Mega Weaver: A Simple Iterative Approach for BAC Consensus Assembly

Daolong Wang [1], Mario Lauria [2], Bo Yuan [3 *], and Fred A. Wright [1, 4 *]

1 Department of Biostatistics, University of North Carolina, Chapel Hill, North Carolina 27599; Departments of 2 Computer and Information Science, and 3 Biomedical Informatics and Pharmacology, The Ohio State University, Columbus, Ohio 43210; 4 The Lineberger Cancer Center, UNC Chapel Hill.

* Corresponding authors: fwright@bios.unc.edu

Abstract

Hierarchical genome assembly can be divided into three distinct stages: sequencing and assembling shotgun reads for each of a series of selected BAC clones; assembling the resulting fragments into BAC consensus sequences; and mapping and orienting the BAC consensus according to external positional information. We report a new approach for BAC consensus assembly that relies on iterative layouts of overlapping sequence, with no need for prior masking of repetitive sequence. The approach includes major steps of quality filtering and an iterative screening algorithm within and between clusters of overlapping BAC fragments. Each step includes numerous minor steps designed to detect false overlaps at minimal expense in true overlaps. In contrast to dynamic algorithms, our approach attempts to minimize false overlaps before attempting to form BAC consensus sequences. We show that false overlaps are reduced to a degree that final BAC consensus assembly is straightforward under a coordinate system described in the paper. Using human chromosome 22 and a range of simulation conditions, an average of 98.1% false overlaps could be removed, while 6.7% of true overlaps were not detected. The final assembled BAC consensus sequences were nearly optimal, and support the usefulness of our approach for future hierarchical sequencing projects.

*Keywords***:** DNA sequence, BAC consensus, assembly, Algorithm

1. Introduction

Despite the rapid progress over the last few years in sequencing large, complex genomes such as the human and mouse, the best strategy for sequencing and assembling such genomes remains a matter of vigorous debate (Green 1997; Weber and Myers 1997; Waterston et al. 2002). Genome researchers are well aware of the trade-offs inherent in a whole-genome shotgun approach vs. a clone-by-clone hierarchical approach (which employs shotgun sequencing at the clone level). Certain economies of scale may favor whole-genome shotgun sequencing, but even recently this view has been questioned (Green 2002). For the future sequencing of repeat-rich genomes, the hierarchical approach using bacterial clones will offer a continued advantage in reducing repeat-induced assembly errors (Waterston et al. 2002). Thus, although the assembly of the human genome has moved far beyond the draft stage, computational improvements in hierarchical assembly will remain of continued relevance.

In the hierarchical strategy, a series of mapped bacterial clones (*bacterial artificial chromosome*s, BACs) are selected with the intent of covering the genome with a minimum of overlap to reduce unnecessary sequencing costs. Each of the clones in this tiling path is shotgun-sequenced at (ideally) 10X or greater coverage (International Human Genome Sequencing Consortium 2001). Initial sequence assembly proceeds with shotgun reads within each BAC, forming one or more contigs (called *BAC fragments* hereafter). The next step, which is the focus of this paper, involves the assembly of BAC fragments from different BACs into longer consensus contigs (the *BAC consensus*). The procedures for high-quality BAC consensus assembly have received relatively little attention, although the BAC consensus provides critical input for the final genome assembly. The final genome assembly involves numerous additional steps that are beyond the scope of this paper, including scaffolding, ordering and orienting the assembled BAC contigs using external positional information and genomic landmarks.

Numerous existing approaches for assembling shotgun reads (~500 bp) within BAC clones of moderate size (~150 Kb) (Peltola et al. 1984; Huang 1992; Huang and Madan 1999; Green 1994; Sutton et al. 1995; Kim et al. 1999; Myers et al. 2000; Pevzner et al. 2001; Batzoglou et al. 2002). However, none of these approaches has been applied in full comparisons of BACs/BAC fragments across the genome, partly due to computational constraints. To our knowledge, only one published approach (GigAssembler, Kent and Haussler 2001) has been designed especially for BAC consensus assembly using pre-assembled BAC fragments as input, and formed the basis of the initial public human genome drafts (International Human Genome Sequencing Consortium 2001; Kent and Haussler 2001). GigAssembler also conducts the further steps of ordering and orienting the BAC consensus into mapped scaffolds (Kent and Haussler 2001).

GigAssembler relies on external mapping information (via fingerprint clone contigs, FPC) to pre-identify groups of BACs with presumed overlap. Each FPC typically includes only 10-100 BACs, and so by restricting BAC consensus assembly to an FPC the computational burden is greatly reduced, and smaller regions may be less prone to error produced by modestly repetitive sequence. However, the FPC is created using a statistical evaluation of restriction enzyme digestion patterns (Gillett et al. 1996; Soderlund et al. 1997), inevitably containing mapping errors. The extent of FPC mapping error has not been carefully documented, and the FPC map repository (http://genome.wustl.edu) is manually edited to accord with other data sources. Several lines of evidence suggest the FPC mapping error is nontrivial: (1) overlapping BAC sequences show that the tiling coverage of the human genome, selected via FPC to achieve uniformity on the genome with no multiple overlaps, often is in fact redundant to a depth of several-fold BAC coverage; (2) by comparing the original FPC map (June 16 2000 freeze) with the NCBI Build 30 of the human genome, we found that the FPC map (except the finished chromosomes 21 and 22) show substantial deviation from the assembly; (3) sequence comparisons of BACs across FPCs on many chromosomes exhibit sequence overlap that can be parsimoniously explained as misplacement of BACs in FPCs. Moreover, the breakpoints between FPC contigs do not necessarily represent sequencing gaps, so that BACs straddling the breakpoint are not compared in GigAssembler; and (4) integration of independently mapped-consensus transcripts into individual BACs show significant inconsistencies (5%) for the order and orientation of corresponding BACs in the published human genome draft (Zhou et al. 2001).

The National Center for Biotechnology Information (NCBI) is now responsible for the regular updating of human genome assembly. The NCBI assembly does not require FPCs for clustering BACs, according to the brief introduction on the NCBI site (http://www.ncbi.nlm.nih.gov/genome/guide/build.html). However, the procedures in creating the BAC consensus are not fully detailed, so it is difficult to assess how improvements may be made.

In addition to computational burden, another major obstacle in large, complex genome assembly arises from the considerable amount of repetitive sequence in the genome, causing misassembly if not handled carefully. Highly repetitive sequences are commonly masked when conducting genome assembly (Kent and Haussler 2001), although at the scale of BAC fragments, these "repeats" may be unique enough to warrant retention for the BAC consensus. Moreover, it may be difficult to specify repeat sequences before attempting the assembly.

In this report, we propose a new algorithm (Mega Weaver) for assembling BAC fragments into a BAC consensus based on all-by-all comparison of BAC fragments, without relying on any predefined clusters of BACs. In general, it applies the "overlap-layout-consensus" framework familiar in genome assembly. However, our approach focuses more on cleaning up false overlaps before final layout and consensus phases, i.e., identifying pairwise matches of BAC fragments that are not consistent with a final layout. After initial filtering of low-quality overlaps, Mega Weaver uses an iterative procedure to remove false overlaps and minimize the influence of repetitive sequences. Mega Weaver does not require repeat-masking of the input BAC fragments. As we will later document, true BAC sequence overlaps can be readily identified in chromosome-wide comparisons using our approach, and the indications are that it will scale up to genome-wide comparisons. Moreover, the great majority of false overlaps can be identified and

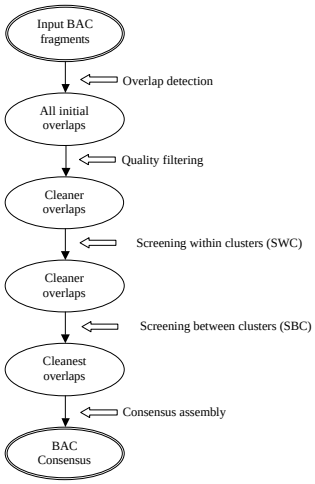

Figure 1. Procedure for BAC consensus assembly

removed. We conducted careful simulation studies to evaluate this algorithm using human chromosome 22 as a hypothetical genome. The results show that Mega Weaver can generate near-optimal BAC consensus assembly for realistic BAC tiling coverage and sequencing error rates.

2. Methods

Mega Weaver takes preassembled BAC fragments as input, and produces BAC consensus sequence as output. It first identifies all pairwise overlaps and conducts quality filtering to eliminate low quality (likely false) overlaps. After quality filtering, two subsequent iterative procedures are performed to identify additional false overlaps. Finally, BAC consensus sequences are generated through sequence layout with the cleaned overlaps. In addition to a genome-wide sequence alignment, the use of iterative procedures allows efficient validation of overlaps across the genome – a key feature that distinguishes Mega Weaver from other greedy assembly algorithms. Figure 1 shows the overall schema of our assembly procedure. Overlapping fragments (identified via Megablast, Zhang et al., 2000) are used to define clusters (described further in Methods) of possibly contiguous sequence, and screening within clusters (SWC) and screening between clusters (SBC) proceed iteratively. The iterations are further detailed in Figure 2, and involve swapping of query/subject identities of fragments to accord with the Megablast input format.

2.1 Sequence Data

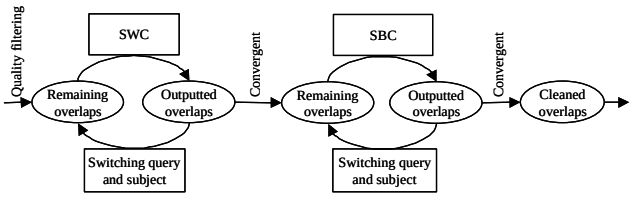

Figure 2. Iterative procedures for screening false overlaps within and between clusters

The input sequence data consists of BAC fragments preassembled from shotgun reads (possibly including full-length BACs as a special case). The assembly of BAC fragments (often with PHRAP, http://www.phrap.org) ensures that for the most part no true overlaps remain between fragments within the same BAC. As this paper was prepared, the average length of BAC fragments from the public human genome assembly had reached ~60 kb, while an ever-increasing number of BACs had reached the finished stage as a single consensus of ~150 kb. The current generation of sequencers has an error rate on the order of 10^{-2} per bp or less (Ewing and Green, 1998). With sequencing depth of coverage at ~8X or more (International Human Genome Sequencing Consortium 2001), the unresolved error rate in BAC fragments is perhaps 10^{-4} or lower, and thus the effects of sequencing errors are not heavily emphasized in this report. However, several of our simulation conditions do consider the effects of conservatively high error rates in the BAC fragments (up to 10^{-2} per bp). No repeat masking or other sequence preprocessing is required. Sequence quality information is not required, although we do account for reduced quality at the ends of each fragment.

2.2 Identifying pairwise overlaps

The first step of our assembly process is to find all pairwise overlaps between BAC fragments across a whole genome or chromosome (assuming BACs have been correctly assigned to chromosomes). Several existing programs can be used to identify overlaps, and provide useful scores for overlap quality. We have found that Megablast (Zhang et al. 2000) can perform this task efficiently, and is easily implemented on standard workstations.

Figure 3 shows the structure of a typical overlap with details of alignment procedure, detectable by Megablast under output option "-D 3". Megablast compares so-called *query* sequences to *subject* (database) sequences. For our purposes we need to perform all $N(N\text{-}1)/2$ pairwise comparisons of N total sequences, and there is no meaningful distinction between query and subject. However, we adopt this nomenclature to adhere to the Megablast format.

In order to obtain highly accurate overlap positions (q_1, q_2, s_1, and s_2, see Figure 3), we set the filter option for running Megablast as "–F F". However, applying this option to the full-length BAC fragments dramatically slows Megablast, and we have devised a two-stage strategy to speed up the process. In the first stage, the two ends of each sequence are cut off at a specified length (e.g. 500 bp) and used as query. These *end query* sequences are then compared with full-length subject sequences. Although the creation of end queries doubles the number of comparisons, the total comparison time is greatly reduced. The choice of end query length involves inherent trade-offs in specificity and computing time, and invalid overlaps will increase the burden for the second stage. We determined from our simulations that end

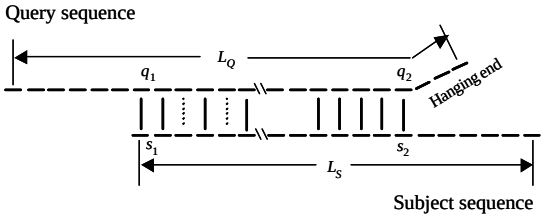

Figure 3. A typical pairwise sequence overlap. L_Q and L_S are lengths in bp of the sequences. Base positions q_1 and q_2, and s_1 and s_2, counted from the left ends of the sequences, define the overlap region with respect to each sequence. Base matches are indicated by vertical bars, along with mismatches (dashed bars) and gaps (no bars). A "hanging end" is defined as the shorter of any unmatched ends of the two sequences.

sequences of 500 bp were sufficient to detect nearly all true overlaps with high efficiency. The second stage is to verify potential pairwise overlaps in Megablast by performing full-length comparisons of only those sequences identified as overlapping in stage one.

The net effect of the two-stage approach was to reduce the computing time for the pairwise comparisons by ~ 5 times or more. Our simulation studies using chromosome 22 as a hypothetical genome indicate that ~99.9% of true overlaps can be identified using the above approach, with Megablast false alignment expectation set to 1e-10 (option "-e"). However, false overlaps caused by repetitive sequences can account for more than half of all detected overlaps, depending on the average sequence length and error rates.

2.3 Initial quality filtering

Simply increasing the stringency (option "-e") cannot remove all false overlaps, especially those with high overlap quality, while higher stringency will exclude many true overlaps of moderate quality. We have considered a number of quality indicators derived from Megablast output, and have found percent identity, length of overlap region, length of hanging ends, and differences in lengths of overlap regions to be the most useful. BAC fragments that are fully contained in several other sequences, or contained in the same sequence multiple times, are considered suspicious and are also removed at this step.

Further removal of false overlaps in the remaining overlaps is only possible by checking for mutual compatibility of overlaps by consensus layout. Mega

Weaver uses iterative procedures to identify and remove incompatible overlaps (described below). Under a variety of scenarios, the algorithm can reliably reduce false overlaps to less than 1% of all presumed overlaps while retaining ~95% of true overlaps (see Results).

2.4 Screening false overlaps within clusters (SWC)

2.4.1 Clustering and hash tables

This step identifies groups (*super clusters*) of BAC fragments that are mutually connected through transitive pairwise overlaps. Pairwise overlapping BAC fragments

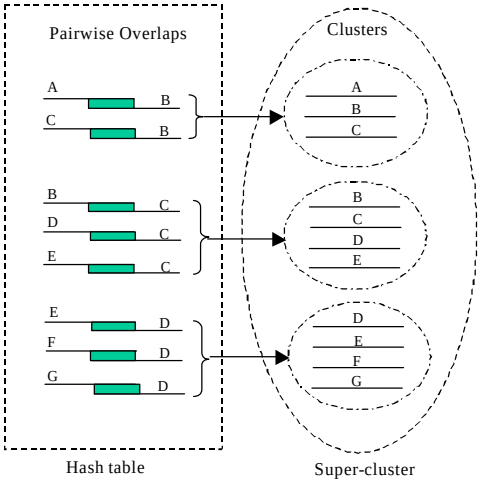

Figure 4. Creation of a super cluster and hash table based on pairwise overlaps.

that remain after the initial quality filtering are first grouped into *clusters* (containing 2 or more fragments), each of which has one common (subject) sequence that overlaps with all other (query) sequences in the cluster. Super clusters are formed by grouping together all clusters that share one or more sequences (query and/or subject) (Figure 4). By definition, BAC fragments from different super clusters do not overlap. Thus, overlap validation via sequence layouts need only be performed within each super cluster, greatly reducing computation. Efficient overlap validation via sequence layout requires rapid retrieval of pairwise overlap information, for which we use hash tables. Once super clusters have been generated, one hash table stores all the pairwise overlaps for each super cluster, using pairwise sequence IDs as the keys.

2.4.2 A coordinate system and fitness scores

Overlap validation is carried out by building a sequence coordinate system for each cluster in the super cluster. The system assigns a pair of coordinate values (*P*, *O*) to each BAC fragment based on its overlap position with the common subject sequence *S* (Figure 5*a*). *P* specifies the base position of the sequence left end in the coordinate system; *O* specifies the sequence orientation. The common sequence *S* is set at the origin with coordinate (*P* = 0, *O* = 1), while the coordinate for any other sequence (Q_i) in the cluster is determined by

$$\begin{cases} P_i = s_1 - q_1, \quad O_i = 1 & \text{if } s_1 < s_2 \\ P_i = s_1 - (L_Q - q_1 + 1), O_i = -1 & \text{if } s_1 > s_2 \end{cases},$$

where s_i, q_i, and L_Q are defined as in Figure 3. Different clusters in the same super cluster have different origins, and thus different coordinate systems. The coordinate system facilitates sequence layout and comparisons.

We also find it useful to define a "fitness score" F_i, a simple measure of how well each BAC fragment *i* fits in a cluster, later used to decide whether the sequence is likely to belong to the cluster. The fitness score is the sum of overlap scores that sequence *i* has with the remaining sequences in the cluster, or $F_i = \sum_{j \neq i} score_{ij}$, where $score_{ij}$ is the Megablast overlap quality score for sequences *i* and *j*.

2.4.3 Compatibility checks

The coordinate system for each cluster assumes that overlaps between queries and the common subject are true, so that all overlaps in the cluster should be mutually compatible. We can easily perform compatibility checks between inferred overlaps (based on the coordinate system) and the observed overlaps obtained from Megablast sequence comparison.

To infer the overlap between two sequences Q_1 and Q_2, we compute the base positions for their right ends (P_{right1} and P_{right2}) in the cluster coordinate system. Given the coordinates for the left ends of Q_1 and Q_2 (Figure 5*b*), it is apparent that

$$P_{right1} = P_{left1} + L_1 - 1,$$
$$P_{right2} = P_{left2} + L_2 - 1,$$

where P_{left1} and P_{left2} are the base positions for the left ends of Q_1 and Q_2, and L_1 and L_2 denote the respective sequence lengths. If $P_{left1} > P_{right2}$ or $P_{left2} > P_{right1}$, then there is no overlap expected between Q_1 and Q_2;

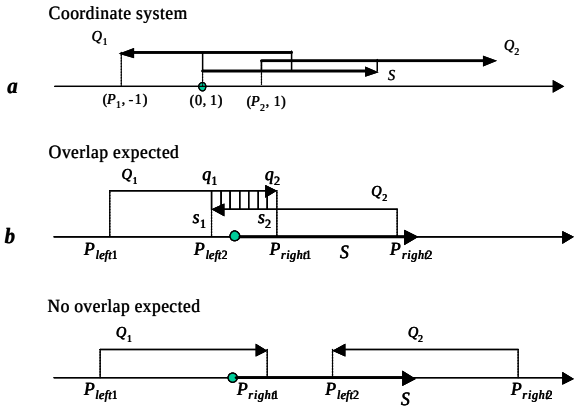

Figure 5. (*a*) Coordinate system built on a subject sequence for a cluster of three sequences. The subject sequence *S* (left end) is placed at origin; query sequences are placed according to their overlaps with the subject sequence. (*b*) Coordinate system giving the expected overlap between two sequences in a cluster.

otherwise they should overlap. If Q_1 and Q_2 are expected to overlap, then the expected overlap region (q_1 and q_2) on sequence Q_1 (as query) is

$$\begin{cases} q_1 = \max(1,\ P_{left2} - P_{left1} + 1) \\ q_2 = \min(L_1, P_{right2} - P_{left1} + 1) \end{cases} \text{ for } O_1 = 1, \text{ or}$$

$$\begin{cases} q_1 = L_1 - \max(1,\ P_{left2} - P_{left1} + 1) + 1 \\ q_2 = L_1 - \min(L_1, P_{right2} - P_{left1} + 1) + 1 \end{cases} \text{ for } O_1 = -1.$$

The expected overlap region (s_1 and s_2) on sequence Q_2 (as subject) is

$$\begin{cases} s_1 = \max(1,\ P_{left1} - P_{left2} + 1) \\ s_2 = \min(L_2, P_{right1} - P_{left2} + 1) \end{cases} \text{ if } O_2 = 1, \text{ or}$$

$$\begin{cases} s_1 = L_2 - \max(1,\ P_{left1} - P_{left2} + 1) + 1 \\ s_2 = L_2 - \min(L_2, P_{right1} - P_{left2} + 1) + 1 \end{cases} \text{ if } O_2 = -1.$$

Note that if $q_1 > q_2$, the assignments of q_1 and q_2 (and s_1 and s_2) must be swapped, because for overlaps detected by Megablast, q_1 is always less than q_2. Also, we require that the inferred overlap satisfy the same quality requirement (overlap region length) used for filtering pairwise overlaps in the initial quality filtering step; otherwise no overlap is inferred.

With the hash table, it is easy to check the compatibility of observed vs. inferred overlaps between Q_1 and Q_2. Compatibility is assumed if there is neither inferred nor observed overlap between Q_1 and Q_2, or the inferred overlap deviates from the detected overlap within allowable limits for boundaries of the overlap regions. Each compatibility check of Q_1 and Q_2 also involves the origin sequence *S*. In case of incompatibility, if the fitness scores of Q_1 and Q_2 are both greater than that of *S*, then both Q_1 and Q_2 are removed from the cluster; otherwise the sequence with the smaller fitness score is removed. In this manner, all pairs of query sequences in the cluster are compared, and incompatible sequences removed. When a sequence is removed, the fitness scores for the remaining sequences are updated to account for this removal.

After the SWC step has removed incompatible sequences for all clusters in the super cluster, the cleaner overlaps are output from the corresponding hash table for the remaining sequences in each cluster. When this is completed for all super clusters, one iteration of SWC is complete. The next iteration of SWC begins by switching the roles of query and subject sequences, so that query sequences from the previous iteration are taken as subject, and the previous subjects become queries. The clustering, hashing and compatibility steps are again performed, and so on. The iteration proceeds until no more (false) overlaps can be removed (Figure 2), and we then describe SWC as "convergent." The iterations do not require the Megablast comparisons to be repeated, and so the entire procedure can be performed efficiently.

2.5 Screening false overlaps between clusters (SBC)

The SWC procedure ensures that BAC fragments within clusters are compatible with each other. However, some of the BAC fragments, in particular sequences at the ends of cluster layouts, or regions with low sequence coverage, may have spurious compatibility with other sequences in the cluster. Screening between clusters (SBC) provides an additional opportunity to identify false overlaps involving such sequences, and merges compatible clusters to form longer BAC consensus sequences. Like SWC, SBC is an iterative procedure (Figure 2). It begins by using the convergent SWC output, and the later iterations include the clustering and hashing steps as described for SWC. The compatibility checks, however, are performed *between* clusters.

For each super cluster, two clusters, denoted C_1 and C_2, are compared at a time to see if they have any directly shared sequences. If there are no shared sequences, then SBC continues with another pair of clusters. If there is more than one sequence shared, compatibility of the shared sequences is checked. This requires that the coordinates of the shared sequences in C_1 be converted to those in C_2 based on one of the shared sequences, called the *anchor* (denoted *A*). Let (P_A^1, O_A^1) be the coordinate/orientation of *A* in C_1, and (P_A^2, O_A^2) the corresponding coordinate in C_2. Then coordinate (P_i^1, O_i^1) for any other sequence S_i in C_1 can be converted to (P_i^2, O_i^2) for C_2 by:

$$P_i^2 = (P_A^2 - P_A^1) + P_i^1, \quad O_i^2 = O_i^1 \text{ , if } O_A^1 = O_A^2,$$

or

$$P_i^2 = (P_A^2 + P_A^1 + L_A) - P_i^1 - L_i, \quad O_i^2 = -O_i^1, \quad \text{if } O_A^1 \neq O_A^2.$$

If the converted coordinate of a shared sequence deviates from its original C_2 coordinate by more than a specified value, a penalty score 1 is recorded for each of the shared sequence and the anchor. Otherwise a penalty score of 0 is recorded.

Each of the shared sequences is taken as anchor sequence in turn, and a total penalty score is calculated for each shared sequence by summing all its penalty scores recorded under all anchor sequences (including itself). The shared sequence with largest penalty is removed, and the total penalty scores for sequences incompatible with the removed sequence (either anchor or non-anchor) are reduced by 1. The process continues until all remaining shared sequences have a zero total penalty score. If only one sequence is shared by C_1 and C_2, there is no need for a compatibility check. The shared sequence may be validated in later comparisons, or (as a conservative measure) dropped from the cluster in which it has the smaller fitness score.

After the compatibility check for shared sequences, coordinates for all unique sequences in C_1 are converted to those of C_2 with any compatible shared sequence as anchor. The conversion merges the two clusters and adopts C_2 sequence in overlapping regions. Comparisons of unique (non-shared) sequences from C_1 and C_2 are conducted in the same manner as for SWC.

The compatibility check iterates by swapping the roles of query sequences and subject sequences in the same manner as for SWC. At the end of the SBC iterative procedure, each super cluster becomes one or more disjoint clusters, which are then ready for assembly. For realistic tiling coverage in a draft assembly stage of a complex genome, the combination of quality filtering and the SWC and SBC iterative procedures reduces false overlaps to < 1% of detected overlaps, while retaining > 95% of true overlaps (see Results).

2.6 Assembly of BAC consensus sequences

With the cleaned overlaps, BAC consensus assembly now becomes straightforward: simply join overlapping BAC fragments, following the coordinate system within each cluster. Mega Weaver outputs the BAC fragments, cluster IDs, and position information. The final step is to perform multiple alignments of BAC fragments to form a BAC consensus. Because the remaining BAC fragments have been extensively cleaned, misassembly is very limited. In our simulations, the assembly results were nearly optimal in the sense of reconstructing true overlaps (see Results).

3. Results

Simulation studies were carried out to assess the performance of Mega Weaver in assembly of BAC consensus sequences. The complete sequence of human chromosome 22, after removal of sequencing gaps, was used as a 33.5 Mb hypothetical genome. The chromosome size and the amount of repetitive sequence (42% tandem and interspersed, Dunham et al. 1999) make it ideal for simulation studies (Batzoglou et al. 2002).

We considered several conditions that influence assembly quality, including *tiling coverage*, *average BAC fragment length*, and *sequence error rates*. Tiling coverage is the total length of all BAC fragments divided by the genome length, which should not be confused with the shotgun coverage used in creating BAC fragments. Each condition corresponds to one set of simulations. The first set of simulations used five degrees of tiling coverage (1X, 1.4X, 1.8X, 2.2X, and 2.6X) with a fixed average BAC fragment length of ~47kb and zero sequence error rate. For the second set of simulations, eight different average fragment lengths (10kb, 20kb, 30kb, 40kb, 60kb, 80kb, 100kb, and 200kb or complete BAC clone) were used with a fixed tiling coverage of ~1.5X and zero sequence error rate. The last set of simulations used five sequence error rates (0%, 0.01%, 0.05%, 0.5%, and 1%) under fixed tiling coverage of 1.5X and average fragment length of 10 kb. The conditions are not exhaustive, but reflect the wide range of coverage likely to be encountered as a genome moves through successive stages of draft assembly.

Each BAC clone was simulated by first choosing a random length in the range 100-300kb, and then choosing a starting bp position uniformly on the genome. Although one intent of the FPC map had been to aid in tiling path selection, we find that our simulations do not appear to be dramatically different from the observed human genome coverage. Each BAC was fragmented by inserting random gaps according to the specified average fragment length. Sequence errors were introduced at a specified rate per bp for each BAC fragment. For each condition, 10 replicate simulations were conducted.

Pairwise overlaps were identified as described in METHODS with Megablast options "-e 1e–10", "-D 3", and "-F F". Quality filtering and the SWC and SBC overlap procedures required a 200 bp minimum overlap length, 3 bp maximum hanging end, 3 bp maximum difference in overlap regions, and 95% minimum overlap identity. Megablast comparisons were performed using a small Linux cluster (8 nodes, 16 processors at 1 GHz), and the remaining steps performed on a single Linux workstation.

3.1 Performance in screening false overlaps

We evaluate the effectiveness of the screening procedures

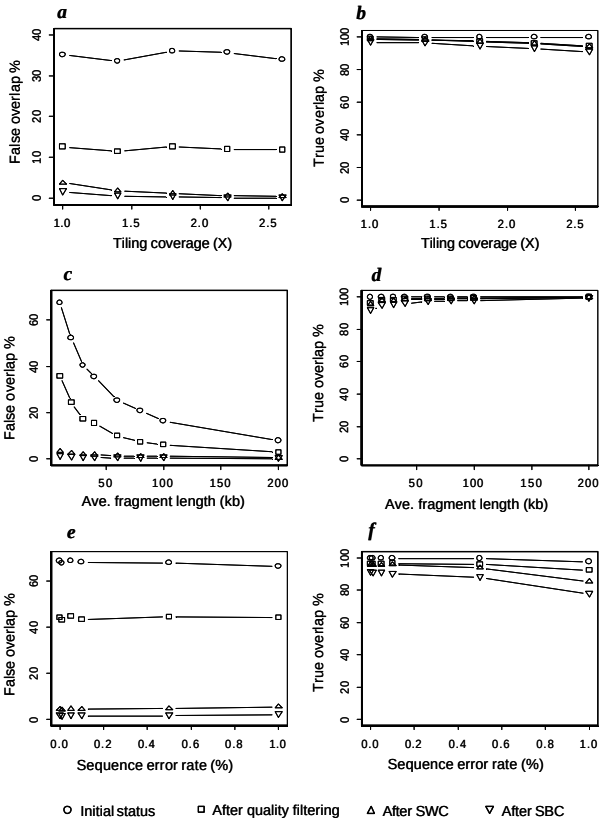

Figure 6. Performance of Mega Weaver in screening false overlaps. (*a, b*) effect of tiling coverage with average fragment length 47kb; (*c, d*) effect of fragment length with tiling coverage 1.5X; (*e, f*) effect of sequence error rate with fragment length 10kb and tiling coverage 1.5X. Results were averaged for 10 replicates under each condition.

by two measures: the proportion of false overlaps among detected overlaps (ideally 0%), and the proportion of true overlaps that were correctly detected (ideally 100%). Figure 6 shows the results for each step of the screening procedure (including initial Megablast output, quality filtering, SWC and SBC) for each of the three sets of simulations. It is apparent that nearly 100% of true overlaps were detected with Megablast under all of the conditions examined (Figure 6b, d, f). However, the initial overlaps contained up to ~68% false overlaps, and 8% false overlaps even when full-length BAC clones were used (Figure 6a, c, e). The proportion of false overlaps in initial Megablast output depended on average BAC fragment length more than tiling coverage and sequence bp error rate (Figure 6a, c, e).

After the entire screening procedure, false overlaps were reduced to 0.19%-1.9% for the entire range of conditions considered. This eventual reduction did not depend much on the conditions examined, an indicator of the robustness of our algorithm. At the same time, the losses in true overlaps were tolerable – an average loss of 6.7% over all simulation conditions. The greatest loss of true overlaps (~15%) occurred with the highest sequence error rates (1% and 0.5%), and short average fragment length (10kb). For full-length BAC clones (the other extreme), the loss was only 0.8%. Sequence error rates seemed to have the greatest influence on loss of true overlaps, assuming a relatively low average fragment length (10kb). Longer fragment lengths would correspond to greater depth of sequencing coverage, for which the remaining errors in BAC fragments would be at the extreme low end of our range.

The steps in the screening procedure did not contribute equally to the reduction of false overlaps and loss of true overlaps (Supplemental Figure S1 on our web site). The most effective steps were quality filtering and SWC, which together reduced false overlaps by 94.2%, but also accounted for 50.5% of the loss in true overlaps. By contrast, SBC succeeded in reducing false overlaps by only 3.8% of the total.

3.2 Performance in BAC consensus assembly

The overall quality of overlaps after the screening procedure largely determines the quality of the final BAC consensus assembly. Figure 7 compares the total number, average length, and coverage of BAC consensus sequences assembled with the cleaned overlaps, compared to the best possible assembly using the known genomic positions of all BAC fragments. It is evident from the figure that our assembly after the cleaning procedures was nearly optimal. The performance did not vary greatly by simulation condition, except in the worst case with sequence error rate 1% and average fragment length 10kb. The influence of the three sets of simulation conditions on the overall assembly quality were as expected.

We further evaluated the quality of each assembled consensus sequence by comparing it to the true consensus inferred from known genomic positions of BAC fragments (supplemental Figure S2 on our web site). Most consensus sequences (average 94%) were correctly assembled, i.e., the corresponding BAC fragments were positioned and oriented correctly (in proper order and positioned to within 3 bp). Moreover, the correct consensus sequences formed an average 95.5% of the total length of assembled consensus sequences. Using full-length BAC clones yielded ~99.8% correct consensus sequences, and 99.6% of the assembly length.

3.3 Performance in larger genome assembly

To assess the performance of our algorithm for BAC consensus assembly of a larger hypothetical genome, we performed additional simulations by joining end-to-end all human chromosome 1 BAC fragments (NCBI Build 30). This "genome" length is 254 Mb, including over 52% repetitive sequence. Simulations were carried out under fixed tiling coverage 1.5 X and sequence bp error rate 0.1%, with two different average fragment lengths, 10 kb and 200 kb (complete BAC sequences). Five sequence data sets were simulated for each of the two conditions. Pairwise overlaps were detected and iteratively screened using the same criteria as above. After quality filtering and iterative screening of false overlaps, false overlaps were reduced to ~6% on average for the shorter fragments (10kb), but only ~0.2% for the longer fragments (200 kb). Loss of true overlaps was ~7% with 10kb fragments, and ~5% for the 200 kb fragments. These results indicate that the algorithm can be scaled up for larger genome assembly without substantial reduction in performance.

4. Discussion

Sequence errors and repetitive sequences are two major obstacles to the assembly of complex genomes. Even using full-length BAC sequences, false overlaps (after quality filtering) caused by repetitive sequence were as great as 3% (Figure 6c). This percentage increases considerably when considering fragmented BAC sequences with sequence errors that remain in the BAC fragments. Some of the issues in BAC consensus assembly are similar to that encountered in whole-genome shotgun assembly (Batzoglou et al. 2002), and the detection of false overlaps is central to genome assembly. However, compared to individual shotgun reads, the greater length of BAC fragments allows the assembly procedure to focus immediately on layout compatibility of fragments rather than more elementary issues of sequence quality. The efficiency of the overlap comparisons detailed here enables the use of iterative procedures not considered in GigAssembler (Kent and Haussler 2001). Most greedy or dynamic algorithms (Staden 1982; Dean and Staden 1991; Kent and Haussler 2001) process one sequence (raw input sequence or existing contig) at a time, either creating a new contig or extending an existing contig. False overlaps are identified and removed as the consensus is extended.

Our simulation study has demonstrated that our iterative approach can remove the majority of false overlaps involving repetitive sequences reliably and effectively, to the point of nearly achieving optimal assembly and

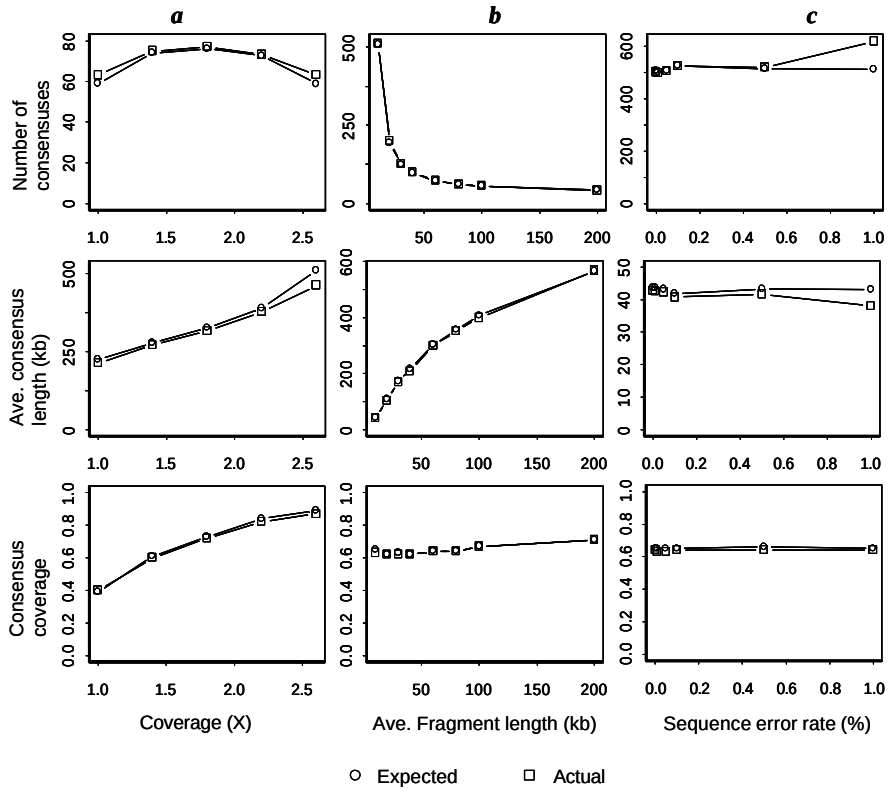

Figure 7. Performance of Mega Weaver in BAC consensuses assembly. Columns *a, b,* and *c* show the effects of coverage, average fragment lengths, and sequence bp error rate, respectively, on the quality of assembly.

including false overlaps to <1% of included sequence (which is further improved in later finishing steps). We found that average fragment length, and (to a lesser extent) tiling coverage and sequence error rate were important determinants of the quality of consensus assembly. Sequence error rate did show some impact when using relatively short fragment sequences (e.g. 10kb).

The assembly appears fairly robust to variations in tiling coverage, average sequence length, and reasonable sequence error rates. Thus, if BAC fragments have been assembled largely correctly, this algorithm can provide a useful intermediate step to full genome assembly, without requiring prior position information. The fact that BAC fragments are sufficiently unique to perform this assembly on the scale of human chromosome 1 (and likely larger) suggests that alternate hierarchical strategies involving random BAC selection may be feasible. For such a strategy, the potential extra costs in sequencing (via redundant sequenced BACs) must be weighed against the costs and accuracy of creating the FPC-based tiling path.

Mega Weaver code is freely available at http://www.bios.unc.edu/~dwang/ .

5. Acknowledgments

Thanks to Dr. William Lemon and Hao Sun for helpful discussion. This research was supported in part by GM 58934 to F.A.W.

6. References

Batzoglou, S., Jaffe, D.B., Stanley, K., Butler, J., Gnerre, S., Mauceli, E., Berger, B., Mesirov, J.P. and Lander, E.S. (2002): ARACHNE: A whole-genome shotgun assembler. *Genome Res.* **12**(1): 177-189.

Dunham, I., Shimizu, N., Roe, B.A., Chissoe, S., Dunham, I., Hunt, A.R., Collins, J.E., Bruskiewich, R., Beare, D.M., Clamp, M., et al. (1999) The DNA sequence of human chromosome 22. *Nature* **402** (6761): 489-495.

Ewing B. and Green P. (1998): Base-calling of automated sequencer traces using Phred. II. Error probabilities. *Genome Res.* **8**(3): 186-194.

Gillett, W., Hanks, L., Wong, G.K., Yu, J., Lim, R. and Olson, M.V. (1996): Assembly of high-resolution restriction maps based on multiple complete digests of a redundant set of overlapping clones. *Genomics* **33** (3): 389-408.

Green, P. (1997): Against a whole-genome shotgun. *Genome Res.* **7**(5): 410-417.

Green, P. (2002): Whole-genome disassembly. *Proc. Natl. Acad. Sci.* **99**(7):4143-4144.

Huang, X. (1992): A contig assembly program based on sensitive detection of fragment overlaps. *Genomics* **14**(1): 18-25.

Huang, X. and Madan, A. (1999): CAP3: A DNA sequence assembly program. *Genome Res.* **9**(9): 868-877.

International Human Genome Sequencing Consortium. 2001. Initial sequencing and analysis of the human genome. *Nature* **409**(6822): 860-921.

Kent, W.J. and Haussler, D. (2001): Assembly of the working draft of the human genome with GigAssembler. *Genome Res.* **11**(9): 1541-1548.

Kim, S. and Segre, A.M. (1999): AMASS: A structured pattern matching approach to shotgun sequence assembly. *J. Comp. Biol.* **6**(2): 163-186.

Myers, E.W., Sutton, G.G., Delcher, A.L., Dew, I.M., Fasulo, D.P., Flanigan, M.J., Kravitz, S.A., Mobarry, C.M., Reinert, K.H.J., Remington, K.A., et al. (2000): A whole-genome assembly of Drosophila. *Science* **287** (5461): 2196-2204.

Peltola, H., Sunderland, H. and Ukkonen, E. 1984. SEQAID: A DNA sequence assembling program based on a mathematical model. *Nucleic Acids Res.* **12**(1 pt 1): 307-321.

Pevzner, P.A., Tang, H.X. and Waterman, M.S. (2001): An Eulerian path approach to DNA fragment assembly. *Proc. Natl. Acad. Sci.* **98**(17): 9748-9753.

Soderlund, C., Longden, I. and Mott, R. (1997): FPC: a system for building contigs from restriction fingerprinted clones. *Comput. Appl. Biosci*. **13**(5): 523-535.

Sutton, G., White, O., Adams, M. and Kerlavage, A. (1995): TIGR assembler: A new tool for assembling large shotgun sequencing projects. *Genome Sci. Technol.* **1**(1): 9-19.

Venter, J.C., Adams, M.D., Myers, E.W., Li, P.W., Mural, R.J., Sutton, G.G., Smith, H.O., Yandell, M., Evans, C.A., Holt, R.A., et al. (2001): The sequencing of the human genome. *Science* **291**(5507): 1304-1351.

Waterston, R.H., Lander, E.S. and Sulston, J.E. 2002. On the sequencing of the human genome. *Proc. Natl. Acad. Sci*. **99**(6): 3712-3716.

Weber, J.L. and Myers, E.W. (1997): Human whole-genome shotgun sequencing. *Genome Res.* **7**(5): 401-409.

Zhang, Z., Schwartz, S., Wagner, L, and Miller, W. (2000): A Greedy Algorithm for Aligning DNA Sequences. *J. Comp. Biol.* **7**(1-2):: 203-214.

Zhuo, D., Zhao, W.D., Wright, F.A., Yang, H.Y., Wang, J.P., Sears, R., Baer, T, Kwon, D.H., Gordon, D., Gibbs, S., et al. (2001): Assembly, annotation, and integration of UNIGENE clusters into the human genome draft. *Genome Res.* **11**(5): 904-918.