Even though the length of the target genomic segment is known, we incorporated a $\pm 10\%$ range for the purpose of testing the proposed solution.

Read length/ Min. overlap	Coverage	String graph size (edges)	Annotated graph size (edges)	N50 kbp	N70 kbp	N80 kbp	N90 kbp
500/150	10x	33	19	104.3	63.1	42.3	22.8
	20x	22	5	503.7	276.2	179.9	179.9
250/100	10x	126	106	16.1	11.8	8.9	6.4
	20x	28	9	503.7	270.4	179.9	179.9
100/60	50x	184	17	269.9	179.7	179.7	80.0
50/40	50x	396	79	101.4	80.0	46.0	27.2

Table I
Assembly results with various read lengths and coverage on a 1 Mbp stretch of the *E. Coli* genome.

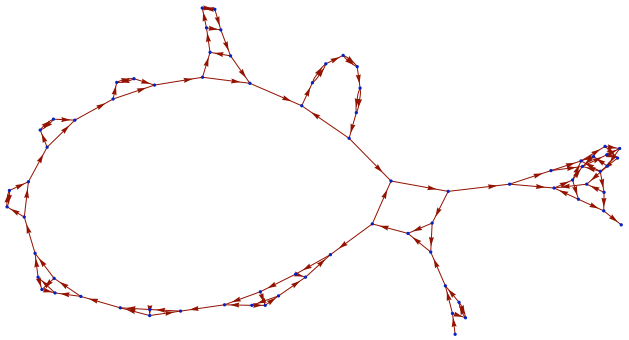

Figure 2. String graph of a 1 Mbp stretch of *E. coli* genome with 100 bp reads and 100x coverage.

Experimental results on the *E. Coli* genomic segment are shown in Table I. For each data set tested, we show the sizes of the resulting string graph and annotated string graph in terms of the number of edges. As expected, the sizes of these graphs increase with decrease in read length as more ambiguities arise from shorter reads. For illustration

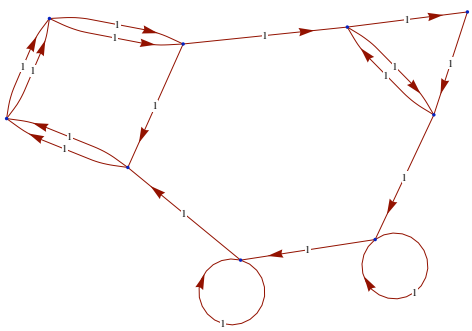

Figure 3. Annotated string graph of a 1 Mbp stretch of *E. coli* genome with 100 bp reads and 100x coverage.

purposes, the string graph and the annotated string graph for the case of 100 bp reads at 100x coverage are shown in Figure 2 and Figure 3, respectively. The annotated string graph becomes smaller due to the elimination of edges with zero count and collapsing of the resulting simple paths. At this stage, we report each edge as a separate contig. Note that our experiments are conducted with unpaired reads. In practice, mate pair information is available and can be used to find longer paths in the graph when a collection of mate pairs unambiguously indicates successive edges.

For each data set, we report the N50, N70, N80 and N90 lengths in kbp. The Nx measure reports the longest contig such that contigs of length no smaller than it can cover at least x% of the genome. For the *E. coli* genomic segment, just one contig covered more than half the target sequence in case of 500/250 bp reads at 20x coverage, while just two contigs covered the same in case of 100 bp reads at 50x coverage. Quality of the generated contigs was tested by aligning the contigs back to the reference genome. In all cases, the contigs generated were substrings of the genome with no differences. Note that the size of the annotated graph also provides the number of contigs generated. It is clearly seen that this approach is able to reproduce the genome as a small number of very large contigs. For example, the target sequence is revealed as 19 contigs at 10x coverage and 5 contigs at 20x coverage for 500 bp reads.

Experimental results on similarly composed data sets for the human *Y* chromosome segment are shown in Table II. Due to the higher repeat content, the string graph sizes are significantly larger than the corresponding ones for *E. Coli*. However, the annotated graph sizes for 500 bp and 250 bp reads are much smaller than the corresponding string graphs, and result in very large contigs. This attests to the power of the maximum likelihood approach in accurately characterizing repeats and deducing their copy count. The annotated graph sizes for 100 bp and 50 bp reads are significantly larger than their *E. Coli* counterparts. This is partly because of increase in the number of repeats larger than the read length, and partly because it is harder to accurately characterize repeats with shorter reads.

As the experimental results demonstrate, the maximum likelihood approach has the potential to lead to vastly improved genome assemblers, both in terms of assembly quality and the goal of inferring the genome with as few contigs as possible. Although the experiments in both this paper and in [20] are presented for error-free reads, they are indicative of the quality of results that can be obtained with this approach under ideal conditions. Any real assembler must contend with reads containing errors. The effect of errors can be mitigated through running error-correction software prior to assembly (for example, see [13], [21]. We further expect that most paths resulting from erroneous reads are likely to receive zero traversal counts in the maximum likelihood formulation. We expect an increase in the number of contigs in practice both due to left over erroneous paths as well as regions of the genome not covered by reads, which is a problem for any genome a ssembler. A practical maximum likelihood assembler that deals with these issues is currently under development.

A significant drawback of the maximum likelihood based methods is they rely on complex optimization or network flow based techniques which are more compute-intensive than the typical linear-time methods currently used in production assemblers. As a result, it is unlikely that maximum likelihood based methods can scale to mammalian or plant genomes in the Gbp range. Nevertheless, given the much higher quality, contig length, and coverage of the assemblies we produced indicates it is a worthwhile approach when it is applicable. For instance, rapid sequencing of microbial organisms is continually underway – and these genomes are typically a few million base pairs at which scale the maximum likelihood approaches can be applied. Furthermore, these methods can be useful in hierarchical sequencing approaches, or genome sequencing projects where pools of BACs are sequenced.

IV. Conclusions

In this paper, we provide a maximum likelihood formulation of the genome assembly problem and show that it can be solved as a convex optimization problem. The overlap-layout-consensus paradigm is susceptible to mistakes in assembly even due to repeats shorter than the read length. Our results highlight the danger: the size of the annotated graph is much smaller than the string graph because our method has detected and removed the corresponding short (smaller than the length of the read) repeats. The contig lengths we report are significantly longer than the results reported in the literature with overlap-layout-consensus assemblers, and our own experience them. They are also a marked improvement over the results in [20], possibly because of our different formulation. Experimental results demonstrate

Read length/ Minimum overlap	Coverage	String graph size (edges)	Annotated graph size (edges)	N50 kbp	N70 kbp	N80 kbp	N90 kbp
500/150	10x	108	12	84.2	65.4	36.7	34.1
	30x	190	5	500.2	500.2	500.2	77.8
250/100	10x	289	75	13.5	10.2	8.5	5.9
	30x	352	19	499.3	499.3	499.3	77.7
100/60	50x	907	99	65.8	34.1	27.6	10.5
50/35	50x	10366	654	2.7	1.3	0.35	—

Table II
ASSEMBLY RESULTS WITH VARIOUS READ LENGTHS AND COVERAGE ON A 581,282 BP CONTIG FROM HUMAN *Y* CHROMOSOME. THE CONTIGS ARE SMALLER THAN THE CASE OF *E. Coli* GENOME DUE TO HIGHER REPEAT CONTENT.

that under the ideal condition of error-free reads, our maximum likelihood method can approach the ideal goal of discovering the genomic sequence as a single contig. This provides confidence that the approach could improve upon current generation assemblers even after deterioration in the results due to read errors, chimeras, non-uniformity of coverage, and poorly sampled regions. The development of a comprehensive assembler based on the maximum likelihood formulation is still an open issue.

ACKNOWLEDGEMENTS

S.A. is funded in part by Swarnajanti Fellowship from the Government of India.

REFERENCES

[1] S. Batzoglou, D. Jaffe, K. Stanley, and *et al.*, "Arachne: A whole-genome shotgun assembler," *Genome Research*, vol. 12, no. 1, pp. 177–189, 2002.

[2] P. Havlak, R. Chen, K. Durbin, and *et al.*, "The atlas genome assembly system," *Genome Research*, vol. 14, pp. 721–732, 2004.

[3] X. Huang and A. Madan, "CAP3: A DNA sequence assembly program," *Genome Research*, vol. 9, no. 9, pp. 868–877, 1999.

[4] E. Myers, G. Sutton, A. Delcher, and *et al.*, "A whole-genome assembly of *drosophila*," *Science*, vol. 287, pp. 2196–2204, 2000.

[5] W. Kent and D. Haussler, "Gigassembler: an algorithm for initial assembly of the human working draft," *Genome Research*, vol. 11, no. 9, pp. 1541–1548, 2001.

[6] X. Huang, J. Wang, S. Aluru, and *et al.*, "Pcap: A whole-genome assembly program," *Genome Research*, vol. 13, no. 9, pp. 2164–2170, 2003.

[7] J. Mullikin and Z. Ning., "The phusion assembler," *Genome Research*, vol. 13, no. 1, pp. 81–90, 2003.

[8] G. Sutton, O. White, M. Adams, and A. Kerlavage, "Tigr assembler: A new tool for asembling large shotgun sequencing projects," *Genome Science and Technology*, vol. 1, pp. 9–19, 1995.

[9] R. Idury and M. Waterman, "A new algorithm for dna sequence assembly," *Journal of Computational Biology*, vol. 2, pp. 291–306, 1995.

[10] H. T. P.A. Pevzner and M. Waterman, "An eulerian path approach to dna fragment assembly," *Proceedings of the National Academy of Sciences USA*, vol. 98, no. 17, pp. 9748–9753, 2001.

[11] E. Myers, "The fragment assembly string graph," *Bioinformatics*, vol. 21, pp. 79–85, 2005.

[12] J. Butler, I. MacCallum, M. Kleber, and *et al.*, "ALLPATHS: De novo assembly of whole-genome shotgun microreads," *Genome Research*, vol. 18, pp. 810–820, 2008.

[13] M. Chaisson and P. Pevzner, "Short fragment assembly of bacterial genomes," *Genome Research*, pp. 18:324–330, 2008.

[14] P. Medvedev and M. Brudno, "Ab initio whole genome shotgun assembly with mated short reads," in *Lecture Notes in Computer Science*, vol. 4955, 2008, pp. 50–64.

[15] D. Zerbino and E. Birney, "Velvet: Algorithms for de novo short read assembly using de Bruijn graphs," *Genome Research*, vol. 18, pp. 821–829, 2008.

[16] B. Jackson, P. Schanble, and S. Aluru, "Parallel short sequence assembly of transcriptomes," *BMC Bioinformatics*, vol. 10, p. S14, 2009.

[17] ——, "Assembly of large genomes from paired short reads," *Springer-Verlag Lecture Notes in Bioinformatics*, vol. 5462, pp. 30–43, 2009.

[18] B. Jackson, M. Regennitter, X. Yang, and *et al.*, "Parallel de novo assembly of large genomes from high-throughput short reads," in *Proc. International Parallel and Distributed Processing Symposium (IPDPS)*, 2010, pp. 1–10.

[19] J. Simpson, K. Wong, S. Jackman, and *et al.*, "Abyss: A parallel assembler for short read sequence data," *Genome Research*, vol. 19, pp. 1117–1123, 2009.

[20] P. Medvedev and M. Brudno, "Maximum likelihood genome assembly," *Journal of computational Biology*, vol. 16, pp. 1–16, 2009.

[21] X. Yang, K. Dorman, and S. Aluru, "Reptile: Representative tiling for short read error correction," *Bioinformatics*.