GENOME SCIENCE & TECHNOLOGY
Volume 1, Number 1, 1995
Mary Ann Liebert, Inc.

TIGR Assembler: A New Tool for Assembling Large Shotgun Sequencing Projects

GRANGER G. SUTTON, OWEN WHITE, MARK D. ADAMS, and ANTHONY R. KERLAVAGE

ABSTRACT

A new approach to assembling large, random shotgun sequencing projects has been developed. The TIGR Assembler overcomes several major obstacles to assembling such projects: the large number of pairwise comparisons required, the presence of repeat regions, chimeras introduced in the cloning process, and sequencing errors. A fast initial comparison of fragments based on oligonucleotide content is used to eliminate the need for a more sensitive comparison between most fragment pairs, thus greatly reducing computer search time. Potential repeat regions are recognized by determining which fragments have more potential overlaps than expected given a random distribution of fragments. Repeat regions are dealt with by increasing the match criteria stringency and by assembling these regions last so that maximum information from nonrepeat regions can be used. The algorithm also incorporates a number of constraints, such as clone length and the placement of sequences from the opposite ends of a clone. TIGR Assembler has been used to assemble the complete 1.8 Mbp *Haemophilus influenzae* (Fleischmann et al., 1995) and 0.58 Mbp *Mycoplasma genitalium* (Fraser et al., 1995) genomes.

INTRODUCTION

The advent of automated DNA sequencing (Smith, L.M., et al., 1986) and recent advances in scaling up this technology (Adams et al., 1994) have led to new approaches in determining the genome sequence of humans and other organisms. Classic approaches have included shotgun sequencing of lambda inserts (~10–20 kbp) or cosmids (~40 kbp), and this approach has been successfully applied to many large-scale projects, such as the genomes of *Escherichia coli* (Sofia et al., 1994), *Saccharomyces cerevisiae* (Levy, 1994), *Caenorhabditis elegans* (Sulston et al., 1992), and humans (Martin-Gallardo et al., 1992; McCombie et al., 1992). The strategy is based on sequencing many small (on the order of 1–2 kbp), randomly chosen subclones of the lambda or cosmid clones. Typically, 300–500 bp of sequence is determined from one or both ends of the subclones. Enough sequence fragments must be obtained and their positions evenly enough distributed so that all or most of the target DNA will be covered by at least one fragment. Ideally, the coverage should be by as many fragments as are required to resolve ambiguities and provide confidence in the accuracy of the final consensus sequence.

The Institute for Genomic Research, Rockville, Maryland.

Many sequence assembly algorithms have been developed to reconstruct the original sequence from these fragments. The reconstruction process involves finding pairs of fragments that overlap, merging as many fragments together as possible, and creating a consensus sequence from the merged fragments. Sequence fragment assembly is a straightforward programming problem in the absence of complicating factors. There are many assembly algorithms available, including XBAP (Gleeson and Staden, 1991), AutoAssembler (Applied Biosystems, Foster City, CA), GCG (Dolz, 1994), and several others (reviewed in Miller and Powell, 1994). Each of these algorithms has its advantages and disadvantages and works reasonably well on lambda- or cosmid-sized assemblies. However, they typically do not perform well on reconstructions of an order of magnitude larger scale. A more recent assembly program, FALCON (Gryan, 1994), uses a fast pairwise comparison method similar to that of TIGR Assembler and can handle very large assembly projects. Most of the other assembly programs could also be scaled up with some redesign effort. However, a feature that most of these programs lack is a strategy to detect and handle repeat regions.

The lack of an adequate assembly engine has been one of the driving forces to maintain a cosmid-based genome sequencing strategy, even for megabase-scale projects. The major problems to be solved in reconstructing much larger assemblies are (1) a dramatic increase in the number of pairwise comparisons required, (2) the increased likelihood of encountering repetitive DNA that confounds true fragment overlaps with false overlaps, and (3) the increased probability of encountering chimeric clones that can cause false overlaps or mask true overlaps. Herein, we explain how TIGR Assembler handles these complicating factors. In addition, we discuss future enhancements to the algorithm to further improve the handling of these factors.

MATERIALS AND METHODS

The basic steps in the TIGR Assembler algorithm are outlined here.

1. Perform pairwise fragment comparisons for the entire dataset to generate a list of potential fragment overlaps.
2. Use the distribution of the number of potential overlaps for each fragment to label fragments as repeat or nonrepeat.
3. Start with a nonrepeat fragment as the initial assembly seed or a repeat fragment if no nonrepeat fragment is left; quit if no fragments remain.
4. Use a potential overlap list to attempt merges between the current assembly and nonrepeat fragments.
5. When no potential overlaps with nonrepeat fragments remain for the current assembly, increase the stringency of the match criteria and enforce clone length constraints when attempting to merge with repeat fragments.
6. If due to a merge with a repeat fragment, a nonrepeat fragment is added to the potential overlap list; go to step 4.
7. When there are no fragments left on the current potential overlap list, output information about the current assembly and go to step 3.

Pairwise comparison

TIGR Assembler compares every fragment to every other fragment to find potential pairwise overlaps. The pairwise comparison of all n fragments involves approximately n^2 comparisons. For the *Haemophilus influenzae* genome project, there were approximately 25,000 fragments requiring 625,000,000 pairwise comparisons. The widely accepted sequence comparison method of choice is the Smith-Waterman algorithm (Smith, T.F., and Waterman, 1981), which finds the optimal alignment of two fragments by maximizing the score determined by a mathematical model of sequence match criteria. Unfortunately, the Smith-Waterman algorithm is relatively slow. To speed up the pairwise comparisons, TIGR Assembler locates all n-mer oligonucleotides shared between fragment pairs (where n is typically 10–12), in a process similar to the initial processing of BLAST (Altschul et al., 1990). TIGR Assembler views fragment pairs with a high degree of n-mer similarity as potential fragment overlaps. This process is much faster but less sensitive than

using the Smith-Waterman algorithm to determine potential fragment overlaps. In the future, TIGR Assembler will use an additional step similar to the initial processing in FASTA (Pearson and Lipman, 1988) to determine if the potential fragment overlaps with n-mer similarity have the n-mers in the same order. This additional step will eliminate some of the false potential overlaps with a minimal speed penalty. If the time penalty is not too steep or an implementation on a parallel processor is undertaken, an additional step of evaluating the much reduced set of potential fragment overlaps using Smith-Waterman will be undertaken. By using a fast comparison method to eliminate fragment pairs that clearly do not overlap, followed by a slower, more accurate comparison method on the much smaller remaining set of fragment pairs for the final determination of pairwise fragment overlap, both reasonable speed and accuracy should be achieved. We have applied this technique successfully to the identification of expressed sequence tags (ESTs) by similarity searching (Adams *et al.*, 1995).

Merging fragments with assemblies

TIGR Assembler uses a modified Smith-Waterman algorithm to evaluate a potential overlap between the current assembly and a fragment. The fragment is merged with the current assembly if the match criteria are satisfied, resulting in a new consensus profile sequence for the current assembly. A consensus profile sequence is the count of the bases and gaps that have been aligned at each position of the assembly. There are four elements of the match criteria: minimum length of overlap, minimum similarity in the overlap region as a percentage of the best possible score, maximum length of overhang, and maximum number of local errors. The Smith-Waterman algorithm produces a local alignment such that bases a through b of a fragment and bases c through d of the current assembly are optimally aligned (Fig. 1a). The length of overlap is (b − a) + 1. The length of overhang is the maximum of the overhang at both ends of the alignment. In Figure 1a, the overhang lengths (diagonal lengths) are a–1 and L–d, where L is the length of the current assembly. The Smith-Waterman algorithm generates a score for the overlap region based on positive values for matching base pairs and negative values for mismatching base pairs. The similarity score is the over-

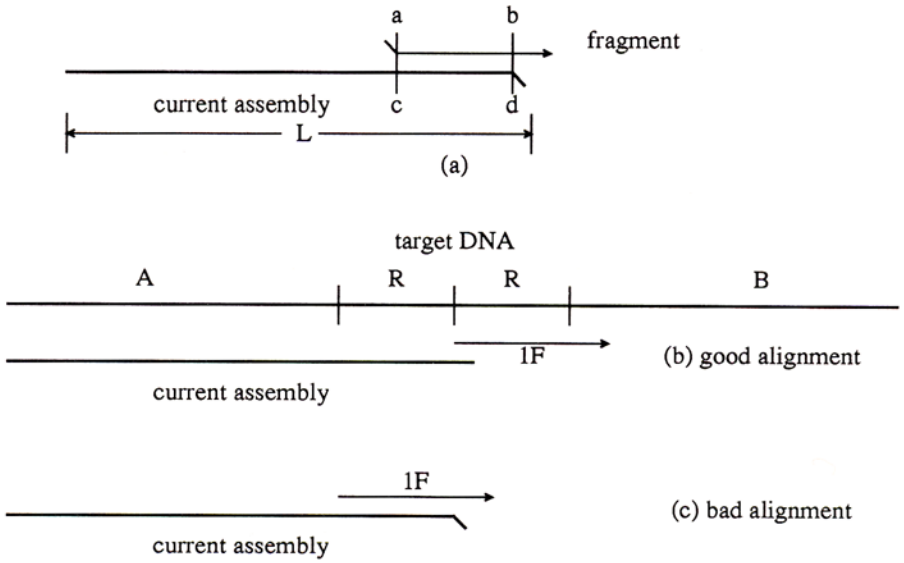

FIG. 1. **a.** A graphic representation of a Smith-Waterman alignment is shown. The overlap is shown between vertical bars, and the diagonal lines represent overhangs where the sequences do not match. The current assembly has a length of L. **b.** A tandem repeat of two copies of repeat region R is shown along with the proper alignment of fragment 1F with the current assembly. **c.** A bad alignment of the current assembly and fragment 1F is produced if the overlap is maximized without regard to the length of the overhang. The bad alignment can result in two outcomes. If the overhang is short, it will be ignored, and the two repeat regions will be compressed into a single region. If the overhang is long, the merge will not be allowed, and the current assembly will not be extended.

lap score divided by the best possible score for any fragment vs the overlap region of the assembly. Local errors are the number of base pair mismatches between the fragment and the assembly in any given contiguous 32 bp in the overlap region. The local errors may seem to be redundant with the similarity percentage, but we have found that, for example, 16 errors spread out evenly over a 400 bp overlap (96% similarity) is more likely to be a real overlap than 8 errors concentrated within a 32 bp window of a 400 bp overlap with no other errors (98% similarity). The stringency of the match criteria is increased when the potential overlap fragment is either a repeat or does not meet clone length constraints. TIGR Assembler immediately rejects any fragment that is both a repeat and does not meet clone length constraints. The stringency of the match criteria is decreased when the fragment is not a repeat and meets clone length constraints.

Experience with failures in properly merging fragments from medium length (same order of magnitude as the fragment length) tandem repeats led to a modification of the alignment algorithm. Tandem repeats are repeat regions that are nearly identical and contiguous where there can be two or more copies (Fig. 1b). The local nature of the Smith-Waterman algorithm produces an alignment with the largest possible overlap score regardless of the length of the resulting overhangs. A simple constraint was added to allow only those solutions with an acceptable length of overhang. This prevents the merge from failing because of incorrect compression of a tandem repeat (Fig. 1c). Unfortunately, this is not a solution for preventing multiple copies of a tandem repeat from being compressed. Fragments that are completely contained within a tandem repeat match each other in many equally likely ways and can be assembled to produce many different numbers of repeat units. If the variance of the clone length is significantly smaller than the length of a single repeat unit, the tandem repeat can be walked up to the clone length (described in *Repeat regions*). Longer tandem repeats and tandem repeats with smaller repeat units must have the number of repeat units determined in a different fashion.

Building a consensus sequence

The consensus sequence for an assembly is built by merging a single fragment at a time to the existing consensus sequence. Each position of the assembly is represented as a history of what bases have been aligned to that position in the past. This results in a column or profile (Gribskov et al., 1987) where the count of bases plus gaps is recorded. For example, position 10 might have 3 As, no Cs, Gs, or Ts, and 1 gap based on the iterative alignment of four fragments across that position. This profile is used for two purposes: to align a fragment with the assembly and to generate a final consensus sequence for the assembly. The profile and not a consensus sequence is used for alignment, primarily as an effort to gather gaps. Suppose, if in addition to position 10 described above, position 11 is composed of 4 As and nothing else. If a new fragment has the choice of having a single A aligned at position 10 or 11 and a gap at the other, the A will be aligned at position 11 and the gap at position 10 because this maximizes the alignment score. The alignment score for a single position of the profile is calculated as the weighted average of the fragment base scored against each of the profile components. The score of base A vs position 11 would be 4 matches and 0 mismatches divided by 4, whereas the score of base A vs position 10 would be 3 matches and 1 mismatch divided by 4. A match might be given a score of 1 and a mismatch a score of -3, resulting in scores for positions 11 and 10 of 1 and 0, respectively. This results in gaps tending to gather in the same position of the assembly profile. Even so, the gap gathering can be nonoptimal in some cases.

TIGR assembler will be enhanced to include a local optimization of gap position before a final consensus sequence is generated. The final consensus sequence is generated for each profile position using a small set of rules that can be summarized as follows: (1) If the largest profile component is greater than two thirds of the total of the components, output an uppercase A, C, G, or T, or nothing in the case of a gap. (2) If there are two nongap components that are prevalent, output a lowercase two-base ambiguity code (m, r, w, s, y, or k). (3) If the prevalent component is greater than half of the total, output a lower case a, c, g, or t, or nothing in the case of a gap. (4) Otherwise, output a lower case n. In addition, if a gap is the prevalent component but less than two thirds of the total, make the next symbol output be in lower case. These rules highlight any ambiguities by outputting lower case. The ambiguities can then be checked manually by looking at the assembly alignments generated by TIGR Assembler and referring back to the original electropherogram data from which the base calls were automatically determined.

Repeat regions

A very important task for TIGR Assembler is identifying repeat regions in the target DNA because these regions have the most potential to cause misassembly of the target. A repeat region is a contiguous piece of DNA that has a very high similarity to another piece of DNA in the target (Fig. 2). They can be exact (or nearly exact) copies of a segment of the target, which are repeated some number of times in a genome, such as the six ribosomal RNA operons in *H. influenzae* (Fleischmann *et al.*, 1995). Alternatively, they may be more highly variable segments that are repeated at high frequency, such as the human Alu repeats, which may be found at a frequency of one per kbp (McCombie et al., 1992; Martin-Gallardo et al., 1992).

Repeat regions are a problem because a fragment from a repeat region can have false overlaps with fragments from other repeat regions, resulting in bad merges and incorrect final assemblies. A true overlap is the region shared between two fragments that span a contiguous piece of the target DNA. A false overlap occurs when two noncontiguous fragments contain very similar regions because, for example, the two fragments are from different copies of a repeat region. A bad merge occurs when a false overlap is incorporated into the current assembly. The fragments from repeat regions are identified by the number of potential overlaps each fragment has based on pairwise comparisons. TIGR Assembler determines the median number of potential overlaps, and any fragment with more than $\rho \times$ median potential overlaps (where ρ is typically set at 1.4 to 1.6) is labeled as being from a repeat region. This arbitrary cutoff does not completely separate repeat from nonrepeat fragments because of the randomness of the fragment coverage of the target DNA and the approximate nature of the n-mer oligonucleotide pairwise comparison. TIGR Assembler tries to err on the side of overlabeling fragments as repeats in order to minimize the chance of treating a repeat fragment as a nonrepeat fragment. Repeat detection is one of the primary reasons to use a more sensitive pairwise comparison method, as we proposed previously.

Once fragments have been labeled repeat or nonrepeat, TIGR Assembler uses two main strategies to minimize bad merges due to false overlaps between repeat regions: increasing the stringency of the match criteria for repeat fragments and strictly enforcing clone length constraints for repeat fragments. Increasing the stringency of the match criteria prevents bad merges when the average difference between two repeat regions is measurably greater than the average number of errors introduced into the sequence of the fragments by the sequencing methodology. For example, Alu repeats, which are on the order of 80% identical, are easily discriminated when sequencing accuracy is on the order of 98%, which is typical in our laboratory (Adams et al., 1995; Fleischmann et al., 1995). This emphasizes the need for the most accurate sequencing possible. Unfortunately, some repeat regions are identical or nearly identical, making bad merges impossible to detect on the basis of sequence mismatches. For short identical repeat regions, a single frag-

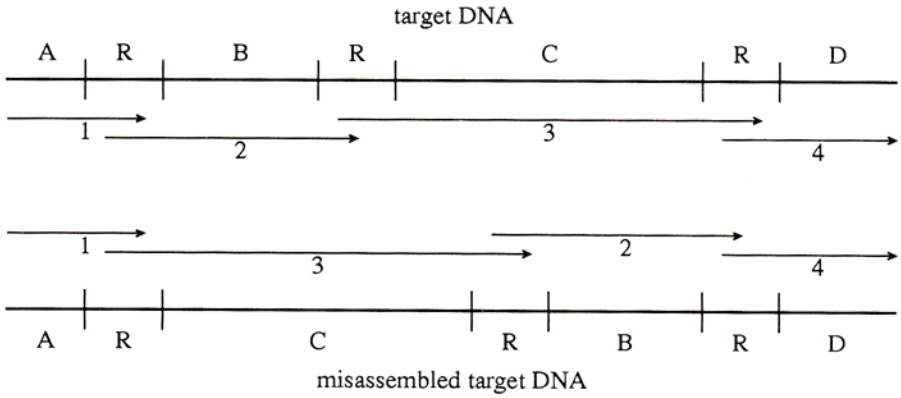

FIG. 2. The target DNA contains three exact copies of a repeat region R. Fragment 1 has a true overlap with fragment 2 and false overlaps with fragments 3 and 4 as a result of these fragments terminating in R repeat regions. This can lead to any of several misassemblies, one of which is shown where regions B and C are transposed due to several bad merges, including merging fragments 1 and 3 and fragments 2 and 4.

ment will often span the repeat region, allowing bad merges to be detected based on the differing sequences surrounding the repeat region. The importance of longer sequence reads to span short repeats should not be overlooked. For longer identical repeat regions, TIGR Assembler uses clone length constraints.

Up to this point, fragments have been treated as independent entities, each a random piece of the target DNA. A fragment is actually the sequence from one end of a clone where the clone is usually longer than the fragment. In order to use clone length constraints, a clone must be sequenced from both ends, and the approximate length of the clone must be known. The use of clone length constraints is illustrated in Figure 3a, where 2F is the "forward" sequence from one end of clone 2 and 2R is the "reverse" sequence from the other end of clone 2, and the length of clone 2 is L. Fragments 2F and 2R are called clone mates. TIGR Assembler knows that fragment 2F should be on the opposite strand from fragment 2R and the first base of 2F should be approximately L bases away from the first base of 2R. Assume that fragments 2F and 8R

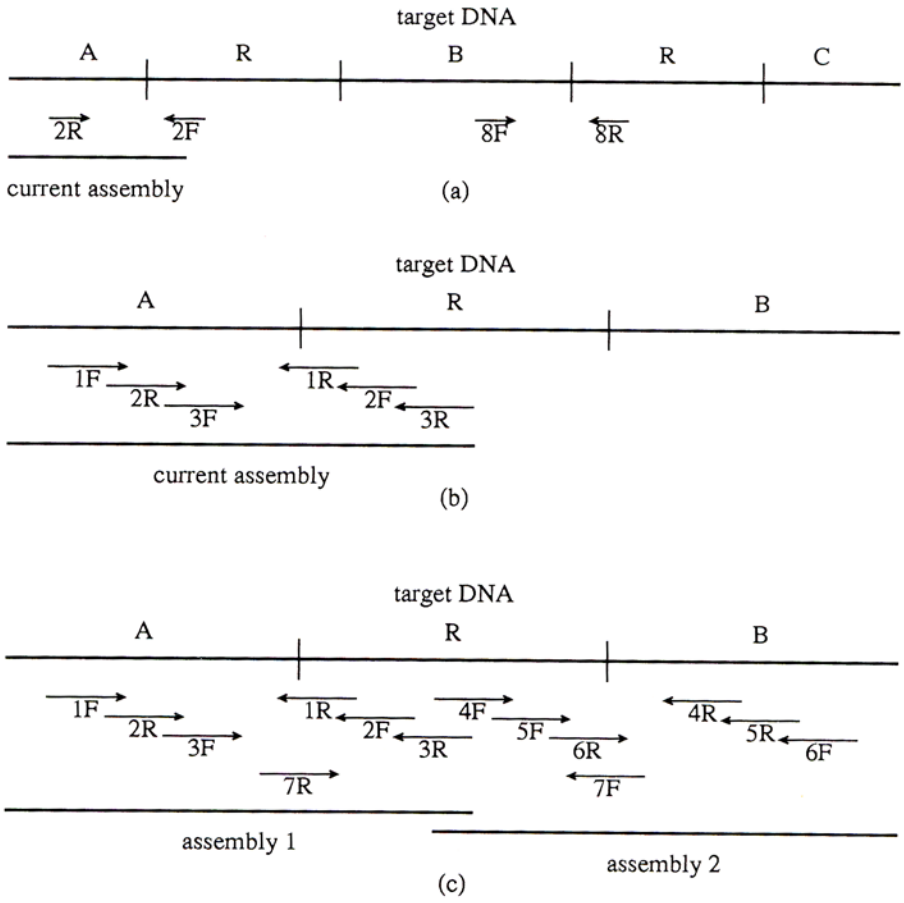

FIG. 3. **a.** TIGR Assembler must choose between adding fragment 2F or 8R to the current assembly. The matches between 2F or 8R and the current assembly are equally good because 2F and 8R lie in exact repeat regions R. TIGR Assembler chooses 2F because 2F's clone mate, 2R, is already in the current assembly at the appropriate distance based on the length of clone 2 and correct direction based on the direction of 2F, whereas, 8R's clone mate 8F is not in the current assembly and should be based on the length of clone 8 and direction of 8R. **b.** TIGR Assembler adds the clone mates of fragments 1F, 2R, and 3F to extend the current assembly into the repeat region R. This extension has a maximum length equal to the maximum length of a clone. **c.** Because assembly 1 is in a repeat region, TIGR Assembler cannot extend it by adding fragment 4F because 4F is a repeat fragment and can only be added if 4F's clone mate is already in assembly 1. TIGR Assembler will be improved to concurrently build assemblies 1 and 2 so that the existence of fragment 7R's clone mate, fragment 7F, in assembly 2 can be used to merge assembly 2 with assembly 1. In addition, if clone 7 is one of a small set of much longer clones, the maximum length of repeat R that can be spanned is twice the maximum clone length of the abundant shorter clones because the repeat region can be walked into from both sides.

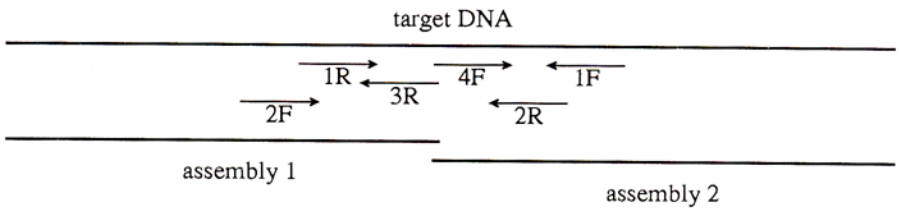

FIG. 4. The match between fragments 3R and 4F does not meet the match criteria because the overlap is too short. Assemblies 1 and 2 can still be joined because the match criteria can be lowered based on the clone mate information from clones 1 and 2.

have identical sequences because they are from two different identical repeat regions and, in addition, that fragments 2R and 8F have different sequences because these ends of clones 2 and 8 are not in the repeat regions. TIGR Assembler makes use of this clone mate information to properly place fragments 2F and 8R in different assemblies even though they have identical sequences. TIGR Assembler accomplishes this by starting with nonrepeat fragments to build up an assembly until one end of the assembly only has potential overlaps with repeat fragments. Repeat fragments with clone mates already in the assembly and at the appropriate distance based on approximate clone length and in the correct direction with respect to their clone mates are given preference for being merged. Repeat fragments with no clone mate in the assembly but that should already have a clone mate in the assembly based on approximate clone length and clone mate direction are excluded as merge candidates. In this way, TIGR Assembler walks into a repeat region by adding the clone mates of fragments already in the current assembly, thus extending the current assembly into the repeat region (Fig. 3b). The maximum length repeat that can be walked across in this manner is equal to the maximum clone length. For very long, nearly identical repeat regions, a relatively small set of much longer clones sequenced from both ends is necessary to determine which flanking regions should be joined. TIGR Assembler can fill the very long repeat regions with a consensus sequence, or the exact sequence can be determined by designing primers to walk the repeat containing clone.

In contrast to repeat regions, fragments with less than the median number of potential overlaps are labeled as low coverage regions. TIGR Assembler reduces the stringency of the match criteria in these regions to allow potentially real merges to be made in the absence of strong fragment overlap data due to the low coverage. The possibility of false overlaps and resulting bad merges is small in low coverage regions because they are unlikely to be repeat regions.

Concurrent assemblies

In addition to the improvements to TIGR Assembler's pairwise comparison method discussed, there is a planned enhancement for building assemblies. TIGR Assembler currently makes use of clone length constraints for the purpose of separating repeat regions as described, but clone mate information can be used more efficiently for this purpose, as well as for other purposes, if multiple assemblies are constructed simultaneously instead of constructing a single assembly at a time. Spanning repeat regions is more efficient with multiple assemblies because more clone mate information is available. This is because a clone may span a repeat and thus connect two assemblies, whereas there is no set of clones that can be used to walk across the repeat (Fig. 3c). This can be described as walking into the repeat region from both ends and relying on a repeat spanning clone to join the two assemblies. By using a relatively small number of longer clones to span long repeat regions, walking from both ends doubles the length of a repeat region that can be crossed to twice the maximum clone length of the abundant shorter length clones.

Another use of clone mate information in conjunction with concurrently built assemblies is to join two assemblies that would not normally be merged. For instance, in Figure 4, the presence of fragments 1R and 2F in assembly 1 and fragments 1F and 2R in assembly 2 at the appropriate distances and directions is strong evidence that assemblies 1 and 2 might be contiguous even though under normal match criteria, fragment 4F could not be added to assembly 1 because of a very short overlap or a poor quality match. TIGR assembler can use this information to reduce the match criteria and attempt to merge assemblies 1 and 2.

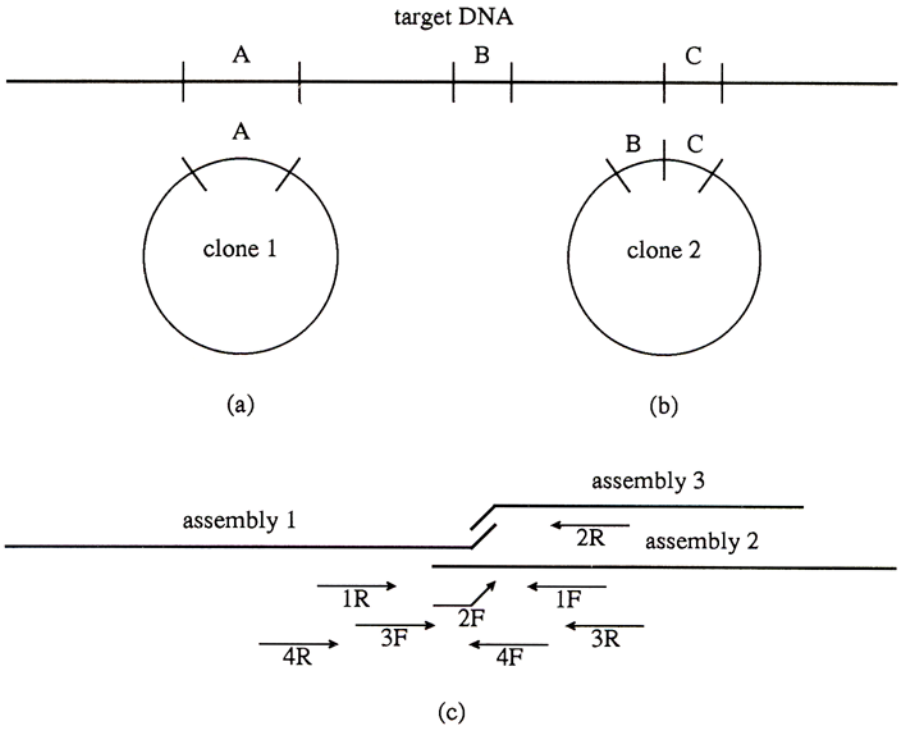

FIG. 5. **a.** A nonchimeric clone has an insert that is a single contiguous piece of target DNA. **b.** A chimeric clone has an insert composed of two or more noncontiguous pieces of target DNA that have become joined during the subcloning process. **c.** Clone 2 is chimeric, with the discontinuity occurring in the middle of fragment 2F. Fragment 2R indicates that assemblies 1 and 3 should be joined, but the overwhelming evidence of clone mates from clones 1, 3, and 4 argues that assemblies 1 and 2 should be joined and that fragment 2F should be discarded as chimeric.

Clone mate information can also be used to detect chimeric clones. A chimeric clone contains two pieces of noncontiguous DNA from the target DNA (Fig. 5). A chimeric clone will give erroneous assembly information, so the number of chimeric clones must be kept to a minimum such that the evidence from nonchimeric clones overwhelms the erroneous information. For example, in Figure 5c, assemblies 1 and 2 were not merged because a fragment from a chimeric clone is at the end of assembly 1. The preponderance of clone mate information often can be used to identify the chimeric clone and remove it from further consideration.

Currently, detection of weak joins and chimeras based on clone mate information is done as a post-processing step requiring user intervention. First, the alignment information generated from TIGR Assembler, including the position of every fragment in assemblies, is loaded into a relational database. Another program, ASM_ALIGN, generates an output file showing which pairs of assemblies are linked by clone mates. The user sees basically the same information as is graphically shown in Figures 3c, 4, and 5c. For weak joins, either more sequence fragments in the join region are obtained or the match criteria are manually lowered to achieve a merge. For chimeric fragments, the chimeric fragments are removed, and the assembly is redone. These manual analysis steps will be integrated into TIGR Assembler in a future version as described.

Optimal clone length

The importance of clone length constraints for assembling repeat regions raises the issue of optimal clone length. Longer clones potentially allow larger repeats to be walked across. The drawback to long clones is that in order for the clone length constraints to be used, the clone mate of a fragment in a repeat region

must be in an assembly contiguous to the repeat region. This means that there must not be an uncovered region, chimeric clone, or repeat region between the two clone mates that prevents assembly. The likelihood of one of these breaks occurring between clone mates increases with clone length and is also dependent on the density of repeats, the frequency of chimeric clones, and the depth of coverage for a specific project. Another constraint is the limited number of clone lengths that are biologically feasible. One approach that we have found valuable is to use a mixture of clone lengths: a large number of shorter clones to minimize breaks between clone mates and a much smaller number of longer clones to span large repeat regions. The choice of optimal clone size is dependent on the total size of the target DNA and the size of typical repeats. For the shorter clones, inserts of up to 5 kbp should work very well. In the *H. influenzae* genome project, we found that lambda clones were ideal for sorting out the six virtually identical 5 kbp ribosomal operons.

Implementation details

TIGR Assembler is written in C for Unix systems and has been compiled under Solaris 4.2 on Sun workstations. The program runs from the command line taking a single input file and optional command line parameters and generates several output files. The input file contains the fragment sequences in FASTA format. In order to use clone length constraints, the FASTA description lines must be in the format: clone_nameF_or-R min_clone_length max_clone_length (e.g., GHIAA01F 1500 2500). The command line parameters can be used to override the defaults for the match criteria and n-mer length. The output files include a GDE (Smith, S.W., et al., 1994) formatted file for the alignment of each assembly, a FASTA file of the consensus sequences of all of the assemblies, and a text file containing the information to build Sybase records to represent the assembly alignments. The assembly alignment files can be viewed with GDE, and the FASTA file can be used for searching. The virtual memory and processing time requirements can be large for large projects. The oligonucleotide pairwise comparisons use 32×4^n bytes of memory, where n is the length of the oligonucleotides. This memory is then freed and reused for the consensus sequence profile, which requires 32 × length of the assembly. For the *H. influenzae* genome, approximately 25,000 fragments were assembled with n = 10 in 30 h on a Sun SPARCstation 5 with 40 Mbytes of RAM and 100 Mbytes of virtual memory. We have also assembled 186,000 ESTs into 20,000 assemblies (with 39,000 nonoverlapping ESTs remaining) on a single processor of a Sun SPARCCenter 2000 with 512 Mbytes of RAM in 7 days.

RESULTS AND DISCUSSION

The lessons learned from assembling the genomes of *H. influenzae* and *Mycoplasma genitalium* can best be summarized as follows. It is critical to identify repeat regions, clone length constraints are necessary to assemble long identical repeat regions, and the accuracy of the fragment sequence data is crucial for assembling target DNA with repeat regions having similarity near the same level as the fragment sequence accuracy. TIGR Assembler does a good job of identifying repeat regions and using clone length constraints to assemble long repeat regions. However, some nonrepeat fragments are labeled as repeat fragments due to the inexact nature of the pairwise comparison method. A more accurate second pairwise comparison step can be added to eliminate this problem. TIGR Assembler uses clone length constraints to walk into and possibly across long repeat regions. In the future, by constructing multiple concurrent assemblies, long repeat regions can be walked from both ends simultaneously, doubling the length of a repeat region that can be spanned. For *H. influenzae,* the most common reason that fragments having true overlaps with an assembly failed to merge with that assembly was poor quality sequence data at the end of either the fragment or the assembly. Trying to correct for this problem by lowering the stringency of the match criteria resulted in repeat regions being misaligned. The problem was ameliorated in two ways. First, TIGR Assembler set the match criteria lower in nonrepeat regions than in repeat regions and even lower in low coverage regions. In addition, the most frequent sequencing errors were discounted. TIGR Assembler corrects for some known biases in the accuracy of fragment sequence data produced by the ABI 373 and 377 sequencers to

effectively increase the accuracy of the data. This is done by discounting errors that are known to be frequently due to sequence read error, such as doubled base calls toward the end of a sequence read. A much better solution is for the base caller to give a measure of quality for each base call, so that errors that are probably due to sequence read error can be ignored. This would allow sequences to be trimmed back to good sequence data, which would alleviate the problem of poor quality data at the end of sequences breaking apart assemblies. A quality measure would also allow dubious base calls in the middle of a sequence to be discounted so that the ability to discriminate between repeat regions would be significantly increased.

The successful application of TIGR Assembler to the assembly of the 1.8 Mbp *H. influenzae* genome and to a very large number of EST sequences has demonstrated that a major hurdle to rapidly completing large genomes has been eliminated. This should allow new strategies, such as shotgun sequencing of bacterial artificial chromosomes (BACs), to be applied to the human genome. Such an approach has the potential to be much more efficient than current cosmid-based strategies.

REFERENCES

ADAMS, M.D., KERLAVAGE, A.R., FLEISCHMANN, R.D., FULDNER, R.A., BULT, C.J., LEE, N.H., et al. (1995). Initial assessment of human gene diversity and expression patterns based upon 83 million nucleotides of cDNA sequence. Nature **377,** in press.

ADAMS, M.D., KERLAVAGE, A.R., KELLEY, J.M., GOCAYNE, J.D., FIELDS, C., FRASER, C.M., and VENTER, J.C. (1994). A model for high-throughput automated DNA sequencing and analysis core facilities. Nature **368,** 474–475.

ALTSCHUL, S.F., GISH, W., MILLER, W., MYERS, E.W., and LIPMAN, D.J. (1990). Basic local alignment search tool. J Mol Biol **215,** 403–410.

DOLZ, R. (1994). GCG: Fragment assembly programs. Methods Mol Biol **24,** 9–23.

FLEISCHMANN, R.D., ADAMS, M.D., WHITE, O., CLAYTON, R.A., KIRKNESS, E.F., KERLAVAGE, A.R., et al. (1995). Whole-genome random sequencing and assembly of *Haemophilus influenzae* Rd. Science **269,** 496–512.

FRASER, C.M., GOCAYNE, J.D., WHITE, O., ADAMS, M.D., CLAYTON, R.A., FLEISCHMANN, R.D., et al. (1995). The *Mycoplasma genitalium* genome sequence revels a minimal gene complement. Submitted.

GLEESON, T.J., and STADEN, R. (1991). An X windows and UNIX implementation of our sequence analysis package. Comput Appl Biosci **7,** 398.

GRIBSKOV, M., McLACHLAN, A.D., and EISENBERG, D. (1987). Profile analysis: Detection of distantly related proteins. Proc Natl Acad Sci USA **84,** 4355–4358.

GRYAN, G. (1994). Faster sequence assembly software for megabase shotgun assemblies. *Genome Sequencing and Analysis Conference VI,* abstract E/B-3 (Mary Ann Liebert, Inc., Larchmont, NY).

LEVY, J. (1994). Sequencing the yeast genome: An international achievement. Yeast **10,** 1689–1706.

MARTIN-GALLARDO, A., McCOMBIE, W.R., GOCAYNE, J., FITZGERALD, M., WALLACE, S., LEE, B.M., et al. (1992). Automated DNA sequence analysis of 106 kilobases from human chromosome 19q13.3. Nature Genet **1,** 34–39.

McCOMBIE, W.R., MARTIN-GALLARDO, A., GOCAYNE, J.D., FITZGERALD, M., DUBNICK, M., KELLEY, J.M., et al. (1992). Expressed genes, *Alu* repeats and polymorphisms in cosmids sequenced form chromosome 4p16.3. Nature Genet **1,** 348–353.

MILLER, M.J., and POWELL, J.I. (1994). A quantitative comparison of DNA sequence assembly programs. J Comp Biol **1,** 257–269.

PEARSON, W.R., and LIPMAN, D.J. (1988). Improved tools for biological sequence comparison. Proc Natl Acad Sci USA **85,** 2444–2448.

SMITH, L.M., SANDERS, J.Z., KAISER, R.J., HUGHES, P., DODD, C., CONNELL, C.R., et al. (1986). Fluorescence detection in automated DNA sequence analysis. Nature **321,** 674–679.

SMITH, S.W., OVERBEEK, R., WOESE, C.R., GILBERT, W., and GILLEVET, P.M. (1994). The genetic data environment: An expandable GUI for multiple sequence analysis. GABIOS **10,** 671–675.

SMITH, T.F., and WATERMAN, M.S. (1981). Identification of common molecular subsequences. J Mol Biol **147,** 195–197.

SOFIA, H.J., BURLAND, V., DANIELS, D.L., PLUNKETT, G., III, and BLATTNER, F.R. (1994). Analysis of the *Escherichia coli* genome. V. DNA sequence of the region from 76.0 to 81.5 minutes. Nucleic Acids Res **22,** 2576–2586.

SULSTON, J., DU, Z., THOMAS, K., WILSON, R., HILLIER, L., STADEN, R., et al. (1992). The *C. elegans* genome sequencing project: A beginning. Nature **356,** 37–41.

Address reprint requests to:
Granger G. Sutton, Ph.D.
The Institute for Genomic Research
9712 Medical Center Drive
Rockville, MD 20850