Comparative genome assembly

Mihai Pop, Adam Phillippy, Arthur L. Delcher and Steven L. Salzberg

Date received (in revised form): 21st June 2004

Mihai Pop
is a Bioinformatics Scientist at the Institute for Genomic Research. He earned his PhD in Computer Science from Johns Hopkins University in 2000. His research interests include genome assembly and comparative genomics.

Adam Phillippy
is a Bioinformatics Software Engineer at The Institute for Genomic Research. His research interests include comparative genomics, whole genome alignment and sequence assembly.

Arthur L. Delcher
is a Bioinformatics Scientist at The Institute for Genomic Research and Professor Emeritus of Computer Science at Loyola College in Maryland. He earned his PhD in Computer Science from Johns Hopkins University in 1989 and was a member of the Informatics Research group at Celera Genomics that developed the human genome assembly program.

Steven L. Salzberg, PhD
is the Senior Director of Bioinformatics at TIGR and is a Research Professor of Computer Science and Biology at Johns Hopkins University.

Keywords: *assembly, shotgun sequencing, comparative genomics, open source*

Mihai Pop,
The Institute for Genomic Research,
9712 Medical Center Drive,
Rockville, MD 20850, USA

Tel: +1 301 795 7000
Fax: +1 301 838 0208
E-mail: mpop@tigr.org

Abstract

One of the most complex and computationally intensive tasks of genome sequence analysis is genome assembly. Even today, few centres have the resources, in both software and hardware, to assemble a genome from the thousands or millions of individual sequences generated in a whole-genome shotgun sequencing project. With the rapid growth in the number of sequenced genomes has come an increase in the number of organisms for which two or more closely related species have been sequenced. This has created the possibility of building a comparative genome assembly algorithm, which can assemble a newly sequenced genome by mapping it onto a reference genome. We describe here a novel algorithm for comparative genome assembly that can accurately assemble a typical bacterial genome in less than four minutes on a standard desktop computer. The software is available as part of the open-source AMOS project.

INTRODUCTION

Most large-scale genome sequencing projects today employ the whole-genome shotgun (WGS) sequencing strategy, in which a genome is shattered into numerous small fragments, and the fragments are then sequenced from both ends. The resulting sequences, ranging from 650 to 850 base pairs (bp) in length using the latest sequencing technology, must then be *assembled* to reconstruct the chromosomes of the target organism. As genome sequencing has become more efficient, the number of organisms sequenced by this strategy has rapidly increased. In many cases, little is known about a genome prior to sequencing; sometimes even the genome size is only a rough estimate. Genome assemblers have been relied on not only to reconstruct the genome, but also to help answer such basic questions as how many chromosomes an organism has.

In addition to sequencing new and completely unknown species, the scientific community has recognised in recent years the value of sequencing two or more very similar species (or strains) from the same genus. Perhaps the most prominent example of this is in biodefence, where multiple strains of *Bacillus anthracis* are currently being sequenced in order to create precise genotyping information that can be used for forensic purposes.[1] Sequencing projects for several major human pathogens, including *Mycobacterium tuberculosis*, *Streptococcus pneumoniae* and *Staphylococcus aureus* (to name but a few) are targeting multiple strains of these bacteria in order to understand virulence, drug resistance and other phenotypic differences between strains. On a larger scale, the mouse, rat and chimpanzee genomes are all being sequenced and mapped to the human genome to better understand human biology, and multiple *Drosophila* species are being sequenced and mapped onto one another.

DNA sequence assembly algorithms have generally followed an algorithmic strategy known as *overlap-layout-consensus*.[2] Although many distinct systems have been developed – among them the popular program *phrap*[3] – all begin by comparing all the shotgun sequences ('reads') to one another and computing which ones overlap each other (the *overlap* step). To avoid the quadratic time requirement of a brute-force approach,

most assemblers use some form of a hashing strategy (or similar indexing technique) based on short oligomers in order to identify those reads that are likely to overlap, and then run a more costly alignment algorithm on any pairs of reads that have a significant intersection in the hash table. Following the overlap step, the layout step positions the reads precisely with respect to one another, producing a multiple alignment of all reads. This multi-alignment is then used to produce the consensus, ie the final DNA sequence. Some algorithms[4–6] also include an additional step in which contigs are joined together using the linking information from the paired-end reads, creating larger structures known as scaffolds. This scaffolding step can also be run as a separate program after the initial assembly.[7]

Reference and target genomes

AMOS Comparative Assembler

In the AMOS Comparative Assembler (AMOS-Cmp), we take a fundamentally different approach: the overlap step is skipped entirely. Instead, reads are aligned to the reference genome using a modified version of the MUMmer algorithm.[8] This strategy can be called *alignment-layout-consensus*, because it produces a layout directly from the alignment. In fact, AMOS goes a step beyond many conventional assembly algorithms, because information is used from the paired-end reads in the alignment step. As described below, this helps in disambiguating the placement of reads that have matches to more than one place in the reference genome.

Alignment-layout-consensus

Read placement with MUMmer

METHODS

All assembly algorithms attempt to recover the information lost through the WGS shearing process; in particular, they try to identify the original placement of the shotgun sequence fragments along the genome. This problem encompasses most of the complexity of shotgun sequence assembly, while the remaining tasks, such as computing the precise multiple alignment of the reads or identifying polymorphisms, are somewhat easier. Our comparative assembler uses the complete sequence of a closely related organism to determine the relative placement of the reads. Thus the overlap and layout stages of a typical assembler are replaced with a module performing the alignment to a *reference genome*. The remaining assembly stages – consensus generation and scaffolding – remain the same as for an overlap-layout-consensus algorithm. Throughout the remainder of this paper, the genome being sequenced is referred to as the *target genome*, the goal being to obtain an assembly of this genome using a reference genome as a template.

The differences between the target genome and the reference genome, coupled with the presence of repeats in the data, create the biggest challenges for a comparative assembler. Even when the two genomes belong to the same species, the differences between them can be significant. For example, Figure 1 shows the alignment between two strains of *Streptococcus agalactiae*.[9,10] It is immediately clear that the strains differ by multiple insertions and deletions (indels, identified by breaks in the diagonal), furthermore a cluster of repeats can be seen in the first 500 kilobases (Kbp) of the two genomes.

The algorithm employed by AMOS-Cmp to overcome these challenges has five major steps.

- **Read alignment**. Each shotgun read is aligned to the reference genome using MUMmer.[8,11,12] Repetitive sequences and polymorphisms between the target and the reference cause some reads to align in a non-contiguous fashion. A modified version of the Longest Increasing Subsequence (LIS) algorithm[13] is used in order to generate chains of mutually consistent matches between each read and the reference. In addition to the longest consistent chain, a set of near-optimal chains is also computed in order to identify reads anchored in repeats. Those reads that are ambiguously placed in the genome (one or more chains are within 2 per cent identity from the best placement)

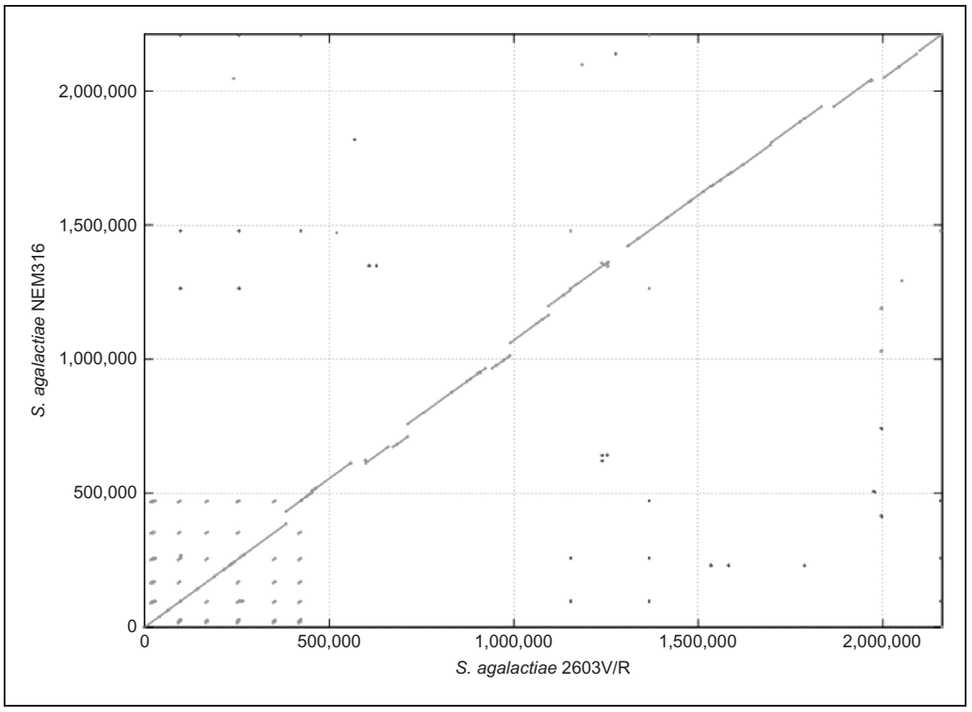

Figure 1: Alignment between two strains of *Streptococcus agalactiae*. The lines represent near-exact matches. The positions of the matches on the two genomes are shown along the axes

are classified as repetitive and resolved later (in some cases) by using mate-pair information.

Repeat solution

- **Repeat resolution**. For each read that cannot be unambiguously placed along the reference genome, a three-stage process is employed to disambiguate its placement. First, it is checked to see if the paired-end sequence (the 'mate') is uniquely anchored in the genome. If it is, the read is placed in the location that satisfies the constraints imposed by the mate-pair information. Second, if a read and its mate are both ambiguously placed, an attempt is made to find whether the mate-pair information allows us to place them both in the assembly. In some cases, there exists only one placement of both a read and its mate that satisfies the mate-pair constraints on distance and orientation. Third, when the first two steps leave us with more than one placement for a pair of reads, one of the possible placements that satisfy the mate-pair constraints is chosen at random. To illustrate how well these steps work, the original shotgun reads from the *S. agalactiae* 2603[9] WGS project were aligned to the final, finished chromosome. Out of a total of 26,099 reads, 25,310 were uniquely anchored, 314 were placed with the help of a uniquely anchored mate, 22 were placed by mate-pair constraints, and the remaining 442 had to be placed in a randomly chosen copy of a repeat.

Layout refinement

The use of mate-pairs

- **Layout refinement**. Indels and rearrangements between the target and the reference genomes complicate the mapping of the reads to the reference. As a result, reads from the target genome may only partially match the reference, or adjacent sections of the reads may match non-adjacent portions of the reference. Because of these issues, layout refinement is one of the most complex parts of AMOS-Cmp. The algorithm is described in detail below.

- **Consensus generation**. For each

Consensus generation

group of overlapping reads in the refined layout, a multi-alignment is computed to generate a consensus sequence for the genomic region covered by those reads. The multi-alignment is computed in a series of rounds. In each round, a pairwise alignment of each read to the current consensus sequence is computed and the resulting multi-alignment is used to generate a new consensus sequence. The process terminates when the new consensus sequence is the same as the one in the previous round. This is essentially the same algorithm described in Anson and Myers.[14]

Scaffolding

- **Scaffolding**. The placement of the reads along the reference genome implicitly defines a set of *contigs* – contiguous regions of the assembly – as well as the relative order and orientation of these contigs — a structure commonly known as a *scaffold*. The Bambus package[15] is used to determine the order and orientation of the contigs as determined by the mate-pair information, and to build scaffolds based on this information. Discrepancies between the two orders highlight rearrangements between the target and the reference genomes.

Handling polymorphisms

Read alignment and scaffolding are accomplished by existing programs (MUMmer and BAMBUS respectively) that have been described in detail elsewhere; our discussion is therefore confined to the complications in the layout refinement algorithm that result from polymorphisms between the target and the reference genomes. These can be divided into four different classes:

- **insertions in the target**: segments of DNA present in the target genome that do not have a counterpart in the reference genome;
- **deletions from the target**: segments of DNA present only in the reference;
- **rearrangements**: segments of DNA that appear in a different order/orientation in the reference and the target; and
- **divergent DNA**: low-similarity regions, where the target and the reference genomes have diverged significantly from each other during evolution.

In regions where the target and reference genomes are similar, the alignment of the reads to the reference genome is sufficient to infer the layout of the reads along the target genome. Any small differences in the relative placement of the reads are easily corrected when computing their precise multiple alignment during the consensus generation stage. A simplistic comparative assembler could, therefore, accept the layout of the reads along the reference genome as an approximation of the layout of the reads along the target genome. In the case of two similar genomes, the only breaks in the resulting assembly would occur in areas of the target genome not contained in any read. The comparative assembly problem is most interesting, however, when differences exist between the target and reference genomes. In the following paragraphs, the major types of polymorphisms that AMOS-Cmp can handle are discussed.

Insertions in the target

In the case of an insertion in the target, the portions of the reads contained in the insertion will not match the reference genome, leading to the situation shown in Figure 2. The alignment to the reference does not provide any information about the relationship between reads A, B and C. This information could be reconstructed by separately assembling these reads with a traditional assembly algorithm. AMOS-Cmp will break the assembly at this point, creating two separate contigs – one that ends at read A, and second one that starts with read C – leaving reads such as B (which is entirely

Insertions in the target

Insertions in the reference

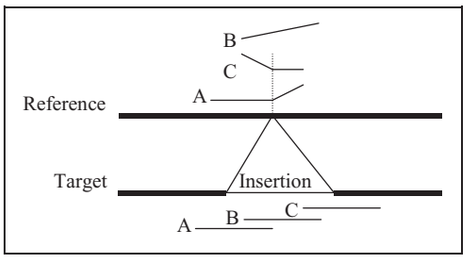

Figure 2: Mapping reads to the reference genome when the target genome contains an insertion. The bottom indicates the true layout of the reads (A, B, C) along the target. The top indicates the alignment of the reads to the reference. Slanted lines depict portions of the read that do not match; in the case of read B, the entire read does not align to the reference

Rearrangements

contained in the insertion) as unassembled individual reads, or singletons. Mate information used during the scaffolding stage should allow us to detect the presence of this insertion and properly order and orient the contigs with respect to each other.

A special case occurs when the insertion is smaller than the length of a read (Figure 3). In this case, the middle of some reads will not match the reference but both ends will. An alignment that breaks and continues at the same position in the reference, but at distant points in the read, is evidence of this type of insertion. AMOS-Cmp can resolve this situation and produce a single contig correctly containing the inserted sequence.

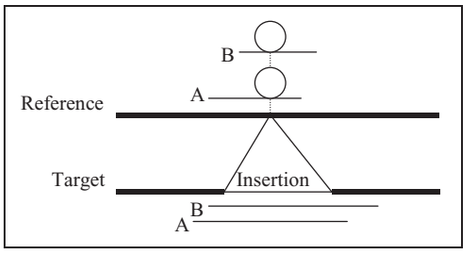

Figure 3: The insertion in the target genome is shorter than a single read. The 'bubbles' identify the portions of the two reads that do not align to the reference

Insertions in the reference

The case of an insertion in the reference (or deletion from the target genome) is easily identified from the characteristic alignment of the reads to this reference genome. The reads that span the insertion point match two disjoint regions of the reference – the areas adjacent to the insertion (see Figure 4). The relative placement of the reads in the target genome is easily determined from the alignments to the reference; therefore all the reads spanning the insertion point, as well as those in the surrounding areas, can be placed into the same contig.

Rearrangement

Rearrangements between the two genomes pose the most difficult challenge to comparative assembly. Our program must cope with the many possible combinations of rearrangements and inversions (a special case of rearrangement) while avoiding mis-assemblies. A conservative approach was chosen by restricting ourselves to resolving areas that have the signature of an insertion in the reference — a situation easily resolved as shown in the previous section. The example shown in Figure 5 identifies a simple rearrangement where sections II and III in the target appear in a different order in the reference. AMOS-Cmp detects this type of rearrangement using the alignment of the reads spanning the boundaries between sections of the target genome. The signature of read A identifies section III as an insertion in the reference between sections I and II;

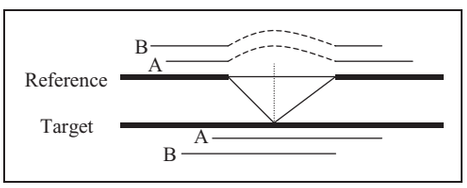

Figure 4: Insertion into the reference. The alignment of reads to the reference (top) indicates the presence of the insertion. Dashed lines indicate the 'stretch' of the reads needed to align to the reference

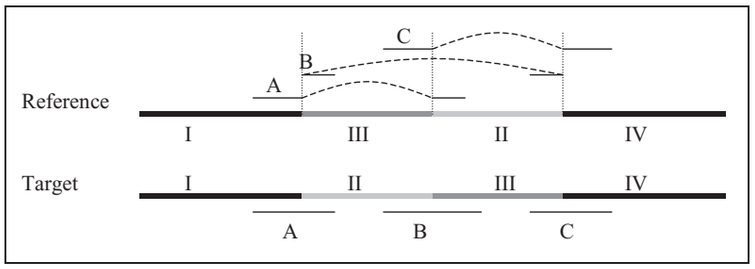

Figure 5: Signature of a genome rearrangement between the target and the reference. Regions II and III from the target appear in a different order in the reference. Reads A, B and C match the reference in disjoint locations – the dashed lines connect sections of a read that are adjacent in the target genome

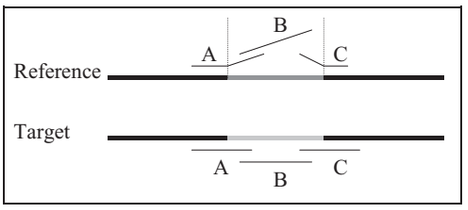

Figure 6: Signature of a region that has diverged significantly between the reference and the target genomes (the grey areas). Portions of the reads not matching the reference are shown at an angle

therefore AMOS-Cmp creates a single contig spanning these two regions. Similarly, read C identifies section II as an insertion between sections III and IV, allowing our assembler to correctly assemble them. Read B's alignment does not fall into the 'insertion into the reference' category, therefore we break the contigs when reaching it. The target genome will thus be reconstructed in two pieces: I+II and III+IV. The scaffolding stage will later join the two pieces together using mate-pair information.

Divergent DNA

Divergent DNA

Regions of divergent DNA occur when one or both genomes have mutated sufficiently that the sequences can no longer be aligned. When the difference between corresponding sections of the reference and target genomes is greater than the alignment threshold, we are faced with the situation depicted in Figure 6. The alignment signature of the reads is identical to that caused by an insertion in the target (Figure 2), causing the comparative assembler to break the region into two contigs (one ending at read A, and the other starting with read C) and to leave out the reads contained in the divergent region as singletons.

Breakpoints in alignments

Distinguishing between sequencing errors and true polymorphisms

Sequencing errors *v.* polymorphisms

Errors occur at a low but non-zero rate in all shotgun sequencing projects. Some of the most common examples of errors are (1) chimeric reads, introduced during the cloning process; (2) sequencing errors, caused by the sequencing or base-calling processes; and (3) clipping errors, introduced during the post-processing of the sequence data. The alignment signatures of reads containing such errors are, unfortunately, similar to those caused by true polymorphisms between the target and reference genomes. Chimeric reads look like rearrangements, while sequencing and clipping errors have the same signature as insertions into the target genome. In order to limit the effects of clipping errors, all the reads were trimmed using the *lucy* package,[16] eliminating both the vector sequence, and the regions with error rates higher than two errors in each 50 base-pair window.

A *breakpoint* is defined as the situation when the best alignment of a read to the reference (as identified by the LIS algorithm) consists of more than one contiguous segment, or a single segment that does not extend to the end of the read. A majority rule is used to identify the reads containing errors. For all the reads whose alignment to the reference contains breakpoints, other reads are identified that contain the same breakpoints, as well as reads whose alignment spans the breakpoint. The hypothesis that has the most support is then chosen. An example is shown in Figure 7. Read A probably contains an error because the alignments of both reads B and C disagree with it. Reads D and E,

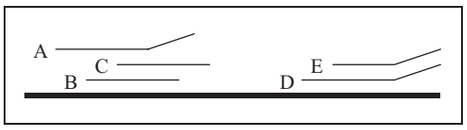

Figure 7: Detecting errors. Read A is probably incorrect since it disagrees with reads B and C. Reads D and E probably indicate a polymorphism between the target and the reference

Test data available at NCBI trace archive

however, probably indicate a true polymorphism because both their alignments agree.

Experimental setup

Another challenge is caused by short repeats that often flank polymorphisms (see, for example, Wu *et al.*[17]). Figure 8 shows the case of an insertion in the reference that is accompanied by the presence of a small repeat flanking the insertion. In order to correctly handle this situation, the LIS algorithm was modified to allow an overlap between adjacent alignments to the reference. When such an overlap is present in the alignment of a read to the reference, the positions of the reads following the insertion are adjusted to allow for the presence of the short repeat.

Implementation details

AMOS-Cmp is implemented as a pipeline executed by the runAmos program – a generic pipeline executor provided by the AMOS package (software available at the website[18]). This pipeline combines an invocation of the MUMmer alignment package with subsequent processing modules written in C++ and Perl. The code was only tested under RedHat Linux on an Intel-based workstation, and under OSF 5.1 on an Alpha server; however, it should be portable to most other UNIX variants with minimal changes.

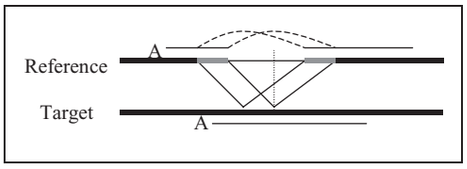

Figure 8: Effect of short flanking repeats on the alignment of a read to the reference in the case of an insertion in the reference. The repeat is shown in grey, and the dashed lines connect sections of read A that occur twice in the reference but once in A and in the target genome

RESULTS AND DISCUSSION

AMOS-Cmp was tested on several genome assembly problems, using the bacterium *Streptococcus agalactiae*, for which more than two distinct strains have been completely sequenced, as a model. *Streptococcus agalactiae* NEM316,[10] a strain sequenced at the Pasteur Institute whose complete sequence is available in GenBank, was used as a reference genome. *Streptococcus agalactiae* 2603 V/R,[9] a strain completed at TIGR for which all the shotgun sequence reads were available (also deposited in the NCBI Trace Archive, accessions 199934405 through 199969788) was assembled as a target. These two strains were chosen because they are very similar at the nucleotide level, yet contain numerous indels and repeats that may confuse a comparative assembly algorithm (see Figure 1). Six sets of reads were created from strain 2603, corresponding to a WGS project at 1, 2, 3, 5, 7 and 9-fold coverage respectively. For each data set the reads were selected in the order they were originally sequenced, to accurately simulate a sequencing project at the specified coverage.

Three different tests were performed; the results are presented in Tables 1 and 2. In the control experiment, an attempt was made to assemble strain 2603 using itself as a reference (columns headed '*v.* 2603' in the tables). This experiment allowed us to concentrate on the algorithm's ability to assemble the genome correctly, and especially to resolve repeats. In the second experiment, strain 2603 was assembled using strain NEM 316 as a reference (columns headed '*v.* NEM 316' in the tables). In a third test the WGS reads were assembled from scratch using Celera Assembler,[5] thus providing a baseline for

Table 1: Overall statistics on assemblies produced by AMOS-Cmp

×	*v.* 2603			*v.* NEM 316			CelAsm		
	N	Total contig size	N50	*N*	Total contig size	N50	*N*	Total contig size	N50
1	604	1,001,743	0	527	839,315	0	585	903,184	0
2	619	1,593,364	2,294	586	1,393,287	1,479	657	1,488,287	1,595
3	443	1,856,394	5,707	450	1,640,231	4,179	506	1,812,266	4,981
5	243	2,043,842	14,915	277	1,829,976	10,395	293	2,046,730	12,458
7	144	2,100,541	27,364	198	1,891,527	18,142	189	2,110,396	21,926
9	86	2,119,579	42,679	155	1,919,237	24,239	130	2,132,490	33,953

Table 2: Overall statistics of alignment to *S. agalactiae* strain 2603 V/R. The total gap size indicates the total number of bases missing from the assembled contigs after mapping them to the finished genome. The column marked LW represents the theoretical estimate of coverage based on Lander–Waterman[19] statistics

×	*v.* 2603			*v.* NEM 316			CelAsm			LW
	N gaps	Total ga size	% genome covered	*N* gaps	Total gap size	% genome covered	*N* gaps	Total gap size	% genome covered	% genome covered
1	588	1,168, 208	45.92	511	1,329,996	38.43	562	1,261,419	41.61	39.31
2	596	577,987	73.24	552	778,491	63.96	601	679,386	68.55	74.10
3	430	301,899	86.02	415	530,417	75.45	455	365,736	83.07	89.88
5	232	119,917	94.45	240	347,697	83.90	257	153,824	92.88	98.56
7	132	62,410	97.11	155	292,068	86.48	146	81,406	96.23	99.79
9	80	43,408	97.99	110	270,210	87.49	97	61,544	97.15	99.97

AMOS-Cmp performance

the expected performance of a typical assembly program (columns headed 'CelAsm' in the tables). For each of these tests we collected statistics on the size of the contigs generated by the assembly programs (number of contigs, total contig size and N50 contig sizes in Table 1) and on how well these contigs cover the target genome (number and size of the gaps in the alignment as well as percentage of the genome covered in Table 2). The consensus sequence of the contigs was aligned to the complete sequence of strain 2603 using the MUMmer package, retaining only those contigs that matched the reference over more than 90 per cent of their length at better than 90 per cent similarity. (The few contigs not satisfying these criteria were removed from the analysis in Table 2; note that these exclusions do not affect the overall statistics as the excluded contigs were few in number and shorter than 2 kbp.)

Unassemblable regions

The comparative assembly using strain 2603 as a reference (the control) significantly outperformed the scratch assembly, producing larger contigs and better coverage of the target genome at all depths of coverage, from 1× to 9×. This result demonstrates that AMOS-Cmp can utilise the alignment to the reference to identify and resolve repeats that cannot be correctly resolved by Celera Assembler. The highly repetitive region in the first 5 Kbp of both test genomes caused the only significant misassembly in the Celera Assembler data – a misassembled 17 Kbp contig – while all comparative assembly experiments (including those with NEM 316 as the reference) correctly resolved the region. Secondly, this result shows that AMOS-Cmp can assemble together reads that overlap by as little as 10 bp,

while Celera Assembler requires a minimum overlap of 40 bp. This greater sensitivity is possible because the read overlaps are inferred from longer (and therefore more specific) alignments to the reference, so avoiding the false overlaps that would result from the use of short overlaps in conventional assembly programs.

Estimating the extent of unassemblable regions

The assemblies using strain NEM 316 as a reference appear, in at least one respect, to be worse than those created by the Celera Assembler. Table 2 shows that the NEM 316 reference assembly consistently left more bases unassembled than the scratch assembly. This apparent shortcoming of our method can be explained by (a) insertions in the 2603 strain with respect to the NEM 316 strain, and (b) regions of significant sequence divergence. By its very design (as explained above), a purely comparative assembler cannot assemble such regions, because they simply do not exist in the reference genome. The total length was calculated of all regions in the 2603 strain that had no homologue in the NEM 316 strain, and 254,657 bases were found that cannot be assembled by a comparative assembler. If this amount is subtracted from the total gap size column in Table 2, it can be seen that AMOS-Cmp consistently outperforms the scratch assembly on those regions for which an assembly is possible.

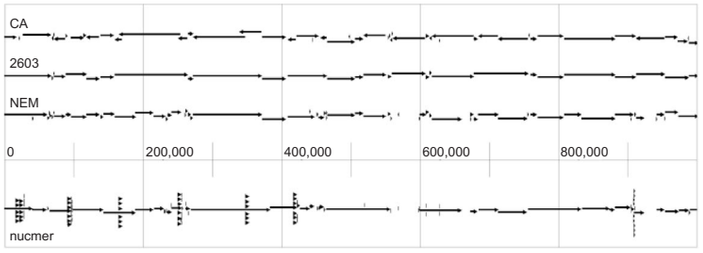

Figure 9: Assemblies of the first megabase of strain 2603 at 9× coverage. The rows correspond (top to bottom) to: CA – scratch assembly contigs created by Celera Assembler; 2603 – AMOS-Cmp contigs created using strain 2603 as a reference; NEM – AMOS-Cmp contigs using strain NEM 316 as a reference; and nucmer – the alignment of strain NEM 316 to strain 2603. Stacked arrows in the bottom row correspond to repeats

To illustrate further the differences between assemblies, all contigs from the three experimental assemblies are mapped to the first megabase of *S. agalactiae* 2603. The top three rows in Figure 9 correspond to the alignment of these contigs to the completed genome of strain 2603. The bottom row shows an alignment of strain NEM 316 to strain 2603, highlighting both repeats (arrows stacked on top of each other) and regions of dissimilarity (gaps in the alignment). Several features are immediately apparent in this figure. First of all, Celera Assembler contigs tend to end at repeat boundaries, while both of the comparative assemblies manage to correctly span many of these repeats. Second, the NEM 316 reference assembly does not cover the regions of dissimilarity

Table 3: Portion of the genome that cannot be assembled through the comparative assembly method computed for four pairs of closely related genomes. For each pair the genome being assembled (target) and the genome used as a reference are identified, together with their sizes. The number of bases that cannot be assembled by AMOS-Cmp as well as the fraction of the target genome contained in these regions are given

Reference genome (# bases)	*Staphylococcus epidermidis* RP62A [unpublished] (2,616,530)	*Streptococcus pyogenes* MGAS315[26] (1,900,521)	*Streptococcus pyogenes* MGAS8232[21] (1,895,017)	*Streptococcus agalactiae* 2603 V/R[9] (2,160,267)
Target genome (# bases)	*Staphylococcus epidermidis ATCC* 12228[22] (2,499,279)	*Streptococcus pyogenes* SF370 serotype M1[23] (1,852,441)	*Streptococcus pyogenes* SF370 serotype M1[23] (1,852,441)	*Streptococcus pyogenes* SF370 serotype M1[23] (1,852,441)
Region that cannot be assembled (# bases, %)	143,007 (5.72%)	148,192 (7.99%)	142,495 (7.69%)	1,640,396 (88.55%)

Effects of assembly quality on scaffolding

between the two strains. The breaks in the contiguity of the NEM assembly occur at locations where either there is a break in the alignment of the two strains, or the break is shared by the other two assemblies (probably because of a lack of coverage in the original WGS sample).

Effects of genome similarity on quality of assembly

These results allow us to start to answer the question: how similar do two genomes have to be in order for the comparative assembly method to work? The answer lies in the nature of the differences between the two genomes. Insertions in the target genome cannot be assembled through this method, and rearrangements will often lead to breaks in the contigs. In order to provide the reader with a sense of what genomes can be assembled with our technique we computed the extent of the regions that cannot be assembled through the comparative assembly method for several pairs of genomes. The results are summarised in Table 3.

AMOS-Cmp uses less resources than traditional assemblers

AMOS-Cmp correctly assembles repeats

In each of the three pairs of bacteria representing different strains of the same species, AMOS-Cmp is able to assemble 92–94 per cent of the target genome. When the target and reference genomes are more distantly related, as is the case with the distinct species *S. agalactiae* 2603V/R and *S. pyogenes* SF370, our method is only useful for a relatively small proportion of the target genome (just 212 out of 1850 Kbp in this case).

Scaffolding

As a final test of our method the contigs produced in the '*v.* NEM' and 'CelAsm' tests were scaffolded using the Bambus package.[5] This test examines a more subtle characteristic of assemblies. Most assemblers will incorrectly place at least some reads from repetitive sequences into a wrong copy of those repeats. The misplaced reads, together with their associated mates, lead to incorrect linking information being provided to the scaffolding program, thereby complicating the scaffolding problem and often decreasing the quality of the scaffolds. The resulting scaffolds were similar in number and size, showing that not only is the assembly of the repeats similar in the two sets of assemblies, but also that mate-pair information is able to offset the smaller size of the '*v.* NEM' assembly (see Table 1). Furthermore, owing to the absence of any significant rearrangements between the chosen genomes, Bambus will produce a single scaffold spanning the entire *S. agalactiae* genome by using the alignment information to infer contig adjacency.

Timing

Because AMOS-Cmp skips the most computationally expensive part of assembly, the overlap step, it is faster than a scratch assembly. Our assembly of *S. agalactiae* 2603 onto the NEM 316 reference, using 31,779 reads as input, took just under 3.5 minutes on a desktop Pentium computer (2.8 GHz) running Linux. It also uses far less memory than a conventional assembler: just 178 megabytes (MB) of memory were used for the most memory-intensive step. In contrast, the Celera Assembler (which is far faster than earlier generation assemblers such as phrap[3] and TIGR Assembler[24]) on the same data took 23 minutes and 1.9 GB of memory on an HP Alpha ES40 computer running OSF1 (on the same machine AMOS-Cmp took just under 9 minutes).

Comparative *v. ab initio* assembly

The results on timing described in the previous section indicate the comparative assembler is significantly faster than a typical *ab initio* assembler. While the difference does not appear significant, 9 versus 23 minutes, the reader should bear in mind the size of the genome being assembled. The assembly process currently takes more than one week for large eukaryotes and, frequently, multiple assemblies are performed in order to produce the best possible reconstruction

of the genome. Comparative assembly will dramatically speed up this process when multiple strains of the same large eukaryote are being sequenced. Furthermore, the simplicity of the process, as well as the limited number of parameters that can be tuned, make comparative assembly accessible to most biologists. The assembly of large eukaryotes is currently performed only at large centres with significant bioinformatics expertise.

At a qualitative level, comparative assembly methods are less sensitive to repeats and haplotype differences – regions that provide a significant challenge to existing assembly algorithms. The comparative assembly method is specifically designed to handle polymorphisms between DNA molecules, therefore it can gracefully handle the presence of multiple, divergent, haplotypes in the data being assembled. Furthermore, repeats are accurately identified and resolved, in contrast to traditional assemblers that rely on statistical tests to identify repeats in the shotgun data (such tests are sometimes incorrect due to non-randomness in the shotgun libraries).

Finally, the comparative assembly method can be used to assess the natural diversity of bacterial strains found through environmental sequencing.[25,26] Thus, from a pool of bacteria found in an environment, one could generate a hybrid assembly of all strains that are similar to a known bacterium – a task difficult to achieve with existing assemblers.

CONCLUSION

The comparative assembly method described in this paper performs very well when compared with a standard assembly program such as Celera Assembler. The AMOS-Cmp assembler outperforms traditional assembly programs in both speed and memory requirements, thereby allowing scientists with limited computing resources (ie a standard desktop computer) to run their own genome assemblies. Our algorithm correctly handles several classes of repeats that sometimes confound other assemblers. In addition, our method allows the assembly of reads that overlap by 10 base pairs or fewer, allowing for the contiguous assembly of low-coverage areas. Finally, the comparative assembler provides useful information even at extremely low-coverage levels where traditional assembly methods have limited use. Knowledge that a particular read matches the reference genome leads to a significant improvement over standard assembly programs where distinguishing between unassembled (singleton) reads due to lack of coverage and those due to contaminants or sequencing errors is practically impossible. For many recent survey sequencing projects, where funds are scarce, low-coverage WGS sequencing (often just 1× to 3×) is becoming increasingly common.[27]

The comparative assembly method does have drawbacks, most obviously its dependence on the sequence of a closely related genome. Rearrangements between the two genomes cause fragmentation of the contigs in the resulting assembly, while some regions cannot be assembled owing to lack of similarity to the reference genome. Nonetheless, the comparative assembly method should have an increasing number of applications with the ever-growing number of genomes being deposited in public databases.

Acknowledgments

This work was supported in part by National Institutes of Health Grants R01-LM06845 and R01-LM007938 to SLS. The authors would like to thank Steven R. Gill for providing them with the sequence of *Staphylococcus epidermidis* RP62A prior to publication.

References

1. Read, T. D., Salzberg, S. L., Pop, M. *et al.* (2002), 'Comparative genome sequencing for discovery of novel polymorphisms in *Bacillus anthracis*', *Science*, Vol. 296(5575), pp. 2028–2033.
2. Peltola, H., Soderlund, H. and Ukkonen, E. (1984), 'SEQAID: A DNA sequence assembling program based on a mathematical

model', *Nucleic Acids Res*., Vol. 12(1), pp. 307–321.

3. Green, P. (1994), 'PHRAP documentation: ALGORITHMS', (URL: http://www.phrap.org).
4. Sutton, G., White, O., Adams, M. D. and Kerlavage, A.R. (1995), 'TIGR Assembler: A new tool for assembling large shotgun sequencing projects', *Genome Sci. Technol.*, Vol. 1, pp. 9–19.
5. Myers, E. W., Sutton, G. G., Delcher, A. L. (2000), 'A whole-genome assembly of *Drosophila*', *Science*, Vol. 287(5461), pp. 2196–2204.
6. Batzoglou, S., Jaffe, D. B., Stanley, K. *et al.* (2002), 'ARACHNE: A whole-genome shotgun assembler', *Genome Res*., Vol. 12(1), pp. 177–189.
7. Pop, M. (2004), 'Shotgun sequence assembly', in 'Advances in Computers', Zelkowitz, M., Ed, Elsevier Science, San Diego, CA, pp. 194–248.
8. Kurtz, S., Phillippy, A., Delcher, A. L. *et al.* (2004), 'Versatile and open software for comparing large genomes', *Genome Biol.*, Vol. 5(2), p. R12.
9. Tettelin, H., Masignani, V., Cieslewicz, M. J. *et al.* (2002), 'Complete genome sequence and comparative genomic analysis of an emerging human pathogen, serotype V S*treptococcus agalactiae*', *Proc. Natl Acad. Sci. USA*, Vol. 99(19), pp. 12391–12396.
10. Glaser, P., Rusniok, C., Buchrieser, C. *et al.* (2002), 'Genome sequence of *Streptococcus agalactiae*, a pathogen causing invasive neonatal disease', *Mol. Microbiol.*, Vol. 45(6), pp. 1499–1513.
11. Delcher, A. L., Phillippy, A., Carlton, J. and Salzberg, S. L. (2002), 'Fast algorithms for large-scale genome alignment and comparison', *Nucleic Acids Res*., Vol. 30(11) pp. 2478–2483.
12. Delcher, A. L., Kasif, S., Fleischmann, R. D. *et al.* (1999), 'Alignment of whole genomes', *Nucleic Acids Res*., Vol. 27(11), pp. 2369–2376.
13. Gusfield, D. (1997), 'Algorithms on strings, trees, and sequences', Cambridge University Press, Cambridge.
14. Anson, E. L. and Myers, E. W. (1997), 'ReAligner: A program for refining DNA sequence multi-alignments', in 'Proc. 1st Int. Conf. Computational Molecular Biology', ACM Press, New York.
15. Pop, M., Kosack, D. S. and Salzberg, S. L. (2004), 'Hierarchical scaffolding with Bambus', *Genome Res*., Vol. 14(1), pp. 149–159.
16. Chou, H. H. and Holmes, M. H. (2001), 'DNA sequence quality trimming and vector removal', *Bioinformatics*, Vol. 17(12), pp. 1093–1104.
17. Wu, M., Sun, L. V., Vamathevan, J. *et al.* (2004),' Phylogenomics of the reproductive parasite *Wolbachia pipientis* wMel: A streamlined genome overrun by mobile genetic elements', *PLoS Biol.*, Vol. 2(3), p. E69.
18. URL: http://www.tigr.org/software/AMOS
19. Lander, E. S. and Waterman, M. S. (1988), 'Genomic mapping by fingerprinting random clones: A mathematical analysis', *Genomics*, Vol. 2(3), p. 231–239.
20. Beres, S. B., Sylva, G. L., Barbian, K. D. *et al.* (2002), 'Genome sequence of a serotype M3 strain of group A *Streptococcus*: Phage-encoded toxins, the high-virulence phenotype, and clone emergence', *Proc. Natl Acad. Sci. USA*, Vol. 99(15), pp. 10078–10083.
21. Smoot, J. C., Barbian, K. D., Van Gompel, J. J. *et al.* (2002), 'Genome sequence and comparative microarray analysis of serotype M18 group A *Streptococcus* strains associated with acute rheumatic fever outbreaks', *Proc. Natl Acad. Sci. USA*, Vol. 99(7), p. 4668–4673.
22. Zhang, Y. Q., Ren, S. X., Li, H. L. *et al.* (2003), 'Genome-based analysis of virulence genes in a non-biofilm-forming *Staphylococcus* epidermidis strain (ATCC 12228)', *Mol. Microbiol.*, Vol. 49(6), pp. 1577–1593.
23. Ferretti, J. J., McShan, W. M., Ajdic, D. *et al.* (2001), 'Complete genome sequence of an M1 strain of *Streptococcus pyogenes*', *Proc. Natl Acad. Sci. USA*, Vol. 98(8), pp. 4658–4663.
24. Pop, M., Salzberg, S. L. and Shumway, M. (2002), 'Genome sequence assembly: algorithmic issues and practical considerations', *IEEE Computer*, Vol. 35(7), pp. 47–54.
25. Tyson, G. W., Chapman, J., Hugenholtz, P. *et al.* (2004), 'Community structure and metabolism through reconstruction of microbial genomes from the environment', *Nature*, Vol. 428(6978), pp. 37–43.
26. Venter, J. C., Remington, K., Heidelberg, J. F. *et al.* (2004), 'Environmental genome shotgun sequencing of the Sargasso Sea', *Science*, Vol. 304(5667), p. 66–74.
27. Kirkness, E. F., Bafna, V., Halpern, A. L. *et al.* (2003), 'The dog genome: Survey sequencing and comparative analysis', *Science*, Vol. 301(5641), pp. 1898–1903.