To appear in *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*, Halkidiki, Greece: AAAI Press. 1997.

Increasing Consensus Accuracy in DNA Fragment Assemblies by Incorporating Fluorescent Trace Representations

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

We present a new method for determining the consensus sequence in DNA fragment assemblies. The new method, *Trace-Evidence*, directly incorporates aligned ABI trace information into consensus calculations via our previously described representation, *Trace-Data Classifications*. The new method extracts and sums evidence indicated by the representation to determine consensus calls. Using the *Trace-Evidence* method results in automatically produced consensus sequences that are more accurate and less ambiguous than those produced with standard majority-voting methods. Additionally, these improvements are achieved with less coverage than required by the standard methods – using *Trace-Evidence* and a coverage of only three, error rates are as low as those with a coverage of over ten sequences.

Introduction

Our goal is to improve the quality and efficiency of automatic DNA sequencing (determining the sequence of bases in DNA molecules). One important task in the sequencing process is to establish the consensus sequence for aligned fragments of DNA. This task, often referred to as *consensus calling*, is the focus of this paper.

The usual method to accomplish consensus calling relies on the fragments' representation as sequences of base calls. Before alignment, the base-call sequences are determined by computer analysis of the fluorescent-dye intensity signal, called *trace data*, that is output by automatic sequencers such as the Applied Biosystems Inc. (ABI) 377. The base calls are used to align the sequences and then a majority-voting scheme is commonly used to identify the consensus for each aligned column of bases. We believe, and have shown with an earlier case study in sequence trimming (Allex et al. 1996), that automatic sequencing processes can be improved by incorporation of descriptive fluorescent trace information.

The new method we have developed for automatic consensus calling, *Trace-Evidence*, incorporates ABI trace information via the *Trace-Data Classifications* we defined in prior work (Allex et al. 1996). The use of this representation is the key to the improvements we see when using our new method. Previous methods use only a very simplified representation of trace data: the sequence of base calls. The representation we use is much more descriptive; it captures shape and intensity visual characteristics of traces.

Our *Trace-Evidence* method results in automatically produced consensus sequences that are more accurate and less ambiguous. In addition, it attains these improvements with fewer aligned sequences. (The number of aligned sequences is known as the *coverage*.) Both an increase in accuracy and a reduction in the coverage needed are significant results.

Higher accuracy using *Trace-Evidence* consensus calling mitigates the problem of errors in DNA sequences. The error rate for sequences in GenBank has been estimated to be from 0.3 to 0.03% (Lawrence & Solovyev 1994). If the DNA is translated into protein, the results are potential errors in 0.1 to 0.01% of the amino acids. The mutation of a single amino acid can have substantial adverse effects on protein-related research. Furthermore, the change, deletion, or insertion of a single base can lead to frame shifts and/or the inability to identify open reading frames. Sequencing accuracy is significantly dependent upon careful human examination and editing of consensus sequences in fragment assemblies. The hand process is time-consuming, expensive, and error-prone. Automatic assemblies with improved consensus accuracies alleviate these problems.

Reducing the needed coverage for accurate sequencing by using *Trace-Evidence* means a reduction in overall sequencing costs. Fragment preparation represents a substantial portion of the expense of sequencing. In large sequencing projects, it is typical to produce a coverage of at least six to ensure accurate results. We show in this work that equivalent accuracy can be achieved with as little coverage as three sequences.

We have implemented our new *Trace-Evidence* consensus calling (as well as previously described *Trace-Class* sequence trimming) in the DNAStar Inc. *SeqMan II* fragment assembly program. Among numerous other software packages available to facilitate sequencing efforts are: GCG *Fragment Assembly System* (Dolz 1994), *TIGR Assembler* (Sutton et al. 1995), *Staden Package* (Bonfield et al. 1995), *PHRED* and *PHRAP* (the University of Washington), and *Sequencher* (Gene Codes Corporation). In contrast to these systems, our approach ignores base calls and derives the consensus directly from the trace data.

The remainder of this paper first describes a common previous method for consensus calling, followed by a detailed description of our new *Trace-Evidence* method. We then present empirical evidence of the effectiveness of using *Trace-Evidence.* A discussion of future work and conclusions complete the paper. For readers who are unfamiliar with DNA sequencing, an appendix provides a brief background.

We make liberal use of figures to illustrate our points throughout this paper. In these figures, all trace data displayed are representations of actual sequencing data supplied by the *E. coli* Genome Project at the University of Wisconsin.

Previous Method: *Majority*

A common, simple method to calculate the consensus counts the number of calls of each base in an aligned column (Staden 1982). If the majority base count is above a given fractional threshold of the total count, that base is called unambiguously (*A*, *C*, *G*, or *T*); otherwise the consensus is called as the appropriate ambiguity (combination of *A*, *C*, *G*, and/or *T*). We refer to this method as *Majority*. Figure 1 contains an example of calculating the consensus by *Majority*.

Since the *Majority* approach examines only the base calls and not the underlying trace data, it is prone to errors. *Majority* also requires a minimum number of sequences to make an unambiguous call when a column of base calls is not in total agreement. Methods that directly analyze the trace data help to avoid these problems.

(a)

```
Consensus   ... AAGAAGCACTWAGGATTTGGT ...
---------------------------------------
            ... AAGAGGCACTAAGGATTTGGT ...
Aligned     ... AAGAAGCACTTAGGATTTGGT ...
Fragments   ... AAGAAGCACTAAGGATTTGGT ...
            ... AAGAAGCACTTAGGATTTGGT ...
                    5     11    17
```

(b)

Column	Majority Base(s)	%	Consensus Call
5	*A*	75	*A*
11	*A*, *T*	50	*W* (*A* or *T*)
17	*T*	100	*T*

Figure 1: *Majority* Consensus Calls. (a) Four sequences are aligned and the consensus computed using *Majority*. (b) In this example, the threshold is set at 75%. The consensus call for column 5 is an *A* since three of four (at least 75%) of the calls are *A.* In column 11, 50% of the calls are A and 50% are *T*; the call is *W* (*A* or *T*) since both percentages are below the threshold. In column 17, all calls are *T*, resulting in a consensus call of *T*.

New Method: *Trace-Evidence*

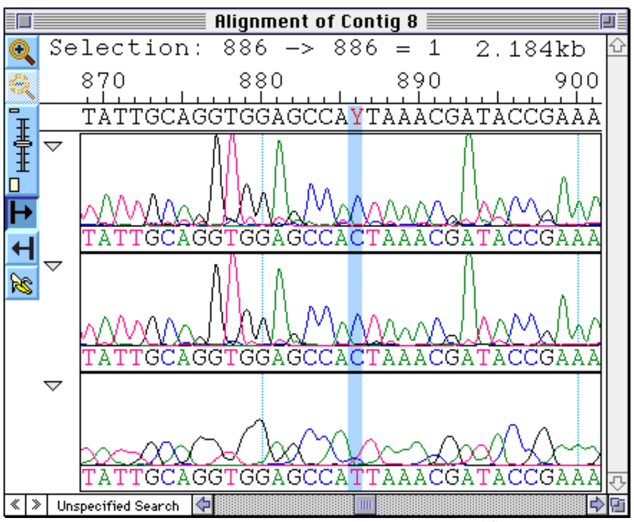
We improve the quality of automatic consensus calling by emulating human analysis of trace data. When an ambiguous consensus call is made by an assembly program, human editors attempt to resolve the ambiguity by a visual examination of the fluorescent trace data. Figure 2 describes an example of resolving an ambiguity by hand. While watching editors working, we observed that many, if not most, of the decisions they make are straight-forward. We believed that we could program a computer to capture the visual characteristics that are relevant to human editors in their work and use this information to improve automatic consensus-calling.

Overview

The new method we have developed for computing a consensus directly incorporates ABI trace-data information via the *Trace-Data Classifications* we developed and described in earlier work (Allex et al. 1996). Figure 3 contains a brief review of our *Trace-Data*

Classifications. The new method for consensus calling sums the evidence supplied for each base by the classification scores. We will refer to the new method as the *Trace-Evidence* consensus method.

The *Trace-Evidence* method is based on the idea that each of the six scores for the *Trace-Data Classifications* supplies an amount of evidence that the associated base should be assigned in the consensus. High *strong peak* (*SP*) scores supply the greatest amount of evidence, high *medium peak* (*MP*) scores supply the next greatest amount of evidence, followed by high *weak peak* (*WP*), *weak*, *medium*, and *strong valley* (*WV*, *MV*, and *SV*) scores in decreasing order. In fact, a high *valley* score actually provides counter-evidence for its base. Figure 4 demonstrates the evidence idea.



DNAStar Inc. *SeqMan II*

Figure 2: Human Editing. The ambiguity in the highlighted column must be resolved to *A*, *C*, *G*, or *T*. Human editors examine the traces and observe that the the first two sequences are of good quality and exhibit sharp, well-defined peaks in the *C* trace. The third sequence, although of poorer quality, also shows a small, discernible peak in the *C* trace, even though the base has been miscalled as a *T*. Human editors determine that the consensus should be a *C*.

To determine the consensus for a column of aligned bases, we sum the evidence, based on *Trace-Data Classification* scores, for each of the four bases. The evidence for each base is weighted by the quality of the trace data in a local area; the quality is also determined by examining *Trace-Data Classification* scores. The base with the highest evidence sum is identified as the *leader* and its evidence sum is the *leading evidence*. The other three bases are *competitors*, and their evidence sums are *competing evidence*. A threshold between 0 and 1 is specified that determines the ignorable fraction of *competing evidence* to *leading evidence*. If the *leader* has no *competitors* with *competing evidence* greater than the threshold, the *leader* is assigned as the consensus. If *competing evidence* for any bases surpasses the threshold, then those bases are included in determining an ambiguous call.

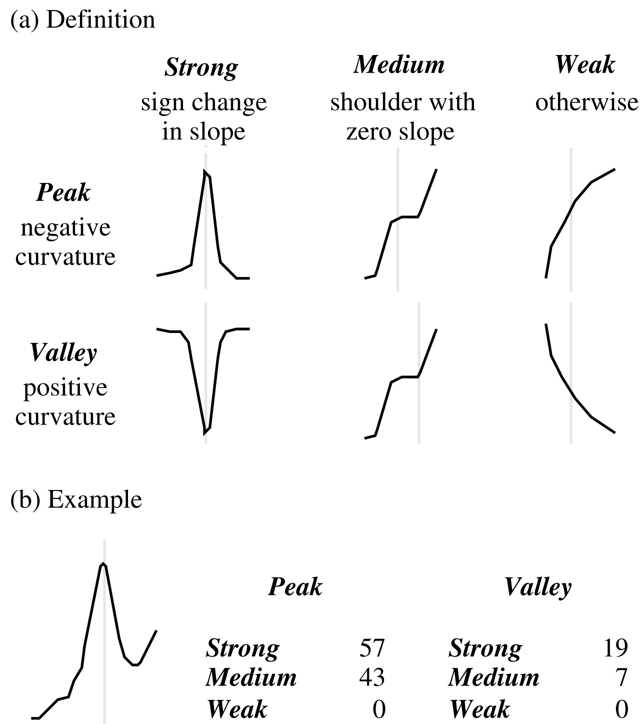


Figure 3: Trace-Data Classifications. The *Trace-Data Classification* representation uses shape and intensity to categorize the trace data for a single base call. (a) The classes and the criteria used to distinguish among them are listed and illustrated. A score from 0 to 100 is assigned for each of six classes that reflects the amount of *strong*, *medium*, and *weak peak* and *valley* characteristic that is exhibited by the data. Both the *peak* and *valley* nearest the base call are identified and scored. Gray lines show the location of the base call. (b) For some calculations we need the classifications for all four sets of trace data while for others we need only the classification of the trace associated with the base that is called. In this example, one of the four sets of trace data is shown. The scores for the trace indicate a combination *strong-medium peak* at the base-call location and a *strong-medium valley* distanced from the base-call location. Scores are adjusted to reflect the distance of the *peak* or *valley* from the base-call location as well as the intensity relative to the other three traces.

Algorithmic Details

To determine the consensus for a column of aligned bases, two types of values must be calculated for each sequence in the column: the *Trace-Data Classification* scores and a measure of the quality of the data. We use the quality of the trace data to weight the evidence supplied by each set of classification scores. That way, more credible, higher quality trace data supplies more evidence than trace data of lower quality.

When gaps occur in a column, a quality measure is also used to decide if the consensus should be called as a gap. To do this, we sum the quality measures for sequences with a gap in the column and compare them with the sum of the quality measures of sequences without a gap. If the gap quality sum exceeds the non-gap sum, the consensus is called as a gap. The problem of calling the consensus when gaps are involved is not always as simply solved as this. We will discuss this further in the *Discussion and Future Work* section.

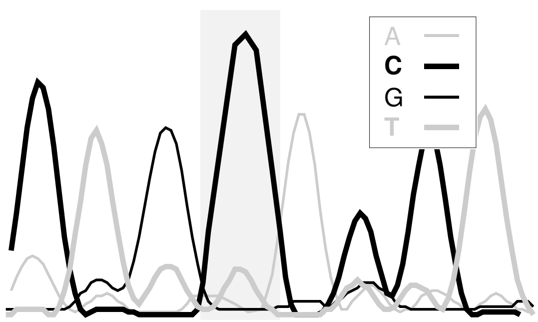


Figure 4: Evidence in Traces. Consider the evidence found in the four traces in the shaded region. The *C* trace will produce a high *strong peak* (*SP*) score, the *T* trace will yield a relatively smaller *SP* score, and both the *A* and *G* traces will produce *valley* scores. A visual examination of the traces supports the premise that the vast majority of the evidence is for a base call of *C* and that there is counter-evidence for a base call of *A* or *G*.

The steps used in the consensus calculation for a single aligned column appear below. Details of calculations mentioned follow the algorithm.

Trace-Evidence Consensus Algorithm

For a single aligned column

1. For each sequence, find the quality of trace data, Q, within a small window centered on the column.
2. Sum Q for each sequence with a gap in the column and compare it to the sum of Q for the remaining sequences. If the gap sum exceeds the non-gap sum, return *gap*.
3. Determine S, the 4 x 6 (six scores for each of four traces) matrix of *Trace-Data Classification* scores for each sequence.
4. Reduce each S to a vector, E, of four values that summarize the evidence for each trace.
5. Multiply each value in E by its corresponding Q to produce a vector E' that has been adjusted by data quality.
6. Sum each of the corresponding E's to produce a vector, T, of the total evidence for each of the four bases.
7. Find the highest evidence (*leading evidence*) in T; its corresponding base is the *leader*.
8. Multiply *leading evidence* by the threshold to compute the maximum ignorable *competing evidence*.
9. Compare *leading evidence* to each *competing evidence*. If no *competing evidence* surpasses the maximum ignorable, then return *leader* as the consensus call, otherwise use all *competitors* who surpass the maximum to determine and return an ambiguity.

We use the *Trace-Data Classifications* as indicators of trace data quality. The idea is similar to that described earlier for establishing evidence. Base calls that are highly reliable are made from trace-data peaks that are sharp and well-defined – those that classify as *strong peaks*. Base calls made from trace data that classify as *medium peaks* are less reliable, and those made from *weak peaks* or *valleys* are increasingly less reliable. Therefore, to determine quality, we examine the *Trace-Data Classification* scores for the traces associated with the called bases. (For example, if the sequence of bases has been called as *GGTACG*, only the *Trace-Data Classifications* for the corresponding *G*, *G*, *T*, *A*, *C*, and *G* traces are calculated.) If the *strong peak* scores are high, it is likely that the data is of good quality – the higher the scores, the better the quality of the data. On the other hand, if the bases have been called with low *peak* scores or non-trivial *valley* scores, the base calls are not as obvious and the data is likely to be less reliable and of lower quality. Figure 5 compares the relative quality of some trace data.

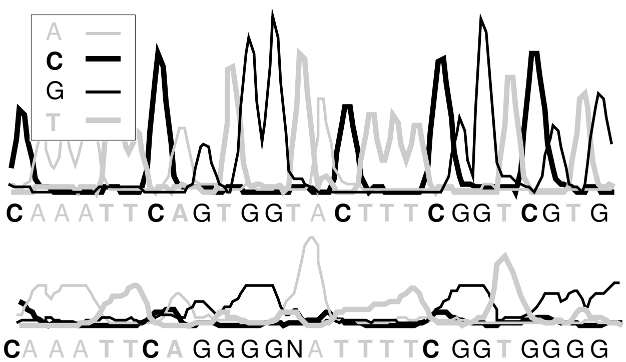


Figure 5: Quality of Trace Data. In the top sequence, the trace associated with each called base exhibits a sharp well-defined peak. The corresponding *Trace-Data Classifications* all show high *strong peak* scores. In contrast, the traces in the bottom sequence are flattened and overlapping; corresponding *strong peak* scores are generally nominal. The top sequence is much more reliable and is of high quality. We want to give the evidence it supplies more weight in consensus calculations.

For use in our calculations of quality, we predefine a constant weight vector, W, such that classes (such as *SP*) that imply better-quality data for a base are given higher values than those (such as *SV*) that imply lower-quality data. The definition follows.

Let

$$W = |W_{SP} \;\; W_{MP} \;\; W_{WP} \;\; W_{WV} \;\; W_{MV} \;\; W_{SV}|$$

where

W_i *is the weight for class i*

and

$$1 \geq W_{SP} \geq W_{MP} \geq W_{WP} \geq W_{WV} \geq W_{MV} \geq W_{SV} \geq 0$$

Using the weight vector, the quality of the data for each sequence in a column is calculated as follows.

1. For each base, i, in a window of size n centered on the column of interest, calculate the vector of *Trace-Data Classification* scores, S_i, for the trace associated with the base that has been called (details of the calculation of S_i appear in Allex et al. 1996):

$$S_i = |SP_i \;\; MP_i \;\; WP_i \;\; WV_i \;\; MV_i \;\; SV_i|$$

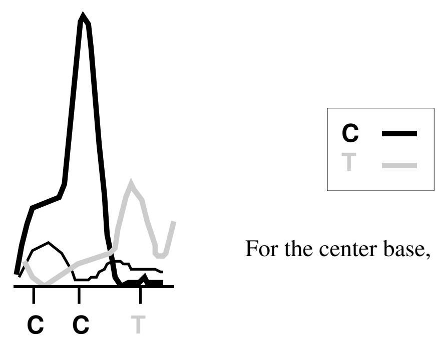
2. The dot product of S_i and W^t (W-transpose) produces a scalar quality measure, Q_i, for base i:

$$Q_i = S_i \bullet W^t$$

3. Average the measures to produce an overall quality score, Q, for the base at the center of the window:

$$Q = (Q_1 + Q_2 + \ldots + Q_n) \;/\; n$$

In our work, we found that $W = |1 \;.67 \;.33 \;0 \;0 \;0|$ yields good results. Using this definition, the quality measures, Q, are between 0 and 100 since *peak* classification scores sum to 100 or less. (Other possible definitions of the weight vector are discussed in the *Discussion and Future Work* section.) Figure 6 contains an example calculation of a quality score.



	SP	*MP*	*WP*	*WV*	*MV*	*SV*	$S \bullet W^t$
C	0	59	41	2	8	0	53
C	83	17	0	1	5	0	94
T	43	21	0	0	2	7	57
					Total		204
					Total / 3		68

Figure 6: Quality Score. In this example, the window size, n, is 3 bases. We want to calculate the quality score, Q, for the center base, C. Three sets of *Trace-Data Classification* scores, S, have been calculated: one for each of the C traces corresponding to the first two C base calls, and a third for the T trace data associated with the T call. The dot product of each set of scores with the weight vector ($|1 \;.67 \;.33 \;0 \;0 \;0|$) is computed. The average of the three is the quality score for the C base in the center of the window.

We use the same weight vector, W, in a similar manner to summarize the *Trace-Data Classification* scores during consensus computation. Multiplication by the weight vector ensures that scores supplying the most evidence (such as those with high SP scores) are given the more weight than those that supply less evidence. Figure 7 demonstrates this idea.

For each sequence in a column, a vector, E, summarizes the evidence for each possible base (A, C, G, and T). For each base, the computed value reflects the amount of evidence that the call should be that base. The vector is computed as follows.

1. Form a 4x6 matrix of *Trace-Data Classification* scores, S, by computing the scores for each trace:

$$S = \begin{vmatrix} SP_A & MP_A & WP_A & WV_A & MV_A & SV_A \\ SP_C & MP_C & WP_C & WV_C & MV_C & SV_C \\ SP_G & MP_G & WP_G & WV_G & MV_G & SV_G \\ SP_T & MP_T & WP_T & WV_T & MV_T & SV_T \end{vmatrix}$$

2. The transpose of the matrix multiplication of S and W^t produces a vector of evidence values, E, for the possible bases:

$$E = (S \times W^t)^t = | E_A \;\; E_C \;\; E_G \;\; E_T |$$

3. Multiply E by the quality of the local trace data, Q, to produce evidence values, E', that have been adjusted by the quality of the data:

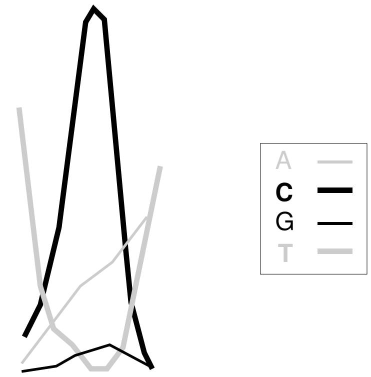
$$E' = E \times Q$$

Finally, we sum the evidence for each base in an aligned column as described next.

Sum corresponding E' values to produce the total evidence, T_i, for each possible base i, where n is number of sequences in the column:

$$\begin{aligned} T_A &= E_{A1}' + E_{A2}' + \ldots + E_{An}' \\ T_C &= E_{C1}' + E_{C2}' + \ldots + E_{Cn}' \\ T_G &= E_{G1}' + E_{G2}' + \ldots + E_{Gn}' \\ T_T &= E_{T1}' + E_{T2}' + \ldots + E_{Tn}' \end{aligned}$$

Once T has been calculated, consensus calling can be completed as described in steps 7-9 of the *Trace-Evidence* Consensus Algorithm. An example determination of a consensus base call appears in Figure 8.



	SP	***MP***	***WP***	***WV***	***MV***	***SV***	$S \bullet W^t$
A	0	4	30	8	1	0	13
C	89	11	0	0	0	0	96
G	3	1	0	0	0	0	4
T	0	0	0	0	13	76	0

Figure 7: Summarizing *Trace-Data Classification* Scores. The *Trace-Data Classification* scores for each of the four traces is computed. When dotted with the weight vector (|1 .67 .33 0 0 0 |), the result is a high value for the C trace – the trace exhibiting highest evidence. The values for the A, G, and T traces are all low. When these summarized values are used to provide evidence, the C trace appropriately has the highest value. Note that in this calculation, the *Trace-Data Classification* scores are computed for each of the four traces in contrast to the calculation of the quality measure (Figure 6) in which only the scores for the trace associated with the called base are computed. Here, we need to know how much evidence each trace supplies.

Testing

All code for testing the new consensus calling method was incorporated into an experimental version of the DNAStar Inc. *SeqMan* fragment assembly program for the Apple *Macintosh PowerPC*. *SeqMan* uses the *Majority* consensus calling method. (*SeqMan* has since been superseded by *SeqMan II*, a more powerful version that incorporates trace analysis as described in this and a previous paper (Allex et al. 1996).)

Method

Fragment assemblies for a 124 kb section of *E. coli* are used to compare correct calls to *Majority* and *Trace-Evidence* calls. The data and correct calls for the assemblies were supplied by the *E. coli* Genome Project at the University of Wisconsin. The original assembly of

2221 ABI sequences ranged in coverage from one to 45 sequences. In order to generate an abundance of test cases with varying amounts of coverage, we developed and applied a minimization algorithm, *Minimize Coverage*, to the assembly.

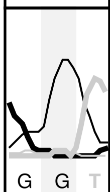
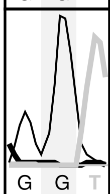
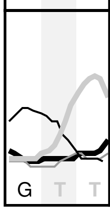
Sequence	*Q*	*E*				*Q* x *E*			
		A	*C*	*G*	*T*	*A*	*C*	*G*	*T*
	.51	0	0	52	0	0	0	26.5	0
	.82	0	0	73	0	0	0	59.9	0
	.26	0	0	21	5	0	0	5.4	1.3
					Total	0	0	91.8	1.3

Figure 8: Trace-Evidence Consensus Example. The consensus base for the center column of three aligned sequences must be called. For each sequence, the evidence for each base is multiplied by the corresponding quality score. When these products are summed for the three sequences, the evidence for *A* and *C* is 0, for *G* is 91.8 and for *T* is 1.3. If the threshold is .50, *G* will be called unambiguously since no competing evidence surpasses 45.9 (91.8 x .50). In contrast, the *Majority* method with a 75% threshold would make an ambiguous call of *K* (*T* or *G*).

With *Minimize Coverage*, sequence fragments are removed from an assembly such that the coverage for any single column does not fall below a specified coverage (unless the coverage is already below the threshold). The idea for the *Minimize Coverage* algorithm is simple. At each pass through the assembly, for each sequence we determine the lowest coverage, *low-coverage*, of any column in which the sequence occurs. We then remove the sequence with the highest *low-coverage*, provided that *low-coverage* is not at or below the threshold. If more than one sequence has the same *low-coverage*, the shorter one is removed. Passes over the assembly are repeated until no more sequences can be removed without violating the coverage threshold restriction. At completion, some columns will have more than the desired coverage (due to the restriction) and some less. The algorithm is summarized next.

Minimize Coverage Algorithm

Let S be the list of all n sequences, S_i, in the assembly.

$$S = \{S_1, \ldots, S_n\}$$

Let L be the list of all n sequences considered for removal. Each sequence, S_i is paired with its *low-coverage*, LC_{Si}.

$$L = \{(S_1, LC_{S1}), \ldots, (S_n, LC_{Sn})\}$$

While *not_empty(L)*

1. Remove from L sequences whose *low-coverage* is at or below the threshold.
2. Remove from S and L the shortest sequence with the highest *low-coverage*.
3. Update *low-coverage* values.

Figure 9 steps through an example execution of the *Minimize Coverage* algorithm.

We repeatedly applied the *Minimize Coverage* algorithm to the original assembly for the range of coverage thresholds from two to ten. This produced nine assemblies with differing coverages, each with an abundance of aligned columns whose coverage corresponded to its threshold. For testing, from each of the nine minimized assemblies, we extracted the statistics for consensus calling only for columns that corresponded to the coverage threshold. For example, for the assembly with a minimum coverage threshold of three, we compiled statistics only for those columns with a coverage of three sequences. The exception is that the statistics for the assembly with the desired coverage of ten include all columns with coverage of ten or greater (rather than just those with exactly ten) since results tend to remain constant with such high coverage. Table 1 lists the number of consensus calls used for each set of results.

(a)

S_1	**GATCGGCTACATCTTACATCACCGTT**
S_2	CTACATCTTACATCACC
S_3	**CGGATCGGCTACATCTTACATCACCGTTGA**
S_4	ATCGGCTACATCTTAC
S_5	ATCTTACATCACC
S_6	CGGCTACATCTTACATCACCGT

(b)

Pass	S	L
0	$\{S_1, S_2, S_3, S_4, S_5, S_6\}$	$\{(S_1, 2), (S_2, 5), (S_3, 1), (S_4, 3), (S_5, 5), (S_6, 3)\}$
1	$\{S_1, S_2, S_3, S_4, S_6\}$	$\{(S_2, 4), (S_4, 3), (S_6, 3)\}$
2	$\{S_1, S_3, S_4, S_6\}$	$\{(S_4, 3), (S_6, 3)\}$
3	$\{S_1, S_3, S_6\}$	$\{(S_6, 3)\}$
4	$\{S_1, S_3\}$	$\{\ \}$

Figure 9: Minimize Coverage Example. (a) Six sequences, S_1 to S_6, are aligned in a fragment assembly. The sequences in bold, S_1 and S_3, provide the optimal minimization when the threshold is set to two. With these two sequences in the assembly, no column has fewer than two sequences (except those that already had fewer in the original assembly). In addition, neither sequence can be removed without causing coverage to fall below the minimum. (b) The algorithm to reduce coverage on the assembly completes after 4 passes. At the outset, all sequences are in S and L. The first pass removes S_5 from both lists since it is the shorter of two sequences with the highest *low-coverage* (5 sequences). Also, S_1 and S_3 are removed from L in the first pass since their *low-coverage* is at or below threshold – these sequences cannot be taken out. At the end of four passes, L is empty and the two desired sequences, S_1 and S_3, remain in the assembly.

Results

We report results that compare correct consensus calls to those made by *Trace-Evidence* and *Majority* for coverage from two to ten or more sequences. The threshold is set to the *SeqMan II* default value of 75% for *Majority* and to 50% for *Trace-Evidence*. Graphs in Figure 10 display the number of correct calls, incorrect calls, and ambiguous calls per kb for the two methods.

The results show a significant improvement with the *Trace-Evidence* method, especially at lower coverages. With a coverage of only three, using *Trace-Evidence* we see a leveling of the number of incorrect calls and a large improvement over the *Majority* method in the number of correct and ambiguous calls. With a coverage of four, the number of ambiguous calls has fallen to nominal values with *Trace-Evidence*.

Discussion and Future Work

We observe striking examples of the utility of the *Trace-Evidence* method when base calls in a column are systematically incorrect. In some instances, a well-defined peak is hidden below a high-intensity valley. The base is often incorrectly called as the one associated with the high-intensity valley. *Majority* methods incorrectly call the consensus as this base. Our new *Trace-Evidence* makes the correct consensus call even when all or most of the bases have been called incorrectly. Figure 11 contains an example of this occurrence.

Table 1: Number of Consensus Calls in Test Results. For each coverage from two sequences to ten or more, the number of consensus calls included in test results is listed.

Coverage	Number of Consensus Calls
2	67,860
3	57,092
4	45,394
5	39,556
6	34,011
7	26,716
8	22,479
9	20,326
≥ 10	47,239

We have identified three situations in which *Trace-Evidence* can make incorrect calls. Overwhelmingly, most problems involve gaps. In rarer cases we have difficulties with low evidence sums or poor-quality data. Next, we briefly describe these three sources of incorrect calls.

In the results reported here, all of the incorrect calls at coverages above three and at least half of those for coverages of two or three involve gaps in the column. The method for determining whether a gap should be inserted in the consensus consists of a simple comparison of gap vs. non-gap sums of the quality of the traces in the column. However, the insertion of a gap affects not only the column it occurs in, but also the columns to either side. When determining a gap call, it is probably necessary to consider more context and examine the data on either side of the base of interest. Finding a solution to calling the consensus when gaps are in the alignment would virtually eliminate incorrect calls made with the *Trace-Evidence* method with a coverage of at least four.

In some instances, incorrect calls can be associated with extremely low evidence sums. When the sums are quite low, even the maximum evidence is often not indicative of the correct call. One solution is to label the consensus as an ambiguous *N* and defer consensus determination to human editors. For the results reported in this paper, this is the solution used (ie. low evidence calls are counted in the ambiguous category). To circumvent the low-evidence problem in the commercial version of *SeqMan II*, consensus calling reverts to *Majority* when the maximum evidence is less than ten. (This number was chosen as one that works well in practice.)

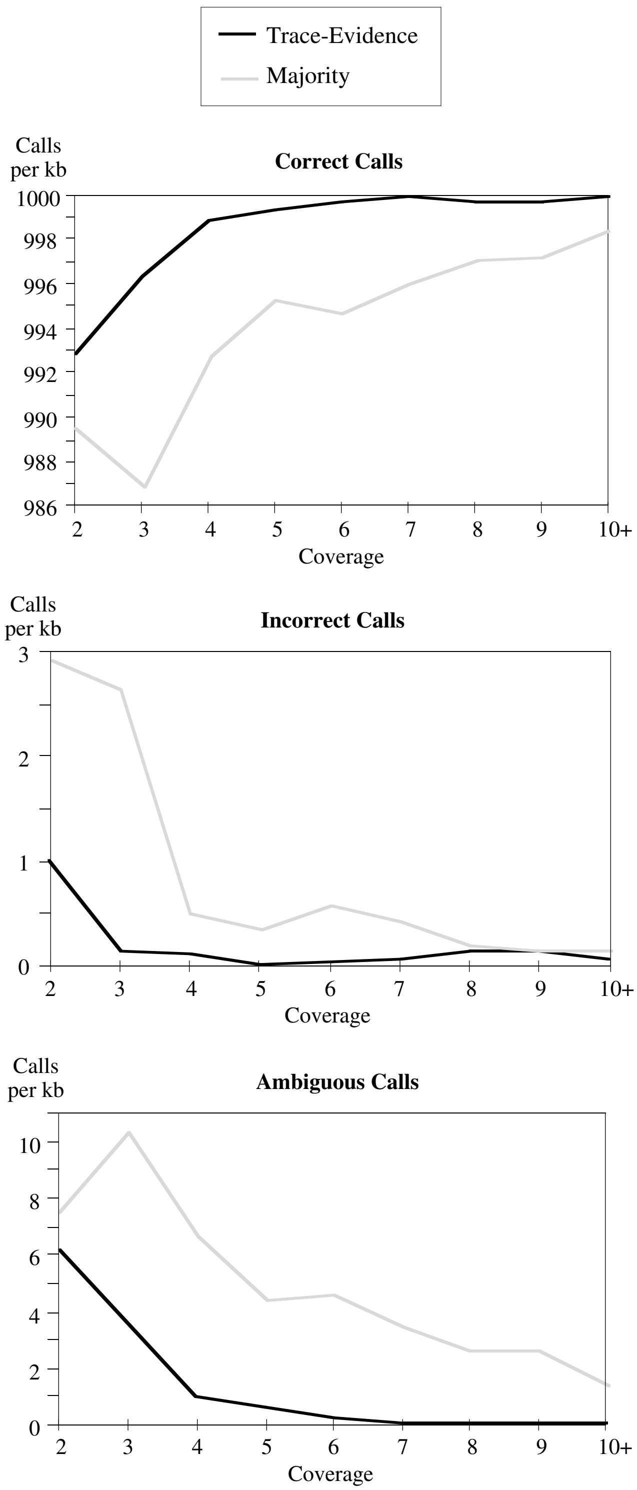


Figure 10: Test Results. Results for calls per kb vs. amount of coverage are graphed. Each data point is based on 20,000-68,000 consensus calls. The new *Trace-Evidence* method produces more correct calls and fewer incorrect and ambiguous calls, especially at low coverages.

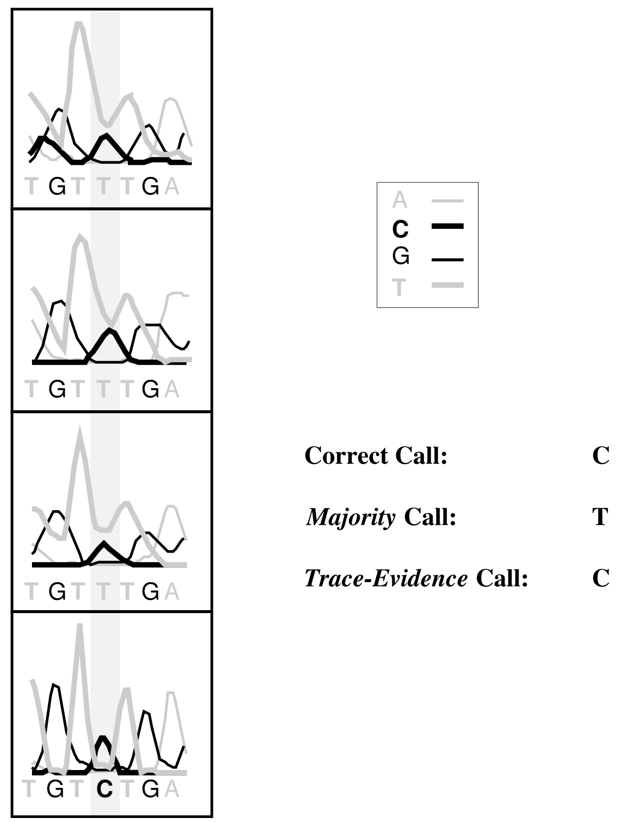


Figure 11: *Trace-Evidence* vs. *Majority* Consensus. In the shaded column, three bases have been incorrectly called as a *T* and one correctly as a *C*. With a 75% threshold, the *Majority* method incorrectly computes the consensus as a *T*. The *Trace-Evidence* method detects no evidence for a *T*, ample evidence for a *C*, and calls the correct consensus. With *Majority* this situation would be even more troublesome if the fourth sequence were not in the assembly. In that case, the call would have no conflicting base calls and would likely go unquestioned during hand-editing. In contrast, *Trace-Evidence* correctly computes a *C*, even in the absence of the fourth sequence.

A few incorrect calls occur in cases that are difficult for both *Majority* and *Trace-Evidence.* These are usually in regions of poorer-quality trace data where peaks are overlapping and ill-defined. The obstacle for *Majority* is that one or more of the base calls is likely to be incorrect in such regions. For *Trace-Evidence* the difficulty lies in the relative locations of the trace peaks. Often the peak associated with the correct base call is significantly offset from the base-call location. The result is that when the *Trace-Data Classifications* are computed, a peak is either not detected or is given a low score due to its distance from the base-call location. Another of the traces may exhibit a small, distinct peak near the base-call location that is scored relatively higher. *Trace-Evidence* then has more evidence associated with the small peak than with the correct trace and calls the consensus incorrectly. This case is illustrated in Figure 12.

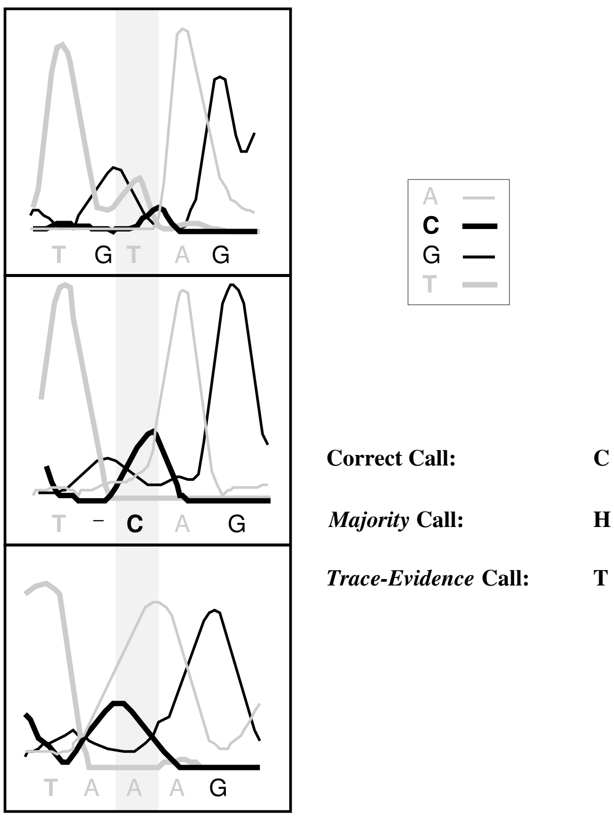


Figure 12: Difficult Consensus Call. Three sequences have been aligned; the correct call for the shaded column is *C*. *Majority* calls an ambiguous *H* for the consensus since the column includes conflicting base calls of *T*, *C*, and *A*. The *Trace-Evidence* method assigns negligible *strong peak* scores to the offset peaks associated with the *C* traces and a high *strong peak* score for the *T* trace in the first sequence. The scores incorrectly sum to adequate evidence for a *T* and insufficient evidence for *C*.

In addition to the occurrence of incorrect calls in the three situations just described, we believe that some errors may be due to the weight vector we chose. The weight vector we used for the calculations in our work ($W = |1\ .67\ .33\ 0\ 0\ 0|$) was chosen empirically from among those listed in Table 2. These vectors conform to the restriction $1 \geq W_{SP} \geq W_{MP} \geq W_{WP} \geq W_{WV} \geq W_{MV} \geq W_{SV} \geq 0$, while varying the emphasis on the scores for different classes. Two of the vectors, described in the table as *linear peaks* and *parabolic peaks*, assign zero values to the three *valley* classes (W_3, W_4, and W_5). We found that these vectors greatly outperformed the others. This observation contradicts the premise that *valley* scores contain useful information. One explanation that reconciles the opposing observation and premise is that these simple functions are not sensitive enough to make use of the information in the *valley* scores. A better approach to finding the weight vector may be a statistical method such as multiple linear regression. In the future, we will compare vectors determined by more rigorous methods such as this to the one we chose empirically.

Table 2: Weight Vectors. Five functions used to generate possible weight vectors are listed. Two of the functions are linear, two are parabolic and one is trigonometric. In two cases, *linear peaks* and *parabolic peaks*, the values for the *valley* classes are all zero. Of the five functions, these two are observed to produce the best fragment assemblies.

Description	Function	$\lvert W_0\ W_1\ W_2\ W_3\ W_4\ W_5 \rvert$
linear	$W_i = \frac{5-i}{5}$	$\lvert 1\ .8\ .6\ .4\ .2\ 0 \rvert$
linear peaks	$W_i = max\left(0, \frac{3-i}{3}\right)$	$\lvert 1\ .67\ .33\ 0\ 0\ 0 \rvert$
parabolic	$W_i = \left(\frac{5-i}{5}\right)^2$	$\lvert 1\ .64\ .36\ .16\ .04\ 0 \rvert$
parabolic peaks	$W_i = \left(max\left(0, \frac{3-i}{3}\right)\right)^2$	$\lvert 1\ .44\ .11\ 0\ 0\ 0 \rvert$
trigonometric	$W_i = \frac{cos\left(\frac{\pi}{5}i\right)+1}{2}$	$\lvert 1\ .9\ .65\ .35\ .1\ 0 \rvert$

As well as improving our algorithmic approaches to sequencing problems, we plan to shift the direction of our research to solutions that involve machine learning. In particular, we plan to use artificial neural networks with *Trace-Data Classifications* to aid in consensus calling and basecalling. In recent years, success in developing neural network solutions for problems in molecular biology has surged. A sampling includes: protein-structure prediction (Rost & Sander 1993), DNA sequence determination (Tibbetts, Bowling & Golden 1994), finding protein binding sites (Heumann, Lapedes & Stormo 1994), and detection of protein-coding regions (Uberbacher & Mural 1991). Neural networks often provide a good solution to biological problems such as these since they involve intricate interactions, and the strength of neural networks lies in their ability to learn to recognize complex patterns.

Conclusions

The overall goal of our work is to improve the quality and efficiency of automatic fragment assemblies. Toward this goal, we have developed a new method for consensus calling, *Trace-Evidence*, that produces more accurate consensus sequences, thereby reducing hand-editing and decreasing the amount of coverage needed. We accomplished this by direct incorporation of trace information into automatic consensus calling via the *Trace-Data Classifications* developed in prior work. In contrast to our new method, less accurate methods use only a limited representation of trace data – base calls – to determine the consensus.

Acknowledgments

This research was supported by National Research Service Award 5 T32 GM08349 from the National Institute of General Medical Sciences and by National Science Foundation grant IRI 95-02990.

Appendix: Sequencing Background

To determine the sequence of bases in a large segment of DNA, a scientist must first produce small, overlapping fragments of the segment, sequence each of the small fragments, and finally align the overlapping regions of the small fragments to establish the overall sequence of the large segment. (The large segment must be broken into smaller fragments since instruments can sequence only fragments of less than one kilo base (kb) in length and the large segments are generally much longer.)

With the Applied Biosystems Inc. (ABI) 377 and other modern sequencers, determining the sequence of the small fragments is made possible by fluorescent-dye labeling (Ansorge et al. 1986, Smith et al. 1986). Fragments of DNA are labeled with one of four dyes (one for each base) that identifies the 3' base in a fragment.

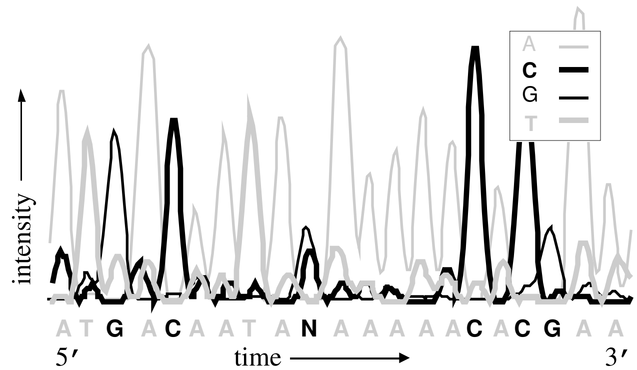


Figure A1: Fluorescent Trace Data. The intensities in four sets of trace data as they vary with time are graphed. Below each peak in intensity is the corresponding base call made by a basecalling program. In general, the base call corresponds to the trace with the highest intensity. Near the center of the graph is an instance where more than one intensity is relatively high – this base is labeled as an *N*.

To prepare to sequence a fragment, sets of labeled sub-fragments are produced such that the 5' ends are identical to the 5' end of the fragment and the 3' ends vary so that all possible sub-lengths of the fragment are represented. The sets of sub-fragments are placed on one end of a gel and an electric current is applied. The current causes the (nearly) linear migration of the DNA across the gel. The sequencing machine scans and records in a computer file the intensities of each of the four dyes as the sub-fragments migrate past the other end of the gel. Since shorter fragments migrate faster than longer ones, the intensities are recorded in the 5' to 3' sequence order of the original small fragment.

A computer program processes the file of intensity data (called *trace* data) to determine the sequence of bases in the original fragment. This process is called *basecalling*. In the simplest case, at any one time the intensity for one particular dye is relatively high, and this identifies the base. If more than one intensity is high, the base is unknown and is labeled with an *N* for *no-call*. Figure A1 contains an example of the 2-D graphs of four sets of trace data and the corresponding base calls.

Table A1: Possible Consensus Calls. A consensus call may be one of the four bases, a gap, or one of 11 ambiguities (combinations of *A*, *C*, *G*, and *T*).

A	*adenine*	***R***	*G or A*	***H***	*not G*
C	*cytosine*	***Y***	*T or C*	***B***	*not A*
G	*guanine*	***M***	*A or C*	***V***	*not T*
T	*thymine*	***S***	*C or G*	***D***	*not C*
		K	*G or T*		
N	*A, C, G, or T*	***W***	*T or A*	**–**	*gap*

When the small fragments have been sequenced, their overlapping regions are aligned in a *fragment assembly*. The sequence for the original segment of interest is the consensus of the aligned small fragments. The problem of determining the consensus for each column in the fragment assembly is called *consensus calling*. In an aligned column, there may be total agreement of base calls or some calls may conflict with the others. A decision must be made to call the consensus as a base (*A*, *C*, *G*, *T*), as a gap (indicating an insertion in one of the sequences), or as one of 11 ambiguities (combinations of *A*, *C*, *G*, and *T*). Table A1 lists possible consensus calls and Figure A2 illustrates an alignment of fragments and a corresponding consensus sequence.

Consensus	…TGCMACGATCTATTGGK-TAAG…
Aligned Fragments	…TGCCACGATCT
	…TGCAACGATCTATTGGT-TAAG…
	…TGCAACGATCTATTGGT-TAAG…
	CGATCTATTGGGNTAAG…

Figure A2: Consensus Calling. Four fragments are aligned by their overlapping regions. Across the top is the consensus that has been computed for the fragments. In addition to *A*, *C*, *G*, and *T* calls, two ambiguous calls (*M* and *K*) and a *gap* call have been made.

References

Allex, C.F., Baldwin, S.F., Shavlik, J.W., and Blattner, F.R. (1996). Improving the quality of automatic DNA sequence assembly using fluorescent trace-data classifications. In States, D.J., Agarwal, P., Gaasterland, T., Hunter, L. and Smith, R., eds., *Proceedings, Fourth International Conference on Intelligent Systems for Molecular Biology,* 3-14. St. Louis, MO: AAAI Press.

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.

Bonfield, J.K., Smith, K.F. and Staden, R. (1995). A new DNA sequence assembly program. *Nucleic Acids Research*, 24:4992-4999 .

Dolz, R. (1994). GCG: Fragment assembly programs. *Methods in Molecular Biology*, 24:9-23.

Lawrence, C.B. and Solovyev, V.V. (1994). Assignment of position-specific error probability to primary DNA sequence data. *Nucleic Acids Research,* 22:7.

Heumann, J.M., Lapedes, A.S. and Stormo, G.D. (1994). Neural networks for determining protein specificity and multiple alignment of binding sites. In Altman, R., Brutlag, D., Lathrop, R. and Searls, D., eds., *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, 188-194. Bethesda, MD: AAAI Press.

Rost, B. and Sander, C. (1993). Prediction of protein secondary structure at better than 70% accuracy. *Journal of Molecular Biology,* 232:584-599.

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. (1982). Automation of the computer handling of gel reading data produced by the shotgun method of DNA sequencing. *Nucleic Acids Research,* 10:4731-4751.

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

Tibbetts, C., Bowling, J.M. and Golden, J.B. III. (1994). Neural networks for automated basecalling 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.

Uberbacher, E.C. and Mural, R.J. (1991). Locating protein-coding regions in human DNA sequences by a multiple sensor-neural network approach. In *Proceedings of the National Academy of Science USA,* 88:11261-11265.