Appears in *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, 3-14. St. Louis, MO: AAAI Press. 1996.

# Improving the Quality of Automatic DNA Sequence Assembly using Fluorescent Trace-Data Classifications

**Carolyn F. Allex**[1,2]
*allex@cs.wisc.edu*

**Schuyler F. Baldwin**[2]
*schuy@dnastar.com*

**Jude W. Shavlik**[1]
*shavlik@cs.wisc.edu*

**Frederick R. Blattner**[2,3]
*fred@genetics.wisc.edu*

[1]Computer Sciences Department, University of Wisconsin – Madison,
1210 West Dayton St., Madison, WI 53706, Tel: (608) 262-1204, FAX: (608) 262-9777

[2]DNAStar Inc., 1228 South Park St., Madison, WI 53715,
Tel: (608) 258-7420, FAX: (608) 258-7439

[3]Genetics Department, University of Wisconsin – Madison, 445 Henry Mall, Madison, WI 53706,
Tel: (608) 262-2534, FAX: (608) 262-2976

**Abstract**

Virtually all large-scale sequencing projects use automatic sequence-assembly programs to aid in the determination of DNA sequences. The computer-generated assemblies require substantial hand-editing to transform them into submissions for GenBank. As the size of sequencing projects increases, it becomes essential to improve the quality of the automated assemblies so that this time-consuming hand-editing may be reduced. Current ABI sequencing technology uses base calls made from fluorescently-labeled DNA fragments run on gels. We present a new representation for the fluorescent trace data associated with individual base calls. This representation can be used before, during, and after fragment assembly to improve the quality of assemblies. We demonstrate one such use – end-trimming of sub-optimal data – that results in a significant improvement in the quality of subsequent assemblies.

## Introduction

A fundamental goal of the Human Genome Project is to determine the sequence of bases in DNA molecules. Since the late 1970's, researchers have been making progress in sequencing human DNA as well as that of several model organisms (Maxam & Gilbert 1977, Sanger 1977). Their methods have evolved from painstaking manual generation and analysis of data to the incorporation of automated and computerized techniques (Ansorge et al. 1986, Smith et al. 1986, Connell et al. 1987, Prober et al. 1987, Dear & Staden 1991, Hunkapiller et al. 1991, Myers 1994).

The key to the use of computers for analysis is that DNA is most naturally represented as a discrete sequence of bases. The sequence of bases can be thought of as a string over an alphabet of four symbols: *A*, *G*, *C*, and *T*. Algorithms for matching and aligning strings have been well-studied in computer science and can be applied to problems in DNA sequencing (Waterman 1989, Kruskal 1983). One critical application involves the alignment of overlapping sequences of bases among DNA fragments; this process is called *sequence assembly*. The sequences of bases used for assembly are determined by an examination of the fluorescent-dye intensity signal, called *trace data*, that is output by automatic sequencers.

We present a more descriptive representation of the trace data that is output by Applied Biosystems Inc. (ABI) automatic sequencers. The output representation of ABI trace data is a sequence of discrete fluorescent-dye intensities. Although the information contained in this data has enormous potential for use in sequence assembly, the representation that is output by sequencers makes the direct use of the data for automatic assembly almost impossible. We believe that our new representation makes trace-data information directly accessible for automatic DNA sequence-assembly programs. To substantiate this belief, we present a case study in which our representation is used in an application that trims sub-optimal data from sequences before assembly. Empirical results show that the inclusion of the trace-data information improves the quality of subsequent assemblies.

The following section of this paper presents a brief background of DNA sequencing and assembly for those readers who are unfamiliar. Next, our new representation for trace data is detailed. This is followed by a presentation of our case study. Finally, ideas for future work and conclusions complete the paper.

## Background

In brief, the sequencing procedure consists of selecting a large segment of DNA, producing overlapping fragments of this segment, sequencing each fragment, and finally

aligning the overlapping areas of the fragments to determine the overall sequence of the original segment of interest. With the ABI 377 sequencer, the large segments of DNA may be as long as several hundred kilobases (kb), and the fragments that can be sequenced are less than one kb long. Our work involves the sequencing and assembly of individual fragments, so these aspects of the procedure will be described in more detail.

## Sequencing Fragments

The basic idea is that for each fragment, we need to produce a set of complementary sub-fragments. The set is complementary since it is generated through replication using polymerase and a primer. At each replication step deoxynucleotides (*A*, *G*, *C*, and *T*) and dideoxynucleotides (*A**, *G**, *C**, and *T**) compete for addition to the growing sequence. Deoxynucleotides permit elongation whereas dideoxynucleotides terminate replication (Prober et al. 1987). The result is a set of sub-fragments that encompasses all possible lengths (except those of the initial primer).

```
Fragment:
   CTTGCTACCCTTCGGATTA
   + primer (GAACG) + polymerase
   + A + G + C + T
   + A* + G* + C* + T*
   =
Complementary sub-fragments:  GAACGA*
   GAACGAT*
   GAACGATG*
   GAACGATGG*
   GAACGATGGG*
   GAACGATGGGA*
   GAACGATGGGAA*
   GAACGATGGGAAG*
   GAACGATGGGAAGC*
   GAACGATGGGAAGCC*
   GAACGATGGGAAGCCT*
   GAACGATGGGAAGCCTA*
   GAACGATGGGAAGCCTAA*
   GAACGATGGGAAGCCTAAT*
```

**Figure 1. The sequence of a fragment of DNA and the corresponding set of complementary sub-fragments for sequencing.** Quantities of primer, polymerase, deoxynucleotides, and dye-labeled dideoxynucleotide terminators are added to copies of the fragment to produce the set of complementary sub-fragments. The asterisks designate fluorescently-labeled dideoxynucleotide terminators.

Each dideoxynucleotide at the end of a sub-fragment is labeled with a fluorescent dye. Since a different dye labels each of the the four bases, all sub-fragments of a given length are labeled with the same dye. (Other methods of labeling also exist, but will not be described in this paper.) Figure 1 shows a fragment and its corresponding set of sub-fragments.

The set of labeled sub-fragments is placed on a plate of polyacrylamide gel and an electric current is applied. The current causes the migration of sub-fragments through the gel. Since smaller pieces of DNA migrate more quickly than larger ones, the sub-fragments become separated by size. The fluorescent labeling then provides the means for determination of the fragment sequence (Ansorge et al. 1986, Smith et al. 1986).

The ABI sequencer reads the intensity trace of each of the four fluorescent dyes as the sub-fragments migrate past. This process is called *reading the trace*, and the data produced is called *trace data*. There is one set of trace data for each of the four fluorescent dyes. Although each set of trace data is composed of discrete measurements, the points can be interpolated to form a continuous curve.

## Base Calling

The sets of trace data are used to determine the sequence of bases in the fragment; this is referred to as *base calling*. The four sets of trace data are kept synchronized as they are scanned during base calling. The sequencer expects to call a base at fairly regular intervals and calls exactly one base for each of these intervals of trace data (Perkin Elmer 1995). There are usually about ten trace-data points per interval, and a record is kept of the points at which the calls are made.

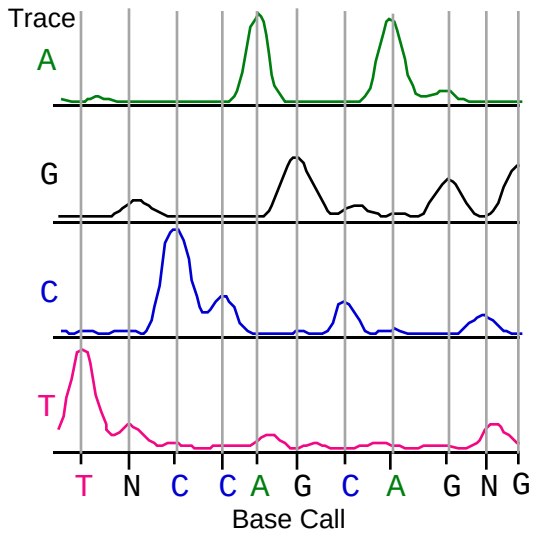



**Figure 2. Sequence base calls and corresponding sets of trace data.** The sequencer calls the base with the highest trace value unless two or more values are similar (in which case it calls an *N*). Gray lines indicate where base calls are made.

The sequencer calls the bases in order as it scans the trace data. Calls are made by examining the values of the trace data. Ideally, the trace values for one base are substantially higher than those for the other three. In this case, the base corresponding to that trace is the one that is called. Sometimes the trace values for two or more possible bases are similar. In this case, the sequencer makes a *no-call* and labels the base with an *N*. The goal is to obtain the exact sequence of bases that is the complement of the fragment. In practice, the accuracy of the base calls made by modern sequencers is 98-99% (Chen 1994, Kelley 1994). A sequence of base calls and corresponding trace data is depicted in Figure 2.

## Sequence Assembly

When all the fragments of the original DNA segment of interest have been sequenced, we proceed to assembling the fragments into larger segments (McCombie & Martin-Gallardo 1994, Myers 1994, Rowen & Koop 1994). The fragments overlap, so we can produce this assembly by aligning the overlapping regions of the sequences. A computer assembly program uses an approximate string-alignment algorithm to find the optimal alignment of the sequences of base calls (Needleman & Wunsch 1970, Martinez 1983). A consensus of the base calls is computed; this forms a contiguous sequence of DNA that is known as a *contig* (Staden 1980). Figure 3 illustrates this idea.

```
Sequence 1:   CCCGGGGCAATT
Sequence 2:      GGGGCAATTAGCCCTTC
Sequence 3:           AATTAGCCCTTCCCACG

Consensus:    CCCGGGGCAATTAGCCCTTCCCACG
```

**Figure 3. Three overlapping fragments aligned to determine the sequence of a larger segment of DNA.** The base sequence of this segment is the consensus of the aligned fragments.

When we assemble sequences that are not entirely correct, we get base locations where sequences align but do not agree completely (McCombie & Martin-Gallardo 1994). A consensus base call in these cases is assigned one of 12 *ambiguity* codes as listed in Figure 4. (An *ambiguity* is any call that is not *A*, *G*, *C*, or *T*.) Figure 5 portrays a multiple sequence alignment containing some ambiguities in the sequence consensus.

In an ideal assembly where the data is flawless and available, the sequences align to form one contig and each consensus base call is *A*, *G*, *C*, or *T*. In fact, this is rarely the case. Difficulties inherent in the preparation and sequencing of fragments lead to incorrect base calls. Also, the quality of the trace data becomes progressively worse near the end of the fragment. Many more incorrect calls and no-calls are in this region (Kelley 1994, Perkin Elmer 1995).

| Base | Code | Base | Code |
|---|---|---|---|
| A or G | R | not A | B |
| A or T | W | not G | H |
| A or C | M | not C | D |
| G or T | K | not T | V |
| G or C | S | no-call | N, X |
| C or T | Y | | |

**Figure 4. Base ambiguity codes.**

After assembly, the ambiguities must be resolved and a single contig formed before a sequence is complete and ready to be submitted to GenBank. This is a time-consuming task performed by human sequence-editors that entails visual analysis of the assembly and data (Rowen & Koop 1994).

```
Sequence 1:  TGTGCGCGGATCCCCTATTTGTTTA
Sequence 2:  TGTCGGCGGAACCCCTATTTGTTTA
Sequence 3:  TGTGCGCGGAACCCCTATTTTTTTA

Consensus:   TGTSSGCGGAWCCCCTATTTKTTTA
```

**Figure 5. Three sequences aligned that are not in perfect agreement.** The four ambiguous base calls in the consensus sequence are underlined.

## Trace-Data Representation

Currently, commercial assembly programs such as DNAStar Inc. *Seqman*, Gene Codes Corp. *Sequencher*, and Genetics Computer Group *Fragment Assembly System*, use only the sequence of base calls, and no trace-data information, in automatic assembly processes (Schroeder 1996, Rosenberg 1996, Edelman 1996). *Seqman* and *Sequencher* do provide a 2-D graph representation of trace data for users, but only for visual examination. Human editors make extensive use of these graphs after assembly to assist in resolving ambiguous calls, fine-tuning alignments, and merging contigs (Rowen & Koop 1994). As the size of sequencing projects continually grows, it becomes increasingly important to reduce these kinds of costly manual operations (McCombie & Martin-Gallardo 1994, Rowen & Koop 1994).

We claim that the need for manual processes can be reduced by allowing the explicit inclusion of trace-data information into the automatic assembly process. Since the existing represention of ABI trace data as four discrete sequences of fluorescent-dye intensities is difficult to incorporate, we have developed an algorithm that

transforms the trace data into a visually-descriptive representation that is usable in assembly programs.

The trace-data output from an ABI DNA sequencer is found in the data files of the ABI *Analysis* program. There are four sets of data for a fragment of DNA – one for each of the four fluorescent dyes. The trace data appears in two forms; one is a sequence of raw intensities, and in the other, the data has been processed such that trace peaks are more distinct and uniform. It is the processed data that is used to produce the graphs made available to users of *Seqman* and *Sequencher.* While studying the graphs, sequence editors pay particular attention to the relative intensities and characteristic shapes of trace data. It is a measure of these shapes and relative intensities found in the graphs of processed data that we describe in our representation. By capturing this information, we can make available to an assembly program the same information that is available to editors.

For our new representation, we are interested in classifying the shape and intensity of the local trace-data that is used for each particular base call. We define this local trace-data to be the data from midway between the previous call and the current call to the data midway between the current call and the next call (Figure 6). We will refer to each of these intervals of data as *base trace-data*. Each set of base trace-data is composed of about ten to 15 data points representing the intensities of the fluorescent dyes. In the following sections, the classification of the trace data always refers to a single set of trace data (the *A*, *G*, *C*, or *T* fluorescent-dye trace-data) for a base.

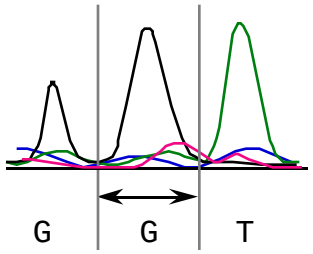



**Figure 6. Base trace-data.** Base trace-data is the trace data associated with a base that extends from midway between the previous call and the current call to midway between the current call and the next call.

## Overview

The trace-data representation we define is composed of classes of shapes, each of which is assigned a score from 0 to 100. Two broad categories are defined that are each then divided into three classes. The two broad categories of base trace-data shapes are *peaks* and *valleys.* Data that curves down is categorized as a peak and data that curves up is categorized as a valley. As illustrated in Figure 7, some peaks or valleys are very sharp and pronounced, others contain a *shoulder*, and others are merely a smooth curve sloping in one direction.

**Figure 7. A variety of shapes occur in trace data.** Trace data takes on a number of characteristic shapes. Some are sharp curves, some contain a shoulder, and others appear as long smooth curves.

Within the peak and valley categories, the data is divided into three classes: *strong*, *medium*, and *weak*. Curves assigned to strong classes are characterized by sharp peaks or dips, those assigned as medium peaks and valleys are characterized by the occurrence of a shoulder in their curves, and curves in weak classes are smooth and slope in only one direction. Stereotypical class shapes are sketched in Figure 8.

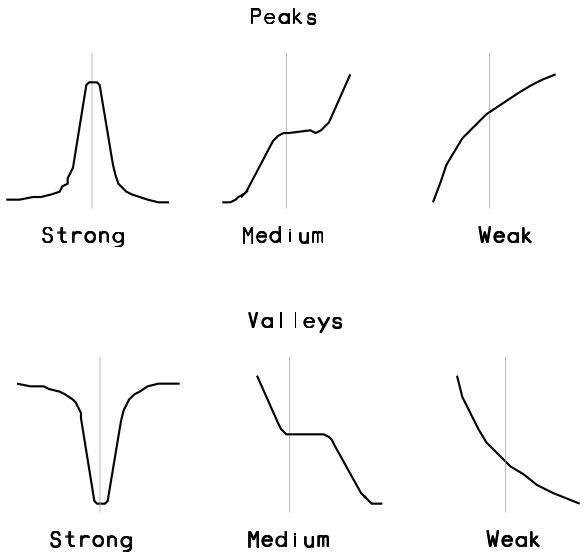



**Figure 8. Stereotypical shapes of class curves.** Gray lines indicate base call locations.

Often, the distinction among the strong, medium, and weak classes is not clear. In these cases, data is assigned a weighted combination of class scores. Each peak and valley is assigned a score that reflects the amount of strong, medium, and weak character that is exhibited.

The trace data associated with a single base may contain a peak, or a valley, or both a peak and a valley. The base is

called at a particular point in the trace data – we assign scores for both the peak and the valley that are the closest to this location. These class scores are weighted by proximity to the base-call location. Peaks or valleys that are closer to where the base is called have a relatively higher score than those that are further away.

Sometimes we may need to make comparisons among the four sets of trace data associated with a single base call. For this situation, the classification scores are adjusted to reflect the relative difference in intensities (heights) of the peaks or valleys; higher peaks score higher than lower peaks, and lower valleys score higher than higher valleys.

## Algorithmic Details

The data is scanned for strong peaks and valleys, then for medium peaks and valleys, and finally, if neither of these is found, a weak peak or valley is assumed. At each step, we look for the peak and valley that are the closest to the point where the base was called. Scores are assigned based on proximity to the base-call location and on the amount of strong, medium, and weak character exhibited.

We first examine the data for strong peaks or valleys. A strong peak is detected when there is a change from a negative to a positive slope, and likewise, a strong valley is detected when there is a change from a positive to a negative slope. The slopes are measured as the change in intensity from one data point to the next. If a strong peak or valley is found, it must be checked for amount of strong and medium character. Peaks that start at the baseline (zero intensity) and return to the baseline are scored as 100% strong and 0% medium. The same is true for valleys that start at the maximum intensity and return to the maximum intensity. Any other peaks or valleys found in this step possess a combination of strong and medium strengths.

To calculate the strong and medium scores, we measure the local size of the peaks and valleys. We do this by looking on either side of the peak or valley to find extremes where the slopes change directions (changing from positive to negative or vice-versa). The values at these locations are used to determine the fraction of the total height of the local area that is the peak or valley. There are three local extremes used in the calculation: one at the center of the peak or valley, and one to each side. The scores for strong and medium classes are computed as follows.

$$SP = 100 * (E - (L + R) / 2) / E$$
$$MP = 100 - SP$$
$$SV = 100 * ((L + R) / 2 - E) / (L + R) / 2$$
$$MV = 100 - SV$$

where

*SP* = *strong peak score*
*MP* = *medium peak score*
*SV* = *strong valley score*
*MV* = *medium valley score*
*E* = *value at peak or valley location*
*L* = *value of extreme to left of E location*
*R* = *value of extreme to right of E location*

If no strong peaks or valleys are found, the data is scanned for peaks or valleys of medium strength. A medium peak is located when the slope has remained (nearly) the same over at least three data points and then changes significantly to a new value for at least three data points. Three data points are used to ensure that a true shoulder in the curve exists.

If a medium peak or valley is found, the amount of medium and weak character is computed. Peaks or valleys that contain a region of zero slope score 100% medium and 0% weak. Other peaks and valleys found in this step are a combination of medium and weak. To assign these strengths we determine the fraction of the overall height of the local area that is the shoulder. We do this by first finding the locations on either side of the peak or valley where the slope changes significantly. These locations and that of the peak or valley are the three slope-change locations used in the following calculation of medium and weak scores.

$$WP = 100 * (max(L,R) - E) / max(L,R)$$
$$MP = 100 - WP$$
$$WV = 100 * (E - min(L,R)) / E$$
$$MV = 100 - WV$$

where

*MP* = *medium peak score*
*WP* = *weak peak score*
*MV* = *medium valley score*
*WV* = *weak valley score*
*E* = *value at peak or valley location*
*L* = *value at slope-change to left of E location*
*R* = *value at slope-change to right of E location*

The computation of the medium class scores defined here do not conflict with the computation given for assigning strong and medium scores since combined strong and medium scores are mutually exclusive with combined medium and weak scores.

Finally, if the data has not yet been classified in the strong or medium assignment steps, a weak peak or valley is assumed. Data with increasingly-positive or decreasingly-negative slopes define data that is assigned as a 100% weak valley. Data with slopes that are decreasingly-positive or increasingly-negative is scored as a 100% weak peak. Partial weak and medium scores are not assigned here since that would have been done in the previous step.

Each class score is adjusted as it is computed to reflect

the proximity of a peak or valley to the location where the base was called. The scores are adjusted as follows.

$$S_{new} = S_{old} * (1 - |E - B| / N)$$

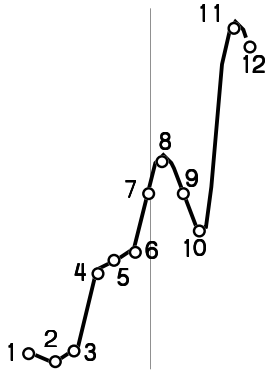
where

$S$ = *a class score*
$E$ = *location of peak or valley*
$B$ = *location of base call*
$N$ = *number of base trace-data points*

Peaks and valleys that are closer to where the base was called get higher scores.

(a) **Existing Representation** (b) **2-D Graph Representation**

| Point | Value |
|---|---|
| 1 | 39 |
| 2 | 14 |
| 3 | 47 |
| 4 | 375 |
| 5 | 404 |
| 6 | 464 |
| 7 | 704 |
| 8 | 833 |
| 9 | 702 |
| 10 | 570 |
| 11 | 1401 |
| 12 | 1321 |




(c) **New Representation**

Base Trace-Data Classification Scores

| SP | MP | WP | SV | MV | WV |
|---|---|---|---|---|---|
| 60 | 32 | 0 | 0 | 74 | 17 |
| (65) | (35) | (0) | (0) | (81) | (19) |

**Figure 9. Sample Base Trace-Data Classification.** (a) The existing representation of trace data is a sequence of intensities associated with a base call. (b) A 2-D graph representation of the trace data is shown as a curve interpolated from the data points. (c) Our new representation is a classification of the trace data based on the visual shape and intensity of the trace data. In this example, a base has been called between a peak and a valley. (The gray line indicates this location.) The base is called at point 7, and the peak and valley are detected at points 8 and 6 respectively. The peak and valley scores have been adjusted to reflect the distance of the peak and valley from the point where the base was called. The scores prior to adjustment are in parenthesis.

An example of our new representation of trace data, based on its visual shape and intensity, is shown in Figure 9. It is contrasted with the existing representation of trace data as a sequence of discrete intensity values. In the new representation, a valley with 81% medium strength and 19% weak strength has been detected to the left of the base-call location. A peak that is 65% strong and 35% medium is to the right of the base call. These scores have then been adjusted to reflect that the peak and valley are not at the base call location.

After classification scores have been computed for all four sets of trace data for a base, the scores are modified to account for the relative intensity differences among them. The following formula accomplishes this.

$$Pnew = Pold * (P / max(T))$$
$$Vnew = Vold * (1 - V / max(T))$$

*where*

$P$ = *a strong, medium, or weak peak score*
$V$ = *a strong, medium, or weak valley score*
$T$ = *base trace-data values*

Higher peaks and lower valleys get higher scores.

Each class in the base trace-data classification representation is now assigned a score between 0 and 100. If desired, a single class may be assigned to base trace-data by selecting *peak* or *valley* according to which has the higher sum of scores, and then *strong*, *medium*, or *weak* according to which has the highest score. For example, if a set of trace data is assigned scores of *SP=75, MP=14, WP=0, SV=0, MV=11,* and *WV=3, peak* has the higher sum of scores (75 + 14 + 0 = 89) compared to *valley* (0 + 11 + 3 = 14), and the highest scoring class is *strong* (75). Given this, the single class assignment is *strong peak*. We anticipate a need for both fine-grained and coarse classifications of trace data.

In summary, we have described a classification of base trace-data as follows.

- Broad categories are defined by curvature and include *peak* and *valley*.
- Peak and valley categories are divided into s*trong, medium,* and *weak* classes.
- Curves assigned to strong classes are characterized by a sharp peak or dip.
- Curves assigned to medium classes are characterized by a shoulder.
- Curves assigned to weak classes are characterized by a smooth slope in one direction.
- Scores reflect the amount of strong, medium, or weak character exhibited.
- Scores reflect the proximity of peaks and valleys to the base-call location.
- Scores reflect relative intensity to corresponding traces.
- A single class may be assigned by choosing the highest scoring class in the category with the higher sum of scores.

We believe that the new base trace-data classification representation we have defined may be used before, during, and after fragment assembly to increase the quality and efficiency of automatic processes. To demonstrate the value of our representation, we next describe a method

that successfully uses base trace-data classifications in an important pre-assembly step.

## Case Study: End-Trimming

The quality of the trace data, and therefore the base calls, decreases dramatically as the read through a gel progresses (Kelley 1994). In good data, peaks are sharp, well-defined, and scaled high (Perkin Elmer 1995). Figure 10 shows a set of trace data as it progresses from good to nearly useless. Since the accuracy of the data we use as input dramatically affects the results of an automatic assembly process, we want to use only data that is of sufficient quality to produce a good assembly. *End-trimming* is a common pre-processing step that helps to ensure that only good data is used in an assembly; it removes sub-optimal data from the 3′ ends of sequences (Seto, Koop & Hood 1993, McCombie & Martin-Gallardo 1994, Rowen & Koop 1994).

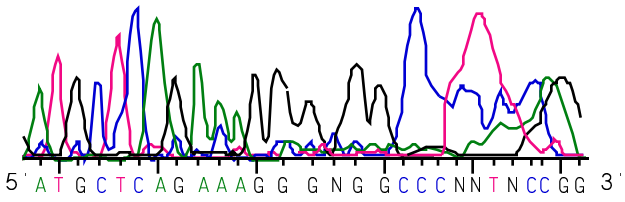



**Figure 10. Deterioration of trace data.** Trace data becomes progressively worse as a gel is read.

For our end-trimmming experiments, we used the DNAStar Inc. *Seqman* sequence-assembly software. In this program, sequences are added one at a time to contigs. New sequences are compared against the consensus of each existing contig. If an acceptable alignment is found with a contig, the sequence is added, otherwise a new contig is created (Burks et al. 1994, McCombie & Martin-Gallardo 1994, DNAStar 1994). Bad data adds many ambiguous and incorrect base calls to, or *poisons*, the consensus for its contig and may prevent subsequent sequences from being added to the contig (McCombie & Martin-Gallardo 1994). Even if bad data does not spoil the consensus enough to prevent the addition of sequences, it still results in a significant number of ambiguities in the consensus that must be resolved manually (McCombie & Martin-Gallardo 1994).

### Existing Methods

Existing methods for end-trimming include the use of *absolute cutoffs* and *N-Trimming*.

The absolute cutoff method trims sequence data after a user-specified number of bases. Often with ABI data, the number of bases is about 500 – this is based on the observation that the quality of trace data generally deteriorates after 500 bases (Kelley 1994). Given this information, data from base 500 to the end of the sequence is trimmed off the sequence. Although trimming the data after 500 base calls is reasonable, the problem is that sometimes good data is trimmed away while at other times poor data is kept.

The other method, N-Trimming, trims off data that exceeds an allowed number of no-calls (*Ns*) in windows of sequence data. The DNAStar *Seqman* sequence-assembly program employs an adaptation of this method described as *End-Clip* in (Seto, Koop & Hood 1993). *Seqman* requires two parameters; one sets a window size in bases and the other specifies the number of *Ns* that are allowed in a window. The sequence of base calls is scanned from the 3' to 5' end until a window of the given size is found such that the number of *Ns* in the window is less than or equal to the maximum number allowed. Data from this window to the end of the sequence is trimmed off. For example, in Figure 10, if we set the window size to 20 and the number of *Ns* allowed to 2, the last seven bases on the 3' end of the sequence would be trimmed away.

Although it is still commonly used, we believe that N-Trimming has been made less useful by contemporary ABI sequencers – the sequencers tend to almost always make a base call even when the trace data is erratic. This results in inferior data that remains untrimmed by the N-Trimming method. Although the number of no-calls is certainly correlated with the quality of trace data (and the base calls made from it), it is advisable to look directly at the trace data to determine its quality. Information contained in the trace data can be used to make a more intelligent decision about the quality of the associated base calls and the best location for trimming.

### Trace-Class Trimming

We want to use the information in trace data to make useful end-trimming decisions. To do this, we examine the base trace-data classifications defined earlier to determine the quality of regions of trace data. For our method, we simply use the single class assignment (i.e. the base trace-data is assigned the class with the highest score in the category with the highest sum of scores). Similar to N-trimming, our algorithm scans data in windows. However, rather than examining the windows for no-calls (as in N-Trimming), we look for sub-optimal base trace-data classifications. As we scan the window, we note only the class of the base trace-data associated with the base that has been called at that location. In general, trace data that falls into the strong peak class is considered optimal; base calls made with corresponding trace data of that classification are likely to be accurate. Medium peaks indicate trace data that is less likely to produce accurate

base calls, and weak peaks and valleys indicate unreliable base calls.

To perform Trace-Class Trimming, three parameters must be set: the size of the window in bases, the maximum number of sub-optimal trace classes to be allowed in the window, and a cutoff that specifies which classes are considered sub-optimal. As the cutoff is adjusted, the trimming stringency is changed correspondingly. For example, the most stringent cutoff would specify that all classifications except strong peaks are sub-optimal. A cutoff identifying all but strong and medium peaks as sub-optimal is less stringent.

The sequence of base trace-data classifications is scanned from the 3′ to 5′ end until a window of the given size is found such that the number of sub-optimal classifications in the window is less than or equal to the maximum number allowed. The data from this window to the end of the sequence is trimmed from the sequence and not used when adding the sequence to the assembly. (The data is not actually "thrown away," but is kept for possible use in manual editing.) Figure 11 gives an example of Trace-Class Trimming of a sequence.

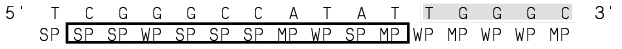



**Figure 11. Trace-Class Trimming example.** In this example, the window size is ten, the maximum number of sub-optimal data classifications allowed is two, and weak peaks (WP) and all valleys are considered sub-optimal. The box encloses the first window from the 3' end of the data that contains two or fewer sub-optimal peaks. The shaded area of the sequence is trimmed off.

We empirically evaluate Trace-Class Trimming and compare it to N-Trimming by optimizing the parameters for each method over one set of data and then testing the best parameters on a second set of data.

## Data Sets

We used data from the *E. coli* Genome Project lab at the University of Wisconsin that was gathered for an assembly of a 243 kb fragment of *E. coli*. Data sets were formed in the following way. The 2021 sequences in the set of data for the assembly were trimmed extensively such that only bases from locations 50 to 200 remained in each sequence. To this set, we added longer *E. coli* sequences from GenBank that were believed to fall in the 243 kb section of the *E. coli* genome. The sequences were then automatically assembled. In this way, only the very best data was used and contigs were formed with sequences that should align (given nearly ideal data).

All contigs containing ten or more sequences were chosen for inclusion in data sets. In these contigs, the GenBank sequences were removed and the full untrimmed length of sequences was reinstated. Each set of sequences in a contig formed a separate data set, called a *project*, that could be independently assembled. The result was 20 projects for evaluating trimming methods. Ten projects form a *training set* used to optimize parameters and the other ten sets form a *test set* used to test the quality of subsequent assemblies using the optimized parameters. Training and test sets were chosen to make the number of projects equal and the total number of sequences similar.

For use in our evaluations, we estimated the expected number of contigs and total contig length for each project. Although each project is formed from a single contig, in some cases, the expected number of contigs is greater than one because regions in the contig were bridged by (now removed) GenBank sequences. To estimate the expected total contig length, we simply use the length of the contigs after they have been extended with complete, untrimmed sequences. The data sets are described in Table 1.

**Table 1. Data sets.** The number of sequences is the actual number in the project. The number of contigs and the contig length are the expected values for the project.

**(a) Training set.**

| Project | #Sequences | #Contigs | Contig Length |
|---|---|---|---|
| 1 | 11 | 2 | 2235 |
| 2 | 14 | 1 | 1715 |
| 3 | 15 | 1 | 2364 |
| 4 | 18 | 1 | 3352 |
| 5 | 20 | 2 | 5229 |
| 6 | 22 | 1 | 1473 |
| 7 | 26 | 1 | 824 |
| 8 | 32 | 3 | 7067 |
| 9 | 69 | 3 | 11,088 |
| 10 | 37 | 3 | 9050 |
| Total | 264 | 18 | 44,397 |

**(b) Test set.**

| Project | #Sequences | #Contigs | Contig Length |
|---|---|---|---|
| 1 | 20 | 2 | 2810 |
| 2 | 16 | 1 | 1221 |
| 3 | 18 | 3 | 4271 |
| 4 | 24 | 3 | 6221 |
| 5 | 27 | 2 | 4503 |
| 6 | 35 | 2 | 6696 |
| 7 | 38 | 1 | 776 |
| 8 | 13 | 2 | 3010 |
| 9 | 15 | 1 | 3408 |
| 10 | 57 | 3 | 11,382 |
| Total | 263 | 20 | 44,298 |

In addition to the projects in the test set, we evaluated our system with an unrelated set of sequences. These are from a 7 kb segment of human DNA. This project has reached completion so the number of contigs and contig length is known. Table 2 describes this set.

**Table 2. Human DNA data set.**

| Project | #Sequences | #Contigs | Contig Length |
|---|---|---|---|
| human | 98 | 2 | 7257 |

## Method and Results

We optimized parameters for the Trace-Class Trimming method and separately for N-Trimming. For N-Trimming, we varied the window size from ten to 50 in increments of five and the number of *N*s to be allowed in a window from zero to five. For Trace-Class Trimming, we varied the window size from ten to 50, the number of sub-optimal peaks to be allowed from zero to five, and the trace class cutoffs over strong peaks, medium peaks, and weak peaks. Valleys were always considered sub-optimal. Each project in the training set was assembled with every combination of parameters and the quality of assemblies was evaluated.

The goal of end-trimming is to produce better-quality automated assemblies of DNA fragments. We used three metrics to measure the quality of assemblies. One is the number of contigs. In general, we want a group of sequences to assemble into a small number of contigs (the ultimate goal is to have only a single contig). The second metric is the number of ambiguities in the consensus sequence. Fewer ambiguities means not only that the sequences align well, but also that less manual work is needed. The third measure is the total length of the contigs. Contigs should be as long as possible without incorporating too many ambiguities.

We measure the number of contigs and contig length as the absolute deviation from the expected values, and the number of ambiguities as the average number of ambiguous calls per kb. To score each set of parameters, we normalize and individually sum the three metrics across all data sets for each set of parameters. The overall score, $S_i$, for parameter set $i$ is

$$S_i = \alpha C_i + \beta T_i + \gamma A_i$$

where $C_i$, $T_i$, and $A_i$ are the normalized sums of the number of contigs, total length of contigs, and number of ambiguities metrics, respectively; $\alpha$, $\beta$, and $\gamma$ are constants. We believe that the order of importance of the metrics is: 1) number of contigs, 2) number of ambiguities, and 3) total length of contigs. Consequently, we set $\alpha = 3$, $\beta = 1$, and $\gamma = 2$ to weight the metrics.

Using the scheme described above, we scored and sorted the parameter sets. We found that, in general, the best Trace-Class Trimming assemblies resulted when the window size was large (40 to 50 bases), the cutoff defined both strong and medium peaks as optimal, and the number of sub-optimal peaks to be allowed was between 5% and 10% of the window size. The best N-Trimming assemblies resulted when the window size was large (40 to 50 bases), and the number of *N*s allowed was small (0 to 2).

The ten minimum scoring parameter sets for N-Trimming and for Trace-Class Trimming were chosen as optimal parameter settings. Next, test set projects were assembled using each of the top ten parameter settings for N-Trimming and Trace-Class Trimming settings. The human DNA project was assembled using only the top-scoring parameter sets. As a baseline, the projects were also assembled with no trimming.

## Discussion

By all three measures of evaluation (number of contigs, total contig length, and number of ambiguities), our new Trace-Class Trimming resulted in assemblies of better quality than those produced after N-Trimming or no trimming. Figure 12 graphs the results for the ten test-set projects. On average over the test-set projects, the absolute deviation from the expected length of contigs falls by about 75% and both the deviation from the expected number of contigs and the number of ambiguities per kb falls by about 50% from assemblies using N-Trimming to those using Trace-Class Trimming. The decrease in the number of ambiguities represents a significant decrease in the amount of hand editing that needs to be done on assembled projects. For example, in a 243 kb project, the number of ambiguities to be resolved would decrease from nearly 10,000 bases using N-Trimming to fewer than 5000 using Trace-Class Trimming.

With the human DNA project, we again see a significant improvement in the assembly done after Trace-Class Trimming over the assemblies done after N-Trimming or no trimming. Table 3 contains the results for the human DNA project. After Trace-Class Trimming, the assembly produces three contigs, compared to five contigs with N-Trimming (the expected number is two). It also results in a 40% reduction in the number of ambiguities per kb over the assembly done after N-Trimming.

The key to the success of Trace-Class Trimming is that it uses the information contained in trace data in the form of base trace-data classifications. These classifications directly reflect the morphology of trace data, and are good indicators of the accuracy of the associated base calls. The N-Trimming method does not use trace data, rather it examines only the sequence of bases for no-calls. Since modern sequencers make base calls even when the trace data is erratic, searching for no-calls may no longer be useful as a method for assessing the accuracy of base calls.

## Future Work

Our trace-data representation is a first attempt at capturing visual qualities of ABI trace data. Although we have had success in using it as described, we believe that it can be enriched to make it more powerful. Relative intensities and relative separations among peaks have been identified as important features in patterns of DNA sequences (Golden, Torgersen & Tibbetts 1993, Tibbetts, Bowling & Golden 1994). We will study the merit of incorporation of these features in our representation. In addition, refinements may be made that define peak sharpness and intensity relative to a global scale.

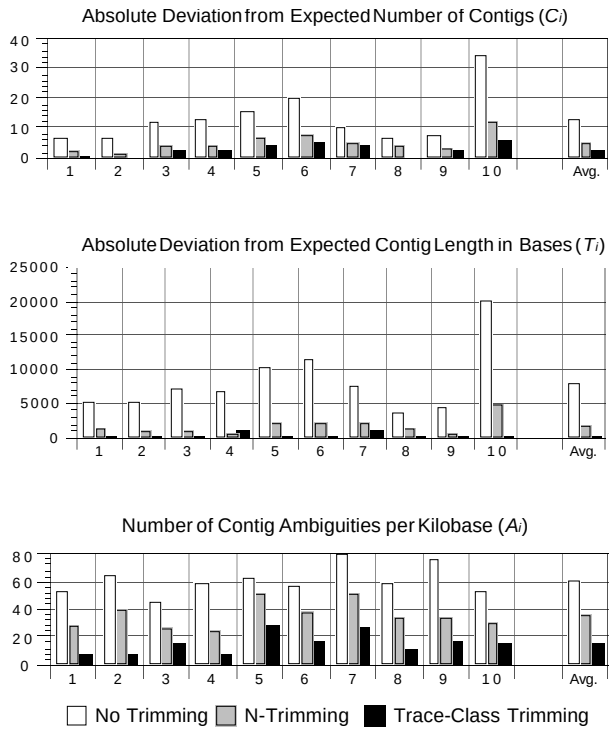



**Figure 12. Results.** Results are graphed individually for the ten test-set projects as well as for the average over all projects.

We also plan to explore other ways that our representation can improve the quality and efficiency of automated sequence-assembly.

The end-trimming method we described works as a pre-processing step to assembly. Another pre-processing step we believe base trace-data classifications may be useful for is base calling. Machine learning systems such as neural networks can be trained to recognize patterns of classification scores that are associated with particular base calls. Once the system has been trained to recognize the patterns, it can make base calls with previously unseen data. Tibbetts, Bowling, and Golden (1994) describe work on one such method that uses trace data as input to simple base-calling neural networks.

**Table 3. Human DNA project test results.** Trace-Class Trimming yields an assembly with one more than the expected number of contigs compared to three more with N-Trimming. The Trace-Class Trimming assembly had 40% fewer ambiguities than the assembly done with N-Trimming.

| Trimming Method | #Contigs Deviation | Contig Length Deviation | Ambiguities per kb |
|---|---|---|---|
| Trace-Class | 1 | 576 | 32 |
| N | 3 | 3113 | 54 |
| None | 13 | 12, 818 | 149 |

Base trace-data classes may also be incorporated into the actual assembly process. In the *Seqman* assembly program, the consensus is computed with a scheme that uses a weight assigned to each base call in each sequence. The idea is that better quality data should have higher weights that will result in a greater contribution to the consensus computation than poorer quality data. The weights can either be assigned uniformly or according to a trapezoidal rule as described in the DNAStar *Lasergene User's Guide* (1994). In either case, sequence-specific information is not used and data may be weighted inappropriately for its quality. As with end-trimming, base trace-data classifications can be used as a measure of the quality of base calls in a sequence. Weights can be assigned to a base call that reflect the quality of data in the local area of the call. This proposed method uses information specific to the sequence as suggested by Rowen and Koop (1994) and Bonfield and Staden (1995).

Alternately, base trace-data classifications may be used in the assembly process to actually compute the consensus sequence. We have encouraging results from preliminary studies in which we use a summing of class scores as evidence for making a particular consensus call.

In the post-assembly process, a significant amount of time is spent in hand-editing. If we can use machine learning techniques such as neural networks to train a system to look for the same trace-data patterns as the editors, we may be able to automate a significant portion of the manual process. Tibbetts, Bowling, and Golden (1994) describe a single-sequence automatic editing system that uses a neural network to confirm calls or suggest changes.

Finally, we believe that a more sophisticated analysis of trace-data classifications can provide useful information to users and assembly programs. In particular, we would like to be able to identify problematic areas in trace data. Some possible causes for the existence of such areas are: the presence of unincorporated dideoxynucleotides, homopolymer regions, gel compressions, and noisy data.

These regions are generally characterized by trace data that exhibits concurrent significant intensities or peaks among the four dye traces (Perkin Elmer 1995). This occurrence can be detected with our trace-data representation and the information gathered can be used by hand-editors or in automatic processes requiring an assessment of data quality.

## Conclusions

The quality and efficiency of automated DNA assembly of ABI-generated sequences can be increased by the incorporation of trace-data information into the process. The visually-oriented base trace-data classes we describe provide a representation of trace data information that makes this incorporation possible. We have shown one such use, trimming of sub-optimal data before assembly, that results in better assemblies. Using the base trace-data classifications for trimming leads to a decrease in the number of contigs, a reduction in ambiguities, and a closer approximation to the expected contig length. Refinements of and other uses of our representation are under investigation.

## Acknowledgments

We thank Mark Craven, Ernest Colantonio, and Guy Plunkett for their useful comments on this document.

This research was supported in part by National Research Service Award 5 T32 GM08349 from the National Institute of General Medical Sciences, and in part by Small Business Innovation Research grant 1 R43 GM51680-01 from the Department of Health and Human Services.

## References

Ansorge, W., Sproat, B.S., Stegemann, J., and Schwager, C. 1986. A non-radioactive automated method for DNA sequence determination. *Journal of Biochemical and Biophysical Methods* 13:315-323.

Burks, C., Engle, M.L., Forrest, S., Parsons, R.J., Soderlund, C.A., and Stolorz, P.E. 1994. Stochastic optimization tools for genomic sequence assembly. In Adams, M.D., Fields, C., and Venter, J.C., eds., *Automated DNA Sequencing and Analysis* 249-259. San Diego, CA: Academic Press.

Chen, E.Y. 1994. The efficiency of automated DNA sequencing. In Adams, M.D., Fields, C., and Venter, J.C., eds., *Automated DNA Sequencing and Analysis* 3-10. San Diego, CA: Academic Press.

Connell, C., Fung, S., Heiner, C., Bridgham, J., Chakerian, V., Heron, E., Jones, B., Menchen, S., Mordan, W., Raff, M., Recknor, M., Smith, L., Springer, J., Woo, S. and Hunkapiller, M. 1987. Automated DNA sequence analysis. *BioTechniques* 5:342-348.

Dear, S. and Staden, R. 1991. A sequence assembly and editing program for efficient management of large projects. *Nucleic Acids Research* 19:3907-3911.

DNAStar. 1994. *Lasergene User's Guide*, Madison, WI.

Edelman, I. 1996. Personal communication.

Golden, J.B. III, Torgersen, D. and Tibbetts, C. 1993. Pattern recognition for automated DNA sequencing: I. On-line signal conditioning and feature extraction for basecalling. In *Proceedings of the First International Conference on Intelligent Systems for Molecular Biology*, 136-134. Bethesda, MD: AAAI Press.

Hunkapiller, T., Kaiser, R.J., Koop, B.F. and Hood, L. 1991. Large-scale and automated DNA sequence determination. *Science* 254: 59-67.

Kelley, J.M. 1994. Automated dye-terminator DNA sequencing. In Adams, M.D., Fields, C., and Venter, J.C., eds., *Automated DNA Sequencing and Analysis*, 175-181. San Diego, CA: Academic Press.

Kruskal, J.B. 1983. An overview of sequence comparison. In Sankoff, D., Kruskal, J.B., eds., *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison* 1-44. Reading, MA: Addison-Wesley Publishing Company, Inc.

Martinez, H.M. 1983. An efficient method for finding repeats in molecular sequences. *Nucleic Acids Research* 11:4629-4634.

Maxam, A.M. and Gilbert, W. 1977. A new method for sequencing DNA. *Proceedings of the National Academy of Science USA* 74:560-564.

McCombie, W.R., and Martin-Gallardo, A. 1994. Large-scale, automated sequencing of human chromosomal regions. In Adams, M.D., Fields, C., and Venter, J.C., eds., *Automated DNA Sequencing and Analysis* 159-166. San Diego, CA: Academic Press.

Myers, E.W. 1994. Advances in sequence assembly. In Adams, M.D., Fields, C., and Venter, J.C., eds., *Automated DNA Sequencing and Analysis* 231-238. San Diego, CA: Academic Press.

Needleman, S.B. and Wunsch, C.D. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* 48:443-453.

Perkin Elmer. 1995. *DNA Sequencing: Chemistry Guide.* Foster City, CA.

Prober, J.M., Trainor, G.L., Dam, R.J., Hobbs, F.W., Robertson, C.W., Zagursky, R.J., Cocuzza, A.J., Jensen, M.A. and Baumeister, K. 1987. A system for rapid DNA sequencing with fluorescent chain-terminating dideoxynucleotides. *Science* 238:336-341.

Rosenberg, D. 1996. Personal Communication.

Rowen, L., and Koop, B.F. 1994. Zen and the art of large-scale genomic sequencing. In *Automated DNA Sequencing and Analysis* 167-174. San Diego, CA: Academic Press.

Schroeder, J. 1996. Personal communication.

Sanger, F., Nicklen, S. and Coulson, A.R. 1977. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Science USA* 74:5463-5467.

Seto, D., Koop, B.F. and Hood, L. 1993. An experimentally derived data set constructed for testing large-scale DNA sequence assembly algorithms. *Genomics* 15:673-676.

Smith, L.M., Sanders, J.Z., Kaiser, R.J., Hughes, P., Dodd, C., Connell, C.R., Heiner, C., Kent, S.B.H. and Hood, L.E. 1986. Fluorescence detection in automated DNA sequence analysis. *Nature* 321:674-679.

Staden, R. 1980. A new computer method for the storage and manipulation of DNA gel reading data. *Nucleic Acids Research* 8:3673-3694.

Tibbetts, C., Bowling, J.M. and Golden, J.B. III. 1994. Neural networks for automated base-calling of gel-based DNA sequencing ladders. In Adams, M.D., Fields, C., and Venter, J.C., eds., *Automated DNA Sequencing and Analysis* 219-230. San Diego, CA: Academic Press.

Waterman, M.S. 1989. Sequence alignments. In Waterman, M.S., ed., *Mathematical Methods for DNA Sequences* 54-92. Boca Raton, FL: CRC Press.