A Case Study in Genome-Level Fragment Assembly

Ting Chen
Department of Genetics
Harvard Medical School
Boston, MA 02115 USA

Steven S. Skiena
Department of Computer Science
State University of New York
Stony Brook, NY 11794 USA

Abstract

Motivation: *We use the fact of two teams independently sequencing the one megabase genome of* Borrelia burgdorferi *as an opportunity to study the accuracy of genome-level assembly.*

Results: *We compare the results of three different assembly programs (PHRAP, TIGR Assembler, and STROLL) on the DNA fragments used in both the Brookhaven and TIGR sequencing projects. We also describe the algorithms and data structures used in our assembly program STROLL, which was used to complete the Brookhaven* Borrelia *project.*

Availability:
http://genetics.med.harvard.edu/~tchen/STROLL.

Contact: *tchen@arep.med.harvard.edu* and *skiena@cs.sunysb.edu.*

Introduction

The problem of fragment assembly for shotgun sequencing is growing in importance and complexity, with new sequencing methodologies being proposed and applied to the larger and problematic genomes of higher organisms. A large number of fragment assembly programs have been developed, including CAP (Huang, 1996), FAK (Myers *et al.*, 1996a and 1996b), FALCON (Gryan and Church, 1994), PHRAP (Green, 1996), Staden (Dear and Staden, 1991), and TIGR Assembler (Sutton *et al.*, 1995). Many of these have similar fundamental designs, but differ substantially on important engineering issues.

In July 1997, TIGR (The Institute for Genome Research) announced successfully sequencing the genome of *Borrelia burgdorferi* the bacterium which causes Lyme disease using a shotgun sequencing strategy. Barely three weeks later, a team at Brookhaven National Laboratory independently completed its sequencing of *Borrelia*, employing somewhat different technology. The circumstances of two groups independently but simultaneously completing the same genome provide us with a unique opportunity to study the state of the art in DNA sequencing, and compare fragment assembly across two different projects.

This paper has two primary missions: First, we describe the algorithms and data structures behind STROLL (Chen and Skiena , 1997a, 1997b, and 1998), a fragment assembler we developed for the Brookhaven National Laboratory team, which was successfully applied to the assembly of *Borrelia burgdorferi* genome. Second, we present the results of extensive experiments with STROLL and two important genome-level fragment assemblers (PHRAP and TIGR Assembler), using data from both the TIGR and Brookhaven groups.

Our goal is *not* to identify the "best" assembly program, but to perform an analysis of the errors made by all three of these programs so as to advance the design of future assemblers. To our knowledge, these are the most thorough experiments ever done in evaluating the accuracy of large-scale assembly. To summarize our results:

- The set of large contigs formed by all the assemblers are sufficiently different that it pays to run multiple assemblers on the same data and merge the resulting contigs. In particular, we recommend running multiple assemblers during the gap-closing stage of any large sequencing project.

- All assemblers have trouble with repeat regions in both projects. In an attempt to resolve these problems, modern assemblers incorporate two additional classes of information; clone order constraints and sequence-quality measures. However, our experiments show that adding this information to the raw sequence data did not improve the sizes of the resulting contigs, although they did lead to a minor improvement on the accuracy of the final consensus sequence.
- We provide a taxonomy of the primary algorithms used in each of the genome-level assemblers, which sheds light on the differences between them.

Methods

STROLL began in collaboration with the Brookhaven National Laboratory Sequencing Group, led by Dr. William Studier, to sequence the genome of *Borrelia burgdorferi*. It originally aimed to provide computational support for the strategy of sequencing by primer walking (Studier *et al.*, 1989, Studier *et al.*, 1992, Studier *et al.*, 1995). However, as the project progressed, STROLL evolved into a general-purpose fragment assembler, which supports a wide variety of sequencing technologies.

Features

Repeat regions have caused troubles in many genome sequencing projects. In sequencing higher organism genomes, we will encounter many longer repeats, but none of the current assembly program can handle them perfectly. The base quality information is one of the most important methods being used in many assemblers to improve sequencing accuracy and to solve repeats. STROLL incorporates it in (1) pairwise comparison to discriminate overlaps, repeats, and chimeras, (2) incremental multiple alignments to merge one fragment into partial assemblies, and (3) consensus generation to determine each consensus base and give an associated confidence level. Besides supporting above features, STROLL includes the following algorithms to improve speed and reduce space:

- The use of a space-efficient data structure, the suffix array (Manber *et al.*1993), to quickly reject non-overlapping fragment pairs, thus reducing the number of calls to the pairwise comparison.
- The use of a fast banded pairwise comparison algorithm, with affine gap penalties and base qualities to search local similarities of two fragments.
- The use of an overlap recovery strategy (exploiting transitive relations) to recover most of the undetected overlaps.
- The use of an incremental multiple alignment algorithm to add fragments into contigs one-by-one in the order of pairwise alignment qualities.

All the algorithms have been chosen purposely for high speed and using low space requirements. In a megabase genome-level sequencing project, the required memory space can easily run to several hundred megabytes and the assembly time to tens of hours. To support the increasing scale of sequencing efforts, all our algorithms run in linear (or close-to-linear) time and take linear space.

Algorithm

Table 1 provides a taxonomy of the algorithms used in PHRAP, FAK, TIGR, and STROLL. The basic assembly algorithm in STROLL is described as follows. Details are provided in the subsequent paragraphs.

1. Build a suffix array for both strands of all fragments and extract all exact matches to form a list of candidate fragment overlaps.
2. Perform banded pairwise comparisons on each fragment pair in the candidate list, generate potential overlaps, and reconstruct missed overlaps using transitive relations.

3. Classify overlaps into multiple levels, using base qualities and alignment qualities.
4. Start with the current cleanest fragment as the initial assembly seed and proceed to step 7 when no seed is left.
5. Select the fragment with the best overlap to continue the current assembly; if none is found, go to step 4, starting a new contig.
6. Add the fragment into the contig; if it fails; go to step 5.
7. Optimize local multiple alignments and generate consensus.

These phases are discussed in greater detail below.

Suffix Arrays and Exact Match Heuristic
STROLL uses an exact match strategy which quickly rejects non-overlap fragment pairs if they do not share exact matches over a threshold length. Compared to suffix trees and hash tables, the suffix array (Manber *et al.*, 1993) is a flexible, space-efficient and time-compatible data structure to extract all exact matches. It consists of an array of sorted suffixes and an array of longest common prefixes, and requires only 6 bytes per character or 12 bytes per base (for both strands). On a SUN Sparc-10 machine with 128 Megabytes RAM, we can construct a suffix array of 200k base pairs in 20 seconds and a suffix array of 4M base pairs in 15 minutes. All the exact matches over a threshold length can be easily obtained by traversing the array of the longest common prefixes and fragment pairs whose exact matches are less than the threshold are rejected. In (Chen *et al.*, 1997b), it has been established that choosing a proper threshold can reduce the calls to pairwise comparison by up to 1,000 times while maintaining a high accuracy.

Pairwise Comparison
The most used sequence comparison method is the Smith-Waterman (Waterman 1995) algorithm, which finds the optimal alignment of two fragments by maximizing the score of a given matching function. However, the Smith-Waterman algorithm requires quadratic time. In order to speed up the pairwise comparison, STROLL heuristically searches for the best possible alignment path of two fragments and then performs a banded Smith-Waterman algorithm with an affine gap penalty to determine the local similarities. STROLL searches for local similarities instead of global similarities for two reasons: (1) the end of a fragment usually contains high rate of errors which should not be counted for similarities, and (2) local similarities help finding repeats and chimeras. Base quality measures (specifically the logarithm of the probability of the given base) are used as weights in the dynamic programming to find the optimal alignment. At this step, we accept alignments with at least 30 base-pair long, and later classify them by the length of the alignment, the percentage of high quality matches, the percentage of high quality mismatches, and the length of high quality overhangs. High quality matches contribute more to the similarities, while high quality mismatches cost more. Two key factors to determine repeats and chimeras are the high quality mismatches and high quality overhangs. Most overlaps missed by the exact match heuristics can be reconstructed through a transitive relation. Our experiments in (Chen *et al.*, 1997b) have shown that most of missed overlaps can be thus reconstructed.

Overlap Classification

The purpose of overlap classification is to give a more accurate measure of the quality of an overlap than its similarity score. Pairwise comparison scores sometimes miss important local information such as high quality mismatches, high quality overhangs, the distribution of the mismatches, and clone constraints. The *clone constraints* account for two fragments being obtained from the same clone. These constraints restrict how far apart these two fragments can be and whether they reside on the same strand or the reverse strand. In most shotgun sequencing projects, the two fragments are located at the two ends of an insert, but in Brookhaven's project additional possibilities result from the primer walking and nested deletion procedures. Low quality mismatches are usually caused by base calling errors: overcalls, undercalls or miscalls, but high quality mismatches indicate a possibility of repeats because a single high quality base is typically more than 99% accurate. All overlaps are evaluated by several criteria including matching percentage, length of overlaps, percentage of high quality mismatches, length of high quality overhangs, and constraint-satisfied overlaps. A repeat is identified if an overlap has one long high quality overhang, or contains a high percentage of high quality mismatches, or violates clone constraints. A candidate chimera, which does not fully overlap other fragments in other clones, involves at least two repeated overlaps and can be broken into two distinct parts by these repeats. Both the repeats and the chimeras are classified into the group with the lowest scores. The remaining overlaps are classified into multiple levels, and specially, the overlaps satisfying clone constraints are clustered into the higher score groups. This guarantees a constraint-satisfied overlap, obtaining by primer walking or nested deletion, having a higher priority. In the later assembly phase of merging fragments, the score of overlap classification determines which fragment should be added next.

Merging the Fragments with the Contigs

The assembly starts with the cleanest available fragment. The cleanliness of a fragment is defined as a combination of the number of good overlaps it involves and the percentage of high quality bases. Any fragments involved in repeats are among the least clean. In the process of merging fragments, every contig starts by the current cleanest seed and extends as far as possible. A contig is formed incrementally: a new fragment which has the best overlap classification score with some fragment at the end of the contig is chosen. If this fragment fails to be added into the contig, the next best one is chosen until no fragment can be added, and this contig is complete. If clone constraints are used, the fragment to be added cannot not violate either the clone-length or clone-strand constraints. STROLL takes advantage of extra constraints imposed by special sequencing technologies. For example, all the within-clone overlaps generated by nested deletion or primer walking are labeled as better overlaps in the classification, and are added first to form a reliable skeleton across the genome, and thus avoid adding repetitive fragments in the assembly.

Incremental Multiple Alignments

Adding a fragment into a multiple alignment can be very slow, because potentially it has to align with every other fragment in the multiple alignment. STROLL first locates the region where the fragment will be added into the multiple alignment, and then performs banded alignment algorithms to find the best alignment. If the alignment fails, a full alignment algorithm is then called to redo it. For good quality data assemblies, this strategy saves time because most fragments can be successfully merged by the banded alignment. Adding a fragment into the multiple alignment is performed by aligning the fragment with the profile sequence generated from the previous multiple alignment. Here we use the base quality measures and the semiglobal alignment. The alignment is rejected if it contains a lot of high quality mismatches.

Optimal Multiple Alignments and Consensus Generation

Incremental multiple alignment does not always give the optimal alignment. The alignment can be adjusted locally by ReAligner program (Myers *et al.*, 1997) for optimal solutions. Currently, the final consensus is determined for each column of the multiple alignment. Base qualities and strand information are used to determine the consensus with a confidence level. STROLL first looks for high quality calls from both strands, and then the number of high quality calls at one strand, and finally the low quality calls.

A Tale of Two Sequencing Projects

Starting at the end of 1995 and the early 1996, The Institute of Genome Research(TIGR) and Brookhaven National Laboratory were independently funded to sequence the genome of *Borrelia burgdorferi*.

Two years later, TIGR released its 910,715 base-pair sequence on July 25, 1997 (the full genome annotations were published in (Fraser *et al.*, 1997)), and Brookhaven reported its 893,370 base-pair sequence of B. burgdorferi (Studier *et al.*, 1997) on August 19, 1997. The two sequences largely agree with each other, however, the TIGR sequence extends 3,728 base-pairs to the left and 14,026 base-pairs to the right of the Brookhaven sequence. Minor differences between the two sequences include approximately 380 mismatches and approximately 50 ambiguities on each of the sequences. Of the 380 discrepancies, 161 are in the 35,193 bases of the Brookhaven sequence where only a single sequence determination has been made, reflecting the lower reliability of single-lane coverage. For more details, see (Studier *et al.*, 1997). For many of the discrepancies between the projects, it is unclear which group's sequence is erroneous, or indeed whether either group is erroneous. In fact, the two groups may be sequencing slightly different strains of Borrelia.

Although similar results were achieved, the two groups used different sequencing strategies. TIGR apparently used a deep shotgun sequencing strategy with approximately 7.5-fold genome coverage and generated 19,078 fragments (including plasmids). For Brookhaven, the Borrelia project was designed as a test case for their primer-walking technology. A framework of dual-end sequenced clones with roughly 2 times coverage was built, with remaining gaps closed using directed sequencing. A total of 8,136 high quality fragments(base-error rates below 1%) were generated for an approximately 4-fold genome coverage. The final multiple alignments and consensus have been carefully and manually checked for mismatches and clone length constraints. All chimera fragments and low quality fragments have been removed. This data provides a good test of how a program handles an assembly with repeats.

Comparing the Assemblers

It proves to be a non-trivial problem to compare the results of different assembly programs on a given project's data. In our case, there are two separate projects which are substantially in agreement. We have run each of the assemblers on both of the projects under a variety of conditions, and measured the edit distance of the large contigs to the human-edited final sequence. In these experiments, the most recent version of the program which was made available was used. PHRAP's last revision to the files was labeled July 25, 1996, and TIGR Assembler's was 1996. It is important to remark that both PHRAP and TIGR Assembler have been significantly upgraded since then, and so our results may not apply to the most current version of their assembler.

Two data sets were used in the tests. One is the whole set of 8,136 fragments from Brookhaven. Another is what we were given from TIGR, a subset of 9,432 fragments. Three test results are shown: (1) gross assembly statistics on Brookhaven data, (2) assembly accuracy on Brookhaven data, and (3) assembly accuracy on TIGR data. No quality data has been included in these tests. The next three paragraphs detail these results. More tests are available in (Chen *et al.*, 1997b).

Gross Statistics

Gross assembly statistics on the Brookhaven data are reported in Table 2. First, Table 2 provides a very rough sense of the speed of each program. We note that these running times may not reflect the "true" efficiency of each program because some assembly programs are error-sensitive, and hence faster on higher quality data, and some assembly programs permit a trade-off between speed and accuracy. Although we have carefully tested each program under a variety of conditions, it is conceivable that we missed the optimal setting, and we may also ignore the extra efforts in some programs to solve problems of repeats/chimeras or to generate more accurate multiple alignments.

All timing runs were conducted on a Sun Sparc10 machine with 128 Mb RAM. Table 2 shows that STROLL and PHRAP have similar speed while TIGR Assembler is about 6 times slower. It is likely to be because TIGR Assembler spends more time on the dynamic programming in pairwise comparison. We also tested PHRAP on data with artificial high base quality, and found no significant difference in time and the results. PHRAP has successfully used base quality measures from PHRED to improve the assembly accuracy. This could be because the data quality is very good. Table 2 also provides gross statistics on the formation of contigs. The number of contigs is a measure of the aggressiveness of the assemblers. PHRAP produces the minimum number of contigs, which is in principle a desirable goal. However, even more important is the size of the largest contigs, since these represent the final desired sequence. In Table 2, we report the sum of the sizes of the four largest contigs, which show PHRAP and STROLL were equally successful at generating large contigs.

Figure 1 illustrates the issue of contig size from a different perspective. The contigs from each program were sorted by size, and the the prefix sums of the sequence computed. Thus the ith point reports the basepair sum of the i largest contigs. The faster this curve approaches the actual genome size, the less effort biologists need to close the gaps. PHRAP and STROLL provide almost identical curves (even though STROLL produces almost twice as many contigs), while TIGR Assembler is the most conservative of the programs.

Assembly on Brookhaven Data

Assessing the accuracy of assembly requires special
treatment of the repeat regions. Brookhaven's reported genome has a long and nearly tandem repeat region: (B) Two copies of a 3,140 basepair region (with only one mismatch) containing Ribosomal RNA which starts at positions 403,700 and 433,940. Table 3 lists the four largest contigs from each program and their locations on the reported Brookhaven genome sequence. Only one copy of the Ribosomal RNA is shown in all programs: Contig 3 of PHRAP, Contig 4 of TIGR Assembler and Contig 3 of STROLL. It demonstrates the difficulty of assembling the ribosomal RNA region, since two repeat copies are longer than a read length and almost identical. Therefore, a pure shotgun sequencing assembly without additional constraints may very well misassemble long and perfect repeats. On the other hand, comparing the contigs formed by STROLL and PHRAP shows that PHRAP's contig 1 is a merge of STROLL's contigs 1 and 4, and STROLL's contig 3 is a merge of PHRAP's contigs 3 and 4. This implies some degree of complementary between the two assembly programs: running both on the same data and comparing the assembly results may eliminate some gaps and get a better result. The error rates of the final consensus sequence for the three programs are lower than the standard .01%: .0025% for PHRAP, .0076% for TIGR Assembler, and .003% for STROLL. We also artificially generated base qualities for PHRAP according to the formula in (Green, 1996) and ran PHRAP again. There were no major difference among the reported contigs, because the data quality is very good.

Assembly on TIGR data

We also tested PHRAP, TIGR Assembler, and STROLL on the TIGR's Borrelia genome sequences. Only the clean region of each fragment was provided, and read qualities were not available for these tests. TIGR's genome sequence contains three noteworthy regions: (T1) Two copies of 3,140 basepairs, containing Ribosomal RNAs, in the region between 465,180 and 472,180; (T2) Seven copies of 161 basepair tandem repeats in the region between 696,462 and 697,576; (T3) One copy of a problematic 3,570 basepair region, from 514,950 to 518,519 in TIGR genome sequence. The results of each program are shown in Table 4. PHRAP reported a total of 3 contigs and STROLL reported a total of 4 contigs and TIGR Assembler reported 35 contigs larger than 10kb. All the top contigs compared to the reported TIGR genome sequence are listed in Table 4. Again, PHRAP is the most aggressive program in forming large contigs with the top two contigs covering most of the genome, but it misassembled both repeat regions (T1) and (T2), and missed (T3) region as well. STROLL formed four large contigs providing similar coverage, and TIGR Assembler remains too conservative with tens of small contigs spread all over the genome. Both of them missed the tandem repeat region (T1) as well and broke the RNA repeat region (T2) into two distinct contigs. All the programs failed to assemble the tandem repeat region, which is longer than a fragment read length. It indicates that tandem repeats are another serious problem in many sequencing projects. The base error percentages of the final consensus sequences are also reported in Table 4, and only correct contigs are taken into account. PHRAP has the lowest error rate (.0015%), which indicates that PHRAP does a better job of discriminating false alignments, constructing multiple alignments, and generating consensus sequences. TIGR Assembler reported a reasonably accurate final consensus sequence (.07% error rate), while STROLL made substantially more mistakes (.14% error rate).

Assembly on Brookhaven data with clone constraints

Both *TIGR Assembler* and *STROLL* use clone-end constraints and clone-length constraints in fragment assembly. We added these constraints to the Brookhaven sequence data and tested them on both programs, hoping that these additional constraints can overcome the problems of repeat regions. The distance for two fragments from two ends of a clone is restricted to be between 2k bps and 10k bps. Our results are shown in Table 5, where the top four largest contigs are listed. However, the results in Table 5 are quite disappointing. They prove inferior to those from Table 3, in that the resulting contigs are smaller. This is because both programs excessively favor the fragment that satisfies clone constraints in the construction of multiple alignments. Thus if the clone constraints are not accurate, the assembly is affected. How best to incorporate clone constraints in the assembly remains an open problem.

Acknowledgments

We thank Bill Studier and the rest of the Brookhaven group for interesting discussions on primer walking and sequencing. We also thank Granger Sutton, Gene Myers, Phil Green, and Xiaoqiu Huang for making their data and assembly programs available for our comparisons.

References

Butler-Loffredo, L., Dunn, J.J. and Studier F.W. (1995) Ligation of hexamer templates to produce primers for cycle sequencing or the polymerase chain reaction. *Analyt. Biochem.*, 228:91–100.

Chen, T. and Skiena, S.S. (1997a) Trie-based data structures for sequence assembly. *The Eighth Symposium on Combinatorial Pattern Matching*, pages 206–223.

Chen, T. and Skiena, S.S. (1997b) How Good is Genome-Level Fragment Assembly? Unpublished manuscript, Department of Computer Science, SUNY at Stony Brook.

Chen, T. and Skiena, S.S. (1998) STROLL WWW Homepage. http://www.cs.sunysb.edu/~tichen/STROLL, Department of Computer Science, SUNY at Stony Brook.

Dear, S. and Staden, R. (1997) A sequence assembly and editing program for efficient manageme nt of large projects. *Nucleic Acids Research*, 19:3907–11.

Fraser, C.M. *et al.*(1997) Genomic sequence of a Lyme disease spirochaete Borrelia burgdorferi. *Nature*, 390:580–6.

Green, P (1996) Documentation for Phrap. http://bozeman.mbt.washington.edu, Genome Center, University of Washington.

Gryan, G. and Church, G.M. (1994) Falcon: Fast assemblies of large contigs. *http://arep.med.harvard.edu/labgc/falcon.html*, Church Lab, Department of Genetics, Harvard Medical School.

Huang, X. (1996) An improved sequence assembly program. *Genomics*, 33:21–31.

Kieleczawa, J., Dunn, J.J. and Studier, F.W.(1992) DNA sequencing by primer walking with strings of contiguous hexamers. *Science*, 258:1787–91.

Manber, U. and Myers, E.W. (1993) Suffix arrays : A new method for on-line string searches. *SIAM J. Computing*, 22:935–948.

Myers, E.W. (1996a) A suite of unix filters for fragment assembly. *Tech. Rep. Dept. of CS, U. of Arizona, Tucson, Az*, TR96-07.

Myers, G., Jain, M., Anson, E., and Larson, S.(1996b) An interface for a fragment assembly kernel. *Tech. Rep. Dept. of CS, U. of Arizona, Tucson, AZ*, TR96-04.

Myers, E.W. and Anson, E. (1997) Realginer: A program for refining dna sequence multi-alignments. *Journal of Computational Biology*, 4:369–383.

Studier, F.W. *et al.*(1997) Genomic sequence of *Borrelia* burgdorferi. *http://www.bio.bnl.gov/htmls/chrom_seq.html*, 1997.

Studier, F.W. *et al.*(1989) A strategy for high-volume sequencing of cosmid DNAs: Random and directed priming with a library of oligonucleotides. *Proc. Natl. Acad. Sci. USA*, 86:6917–21.

Sutton, G.G., White, O., Adams, M.D. and Kerlavage, A.R. (1995) TIGR Assembler: a new tool for assembling large shotgun sequencing projects. *Genome Science and Technology* Vol.**1**, 9-19.

Waterman, M.S. (1995) *Introduction to Computational Biology.* Chapman & Hall, London, UK.

Tables

		PHRAP	FAK	TIGR	STROLL
	Heuristic	Exact Matches	Inexact Matches	Exact Matches	Exact Matches
Overlap	Implementation	Sorted Suffixes	Unknown	Hash Table	Suffix Array
Detection	Pairwise Comparison	Banded S-W	Incremental align	Smith-Waterman	FASTA-like
	Search Restriction	Band DP	Error sensitive	DP	Band DP
	Time Complexity	Adjusted	Adjusted	Quadratic	Adjusted
Assembly	Layout Algorithm	Greedy	Greedy	Greedy	Greedy
	Multiple Alignment	Heuristic	Refined	Refined	Heuristic
	Uses Qualities?	Yes(Phred)	No	was No	Yes
Features	Uses Constraints?	No	No	Yes	Yes
	Detects Repeats?	Yes	Yes	Yes	Yes
	Detects Chimeras?	Yes	Yes	Yes	Yes

Table 1: Designs of PHRAP, FAK, TIGR Assembler and STROLL.

	PHRAP	STROLL	TIGR
Assembly time in minutes	71.5	81.0	455.8
Number of contigs formed	56	89	129
Length of four largest contigs	857,113	858,448	332,700

Table 2: Gross statistics of various assemblers on the Brookhaven data.

	Contig	Base-pairs	Genome start	Genome end	Errors
PHRAP	1	390,227	503,141	893,369	12
	2	275,656	255	275,909	4
	3	166,130	307,382	476,752	Miss (B)
	4	25,100	477,372	502,470	1
	Sum	857,113			17 (.0025%)
TIGR	1	118,225	735,638	853,862	7
	2	99,124	611,885	711,008	8
	3	70,917	307,048	377,964	7
	4	44,434	420,759	468,433	Miss (B)
	Sum	332,700			22 (.0076%)
STROLL	1	339,458	553,907	893,368	15
	2	276,292	0	276,290	4
	3	191,268	307,050	502,695	Miss (B)
	4	51,430	502,676	554,105	3
	Sum	858,448			22 (.003%)

Table 3: Assembly accuracy of PHRAP, TIGR Assembler, and STROLL on Brookhaven data.

	Contig	Basepairs	Genome start	Genome end	Errors(bps)
PHRAP	1	532,940	370,691	910,715	Miss (T1) (T2) (T3)
	2	369,488	72	369,537	58
	3	3,088	515,088	518,175	0
	Sum	905,516			(.015%)
TIGR	1	45,754	764,201	809,950	18
	2	37,014	294,185	331,190	31
	3	32,279	551,861	584,139	10
	4	27,806	113,716	141,524	24
	Sum	142,853			(.07%)
TIGR	6	24,737	446,796	471,518	Break (T1)
	15	19,599	679,506	699,963	Miss (T2)
STROLL	1	392,250	76,955	469,048	612
	2	266,571	471,965	738,765	Break (T1) & Miss (T2)
	3	172,156	738,638	910,713	258
	4	77,023	0	77,013	36
	Sum	908,000			(.14%)

Table 4: Assembly accuracy of PHRAP, TIGR Assembler and STROLL on TIGR data.

	Contig	Basepairs	Genome start	Genome end	Errors (bps)
TIGR	1	59,382	318,582	377,962	5
	2	50,639	660,369	710,969	0
	3	49,457	804,170	853,627	11
	4	44,211	154,563	198,772	0
	Sum	203,689			16 (.008%)
	25	13,564	420,759	433,835	Miss (B)
STROLL	1	199,443	10,006	209,448	4
	2	128,077	502,676	630,752	5
	3	102,514	790,856	893,368	7
	4	83,435	646,870	730,303	0
	Sum	513,469			16 (.003%)
	5	81,816	391,520	476,576	Miss (B)

Table 5: Tests of TIGR Assembler and STROLL
on Brookhaven data with clone constraints.

Figures

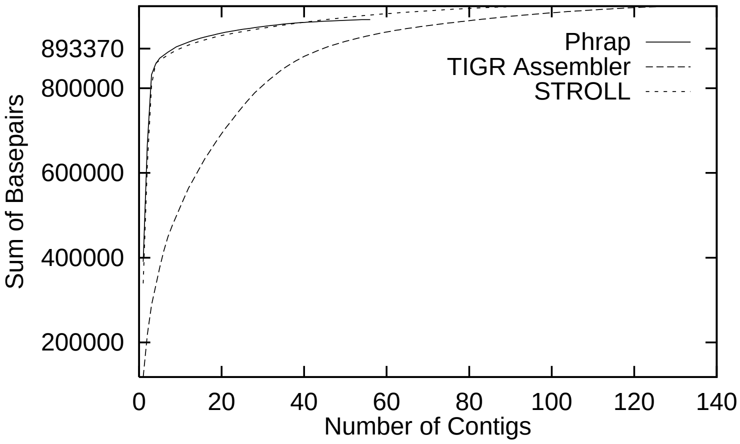

Figure 1: Base-pair sum sizes of sorted contigs.