AMASS: A Structured Pattern Matching Approach to Shotgun Sequence Assembly

Sun Kim†
sunkim@uiuc.edu
The Biotechnology Center
The University of Illinois at Urbana-Champaign
Urbana, IL 61801

Alberto Maria Segre
segre@cs.uiowa.edu
Department of Management Sciences
The University of Iowa
Iowa City, IA 52242

Abstract

In this paper, we propose an efficient, reliable shotgun sequence assembly algorithm based on a fingerprinting scheme that is robust to both noise and repetitive sequences in the data, two primary roadblocks to effective whole-genome shotgun sequencing. Our algorithm uses exact matches of short patterns randomly selected from fragment data to identify fragment overlaps, construct an overlap map, and deliver a consensus sequence. We show how statistical clues made explicit in our approach can easily be exploited to correctly assemble results even in the presence of extensive repetitive sequences. Our approach is both accurate and exceptionally fast in practice: *e.g.*, we have correctly assembled the whole *Mycoplasma genitalium* genome (approximately 580kbp) in roughly 8 minutes of 64MB 200MHz Pentium Pro CPU time from real shotgun data, where most existing algorithms can be expected to run for several hours to a day on the same data. Moreover, experiments with artificially-shotgunned data prepared from real DNA sequences from a wide range of organisms (including human DNA) and containing complex repeating regions demonstrate our algorithm's robustness to input noise and the presence of repetitive sequences. For example, we have correctly assembled a 238kbp human DNA sequence in less than 3 minutes of 64MB 200MHz Pentium Pro CPU time.

1. Introduction

The primary goal of the Human Genome Project is to determine the nucleotide sequence (and thus the information content) of human DNA. Unfortunately, long strands of DNA (the human genome contains on the order of 10^9 nucleotides) cannot be ''read'' directly: due to biological limitations, scientists can read DNA fragments of at most on the order of 1000 nucleotides (*i.e.*, 1kbp) at a time [Waterman95]. Shotgun sequencing is a commonly-used technique for circumventing this limitation. By

† Current affiliation: BCS&E, Blg 328 & B54; Central Research and Development; DuPont Experimental Station; Wilmington, DE 19880 (email: sunkim@micro5.es.dupont.com).

From: *Journal of Computational Biology* **6**:2 (1999)

combining information obtained from reading many randomly-selected fragments taken from multiple copies of the target DNA, it should be possible to assemble the original target sequence. Much existing research on this problem can be found in the literature [Bonfield95, Foulser90, Gingeras79, Green96, Huang92, Huang96, Idury95, Kececioglu95, Parsons95, Peltola84, Staden80, Sutton95]. This area is once again in focus because of a recently announced independent effort to sequence the human genome using shotgun sequencing, as opposed to the hybrid shotgunning/directed sequencing strategy used in the 15-year NIH/DOE Human Genome Project [Venter98]. The success or failure of this new effort depends largely on whether the best assembly algorithms are up to the task of assembling complex, genome-length human DNA sequences effectively and efficiently. To date, very few of these algorithms (most notably the TIGR Assembler, although arguably PHRAP has seen greater use) have actually been successfully applied to whole-genome shotgunning, and even then only at great computational expense on relatively short bacterial genomes containing only relatively simple repeating structure. In this paper, we propose a new algorithm we call AMASS whose primary contribution is the ability to assemble real sequence data containing complex repeating regions in the presence of noise (*i.e.*, the insertion, deletion or substitution of nucleotides during the fragment reading process) and at an exceptionally high rate of speed without sacrificing accuracy. Moreover, experiments with shotgun data from a wide range of organisms (including human DNA) and containing extensive repeating regions demonstrate our algorithm's robustness to noise and the presence of repetitive sequences.

In the next section, we provide background information on the shotgun sequence assembly problem, and, in particular, we focus on the difficulties associated with whole-genome sequencing. In Section 3, we introduce a streamlined version of AMASS, our new sequence assembly algorithm, and, in Section 4, we describe in detail how this algorithm can be extended to reliably handle repetitive sequences. In Section 5, we review the full version of the algorithm (including repeat handling) and, in Section 6, we give experimental results demonstrating the effectiveness and efficiency of our approach.

2. Whole Genome Shotgun Sequencing

The landmark *Haemophilus influenzae* Rd paper provides an excellent overview of the process used for whole-genome shotgun sequencing [Fleischmann95]. Roughly speaking, shotgun sequencing is a three-step process: (1) *sequencing*, (2) *assembly*, and (3) *finishing*. Large-scale automatic sequencing (generally via *gel electrophoresis*) of randomly selected fragments (step 1) is followed by use of automated tools (*i.e.*, the TIGR Assembler [Sutton95] in this case) to ‘‘cluster and assemble’’ fragments of the genome (step 2). The output from this second step generally consists of a relatively large number of *contigs* (assembled regions of the target DNA separated by gaps) which must then be manually bridged and closed in the finishing phase (step 3). While the overall cost of the process is generally dominated by the first and last steps, it is the development of reliable and reasonably efficient computational procedures for the second step that makes the whole process feasible in the first place.

The *shotgun sequence assembly problem* is to reconstruct a long DNA segment from randomly-sampled short fragment data by identifying overlaps between fragments. It is tempting, perhaps, to cast this problem as an optimization problem. Indeed, this is exactly the approach taken by much of the previously-cited research literature, but such a rigorous formulation is just not consistent with real world constraints:

> ‘‘...our goal is not to find an optimal solution according to some objective function, but rather to determine the solution that nature has selected.’’ [Karp98]

Any rigorous formulation of sequence assembly as an optimization problem inevitably must rely on underlying assumptions; yet assumptions about the structure of repeating regions, or even the distribution of nucleotides in the target DNA are, more often than not, simply not biologically plausible. It’s no accident that neither of the two most commonly used systems in practice, PHRAP [Green96] and the TIGR Assembler [Sutton95], formulate sequence assembly as optimization. Even so, it is not entirely clear whether even these algorithms are up to the whole-genome assembly task in general [Green97].

There are several major issues on the road to routine whole-genome shotgun assembly, including gap filling, and the handling of repeating regions, data anomalies, and polymorphisms. When taken together, these problems make the whole-genome shotgun prohibitively expensive, typically by increasing the degree of manual intervention required to produce a high-quality final sequence. Of primary concern is the issue of handling complex and extensive repeating structures. The successful shotgunning of whole bacterial genomes (*e.g.*, the aforementioned *Haemophilus influenzae* work) — which tend to have few repeating regions of limited size and complexity — is simply not likely to scale to human DNA with its complex and extensive repeating structure, as a greater number of more complex repeating regions leads inevitably to higher finishing costs.

Of course, cost-based arguments are based on existing technology; if at least some of these problems are addressed by new technology, *e.g.*, a new and improved sequence assembly engine, then the cost profile will change, and whole-genome sequencing may well become the strategy of choice (indeed, this is more or less the argument made in [Venter98]). The work presented in this paper addresses precisely this issue: efficient and robust assembly in the presence of noise and complex repeating regions.

Why is speed so important? Since the overall cost of the shotgun process is dominated by the first and last steps (*i.e.*, sequencing and finishing), a faster assembler that can provide essentially ‘‘real time’’ assembly running on commodity PC hardware might be integrated right into sequencing equipment, providing an ‘‘on-line’’ assembly capability. Current practice is to sequence to some predetermined amount of fragment data, regardless of whether that much data is actually needed for successful assembly; with on-line assembly, sequencing stops when the assembly is sufficiently complete, reducing the cost of the first step of the shotgun process. Of course, none of this matters if the assembler is unable to handle repeating structure or is not robust to noise.

3. A New Sequence Assembly Algorithm

We propose an efficient, reliable shotgun sequence assembly algorithm we call AMASS that is robust to both noise and repetitive sequences in the data, two of the primary roadblocks to effective whole-genome shotgun sequencing. AMASS takes as input a set of fragments[1], subject to noise, and a *coverage parameter* (the ratio of total fragment length to target sequence length), and produces an *overlap map* describing the relative positions of the input fragments in the target DNA. A *consensus sequence* is then generated, yielding an ordering of nucleotides corresponding to each *contig*, or mutually overlapping subset of the input fragments.

The inspiration for AMASS comes from hybridization fingerprinting, where biological probes are used to identify overlaps between DNA clones. In a similar fashion, we randomly select short patterns, or *probes*, from the input fragment data, where each probe is long enough not to occur by chance while still short enough to identify the part of the target sequence from which it comes even in the presence of noise. If noise is low, many of these probes can be found (via an efficient exact match algorithm) at the appropriate point in the overlapping fragments. Moreover, if we sample several probes at arbitrary positions from each fragment, longer fragment overlaps are likely to contain multiple matching probes with a specific ordering and spacing. A *structured matching* process that relies on the pattern of probe occurrences and interprobe spacing can then be used to find evidence of ‘‘real’’ overlaps even in the presence of insertion, deletion, and substitution errors.

Before we present a more detailed description of our algorithm, it is instructive to compare and contrast AMASS with other existing assembly algorithms. Indeed, the general framework used in AMASS

[1] Since DNA is double-stranded, and we don’t know *a priori* which strand each fragment comes from, we double the size of the input fragment set by adding the reverse complement sequence of each fragment. This is typically what other assembly algorithms do; note that while increased computation time and space result, we can mitigate this penalty by eliminating the dual of a fragment from the remaining fragment set during the assembly process once a fragment’s actual orientation becomes apparent.

is not entirely different from that of other algorithms in the literature. Like many other assembly algorithms, AMASS constructs contigs in a greedy fashion, guided by some measure of fragment alignment. The primary difference is that AMASS uses a heuristic metric based on patterns of probe occurrences and interprobe spacings as opposed to, *e.g.*, a metric based on pairwise fragment alignments [Smith81]. This heuristic metric can be used much more efficiently and can yield more information than a metric based on pairwise fragment alignments. More specifically:

(1) *More efficient:* At the most fundamental level, evidence of fragment/fragment or fragment contig overlaps is provided by exact matches of short patterns. Other algorithms resort to dynamic programming for computing pairwise alignments because the presence of noise precludes the use of exact string matching algorithms. But when many probes are sampled from a fragment, the generally low noise rate encountered in real sequencing facilities along with the relatively short lengths of the probes themselves ensures that most of these probes will be found. Since exact pattern matching is considerably more efficient than dynamic programming, AMASS enjoys a significant performance advantage.

(2) *More compact:* AMASS employs a compact internal representation for both fragments and contigs consisting of an ordered set of probe identifiers along with the interprobe spacing information. The compactness of the representation yields significant performance advantages over approaches that must instead rely on explicit sequence representations; when paged memory hardware thrashes, little real work gets done.

(3) *More informed:* The uniform representation for fragments and contigs, combined with the low computational burden imposed by the heuristic metric, allows the metric to be recomputed each time a fragment is added to a contig. Thus fragment placement decisions are based on a dynamic judgment of how a fragment scores against a collection of fragments, rather than how a fragment scores against each individual constituent fragment in a contig. Furthermore, AMASS needn't commit to contig consensus sequence until all fragments are assigned to a contig, permitting a consensus sequence construction algorithm to fully exploit the higher degree of coverage typical in shotgun sequencing data.

(4) *More robust:* In addition to tolerance to fragment reading errors provided by using patterns of probe occurrences instead of pairwise sequence alignment, statistical information about the frequency of probe occurrences is easily collected and yields valuable clues into the repeating structure of the target DNA. This represents one of the primary contributions of our work (see Section 4).

We now turn our attention to a more detailed description of the AMASS system. AMASS operates in three distinct phases: (1) probe selection and matching, (2) overlap map construction, and (3) consensus sequence generation. Each of these phases will be discussed separately, although — for the sake of expository clarity — we first describe a reduced version of AMASS, postponing our discussion of repeat handling until Section 4.

3.1. Probe Selection and Matching

The first step in the assembly process is to select probes and then identify all of their occurrences in the input fragment data (see Figure 1). We randomly select a fixed number of probes from each input fragments and then find all occurrences of every probe using a multipattern search algorithm.[2]

Of course, any multipattern string match algorithm could be used to find all occurrences of the probes in the fragment data; this is essentially an exact string matching problem, for which many efficient algorithms are known. The multipattern string match algorithm used in our implementation (see [Kim97] for details) exploits a compact 2-bit encoding of nucleotides as well as the short, fixed length, nature of

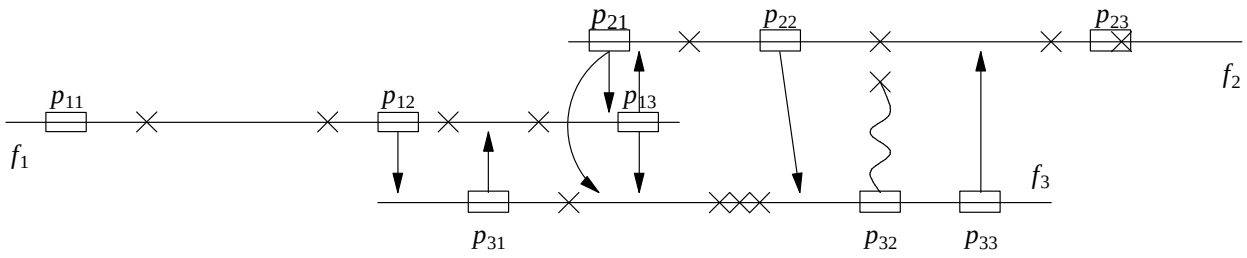

Figure 1: Probe matching and fragment representation. Three probes are sampled from each of three fragments $\{f_1, f_2, f_3\}$. A multipattern search algorithm is then used to find all exact matches for all probes in each fragment, and an internal representation is constructed for each fragment as an ordered set of probe occurrences (*i.e.*, $f_1 = \{p_{11}{:}\, n_1, p_{12}{:}\, n_2, p_{31}{:}\, n_3, p_{21}{:}\, n_4, p_{13}, n_5\}$, where $p_{ij}{:}\, n_k$, represents the pattern p_{ij} at position n_k). The low sequence reading noise rate (each $\times$ represents an insertion, deletion, or substitution error) ensures that most probes are found in their corresponding positions on overlapping fragments, with perhaps just some minor variation in interprobe distances.

[2] While the current implementation of AMASS simply selects probes at random, future implementations could well exploit a more sophisticated probe sampling model that exploits expected noise rates for sequence reading. It is well known that reading errors are more likely to occur nearer the end of the fragment, *i.e.*, that there is a ''clean'' end and a ''noisy'' end to each fragment, and that the precise error profile varies by sequencing operation. Instead of selecting probe locations with uniform probability, one could easily incorporate laboratory-specific noise models to improve the quality of probes sampled. Note that this procedure would entail some significant modification of the repeat handling mechanism of Section 4.

probes and is one key aspect of our system's efficiency. It uses word-length bit operations to identify matches and runs very quickly (for example, it takes just over 2 seconds of 64MB 200MHz Pentium Pro CPU time to find all 256,742 occurrences of 54,621 16bp probes in 5601 fragments averaging 500bp taken from a 400kbp sequence).

Underlying our approach is the notion that probes which are ''long enough'' effectively identify most true overlaps, while identifying false overlaps only with low probability. While formal probabilistic analysis might, with some difficulty, be used to justify this insight, an empirical demonstration showing how long ''long enough'' is likely to be under different probe lengths and noise conditions seems more appropriate here.

For this demonstration, two fragments are said to overlap if they share a single probe occurrence (note that AMASS actually uses a higher — and more effective — standard requiring multiple probe occurrences with appropriate interprobe spacing). We use two empirical metrics, *sensitivity* and *specificity*, to measure the effectiveness of this simple overlap criteria. Sensitivity describes how well the overlap criteria identifies true overlaps, while specificity describes how well the overlap criteria rejects false overlaps. More formally, let *true positive* represent the number of (correctly) identified ''real'' overlaps, *true negative* represent the number of possible false overlaps which are (correctly) not identified, *false positive* represent the number of overlaps detected which do not correspond to ''real'' overlaps, and *false negative* represent the number of ''real'' overlaps which should have been detected but were missed instead. Thus:

$$sensitivity = \frac{true\ positive}{true\ positive + false\ negative}$$

and

$$specificity = \frac{true\ negative}{true\ negative + false\ positive}\ .$$

The higher the value of each metric is, the more effective the overlap criteria. The purpose of this demonstration, then, is to see how the two metrics change as the probe length and the noise rate are varied.

We proceed as follows. First, we take a 100kbp section of a 238kbp human DNA sequence and artificially shotgun this data (average fragment length 500bp, average coverage 8) using GENFRAG [Engle93, Engle94]. Next, just 5 fixed-length probes are randomly selected from each fragment and matched against the input fragment data in a single pass using an efficient multipattern search algorithm; we compute sensitivity and specificity over a range of noise rates and probe lengths (note that, for this demonstration, we define an overlap to be true if its length exceeds the probe length).

Table 1
Single Probe Overlap Detection Sensitivity and Specificity

Noise rate	Probe length (bp)	Sensitivity	Specificity
0.00	8	1.00	0.25
	12	0.95	0.97
	16	0.95	0.99
0.02	8	1.00	0.29
	12	0.9	0.98
	16	0.87	0.99
0.04	8	0.99	0.32
	12	0.83	0.98
	16	0.71	1.00
0.08	8	0.98	0.4
	12	0.54	0.99
	16	0.29	1.00

The results in Table 1 are of interest because they illustrate some general trends. First, as probe length increases, sensitivity decreases; an effect that is exacerbated by noise. This makes intuitive sense, since insertion, deletion or substitution errors lead to missed probe occurrences, which leads to reduced sensitivity; the longer the probe, the more likely it is to be affected by noise. Second, as probe length increases, specificity increases. This also makes intuitive sense, since longer probes are less likely to arise

by chance, or to correspond exactly to more than one copy of a repeated region. Thus sensitivity and specificity define a tradeoff; the best performance is obtained when maximizing both properties. For this particular demonstration, 12bp seems about right, since 12bp probes are ‘‘long enough’’ to occur rarely by chance in a 100kbp sequence, yet ‘‘short enough’’ to be (relatively) insensitive to noise at the rates shown here. In general, the probe length required depends on the length of the target sequence; the longer the target sequence, the longer the probe required to uniquely identify true overlaps. Unfortunately, as probe length and noise rate increase, sensitivity decreases, meaning more true overlaps are missed.

AMASS addresses this issue in two ways. First, rather than relying entirely on singleton probes in common, we use a more sophisticated *structured probe matching* technique — described in the next section — which bases overlap determinations on occurrences of multiple probes at appropriate intervals combined with a secondary search technique called *satellite matching*. By relying on multiple probes, we reduce the number of false negatives (missed overlaps), thereby increasing sensitivity. Similarly, by rejecting spurious overlaps, satellite matching reduces the number of false positives, thereby increasing specificity. Both of these observations are supported by the experimental evidence presented later in this paper.

3.2. Overlap Map Construction

Once probes have been selected and the input fragments have been internally represented as ordered sets of probe occurrences and their interprobe spacing, we move to the construction of the overlap map. Recall the overlap map describes the relative positions of the input fragments in the target DNA. In practice, since the input fragment set may not actually cover all of the target DNA sequence itself, we expect that the overlap map will consist of a relatively small number of contigs, also represented as ordered sets of probe occurrences and their interprobe spacings.

Overlap map construction is based on an *overlap table* which initially scores each fragment against every other fragment in the input using the heuristic scoring metric described below. Map construction is performed greedily (see Figure 2); first, we select the fragment pair with the highest score and construct an initial contig containing just these two fragments. The resulting contig is added to the overlap table and scored against all remaining fragments. The process continues, adding the best-scoring fragment to the contig and updating the modified contig's scores against all remaining fragments. When no remaining fragment exhibits a significant overlap score (defined by a prespecified parameter) with the current contig, a new contig is constructed, or, if no fragments remain, the process terminates.

The common representation used for fragments and contigs (*i.e.*, as an ordered set of probe identifiers and their interprobe distances) is critical to the success of this greedy algorithm, as comparing a fragment to a contig inevitably yields a more sensitive overlap test. Merging a new fragment into an existing contig is quite straightforward; while small differences in interprobe distances might occur, in practice reconciling these differences in just about any reasonable fashion is fine since the scoring function is insensitive to small distance errors. Furthermore, note that fragments can be incorporated into contigs without resorting to constructing the underlying base sequence. Constructing a consensus sequence is therefore always postponed until all fragments are mapped, allowing the consensus sequence generation process to resolve differences using the full coverage of the data.

What remains to be specified is the heuristic structured mapping scoring metric used to rate fragment/fragment or fragment/contig overlaps. The AMASS scoring metric is based mostly on the number of matched probes, appropriately weighted by the interprobe distances. Note that there might on rare occasion be more than one alternative mapping between two fragments; in this case, the overlap score is the highest score among the alternatives.

A second concern is the effect of repeating sections on the scoring, especially in regions densely populated by probes. This problem is particularly troublesome when interleaved probes (*i.e.*, probes that

---

Initialize set S to the set of input fragments.
1. $S = \{ f_1, f_2 \cdots f_F \}$
Initialize overlap table by scoring all fragment pairs.
2. $overlap(s_i, s_j) = score(s_i, s_j)$ where $s_i, s_j \in S$ and $i < j$
Find largest entry in overlap table.
3. $overlap(s_{imax}, s_{jmax}) = max(overlap(s_i, s_j))$ where $s_i, s_j \in S$ and $i < j$
Check overlap against threshold.
4. if $overlap(s_{imax}, s_{jmax}) < \sigma$ then **return**(S)
Merge best matches together, forming seed contig, then update S.
5. $s_{new} = merge(s_{imax}, s_{jmax})$
6. $S = S \setminus \{s_{imax}, s_{jmax}\}$
Rescore new element against existing elements of S.
7. $overlap(s_k, s_{new}) = score(s_k, s_{new})$ where $s_k \in S$
Find highest scoring element of S.
8. $overlap(s_{max}, s_{new}) = max(overlap(s_k, s_{new}))$ where $s_k \in S$
Check overlap against threshold.
9. if $overlap(s_{max}, s_{new}) < \sigma$ then
    # Add new contig to S and look for another contig
    10. $S = S \bigcup \{s_{new}\}$
    11. **goto 3**
  else
    # Update contig and S, then look for more.
    12. $s_{new} = merge(s_{max}, s_{new})$
    13. $S = S \setminus \{s_{max}\}$
    14. **goto 7**

Figure 2: Overlap map construction algorithm. Input consists of a set of fragments $\{ f_1, f_2 \cdots f_F \}$ and a significance threshold, σ. The output consists of S, a set of contigs.

---

physically share portions of the DNA sequence) are present. For example, one might find several interleaved probes within a small repeat (say 30bp). If we counted these interleaved probes individually, it would artificially inflate the corresponding overlap score. Since short repeats are relatively common, we must be careful not to be misled in this manner; hence, we ‘‘collapse’’ interleaved probes and use instead the number of disjoint probe matches in computing overlap scores.

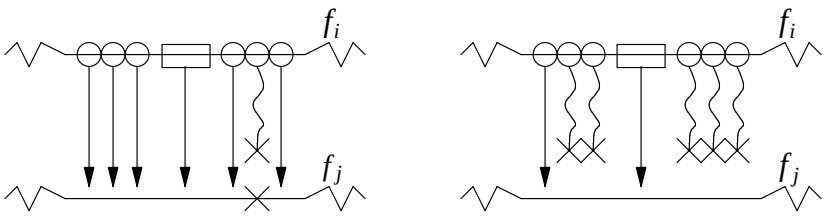

Figure 3: Satellite matching. Short patterns are sampled from the area around a candidate overlap's probe matches. If the overlap is a true overlap (left), most such short patterns will be easily found in the other fragment, thanks to the low sequence reading noise rate. If the overlap is a spurious overlap (right), then most, if not all, of the short patterns will be absent, causing the candidate overlap to be rejected.

Finally, as we have seen previously, spurious matches of probes — while rare if the probes are long enough — may still arise due to the presence of (relatively longer) repeating regions or even, albeit exceedingly rarely, from simple reading errors. To exclude such cases from consideration, AMASS uses a form of secondary search called *satellite matching* to confirm candidate overlaps identified on the basis of probe occurrences.

Satellite matching (see Figure 3) confirms the validity of a pattern of probe occurrences by searching for secondary, shorter, probes in the interprobe gaps. Because satellite matching exploits existing positional information, it can be implemented very efficiently. When we search for probe occurrences in the probe matching phase, we have to search for every probe in every possible location on every fragment since we do not have any *a priori* information about a possible overlap. Once an overlap is being considered, however, we need only search for matches of short patterns at approximately fixed distances (within a small error bound) from the known probe occurrences. When properly used, satellite matching is useful in ultimately rejecting false overlaps (*i.e.*, those based on very few (or even just one) probe occurrences), which inevitably leads to a more specific overlap test.

Structured matching combines disjoint probe occurrences, total probe occurrences, interprobe distances and satellite matching to give a heuristic estimate of fragment/fragment or fragment/contig overlap quality. More precisely, the metric used by the current version of AMASS is:

$$N_{sat}(\omega N_{dis} + N_{mat})$$

where N_{mat} is the number of probe matches, N_{dis} is the number of disjoint probe matches, ω is the distance (in base pairs) between the first and last common probe occurrences, and N_{sat} is the number of successful satellite matches.

A number of observations are in order. Note that this particular metric does not penalize ‘‘missing’’ or unmatched probes, but rather only rewards common occurrences at mutually-consistent locations.[3] Missing probes, of course, can arise due to insertion, deletion, and substitution errors which occur in the course of reading a fragment; ignoring a few missing probes here and there makes the metric robust in the face of this type of noise. Note furthermore that this score metric requires only integer operations, and can therefore be performed more efficiently than a score that relies on any non-integer operations. Thus structured matching, when combined with the uniform and compact representation of fragments and contigs described previously, is quite efficient in practice. For example, using the greedy algorithm just described and notwithstanding the fact that all contig/fragment overlaps are rescored every time a new fragment is added to a contig, constructing a complete overlap map of a 300kbp sequence from 4801 fragments takes only about 2 minutes on a 64MB 200MHz Pentium Pro CPU.

3.3. Consensus Sequence Generation

Once the overlap map is complete, the next step is to construct a consensus sequence. Recall each contig is represented as a collection of probe occurrences along with the appropriate interprobe distances,

[3] Indeed, several metrics that involved a naive penalization scheme were found not to be effective in practice, although alternative scoring metrics remain a topic of interest for future research.

as well as the identities of fragments which comprise the contig. Thus to determine the consensus sequence for the contig, we must reconcile all the individual fragment sequences. One view of this problem, which is consistent with previous work in the area, is to see it as an instance of multiple sequence alignment problem [Huang92, Huang96, Kececioglu95, Sutton95]. Indeed, just about any multiple sequence alignment algorithm could be used to provide a consensus sequence using the overlap just constructed as a guide. In contrast, the current version of AMASS uses a natural extension of structured matching to produce a consensus sequence.

Recall the basic idea of structured matching is that when probes, which are long enough not to occur by chance, occur in the same order and with sufficiently similar spacing in two or more fragments, then this is evidence that the fragments overlap. Furthermore, these occurrences show roughly how the fragments align. We can extend this alignment by selecting new probes from the spaces between the existing probes, aligning multiple fragments where the new probes match. Iteratively decreasing the size of the new probes results in an alignment of patterns which covers most parts of the fragments. This process generates a sequence of matched probes of various lengths which are free of errors with high probability, since the probability of chance occurrences of the same probe at approximately the same locations in short fragments which are already known to overlap is vanishingly small. The matched probes in the alignment constitute the consensus sequence.

The remaining issue, then, is how to go about closing whatever gaps may remain between matched probes. By construction, such gaps must correspond to a divergence caused either by errors in one or more fragments or by the presence of a fragment which doesn't really belong there. Consider the situation shown in Figure 4; after applying the iterative probe matching procedure just described, the remaining short gaps correspond to noise in the original fragment data (assuming no misplaced fragments

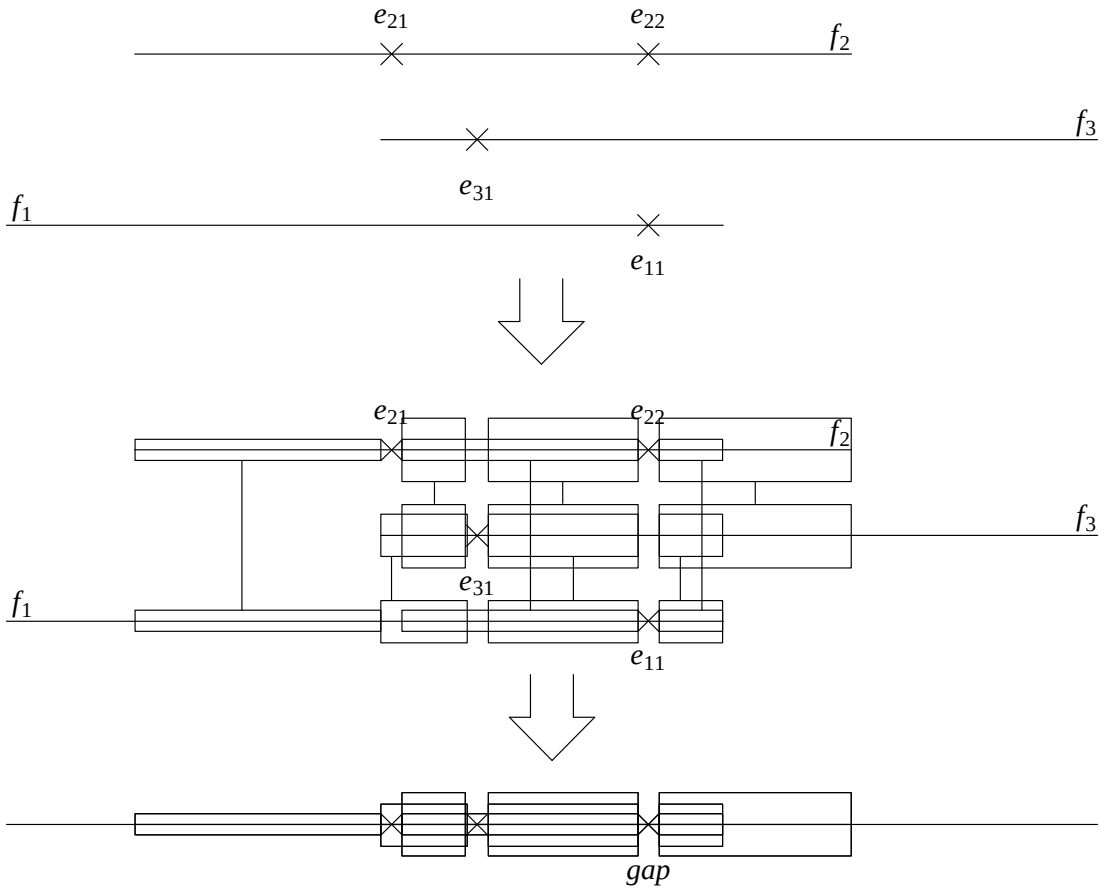

Figure 4: Consensus sequence generation example. Top picture shows a set of three fragments, each containing one or more insertion, deletion or substitution error (denoted ×). Pairwise alignments, denoted by linked boxes, are performed (middle picture), and then the consensus sequence is derived via majority rule (bottom picture). Only regions with appropriate patterns of multiple coincident errors (marked *gap*) cannot be closed in this fashion.

here). Since the gaps here all correspond to a region with coverage of at least three[4], a simple majority rule algorithm (essentially based on pairwise alignments between each pair of fragment present) can be used to close all single-error gaps. Multiple coincident errors (*e.g.*, between e_{11} and e_{22}) will not be

[4] Any region with coverage below three is underspecified and cannot be corrected. In such cases, we simply select the nucleotide sequence of one of the two fragments as the consensus sequence, since we have no basis to *a priori* prefer one fragment over the other.

resolved so easily; in such cases — which occur only infrequently — one might use a computationally-expensive multiple alignment algorithm to close the remaining gap, or one might simply accept one fragment's nucleotide sequence as the true consensus sequence.[5]

4. Handling Repetitive Sequences

In real DNA, complex mixtures of multiple repetitive sequences are not uncommon. Individual repeats differ in both length (measured in base pairs), similarity (*i.e.*, are multiple copies identical, or do they differ slightly), and copy number (*i.e.*, the number of times the same sequence occurs). For this reason, correctly assembling sequences containing repeats is a very complex task. Yet although proper handling of repeats is of prime importance when assembling real sequences, the problem has been ignored in many sequencing systems. Only recently have three sequence assembly algorithms, CAP2 [Huang96], TIGR Auto Assembler [Sutton95], and Phrap [Green96], proposed ways to handle repeats. Unfortunately, the techniques proposed in these algorithms are either limited to small copy numbers [Huang96] or require additional information (outside of the input fragments) to help disentangle the repeats [Green96, Sutton95].

Here, we propose a systematic way to handle a complex mixtures of short and long repeats without the use of additional information. The underlying intuition comes from an observation that, if the coverage is uniform (that is, sequenced fragments are randomly selected with uniform probability), then probes with a larger than expected number of occurrences are — with high probability — from repeating regions. Since probe occurrence counts and other, similar, statistics are easily collected during the probe

[5] In practice, with the relatively low noise rates commonly found in real sequencing data, multiple coincident errors will occur with very low probability. Hence unusually large, or simply difficult-to-close gaps may well be indicative of misplaced fragments, which are also highly unlikely (since such a false match would have had to both return a high overlap score and survive satellite matching). When misplaced fragments occur, there are usually other factors in play, such as the presence of a repeating region. While our philosophy is to do our best to avoid misplaced fragments in the first place, it may well be possible in the future to use gap size and patterns of gap occurrences to detect and correct any misplaced fragments that remain at consensus sequence generation time.

matching phase, we can use these statistics to identify and properly assemble repetitive sequences.

Our general strategy is as follows. First, we collect statistics (such as probe occurrence counts) during the probe matching phase. Second, we identify probes that are likely to correspond to short repetitive sequences and remove these probes so that they are not considered during assembly. Third, using the remaining probes only, we identify fragments that are likely to correspond to longer repetitive sequences, and also exclude these fragments from the assembly. The assembly then proceeds normally using only the surviving fragments and producing contigs that correspond to non-repeating regions. Finally, the excluded fragments are used to assemble the repeating regions, which are then merged with the previously constructed contigs to recover the original sequence.

In the remainder of this section, we examine in closer detail our approach to handling both short and long repetitive sequences, respectively.

4.1. Handling Short Repetitive Sequences

Short repeats may occur frequently within a relatively short section of DNA; for example, short repeats of 85bp occur 22 times in a particular section of *Caenorhabditis elegans* [Agarwal94]. To avoid misleading the structured matching heuristic, we would like to set aside probes from such repeating regions of the original sequence during the assembly process. The question, then, is how to identify probes as coming, with high likelihood, from these short repeating regions. Our approach relies on several statistical cues which are easily and quite naturally collected during the probe matching phase, before fragment assembly begins.

The first statistic used to detect repeats is the *probe count distribution*, which characterizes each probe according to its frequency of occurrence in the input fragment set. A sample probe count distribution derived from an artificially shotgunned *Mycobacterium leprae* cosmid B1496 (41kbp) with coverage of 8 and a 5% noise rate is shown in Figure 5 (the shotgun data consists of 661 fragments, and

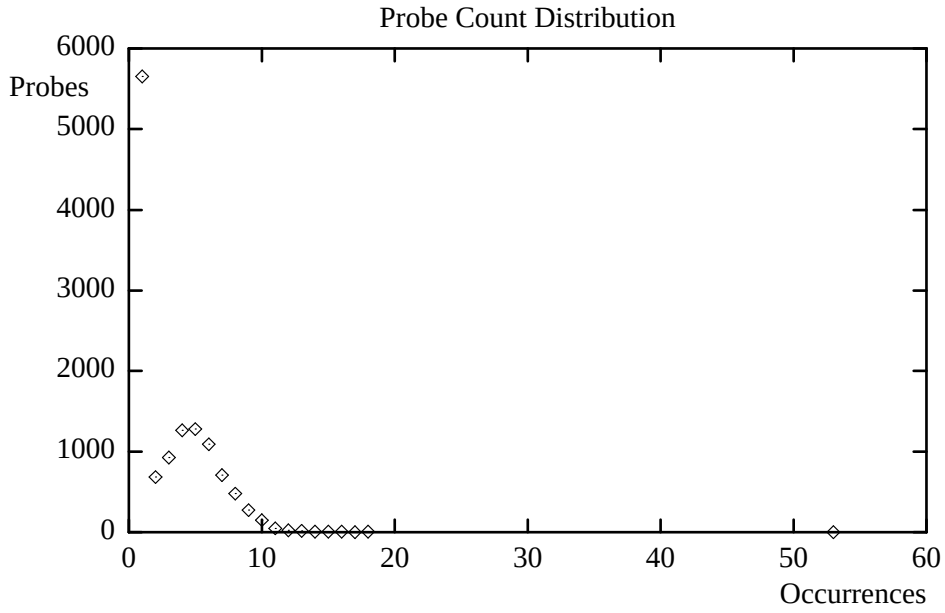

Figure 5: Probe count distribution for artificially shotgunned *Mycobacterium leprae* cosmid B1496 (41kbp). The large number of probes with only one occurrence correspond to probes containing errors due to noise, while the singleton outlier with occurrence count of 53 corresponds to a short repeating region.

10 13bp probes were selected from each fragment). Two immediately obvious features of this plot are worthy of more discussion. First, we note that almost 6,000 probes have only one occurrence. Since the noise rate is 5%, the chance that a 13bp probe contains at least one error is almost 50%. Yet given that the probes are 13bp long and the sequence length is 41kbp, such ‘‘flawed’’ probes are unlikely to occur elsewhere in the data by chance. Thus we can attribute these singleton probes to noise and simply ignore them, excluding them from the assembly process.[6]

[6] Of course, singleton probes might also come from low coverage regions of the target sequence. Nevertheless, since they don’t match anything and cannot therefore contribute to the assembly process, they can safely be excluded.

Once singleton probes are excluded, we might expect what remains of the probe count distribution to approximate a normal distribution with mean dependent on the length of the probes, the length of the target sequence, and the coverage and noise rate of the shotgun data (indeed, we note the peak in the distribution is at 5, just below the input data's average coverage value 8). Instead, we note that the distribution has a long tail, indicating that probes in the tail region have a higher number of occurrences than expected. We claim such probes correspond, with high probability, to repeating regions in the input data.

To see why this is so, consider a lone outlier data point with an occurrence count of about 53: if the probe occurs only once in the target sequence, then it must come from an area of the target sequence whose coverage is almost 7 times the average coverage value. Given that this is highly unlikely, the only logical conclusion is that probes corresponding to outliers must come from repeating regions.

We set a threshold value for lone outliers in the probe count distribution and remove singleton outliers if their values are beyond this threshold value. We call this threshold value the *probe distribution threshold*, and an appropriate value depends on, *e.g.*, the coverage of the data (see [Kim97] for some guidelines on setting this threshold appropriately). Note that we only remove singleton outliers, and not outliers which are closely clustered together. This is because long — rather than short — repeats will span multiple probes, and so the related probe distribution counts can be expected to cluster about a value related to the copy number for that particular repeating region. Such probes are not removed at this stage since they provide valuable clues to long repeating regions, which are discussed in Section 4.2.[7]

Another statistical clue can also be used to help detect short repeats. Recall that initial construction of the overlap map requires scoring all pairs of fragments, which entails performing satellite matching for

[7] Indeed, this particular sequence contains two copies of a 235bp repeat in addition to the short repeats just described [Huang96].

any common probe occurrences between two fragments. If two occurrences of the same probe are from true overlapping fragments, then the satellite matching test will succeed; otherwise, the satellite matching test is likely to fail. In the course of constructing the overlap table, we can easily collect statistics for each probe counting how many times satellite matching tests between two different occurrences succeeded and failed.

What does a high failure ratio for satellite matching tests based on a given probe mean? Since the goal of satellite matching is to see whether two occurrences of the same probe are from true overlaps, a higher failure ratio means that many of the occurrences of the probe are not likely to come from true overlaps (provided the noise level is low and relatively uniform across fragments). It is easy to conclude that those probes must instead be from short, highly repetitive regions of the target sequence.

We set a threshold value for the satellite matching test failure/success ratio and remove probes if their ratios are beyond this threshold value. We call this threshold value the *divergence threshold*, and an appropriate value depends on the noise rate of the shotgun data (since more noise will also cause an increase in failed satellite matching tests; see [Kim97] for some guidelines on setting this threshold appropriately). Of course, by removing highly divergent probes, we may lose some probes that actually come from true overlapping regions. While this is true, our assembly procedure — exploiting multiple coverage by repeatedly computing overlap scores against the current contigs — does not in practice miss many true overlaps.

In summary, when constructing the pairwise overlap table, we exclude singleton probes (most likely due to noise), probes whose lone occurrence counts exceed a preset probe distribution threshold value, and probes whose satellite matching failure/success ratio exceeds a preset divergence threshold value. Such probes are most likely to come from short repeating regions.

4.2. Handling Long Repetitive Sequences

Next we focus on handling long repeating regions using only probes which have survived the tests for short repeating regions outlined in the previous section. Our general strategy is to identify fragments from long repeating regions and exclude them from the assembly. The excluded fragments are then used to construct *repeat clusters*, which are conceptually much like contigs but may contain fragments from different copies of the repeating region. Once constructed, the different copies are ''teased apart'' from each repeat cluster to produce contigs which can be merged together with the contigs constructed from the remaining fragments in order to produce a final assembly.

As with short repeats, we use statistical criteria on the remaining probes to identify long repeats. Unlike with short repeats, however, the statistics used here are fragment-centric as opposed to probe-centric, in order to better exploit the fact that longer repeats will by definition contain multiple probes. The first clue used to detect longer repeats is the *fragment distribution*, which characterizes each fragment according to the number of probes it contains: each (x, y) data point in the fragment distribution denotes that there are y fragments with x constituent probe occurrences. Given our representation of fragments as ordered sets of probe occurrences, it is an easy task to count and monitor the number of probe occurrences in each fragment. Simple analysis reveals that — if probes are uniformly distributed along a fragment — long repeating regions produce fragments with a higher than expected number of probes per fragment.

An example should help make this clear (see Figure 6). Consider four fragment distributions obtained from two copies of an artificial 50kbp sequence, one without repeats (top of Figure 6) and one with 5 occurrences of a 500bp repeat inserted (bottom of Figure 6). Both sequences were artificially shotgunned (average fragment length 500bp, average coverage 8) using GENFRAG, and fragment distributions were obtained using 10 8bp probes per fragment (left side of Figure 6) and 10 15bp probes per fragment (right side of Figure 6). The fragment distributions from the nonrepeating sequences (top of Figure 6) look symmetric; since low coverage is as likely as high coverage and the number of probe

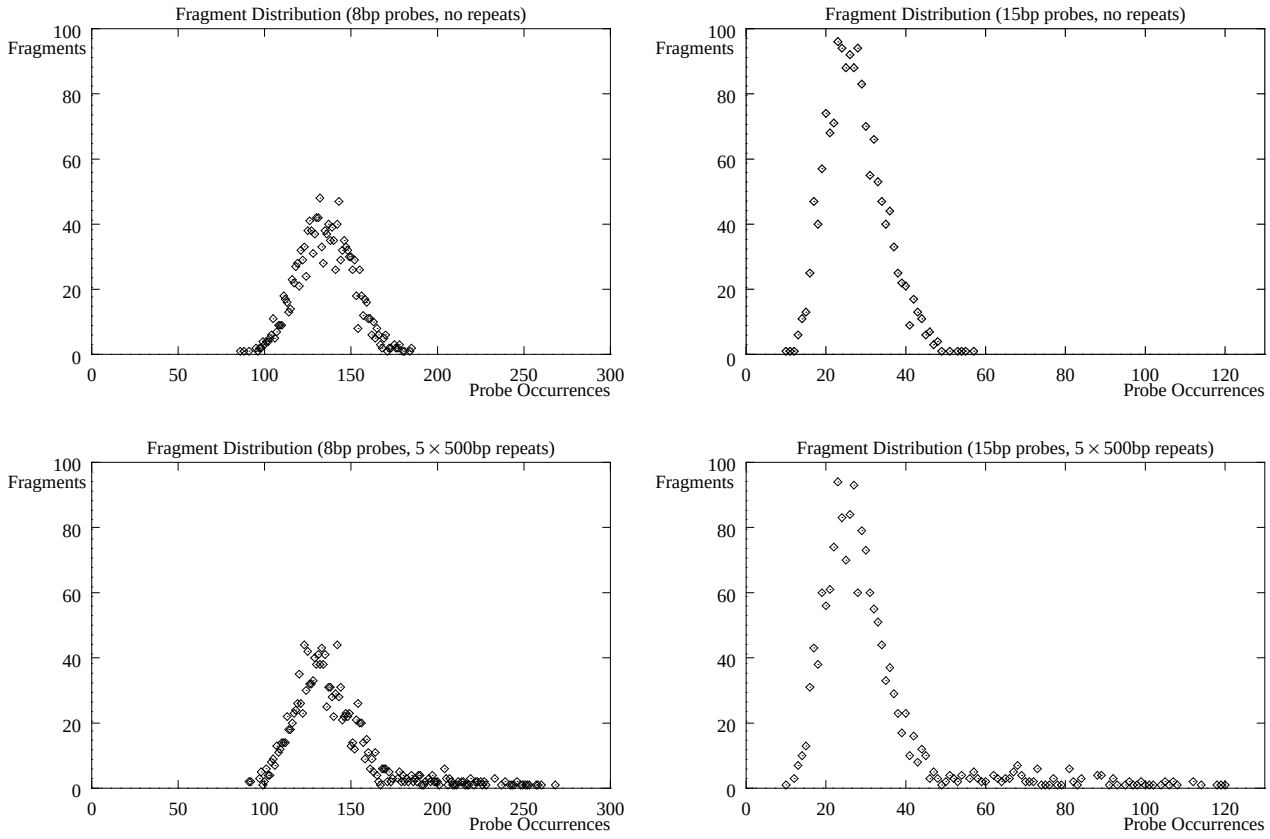

Figure 6: Fragment distributions for sequences without repeats (top) clearly differ from fragment distributions from similar sequences but containing multiple copies of relatively long repeating regions (bottom).

occurrences in a fragment roughly depends on coverage, the number of fragments with low probe occurrences is roughly the same as the number of fragments with high probe occurrences. However, fragment distributions from the sequence containing repeats (bottom of Figure 6) are clearly not symmetric, and look a bit like normal distributions with long tails to one side. We claim that the ‘‘tail’’ regions so observed correspond to fragments from long repeating regions; in this example, the length of the tail is proportional to the number of occurrences of the 500bp repeat. Unfortunately, since real DNA contains several different kinds of repeats of varying similarity, it is in general very difficult to predict the

number of occurrences of long repeats from the fragment distribution alone.

Of course, real DNA contains confounding factors that make finding long repeats less clear than this simple example. Even so, the fragment distribution does yield valuable information about the existence and number of long repeats; simply setting a threshold to separate the ‘‘tail’’ from the ‘‘body’’ of the distribution could provide enough information to identify long repeats. Unfortunately, although fragments from long repeats should tend to have higher than expected values in the fragment distribution, we cannot expect that all long repeats will yield greater-than-expected values, nor even that all fragments with greater-than-expected values are necessarily from long repeats. This is because some repeats may not be long enough — or may have such low coverage — that fragments from these repeats do not exhibit higher probe occurrences than fragments from nonrepetitive portions of the DNA. Furthermore, some portions of the DNA may display unusually high coverage, resulting in fragments that exhibit unusually high fragment distribution values.

One solution is to change the definition of ‘‘probe occurrences per fragment’’ which is used in constructing the fragment distribution. By counting probe occurrences in a subsection of a fragment, *e.g.*, 100bp rather than the whole fragment, the statistic becomes more sensitive to relatively shorter long repeats. Using a ‘‘sliding window,’’ we scan through each fragment, counting probe occurrences within the window. The probe occurrence count of a fragment is then defined as the highest probe occurrence count obtained within the window while scanning through the fragment; it is this value which is used to construct the fragment distribution.

We set a threshold value for separating the tail region of the fragment distribution from the body of the distribution; we call this threshold value the *fragment distribution threshold*. Note that a lower fragment distribution threshold will be able to identify fragments from low-coverage repeating regions; of course, a fragment distribution threshold which is too low will wrongly identify some fragments as coming from repeating regions. The current implementation of AMASS uses a fragment distribution

threshold value corresponding to the 60th percentile of the fragment distribution. While this relatively simplistic threshold is admittedly *ad hoc*, it has been used to obtain all of the results described in this paper.

Once fragments with probe occurrences higher than the fragment distribution threshold value are marked, we use these fragments to construct *repeat clusters*. A repeat cluster is essentially a contig corresponding to a repeating region, where fragments from multiple copies of the same repeating region may be forced to artificially coexist. A slightly modified version of the overlap map construction algorithm (Figure 2) is used to construct repeat clusters. More precisely, repeat clusters are initially seeded using high-scoring overlaps that contain at least one marked fragment. New fragments added to a cluster are also marked, but only if they overlap other marked fragments in the cluster by at least 90%, the *repeat fragment overlap threshold*.[8] Finally, repeat cluster construction terminates when there are no more overlaps containing at least one marked fragment available.

As noted previously, depending on the fragment distribution threshold value used, the repeat clusters so constructed may not all correspond to long repeats; some may instead come from very high coverage regions. Thus we need a way to check whether a repeat cluster does indeed contain long repeats. The basic idea is that intersecting regions of any two overlapping fragments in a repeat cluster should be almost identical sequences with just a few errors (exactly how many depends on the noise rate of the shotgun data), while fragments from different copies can be expected to display more differences, especially in the regions just on either side of the repeat itself. If we could somehow compute all pairwise alignments between fragments in the repeat cluster and count how many times they diverge, we'd be in a position to judge whether the fragments came from different copies of a repeat or whether they represent a

[8] As with the fragment distribution threshold, setting the repeat fragment overlap threshold to 90% is admittedly *ad hoc*, but has been used to obtain all of the results described in this paper.

single, high coverage, region of the original sequence.

Unfortunately, computing combinatorially-many pairwise alignments would be much too expensive, even if we resort to the computationally efficient heuristic alignment algorithm of Section 3.3. Instead, we use a simpler test with much lower computational cost. Recall that when constructing a repeat cluster, new fragments are positioned within the repeat cluster according to the highest scoring (heuristic) alignment. We sort fragments in the contig according to the position of their left ends (in ascending order) (see Figure 7) and perform pairwise pattern alignment only on successive pairs of fragments in the sorted repeat cluster. If all successive pairwise alignment succeed, then we conclude that a repeat cluster does not in fact represent a long repeat, but rather just region with abnormally high coverage.

The question, then, is precisely what metric should be used to evaluate the alignment of successively ordered fragments in a repeat cluster. The current version of AMASS uses a threshold based on the ratio of aligned base pairs to overlap base pairs; we call this threshold the *pairwise alignment threshold*, where an appropriate value depends on the rate, distribution, and variance of errors in the input fragment data (as before, see [Kim97] for some guidelines on setting this threshold appropriately).

The following example should help make this process clear. Consider the repeat cluster shown in Figure 7, and assume a pairwise alignment threshold of 0.8. We first sort all fragments according to their left end positions. Taking each successive pair of fragments in the sorted repeat cluster, we evaluate each successive pair of overlapping fragments, and obtain values of $0.9, 0.85, 0.89, 0.92$, and 0.8, respectively. Since we had set the threshold value to 0.8, we conclude that each successive pair of fragments represents a true overlap, and that the repeat cluster in Figure 7 does not correspond to multiple copies from a repeating region.

If a repeat cluster is found to correspond to a long repeating region, we next attempt to ‘‘tease out’’ the individual contigs from the repeat cluster, where each contig corresponds to a copy of the repeating region. The current implementation of AMASS uses an imperfect but efficient heuristic to separate copies

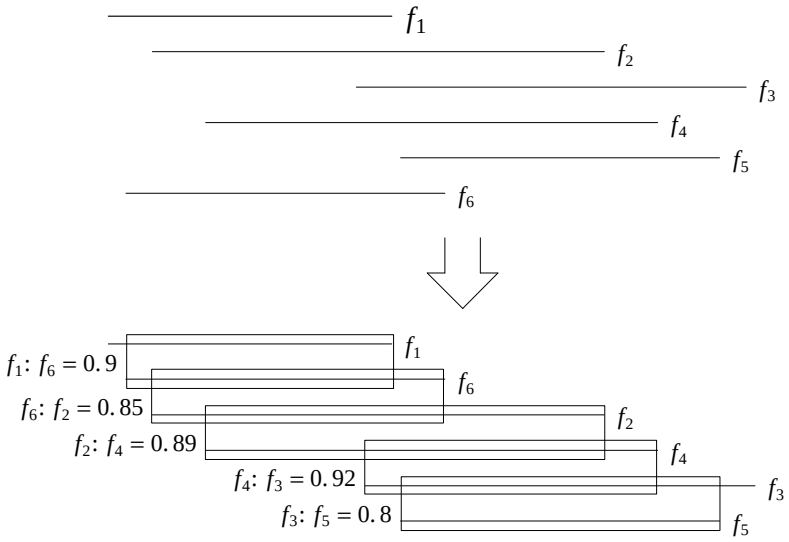

Figure 7: A repeat cluster, its corresponding sorted repeat cluster, and the resulting scored pairwise alignments. Given the pairwise alignment scores obtained and a pairwise alignment threshold of 0.8, AMASS concludes that this particular repeat cluster does not, in fact, correspond to a repeating region but rather to a high-coverage region of the target sequence.

of a given repeat cluster; a simple version of this heuristic is easily introduced by example.[9] Consider again the repeat cluster shown in Figure 7. Assume that the first pairwise alignment performed (between f_1 and f_6) had returned a 0.5 instead of 0.9. Since this value is below the pairwise alignment threshold 0.8, the overlap is rejected and f_6 is set aside. We then align f_1 and the next remaining fragment, here f_2, obtaining a value of 0.9, which exceeds the threshold. Continuing in this fashion, assume we find f_2: $f_4 = 0.85$ (accept), f_4: $f_3 = 0.65$ (reject), and f_4: $f_5 = 0.34$ (reject). We conclude that f_1, f_2, and f_4 are from the same contig. A second pass through the cluster starting with the first discarded fragment (f_6) but considering all fragments (even f_1, f_2, and f_4 which have already been teased out) might yield

[9] A more detailed description of how repeat segments are teased out from a repeat cluster can be found in [Kim97].

f_6: $f_2 = 0.5$ (reject), f_6: $f_4 = 0.4$ (reject), f_6: $f_3 = 0.82$ (accept), and f_3: $f_5 = 0.87$ (accept), producing a second contig containing f_6, f_3, and f_5. We continue teasing out contigs until all fragments are accounted for.

5. The AMASS Sequence Assembly Algorithm

Now that we have all the components, we are ready to describe the operation of the AMASS system, including its repeat handling features:

(1) Select probes and search for probes in fragment data, collecting statistics on probe occurrences, satellite matching failure/success ratio, and maximum number of probes per window in each fragment (see Section 3.1).

(2) Exclude singleton occurrence probes, which must be due to noise or low-coverage regions (see Section 4.1).

(3) Use the probe distribution threshold and the divergence threshold to exclude probes due to short repeating regions (see Section 4.1).

(4) Construct the overlap map (see Section 3.2).

(5) Use the fragment distribution threshold to identify fragments which may arise from relatively longer repeating regions. Construct repeat clusters from these fragments and use the pairwise alignment threshold to determine which repeat clusters correspond to repeating regions (see Section 4.2).

(6) Disable all fragments in repeat clusters, and grow contigs from the remaining set of fragments (see Section 4.2).

(7) Attempt to tease apart different copies of each repeating region from the repeat clusters, producing one contig per copy (see Section 4.2).

(8) Attempt to close remaining gaps by merging contigs from non repeating regions with contigs teased out from repeat clusters.

(9) Generate the consensus sequence (see Section 3.3).

The last unsettled question, then, is precisely how to go about merging contigs in Step 8.

It is tempting to close any remaining gaps aggressively in the hope of delivering as fully complete an assembled sequence as possible (of course, a few gaps will inevitably remain, since shotgun sequencing of any sequence of reasonable length will leave a few regions with zero coverage). On the other hand, reducing the number of gaps in the output is not necessarily advantageous from a final result perspective; such decisions may well be best left to hand editing by biologists in the finishing stage. Thus

the degree to which any assembler pursues gap closing is generally governed by parameter setting, which can be regulated accordingly; AMASS is no exception.

Recall that, at this point, we have a set of contigs, each containing at least one fragment (note that any remaining fragments are simply considered singleton contigs). Some of these contigs correspond to regular contigs, while others may have just been teased out of repeat clusters. Restricting our attention to only the first and last fragments of each contig and their reverse complements, and consider all possible pairwise alignments among these fragments (we use the efficient pairwise alignment algorithm of Section 3.3, but any pairwise alignment algorithm would do). We then decide which contigs to merge using a threshold, called the *gap closing threshold*, based on the ratio of aligned base pairs to overlap base pairs (this is the same metric used when evaluating the alignment of successively ordered fragments in a repeat cluster in Section 4.2). A second threshold, the *minimum overlap thershold*, places a lower bound on the overlap length; fragment overlaps below this threshold are simply not considered.

6. Evaluation

In this section, we describe a series of experiments designed to evaluate the operation a prototype implementation of AMASS.[10] Why an empirical evaluation? We believe that there are many instances where proper and carefully performed experimental evaluations of algorithms on real data sets are more meaningful and more relevant than worst-case (or even average-case) formal arguments. Theoretical arguments are necessarily based on a set of assumptions about the input data, and such assumptions (about, *e.g.*, the character of repeating regions or the distribution and sequencing of bases) are generally not biologically plausible. In short, given the complexity of nature, well-crafted experiments are simply more informative.

[10] All results reported here are collected using a prototype implementation of AMASS, consisting of approximately 14,000 lines of ANSI C code and operating on a 64MB 200MHz Pentium Pro CPU running the RedHat distribution of Linux.

Our experiments fall roughly into two categories: the first set relies on real DNA sequences which have been artificially shotgunned using GENFRAG [Engle93, Engle94], while the second set relies on real shotgun sequence data obtained from multiple sources. The use of real DNA is essential to obtaining representative results. Since one of our claims is that AMASS performs well on real DNA with complex repeating regions, using artificially generated sequences on $\{a, c, g, t\}$ would not provide support for this claim. Of course, with real shotgun data, we can't know where the input fragments are supposed to go in the final result, making checking the correctness of assembly results difficult; hence we rely on easier to control artificially-shotgunned real DNA for at least some of our experiments.

6.1. Artificially-Shotgunned DNA Sequence Data

In this set of experiments, we assemble four different artificially-shotgunned DNA sequences:

(1) *H-7E17*: a 238,939bp human DNA sequence obtained from the Washington University (St. Louis) Genome Sequencing Center.

(2) *TIGR-MJ*: A randomly-selected 500kbp subsequence of the *Methanococcus jannaschii* chromosome obtained from *The Institute for Genomic Research* (TIGR).[11] The selected subsequence spans base positions 5,432 through 505,431 of the original 1,739,896bp.

(3) *TIGR-GHI*: A randomly-selected 500kbp subsequence of the previously cited *Haemophilus influenzae* Rd genome, also obtained from TIGR. The selected subsequence spans base positions 205,335 through 705,334 of the original 1,830,071bp.

(4) *TIGR-GMG*: The entire *Mycoplasma genitalium* genome, consisting of 580,078bp, also obtained from TIGR.

These four sequences are shotgunned using GENFRAG with average fragment length of 500bp, average coverage 8, and average noise rate of 2%, parameters which were chosen to reflect actual shotgun parameters used in practice [Sutton95]. Table 2 gives descriptive statistics for the resulting shotgun data sets (of particular interest are the number of regions with zero or low coverage, where shotgun assembly algorithms are likely to have difficulty).

[11] All sequences obtained from TIGR are courtesy Granger Sutton and are available from http://www.tigr.org.

Table 2
Experimental Shotgun Data Descriptive Statistics

Sequence	Length	Noise rate	Fragments	Zero Coverage	Low Coverage (≤2)
H-7E17	238,939	0.02	3,822	2	54
TIGR-MJ	500,000	0.02	8,001	4	91
TIGR-GHI	500,000	0.02	8,001	4	91
TIGR-GMG	580,078	0.02	9,282	5	129

For these experiments, the AMASS parameters and threshold values are fixed as follows:

probes per fragment	10
probe length	15bp
probe distribution threshold	15
divergence threshold	0.8
fragment distribution threshold	60th percentile
pairwise alignment threshold	0.8
repeat fragment overlap threshold	0.9
gap closing threshold	0.8
minimum overlap threshold	50bp

Note that only overlaps of 50bp or larger are considered for final gap closing.

For each data set, we count the number of contigs produced (both before and after merging contigs), the percentage of the sequence assembled, and the CPU time. We also examine placement of each fragment and compare the location with the actual location of the fragment (one of the unique advantages of artificially-shotgunned data is that we can actually make this comparison). Summary results are shown in Table 3, while Figures 8 through 11 provide schematic detail of each individual assembly.

Table 3
Summary Results

Sequence	Time (sec)	Pre-merging results		Post-merging results	
		Contigs	Percent assembled	Contigs	Percent assembled
H-7E17	142.38	24	96.3	12	96.3
TIGR-MJ	358.40	35	94.1	20	94.7
TIGR-GHI	340.23	28	94.6	14	94.9
TIGR-GMG	446.50	37	90.4	18	90.8

Even without adjusting parameters and thresholds for a particular sequence, the assembly results cover more than 90% of each sequence, and span many of the low coverage regions in the shotgun data (*i.e.*, where coverage is 2 or less). But perhaps most impressive is the fact that every fragment in every assembly, when checked against its original location in the sequence, was found to be correctly placed.

It is interesting to note that — all other parameters being equal — the CPU time required for assembly appears to be linearly related to the length of the target sequence. This linear relationship is critical if the algorithm is to be suitable for long, *e.g.*, genome-length, target sequences. To explore the relationship further, we assembled six nested subsequences of increasing length (from 50kbp to 550kbp in 100kbp increments) taken from the beginning of TIGR-GMG and measured their CPU usage; the results of this simple — albeit somewhat limited — experiment confirm that execution time grows roughly linearly with the length of the target sequence.

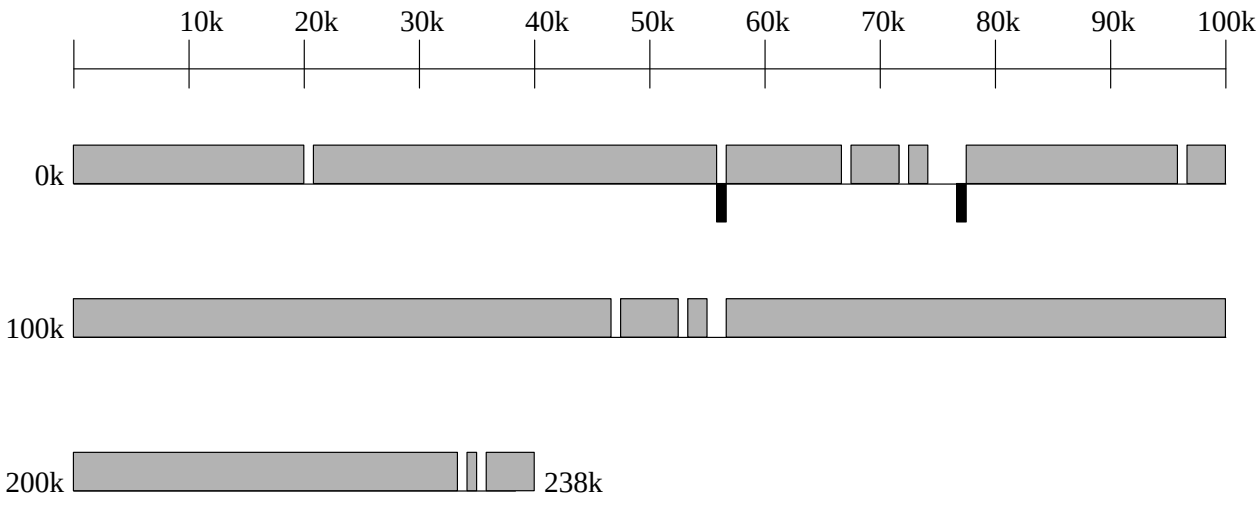

Figure 8: Assembly result for H-7E17 (238,939bp). Gray boxes denote correctly-assembled contigs, while black boxes below the lines denote gaps in the input shotgun data corresponding to regions which cannot be recovered by any assembly algorithm. Every fragment in every contig was found to be correctly placed when compared to the original target sequence.

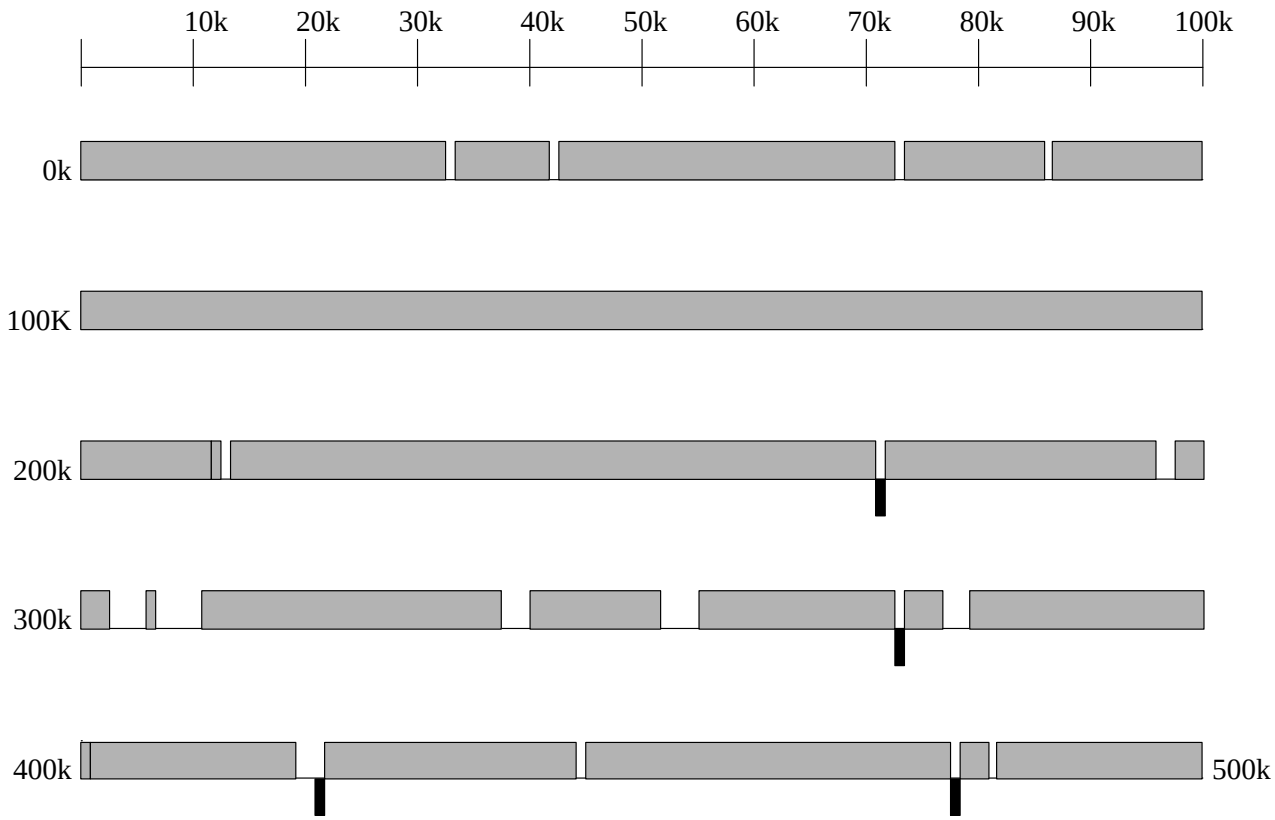

Figure 9: Assembly result for a randomly-selected 500kbp subsequence of TIGR-MJ, the *Methanococcus jannaschii* chromosome. Gray boxes denote correctly-assembled contigs, while black boxes below the lines denote gaps in the input shotgun data corresponding to regions which cannot be recovered by any assembly algorithm. Every fragment in every contig was found to be correctly placed when compared to the original target sequence.

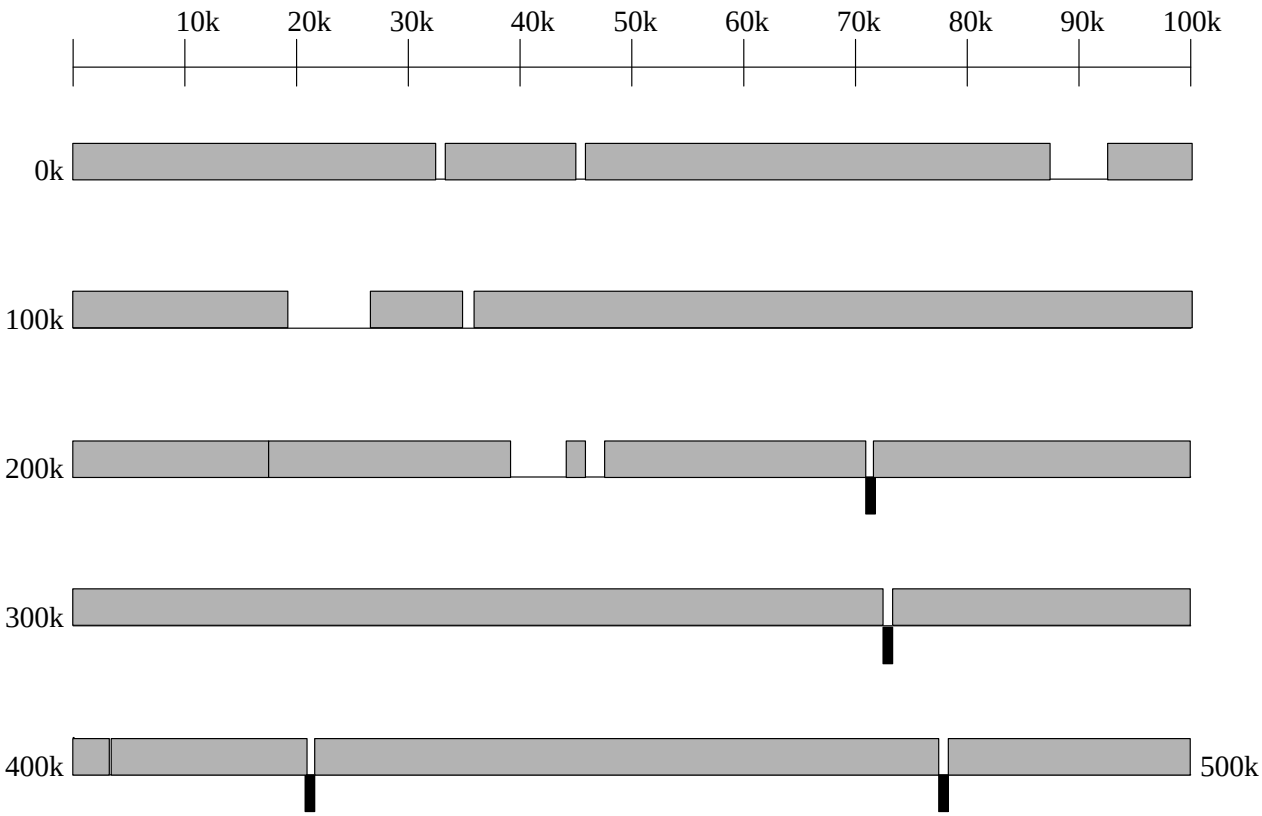

Figure 10: Assembly results for a randomly-selected 500kbp subsequence of TIGR-GHI, the *Haemophilus influenzae* Rd genome. Gray boxes denote correctly-assembled contigs, while black boxes below the lines denote gaps in the input shotgun data corresponding to regions which cannot be recovered by any assembly algorithm. Every fragment in every contig was found to be correctly placed when compared to the original target sequence.

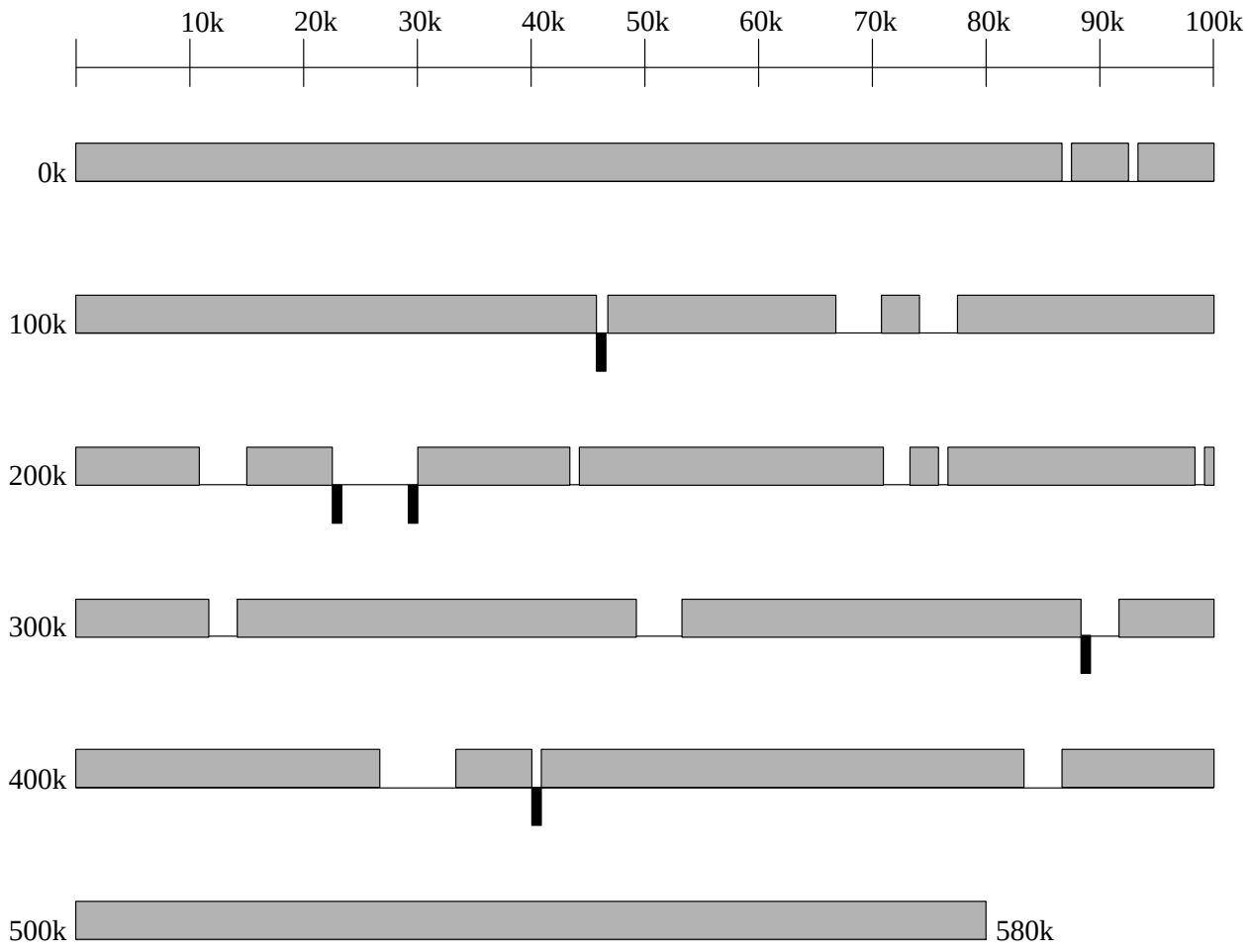

Figure 11: Assembly result for TIGR-GMG, the *Mycoplasma genitalium* genome (580,078bp). Gray boxes denote correctly-assembled contigs, while black boxes below the lines denote gaps in the input shotgun data corresponding to regions which cannot be recovered by any assembly algorithm. Every fragment in every contig was found to be correctly placed when compared to the original target sequence.

6.2. Real DNA Shotgun Sequence Data

Next we turn our attention to four real shotgun sequence data sets, three sets of cosmid data and one whole genome data set. The four data sets used are:

(1) *GT-B2126*: The *Mycobacterium leprae* cosmid B2126, consisting of 44,604bp obtained from *Genome Theraputics*.[12]

(2) *GT-L247*: The *Mycobacterium leprae* cosmid L247, consisting of 44,438bp, also obtained from Genome Theraputics.

(3) *GT-L518*: The *Mycobacterium leprae* cosmid L518, consisting of 41,412bp, also obtained from Genome Theraputics.

(4) *TIGR-GMG*: The entire *Mycoplasma genitalium* genome, consisting of 580,078bp (also used in the artificially-shotgunned experiments of Section 6.1).

For these experiments, we use the same AMASS parameters and threshold values as in Section 6.1, except that here we sample only 5 probes per fragment (as opposed to the 10 probes per fragment used in Section 6.1). Note also that the three cosmid data sets also contain clone length information, which is not used by AMASS, although in theory using such additional information might improve the results obtained.

As previously mentioned, using real shotgun data sets presents an interesting experimental challenge. First, we do not know precisely what the noise rate is for these data sets, since such information is not generally made available. A reasonable guess, based on typical values for most high-throughput sequencing laboratories, is about 2% [Sutton95].

Second, and perhaps more important, it is extraordinarily difficult to evaluate the quality of the assembly result obtained. This is because we cannot be entirely sure that the target sequence provided with real data is actually recoverable from the shotgun data alone. Generally speaking, the published target sequence is produced not only by an assembly algorithm operating on the shotgun data, but also via a number of manual ‘‘finishing’’ operations, each of which may rely on additional information (*e.g.*, clone length constraints) or may even require additional sequencing. Compare this situation with artificially-shotgunned data, which is derived directly from the target sequence in the first place.

---

[12] All sequences obtained from Genome Theraputics are courtesy Hershel Safer and are available from http://www.genomecorp.com.

For these reasons, we will need to be somewhat more creative in checking our assembly results. First, we align the consensus sequences produced from the contigs we assembled with the given target sequence and check for differences between them. This gives us some measure of how much of the target sequence information we are able to recover from the input data (for the reasons just outlined, however, we should not be too alarmed if somewhat less than 100% of the data is recovered). Second, we align every fragment belonging to one of our contigs with the given target sequence and compare the alignment produced with the fragment's placement in our contigs. This gives us a measure of how accurately our algorithm identifies the eventual fragment locations (note that the consensus sequence produced may well be correct even though some fragments in a contig are not correctly aligned, since consensus sequences are based on information from multiple overlapping fragments).

One final note. For all experiments reported in this section, the AMASS parameters were set so as not to be overly aggressive in merging contigs. This is because in practice, such decisions are generally best made by biologists in the finishing stage. Indeed, most assembly algorithms simply produce contigs for finishing, leaving consensus sequence generation to specialized algorithms after manual editing is complete.

6.2.1. *Mycobacterium leprae* Cosmid B2126

The *Mycobacterium leprae* cosmid B2126 shotgun data set consists of 1,504 fragments averaging 232bp. The given target sequence (44,604bp) is known to contain at least one long repeat of 1,550bp, almost seven times longer than the average fragment size [Huang96]. In fact, manual examination of the target sequence shows two copies of a nearly identical repeating region (one ranging from position 30063 to position 31613 and the other ranging from position 31730 to position 33279).

AMASS assembled all 1,504 input fragments into 4 contigs and 1 repeat cluster in 73.10 seconds of 64 MB 200MHz Pentium CPU time. By merging contigs with more than 50bp of overlap that exceed the

0.8 gap closing threshold, AMASS reduces 4 contigs to 2, spanning most (99%) of the nonrepetitive section of the target sequence (see Table 4).

Table 4
Summary Results for *Mycobacterium leprae* Cosmid B2126

Contig	Start	End	Length (bp)	Error rate
C_1	38	29871	29834	0.0045
C_2	33209	44604	11396	0.0028

From the lone repeat cluster, AMASS is able to tease out two additional contigs and bridge the gap between the two remaining contigs. Thus the final sequence produced by AMASS spans all but the first 38bp of the target sequence with a final error rate of 0.0040. Furthermore, every single fragment in the final assembly was aligned with the given target sequence and was found to be positioned correctly by AMASS.

Post-assembly manual examination reveals that AMASS had excluded a single fragment containing the missing 38bp due to an abnormally short overlap. The excluded fragment formed a singleton contig with a 23bp overlap; since our merging criteria parameters required at least a 50bp overlap, the fragment was excluded. Thus more aggressive gap closing parameters would have easily produced the complete sequence (see Section 6.3 for more on parameter settings).

Note that this particular data set contained clone length information which was, in fact, used by the sequence provider to construct the target sequence. The results produced by AMASS do not rely on this additional information; repeats were handled on the basis of the input fragment data alone, notwithstanding the fact that the longest repeat in this sequence was seven times longer than the average fragment length.

6.2.2. *Mycobacterium leprae* Cosmid L247

The *Mycobacterium leprae* cosmid L247 shotgun data set consists of 1,719 fragments averaging 245bp. The given target sequence (44,438bp) is known to contain long reverse repeats of about 640bp, more than twice the average fragment size [Huang96].

AMASS assembled 1,711 of the 1,719 input fragments into 11 contigs and 1 repeat cluster in 63.73 seconds of 64 MB 200MHz Pentium CPU time. By merging contigs with more than 50bp of overlap that exceed the 0.8 gap closing threshold, AMASS reduces 11 contigs to 4, spanning roughly 88% of the target sequence (see Table 5).

Table 5
Summary Results for *Mycobacterium leprae* Cosmid L247

Contig	Start	End	Length (bp)	Error rate
C_1	1	3398	3398	0.0047
C_2	6549	8036	1488	0.0047
C_3	9680	35641	25962	0.0042
C_4	36212	44436	8225	0.0047

From the lone repeat cluster (205 fragments), AMASS is able to tease out two reverse copies of a 640bp repeat; these 2 contigs are used to bridge $C_1{:}C_2$ and $C_2{:}C_3$, leaving $C_3{:}C_4$ (571bp) as the only unremediated gap. The final error rate is 0.0044; every single fragment in the final assembly was aligned with the given target sequence and was found to be positioned correctly by AMASS. As before, we did not exploit the clone length information provided with the input fragment data to obtain these results.

6.2.3. *Mycobacterium leprae* Cosmid L518

The *Mycobacterium leprae* cosmid L518 shotgun data set consists of 1,545 fragments averaging 227bp. The given target sequence (41,412 bp) is known to contain long repeats of about 1387bp, almost seven times the average fragment size [Huang96].

AMASS assembled 1,439 of the 1,545 input fragments into 16 contigs and 2 repeat clusters in 52.31 seconds of 64 MB 200MHz Pentium CPU time. By merging contigs with more than 50bp of overlap that exceed the 0.8 gap closing threshold, AMASS reduces 16 contigs to 13, spanning roughly 83% of the target sequence (see Table 6).

Table 6
Summary Results for *Mycobacterium leprae* Cosmid L518

Contig	Start	End	Length (bp)	Error rate
C_1	1	612	612	0.0016
C_2	620	1537	918	0.0022
C_3	1641	2530	890	0.0034
C_4	2511	3307	797	0.0025
C_5	7258	8720	1463	0.0027
C_6	11449	15894	4446	0.0036
C_7	15905	16297	393	0.0051
C_8	16312	19391	3080	0.0032
C_9	19399	22262	2864	0.0038
C_{10}	22241	23239	999	0.0040
C_{11}	23300	35448	12149	0.0043
C_{12}	35493	40114	4622	0.0037
C_{13}	40112	41411	1300	0.0023

The $C_4{:}C_5$ and $C_5{:}C_6$ gaps are due to a 1,387bp repeat. From the first repeat cluster (254 fragments), AMASS is able to tease out 2 contigs to bridge these gaps. However, for this assembly, the final result still has numerous gaps. Manual inspection reveals that many of the remaining gaps can not be closed due to the less-than-aggressive parameters supplied to AMASS; several of the gaps require small fragments (less than 50bp) to bridge over them; such small overlaps are currently ignored. The final error rate is 0.0037: every single fragment in the final assembly was aligned with the given target sequence and was found to be positioned correctly by AMASS. As before, we did not exploit the clone length information provided with the input fragment data to obtain these results.

6.2.4. *Mycoplasma genitalium* Genome

The *Mycoplasma genitalium* genome shotgun data set consists of 8,921 fragments averaging 575bp. The given target sequence (580,078 bp) contains several repeating regions of various lengths.

AMASS assembled 8,263 of the 8,921 input fragments into 54 contigs and 7 repeat clusters in 338.26 seconds of 64 MB 200MHz Pentium CPU time. By merging contigs with more than 50bp of overlap that exceed the 0.8 gap closing threshold, AMASS reduces 54 contigs to 16, spanning roughly 94% of the target sequence (see Table 7).

Table 7
Summary Results for *Mycoplasma genitalium* Genome

Contig	Start	End	Length (bp)	Error rate
C_1	1	21116	21116	0.0018
C_2	25399	166581	141183	0.0018
C_3	175302	201679	26378	0.0017
C_4	209058	212777	3720	0.0019
C_5	216114	224323	8210	0.0016
C_6	224515	226445	1931	0.0021
C_7	227049	291365	64317	0.0017
C_8	291921	350693	58773	0.0018
C_9	351063	389379	38317	0.0017
C_{10}	390621	419714	29094	0.0018
C_{11}	419700	428854	9155	0.0016
C_{12}	429942	467228	37287	0.0019
C_{13}	468663	471709	3047	0.0016
C_{14}	474688	522265	47578	0.0018
C_{15}	522290	554893	32604	0.0018
C_{16}	557241	580078	22838	0.0017

The final error rate is 0.0018: every single fragment in the final assembly was aligned with the given target sequence and was found to be positioned correctly by AMASS. As before, we did not exploit the clone length information provided with the input fragment data to obtain these results.

6.3. How Good is "Good Enough"?

It seems opportune to step away from the details of the experiments just provided and ask just how good these results really are. The stated goal of the NIH/DOE Human Genome Project is to produce finished sequence data having an error rate of 1 in 5,000bp, or roughly 0.0002. This desired error rate is for manually polished, post-finishing sequence; not the same thing as the prefinishing error rates we report here for AMASS. Obviously, prefinishing error rates will always be higher, since it is in the finishing stage that remaining gaps are closed and base calling errors are resolved. While no assembler's output should be held to the same standards as finished sequence, the prefinishing error rate we report for the *Mycoplasma genitalium* genome (0.0018) is right at 1 in 568bp, a difference easily spanned in the finishing stage.

Another tempting mistake is to assume that it is always preferable to generate output containing fewer gaps. Assemblers in general produce output containing numerous gaps in the sequence, which are then usually closed by hand editing in the finishing stage. While it is often possible to produce output with fewer gaps (*e.g.*, by adjusting parameters of the assembler), assemblers don't do so because of the natural tension between fewer gaps and increased errors. To see how this is the case, consider most assemblers have a parameter that defines the minimum length (in bases) that is allowed to define an overlap between two fragments (both AMASS and the TIGR Assembler have this kind of parameter). Recall that in our assembly of *Mycobacterium leprae* cosmid B2126 (Section 6.2.1), the first 38bp of the sequence were missed in the final assembly because of just such a short overlap (23bp). In this case, a shorter minimum overlap would have produced a more complete assembly, but with a higher risk of being misled by short repeating regions. In practice, the proper behavior would appear to be to avoid overlaps that are too short and accept a concomitantly larger number of gaps; gaps which are then closed in the finishing stage.

How many gaps should one expect to see in assembler output? While only post-finishing statistics are usually published, one account cites a typical prefinishing output gap rate of 1 in 7,500bp for data with

relatively high coverage 10 using current technology [Green97]. For the original *Haemophilus influenzae* Rd genome assembly, 210 contigs were produced by the TIGR Assembler, a rate of roughly 1 gap every 8,750bp (perhaps, at least in part, due to a lower coverage value of 8 for this data) [Fleischmann95]. In comparison, AMASS usually leaves a far smaller number of gaps; for the artificially-shotgunned fragment sets of Section 6.1, the AMASS output gap rates range between 1 in 21,000bp and 1 in 38,000bp, only about a fourth of the gaps one might expect (again, note these lower gap rates are achieved for data with coverage of only 8).

While it is difficult to estimate gap rates for the real shotgun data sets of Section 6.2 because we don't know how many gaps exist in the data sets themselves, some meaningful head-to-head comparisons can still be made. Consider our assembly of the *Mycoplasma genitalium* genome in Section 6.2.4; since we obtained the data set from TIGR, we assume that this is (at least roughly) the same data set used for the original assembly [Fraser95]. For the original assembly, a total of 39 contigs were reported (11 of these gaps were subsequently closed by detecting missing overlaps during finishing); compare with only 15 gaps left by AMASS.

Similar back-of-the-envelope comparisons can be made for speed of assembly. The *Haemophilus influenza* Rd genome (1,830,071bp) assembly required 30 hours of 512MB SPARCenter 2000 CPU time. Since we don't have access to processors with 512MB, we elected to assemble an artificially-shotgunned version of about one third of this sequence (8,001 fragments corresponding to 500kbp); using fragment length and noise rates comparable to those for the original assembly, AMASS produced 16 contigs in only less than 6 minutes of 64MB 200MHz Pentium Pro CPU time. Thus, all other things being equal, the TIGR Assembler is assembling data at about 1,000bp/minute, leaving just over 0.1 gaps/1,000bp. In comparison, AMASS is assembling data at over 83,000 bp/minute and leaving about 0.03 gaps/1,000bp. This, of course, assumes that our 3-year-old 64MB Pentium Pro is as fast or faster than the 3-to-4-year-old SPARCenter 2000 used by TIGR (actually, we expect that the PC is significantly slower, which makes

AMASS look even better), and that AMASS performance scales roughly linearly with sequence length (a claim supported experimentally in Section 6.1).

Having said all this, there are several things AMASS might well do better. For example, the consensus sequence generation algorithm used by the current prototype is simply not as accurate as that used by some other assembly systems, such as, for example, PHRAP [Green96]. PHRAP uses a different heuristic for consensus sequence generation that essentially walks the highest quality fragment, exploiting certainty factors that are provided for each nucleotide by the sequence calling software. Of course, one might argue that the difficult part of the problem is correctly placing the fragments from which a consensus sequence can be derived using any available consensus sequence generation algorithm; in fact, one might contemplate integrating PHRAP's consensus sequence generation algorithm with the AMASS overlap map construction and repeat handling mechanism in order to get the best of both worlds.

7. Conclusion

In this paper, we introduced a new sequence assembly algorithm that exploits the relatively low noise rate and high coverage typically found in shotgun sequencing data. The distinguishing features of our algorithm are as follows:

(1) Our algorithm uses exact pattern matching as opposed to approximate pattern matching in determining both the relative positions of the input fragments and the final consensus sequence. The use of an efficient multipattern search algorithm results in reliable sequence determination that is very fast in practice, yet robust to the levels of insertion, deletion, and substitution error that are common in practice.

(2) The representation of a fragment as an ordered set of probes and associated interprobe distances is quite compact, yielding significant performance benefits over approaches that rely on explicit sequence representation.

(3) Our algorithm does not solely rely on scoring pairwise fragment alignments; instead, each fragment is dynamically scored against each contig, which represents a set of fragments. Comparing a fragment to a contig exploits the multiple coverage characteristic of shotgun sequencing data and allows AMASS to construct contigs without committing to a pairwise consensus sequence until all of that contig's constituent fragments are found. In this fashion, production of a consensus sequence is delayed, allowing the system to exploit the full coverage of any given segment of the target sequence.

(4) Empirically, the algorithm is highly reliable and accurate, and demonstrates approximately linear execution time in the length of the target sequence.

(5) The algorithm easily handles noise rates typical of modern sequencing operations, and has also been tested successfully with much higher noise rates as well (see [Kim97] for experiments with shotgun data sets with higher noise rates). Moreover, as noise decreases, due, for example, to improvements in sequencing equipment or laboratory procedures, execution time also decreases, as lower noise rates exhibit longer regions of exact matches.

Of course, more work remains to be done to further refine AMASS: specifically, we wish to assimilate additional features of other state-of-the-art assembly algorithms such as the use of clone length constraints [Sutton95], or incorporate the use of base calling quality information, especially in the consensus sequence generation step [Green96]. However, even without these additional refinements, AMASS clearly demonstrates its very practical advantages. In summary, the primary contribution of AMASS is to achieve exceptionally fast sequence assembly of DNA sequences while providing an intuitive, robust repeat handling procedure capable of handling complex repeats such as those exhibited in human DNA: our claims regarding the accuracy, robustness, and execution efficiency of our system are supported with empirical evidence on real sequence data.

Acknowledgements

The authors wish to thank Jeffrey Murray (The University of Iowa Department of Pediatrics), Michael Kozal (The University of Iowa Department of Internal Medicine), and Harris Lewin (The University of Illinois Biotechnology Center) for their help and advice during the course of this research. In addition, we would like to acknowledge the contributions of both anonymous reviewers and the editor, Michael Waterman; thanks to their efforts, a considerably more cogent paper has emerged. Support for this research was provided by the Office of Naval Research under grant N0014-94-1-1178, the National Science Foundation under grants CDA-9529518 and IRIS-9729807, the University of Illinois Critical Research Initiatives, and the University of Iowa Department of Sponsored Programs.

References

[Agarwal94]
P. Agarwal and D.J. States, “The Repeat Pattern Toolkit (RPT): Analyzing the Structure of Evolution of the *C. elegans* Genome,” *Proceedings of the 2nd International Conference on Intelligent Systems for Molecular Biology*, AAAI Press (August 1994), pp. 1-9.

[Bonfield95]
J. K. Bonfield, K. Smith, and R. Staden, “A New DNA Sequence Assembly Program,” *Nucleic Acids Research* **23**:24 (December 25, 1995), pp. 4992-4999.

[Engle93]
M.L. Engle and C. Burks, “Artificially Generated Data Sets for Testing DNA Fragment Assembly

Algorithms," *Genomics* **16** (1993), pp. 286-288.

[Engle94]
M.L. Engle and C. Burks, "GenFrag 2.1: New Features for More Robust Fragment Assembly Benchmarks," *Genomics* **10** (1994), pp. 567-568.

[Fleischmann95]
R.D. Fleischmann, M.D. Adams, O. White, R.A. Clayton, E.F. Kirkness, A.R. Kerlavage, C.J. Bult, J.F. Tomb, B.A. Dougherty, J.M. Merrick, K. McKenney, G. Sutton, W. Fitzhugh, C. Fields, J.D. Gocayne, J. Scott, R. Shirley, L.I. Liu, A. Glodek, J.M. Kelley, J.F. Weidman, C.A. Phillips, T. Spriggs, E. Hedblom, M.D. Cotton, T.R. Utterback, M.C. Hanna, D.T. Nguyen, D.M. Saudek, R.C. Brandon, L.D. Fine, J.L. Fritchman, J.L. Fuhrmann, N.S.M. Geoghagen, C.L. Gnehm, L.A. McDonald, K.V. Small, C.M. Fraser, H.O. Smith, and J.C. Venter, "Whole-Genome Random Sequencing and Assembly of *Haemophilus influenzae* Rd," *Science* **269** (July 28, 1995), pp. 496-512.

[Foulser90]
D. Foulser, "A Linear Time Algorithm for DNA Sequencing," Technical Report 812, Department of Computer Science, Yale University, New Haven, CT (July 1990).

[Fraser95]
C.M. Fraser, J.D. Gocayne, O. White, M.D. Adams, R.A. Clayton, R.D. Fleischmann, C.J. Bult, A.R. Kerlavage, G. Sutton, J.M. Kelley, J.L. Fritchman, J.F. Weidman, K.V. Small, M. Sandusky, J. Fuhrmann, D. Nguyen, T.R. Utterback, D.M. Saudek, C.A. Phillips, J.M. Merrick, J.F. Tomb, B.A. Dougherty, K.F. Botth, P.C. Hu, T.S. Lucier, S.N. Peterson, H.O. Smith, C.A. Hutchison, and J.C. Venter, "The Minimal Gene Complement of *Mycoplasma Genitalium*," *Science* **270** (October 20, 1995), pp. 397-403.

[Gingeras79]
T. Gingeras, J. Milazzo, D. Sciaky, and R. Roberts, "Computer Program for the Assembly of DNA Sequences," *Nucleic Acids Research* **7**:2 (September 25, 1979), pp. 529-545.

[Green96]
P. Green, "PHRAP Documentation," http://bozeman.mbt.washington.edu/phrap.docs/phrap.html, University of Washington, Seattle, WA (1996).

[Green97]
P. Green, "Against a Whole-Genome Shotgun," *Genome Research* **7** (1997), pp. 410-417.

[Huang92]
X. Huang, "A Contig Assembly Program Based on Sensitive Detection of Fragment Overlaps," *Genomics* **14** (1992), pp. 18-25.

[Huang96]
X. Huang, "An Improved Sequence Assembly Program," *Genomics* **33** (1996), pp. 21-31.

[Idury95]
R.M. Idury and M.S. Waterman, "A New Algorithm for DNA Sequence Assembly," *Journal of Computational Biology* **2**:2 (Summer 1995), pp. 291-306.

[Karp98]
R.M. Karp, "Sequencing the Genome: A New Application Domain for the Mathematics of Operations Research," Omega Rho Distinguished Lecture, Fall Meeting of the Institute for Operations Research and Management Sciences, Seattle, WA (October 1998).

[Kececioglu95]
J.D. Kececioglu and E.W. Myers, "Combinatorial Algorithms for DNA Sequence Assembly," *Algorithmica* **13** (1995), pp. 7-51.

[Kim97]
S. Kim, "A Structured Pattern Matching Approach to Shotgun Sequence Assembly," Ph.D. Thesis, Department of Computer Science, The University of Iowa, Iowa City, IA (August 1997).

[Parsons95]
R.J. Parsons, S. Forrest, and C. Burks, "Genetic Algorithms, Operators, and DNA Fragment Assembly," *Machine Learning* **21**:1-2, Kluwer Academic (October/November 1995), pp. 11-34.

[Peltola84]
H. Peltola, H. Soderlund, and E. Ukonnen, "SEQAID: A DNA Sequence Assembly Program Based on a Mathematical Model," *Nucleic Acids Research* **12**:1 (January 11, 1984), pp. 307-321.

[Smith81]
T. Smith and M. Waterman, "Identification of Common Molecular Subsequences," *Journal of Molecular Biology* **147** (1981), pp. 195-197.

[Staden80]
R. Staden, "A New Computer Method for the Storage and Manipulation of DNA Gel Reading Data," *Nucleic Acids Research* **8**:16 (August 25, 1980), pp. 3673-3694.

[Sutton95]
G. Sutton, O. White, M. Adams, and A. Kerlavage, "TIGR Assembler: A New Tool for Assembling Large Shotgun Sequencing Projects," *Genome Science and Technology* **1**:1 (1995), pp. 9-19.

[Venter98]
J.C. Venter, M.D. Adams, G. Sutton, A.R. Kerlavage, H.O. Smith, and M. Hunkapillar, "Shotgun Sequencing of the Human Genome," *Science* **280** (June 5, 1998), pp. 1540-1542.

[Waterman95]
M. Waterman, *An Introduction to Computational Biology*, Chapman & Hall, London, UK (1995).