An adaptive, object oriented strategy for base calling in DNA sequence analysis

Michael C.Giddings, Robert L.Brumley Jr, Michael Haker and Lloyd M.Smith*

Chemistry Department, University of Wisconsin, 1101 University Avenue, Madison, WI 53706, USA

Received June 18, 1993; Revised and Accepted August 18, 1993

ABSTRACT

An algorithm has been developed for the determination of nucleotide sequence from data produced in fluorescence-based automated DNA sequencing instruments employing the four-color strategy. This algorithm takes advantage of object oriented programming techniques for modularity and extensibility. The algorithm is adaptive in that data sets from a wide variety of instruments and sequencing conditions can be used with good results. Confidence values are provided on the base calls as an estimate of accuracy. The algorithm iteratively employs confidence determinations from several different modules, each of which examines a different feature of the data for accurate peak identification. Modules within this system can be added or removed for increased performance or for application to a different task. In comparisons with commercial software, the algorithm performed well.

INTRODUCTION

One of the important goals of the Human Genome Initiative is the development of technologies for automated, high-speed DNA sequencing. Such technologies are essential for sequencing the entire genome. A significant part of this development is in the area of data handling and analysis.

Currently, most DNA sequencing is performed by polyacrylamide gel electrophoresis in manual or semi-automated systems (1). In semi-automated systems, data is collected through a detection apparatus that yields an analog signal. The signal represents the time varying concentration of fluorescently-labelled DNA fragments travelling through a detection region. The base calling process consists of determining the base sequence from this analog signal.

In conventional radioisotopic DNA sequencing, base calling is generally performed by a human reading autoradiograms of the sequencing data. A human gel-reader is exceptionally good at performing the complex pattern-recognition needed to correctly analyze base sequence from such data. This may be aided with a scanning device, but nonetheless, the process is tedious and time-consuming. The automation of this process is an important aspect of sequencing technology, particularly for high speed fluorescence based sequencing instruments. Signal variability and noise present in the analog data can make it difficult to produce

a computer algorithm that provides accurate base calls (2). The algorithm must be able to recognize real fragment peaks of varying height and shape in the presence of false peaks due to noise and other spurious signal variations.

Most of the previous work in this area consists of gel-scanning software for reading autoradiographic films (3,4) and software that has been implemented for use in fluorescence-based commercial instruments, such as the Applied Biosystems Incorporated (ABI) 373A [Applied Biosystems, Foster City, CA] or the Pharmacia Alf [Pharmacia, Uppsala, Sweden]. Although the latter commercial software is quite effective, the lack of any published description of its operation has impeded progress in the development of improved software or new algorithms for this and related tasks. It is necessary for these algorithms to be in the public domain for work to proceed on their integration into higher level aspects of sequence data handling such as fragment assembly, cross-checking, and the assignment of confidence values of finished data.

In this paper, a data analysis program is described which has an object-oriented, modular algorithm as the foundation of the base calling process. The program can analyze data from a variety of DNA sequencing instruments, rapidly and accurately base label the fragments, and provide confidence levels on each of these base calls. The base calling algorithm works independently of any pre-defined calibrations to specific instruments or data collection methods.

Object-oriented programming has been used here due to the modularity and extensibility provided by this technique. Object oriented programming is a paradigm that promotes the modularization of code through the use of 'objects' (5). An object consists of code and data related to one task within a program or algorithm. Objects are arranged in a hierarchy of inheritance so that objects lower in the hierarchy 'inherit' all the functionality and features of ancestors. This makes it simple to develop sets of objects that have common features due to a shared ancestor, yet add their own specific functionality to the inherited features for a particular application. Additionally, objects can be re-used for other purposes, or new objects developed to replace old ones.

This object-oriented modularity was utilized to create 'feature detectors', each of which is an independent object that analyzes a particular feature of the analog data to aid in base identification. For example, we have created detectors that examine the features of peak height, peak spacing, and peak width. Other feature detectors can easily be added, or the current ones replaced. This

* To whom correspondence should be addressed

framework is highly flexible and can potentially be used for tasks besides base calling with the correct modules in place.

The algorithm presented here is based on some of the concepts used in Blackboard systems (6). These systems are an artificial intelligence technique which use independent 'knowledge sources', each with their own expertise, to collaborate on solving a problem. Blackboard methods were originally conceived for pattern analysis tasks such as voice recognition, and therefore seem well suited to pattern recognition tasks like base calling.

An important aspect of this work is the assignment of confidences to each base label. This is one step toward improving the speed and accuracy of the fragment alignment process (7). Low confidence labels indicate regions that may warrant further examination by a human due to some problem with the analog data quality, and if the confidences are carried through the fragment assembly process, they can be used to indicate the quality of sequences that are stored in public databases such as GenBank.

In this paper, we give an overview of the data collection and handling, the pre-processing that occurs before base calling, and a description of the algorithm developed for base calling. In addition, data quality and the value of providing confidence values with base labels is discussed.

Data collection and preparation

In the four-color strategy, DNA fragments are labelled with one of four fluorophores depending upon the nucleotide at which they terminate (A, C, G, or T). This allows all four reaction mixtures to be electrophoresed concurrently in one lane on a gel. Data is collected with a detection system at four distinct wavelengths to allow determination of the concentration of each dye present in the detection region at any time during electrophoresis (8).

Four color fluorescence based sequence data from three systems was analyzed in this study: a) horizontal ultrathin gel electrophoresis (HUGE) (9,10), b) a capillary gel apparatus (11), and c) a commercial sequencer (ABI 373A). Different optical and electronic configurations of these instruments result in differences in the acquired data. However, in each case, pre-processing software reduces the 'raw data' to a 'trace file' for input to the base calling algorithm. Each trace file consists of four channels of data, each channel representing the time-varying concentration of one of the dyes labelling the four different end terminated fragments. One trace file is generated for each sample lane (or single capillary). The next section outlines the steps we use for producing a trace file from the raw data collected on the HUGE sequencing instrument (10).

Pre-processing the data

In each of the instrument configurations mentioned, data is collected at four wavelengths in the detection region at specific intervals. This produces an 'image' file which is four $n \times m$ of intensities collected at the four different frequencies, where n is the total number of sample intervals during the run, and m is the number of data points collected across the gel in each sample interval. Figure 1 shows a section of an image file where several sample lanes can be seen.

From this type of image file, there are several steps involved to produce each trace file. These steps are: lane finding, data editing, multi-component analysis, noise filtering, dye-mobility correction, and normalization of the data. Figures 2 a-c show representative data at different stages of this pre-processing.

Base 110 - 25 min.

Base 250 - 44 min.

Figure 1. A section of a four-color, fluorescence based gel image file. The vertical axis represents time during the run, and the horizontal axis represents the detection region across the gel.

Lane finding. The lane finding process consists of determining the regions, or 'lanes', in the image file corresponding to samples undergoing electrophoresis. Each lane is seen in the image file as a sequence of colored bands along the time axis as shown in Figure 1.

Lanes are demarcated manually on the gel image file displayed on the computer. Although manual, this process is straightforward. A four channel raw data file is produced for each lane marked. This raw data file represents the signal intensity at all four wavelengths for each sampling interval. Each of the subsequent steps of pre-processing and base calling are applied in turn to each of the raw data files produced through this process.

Data editing. The data produced from the lane finding process begins at the start of electrophoretic separation, well before the primer peak, and often continues to a point at which most or all of the fragments have passed the detection region. It is generally desirable to eliminate those portions of the data that are irrelevant to analysis, which includes the data before the primer peak and data near the end of the run where resolution or sensitivity is low. This is currently a manual process where the user selects these undesirable regions of the raw data file with the mouse and deletes them.

Multi-component analysis. The fluorophores employed in the four-dye sequencing strategy have distinct emission spectra. These spectra exhibit significant overlap (8, 12). This necessitates a transformation by multi-component analysis to determine the relative concentrations of the four dyes from the fluorescence intensity measured at four different wavelengths (13).

Every time point in the raw data file consists of intensity values at the four wavelengths being detected. These four intensities at

Figure 2. Three phases in data analysis: **a.** Raw data file after lane finding. **b.** The same data after multi-component analysis and mobility shift correction. **c.** The same data after filtering and baseline cutoff. The $F(f)$ parameters used were $s_1 = 100$ and $s_2 = 60$, with the baseline adjusted to 0.45 of the maximal value for background elimination. The filtering and cutoff level have been overemphasized here to illustrate their effects—typical baseline adjustment and s_2 values are smaller.

each time point can be considered as a vector $\bar{s}$ in 'detector space'. For each vector $\bar{s}$, we wish to determine a vector $\bar{f}$ in 'dye space' which indicates the concentration of each dye at that time point.

This transformation is performed by means of a matrix $\mathbf{M}$. $\mathbf{M}$ is a 4×4 matrix in which each column represents the relative signal intensity at each wavelength in detector space for one of the dyes. The transformation from 'dye space' to 'detector space' is given by

$$\bar{s} = \mathbf{M}\bar{f} \quad (1)$$

For the purposes here, it is necessary to perform the inverse operation, converting the 'detector space' vectors to 'dye space' vectors. To do this, $\mathbf{M}$ is inverted numerically and used in the equation,

$$\bar{f} = \mathbf{M}^{-1}\bar{s} \quad (2)$$

The components of $\mathbf{M}$ may be determined by identifying a known peak in the raw trace (detector space) for each of the four different fluorophores. For each of the identified peaks, the relative signal intensities at the four wavelengths are entered into a column of the matrix $\mathbf{M}$. $\mathbf{M}$ is then normalized, and is readily inverted to obtain $\mathbf{M}^{-1}$.

The entries for $\mathbf{M}$ do not change between runs or samples unless different fluorophores are used or the detection apparatus is modified. Thus once a suitable matrix is found it can be used without modification for subsequent analyses. In our program,

Figure 3. The dual-gaussian bandpass filter with the parameters for the function $F(f)$ being $\sigma_1 = 15$ and $\sigma_2 = 1.5$.

$\mathbf{M}$ is determined by allowing the user to mark and identify four different dye-labelled peaks using the mouse. The intensity values for each of these peaks are used as the entries for $\mathbf{M}$. The program allows easy storage and retrieval of various matrices for different dye-sets. Figure 2 shows representative data before and after multi-component analysis in panels a and b, respectively. A more extensive discussion of multi-component analysis techniques is given in (14).

Noise filtering. For better performance in base calling, noise should be reduced as much as possible. Noise can originate from several sources, including background noise, detector noise

(Johnson noise), and other noise from the operating environment. This aperiodic noise typically has a white spectrum. Another type of noise encountered is low-frequency variation due to changes in the background light level during collection. Such variations might be caused by deformation of the gel due to heating, the formation of bubbles in the path of the laser, or variations in laser output power.

We have experimented with several types of noise-removal techniques, and found an effective bandpass filter utilizing the Fast Fourier Transform that removes both the high frequency noise and some of the low-frequency background variation. This filter was inspired by the zero-area convolution filters proposed by Janssens and Francois for use in filtering spectral data (15).

The filtering process proceeds as follows for each of the four channels of raw data: the data is transformed to Fourier space by means of a Fast Fourier Transform (FFT), the resulting complex array is then multiplied point by point against the filter function below, and finally the complex data is conjugated and the FFT is applied to transform the filtered data back to the time domain. The bandpass filter applied in the frequency domain is the product of two gaussian functions. This function is described by,

$$F(f) = e^{-f^2/\sigma_1^2} \left(1 - e^{-f^2/\sigma_2^2} \right) \quad (3)$$

where σ_2 determines the low-frequency cutoff, and σ_1 determines the high-frequency cutoff respectively. These parameters are chosen based upon the sampling rate and amount of noise. This filter is illustrated in Figure 3. An important aspect of the filter is that it allows a very sharp low-frequency cutoff in conjunction with a more gradual high-frequency cutoff. This is important because the removal of baseline variations requires the elimination of only very low frequency components in the frequency spectrum. The low frequency cutoff needs to be sharp to avoid loss of important information in wider or multiple peaks that may have low frequency components near the cutoff value.

This filter greatly reduces low and high frequency noise, leaving the primary signal in the target frequency range for fragments traveling through the detector. The filtering process creates a new, globally flattened baseline. Locally, there still may be small fluctuations around the baseline value that are due to noise in the bandpass frequency range. These are typically only small fluctuations, and can be simply removed by readjusting the baseline to a slightly higher level. This is done after the data has been bandpass filtered, assuming that relevant signal is not lost in readjustment because low frequency baseline fluctuations have been removed. Figure 2 panels b and c show data before and after this filter has been applied, with the baseline in panel c moved to illustrate the elimination of the remaining background noise. In applying this filter, care must be taken by the user to choose the correct frequency cutoffs so that useful signal information is not lost due to over-filtering. For a given system and set of conditions, a set of filter cutoff points can be chosen and re-used because the filtering requirements do not typically change between runs. The process of choosing these parameters might be automated by the application of Weiner filtering (16) or similar technique.

Mobility shift corrections. The four fluorophores used for labelling the DNA fragments have different electrophoretic mobilities (17). These variations impart different mobilities to

the DNA molecules to which they are attached. This can result in a small offset of peaks in one or more of the raw data file channels with respect to the others.

The dye-mobility shift is nearly linear over large regions (13), and thus can be compensated for by using a small, constant offset in one or more of the channels to produce a correct alignment. Fujii and Kashiwagi (18) have presented an automated technique for determining the necessary mobility shifts. However, when using the same instrument and experimental conditions, the offsets do not significantly vary from one run to the next. Thus we have utilized a simple technique where the user of the program can selectively offset any of the data channels with respect to the others using the mouse, and once a set of satisfactory shifts is settled upon, they are recorded and available for use in any subsequent experiment.

Normalization. Signal strength will generally vary during a run from channel to channel or over time within a channel. The most common effect is a drop in signal strength as a run progresses due to the decreased concentration of the larger fragments. The base calling algorithm is able to compensate for such changes, however, the variations observed are often big enough that normalization of the data before input to the base calling routine brings increased accuracy, as well as easier visual identification of features in the data should that be desired.

For this purpose, a routine is used that performs piecewise linear normalization. The data is divided into several windows of size N, and for each window a scaling factor is found such that the data in the window would be normalized to the range [0,1] if scaled by this factor. If the data in each of the windows was scaled by these factors, discontinuities would be produced where the windows join. To avoid discontinuities, linear interpolation is used to perform the normalization. The normalization factors are each assigned to the center of the window in which they were calculated, and a connecting line is calculated for each adjacent set of factors. Normalization of the data is performed by scaling each data point by the factor indicated from the line that covers the region where the point occurs. For example, a point that is on the edge of two windows (halfway between the centers of the windows) would be assigned a factor that is the mean of the normalization factors for the two windows. A point exactly in the center of a window is scaled by the normalization factor for that window. The size of the window N determines how localized the normalization will be. If N is equal to the total number of points, the whole data set is scaled by a single factor. If N is set to one, the normalization will produce a flat line.

Base calling

The pre-processing produces a finished trace file, for which typical data appears in panel c of Figure 2. The base calling task consists of identifying peaks, determining those that are likely to represent fragments, and assigning a confidence value to the peaks that are labelled as a particular base.

The difficulty in determining the base labels for these peaks is due to several things: peak height variability, noise, poor peak resolution, peak overlap, and peak spacing anomalies due to secondary structure in the DNA fragments.

For base calling purposes, several consistent features of the data can be used to identify likely peaks. Peak spacing exhibits consistent behavior except in regions of GC compressions. The peak spacing typically increases to a maximum and then decreases

Figure 4. The configuration of the object-modules in the program, and flow of data between them.

until the end of the run (see below). This can be modelled over large regions by a polynomial. Individual peak heights are less consistent, but the average height tends to vary continuously (usually decreasing) through a run. Additionally, other features of the data such as peak width, inflection points, shape, etc. can be used to recognize peaks.

The various peak features exhibited in the data highlight the need for a modular system where peak recognition and judgement modules can be interchanged, added or removed. Our base calling system provides a framework for this modularity by using Object Oriented Programming techniques.

A central coordinator unit provides several modules, or 'feature detectors', with a list of potential peaks, and receives from each module confidence judgements on each peak based on one particular feature (Figure 4). The judgements of the modules can be averaged to assign a final confidence to each peak. Each module is implemented as a separate 'Object' using the Objective-C language, and can be added to or removed from the program structure easily and quickly.

There are currently three modules in the system: one based upon peak spacing, another that is based upon peak height, and a third that is based upon peak width. The modules all have the same set of standard procedures (or methods) that can be called by the controller. Thus, a simple list of available modules can be made, and for each module in the list, a standard sequence of procedure calls can be performed. The implementation details for the specific modules are described below.

Algorithm. Figure 5 shows the steps of the algorithm, which are implemented in the controller object. This iterative process includes the steps of: peak finding, module calibration, peak assessment, confidence assignment, and peak filtering.

The algorithm starts with the creation of a peak list by finding all maxima present in each of the four channels of the trace file. The maxima are sorted in order of occurrence with respect to location (time), and a list is created of all the maxima specifying their location and the channel in which they occur. The algorithm

Figure 5. Outline of the base calling algorithm.

then goes through this list iteratively, calculating confidences for each one by utilizing the judgements of the different feature recognition modules, and eliminating those peaks that are below a specified confidence cutoff value.

In the first step of each iteration, the peak list and trace data are provided to each of the recognition modules to allow any calibration or configuration necessary for the modules based on this data. Depending on module design, this can be an important step because the calibration determines, at run time, what the parameters are for the feature of the data that the module is examining. For example, the spacing module determines at this step the distribution of peak spacing versus time in the trace being analyzed. This avoids pre-calibrating the algorithm to data from a specific instrument or set of run conditions. The module calibrations are expected to be more accurate on each successive iteration because unlikely peaks have been filtered out in the

previous iterations, leaving only strong, likely peaks for calibration. The calibration step may be utilized by different modules in various ways, and some may not use it at all. However, for all of the modules we have thus far developed, this step is essential. The specific calibration techniques used by the current modules are described in the next section.

After the calibration step, the modules provide confidence judgements for specified peaks based upon their design and the calibration determined. The algorithm utilizes these judgements by assessing each peak in the list in turn. The confidence for each peak is given by the geometric mean of the individual assessments of the modules, as expressed by,

$$c_{avg} = \frac{N}{\sum_1^N \frac{1}{c_i}} \quad (4)$$

where N is the total number of modules with judgements to incorporate. We have modified this equation to allow for different weighting of the input from the different modules. This allows the placing of more or less emphasis on the input of a module based on its accuracy or other factors. Given a set of module weightings m_i ranging between zero and one, this modified geometric mean can be expressed as,

$$c_{wgtavg} = \frac{\sum_1^N m_i}{\sum_1^N \left(\frac{1}{c_i} m_i \right)} \quad (5)$$

This equation is the same as (4) in the case where all module weightings are one, and for a module with a weighting of zero, the confidence assessment is unaffected by any of that module's input.

Once the confidence judgements are made for each peak in the list, peak filtering occurs based on a user-settable parameter for the minimum confidence level at which potential bases are to be discarded. After the low confidence peaks are discarded, an average confidence is calculated for this iteration. Iteration will continue depending on several conditions:

1. A user setting for the maximum number of iterations,
2. A user settable average confidence has been reached, or,
3. No change has occurred in the average confidence from the last iteration.

The iterative aspect of the algorithm provides its 'adaptive' nature. The algorithm starts with no pre-defined knowledge about the spacing, size, or other features of peaks in the trace file, and through successive iterations, refines its estimation of each feature in the module calibration step. This allows the utilization of information from peaks that are easily identifiable in the trace file to guide labelling in areas where peak distinguishability is poor.

Modules. Each of the individual feature-detecting modules is implemented in a common framework. Each has a set of standard 'methods' that it responds to so that the controller object can direct each without regard to particular implementation. This way, new modules can be developed and easily implemented. The standard set of methods include instructions for performing the calibration step as well as for determining confidence judgements for peaks in the trace file.

Figure 6. A typical spacing calibration for a HUGE/CCD sequencing run. The points shown are the actual peak-spacings derived from the trace file in this example, and the curve is a polynomial least-squares fit to the spacing data. The polynomial for the curve is: $\Delta x(x) = 12.13 + 0.0020x - 3.01 \times 10^{-7}x^2$.

The three modules currently implemented are similar in design. Each uses a fitted polynomial for calibration, and assigns confidences based upon the polynomial determined. Because the modules are similar, the first will be described in more detail as an example of the framework used, with the descriptions of the other objects being briefer, noting only their particular features. The spacing object will be described first, as it is the most complicated of the modules.

Peak spacing module. The spacing module utilizes information about the expected spacing between successive fragments in the run. A polynomial is used to model the spacing of fragments in each run, which is determined by means of a least squares fit. An array of values is created with entries of Δx_i versus x_i , where Δx_i represents the spacing between peak i at location x and the adjacent peak $(i+1)$ in any of the four channels, which is used for the least squares fit. Figure 6 shows a typical curve obtained from the HUGE system employed in our lab (10). In this curve the maximal spacing between peaks is at about 4400 points into the run (36 minutes). At this location, the average space between adjacent peaks is 14 points (7 seconds).

Noisy data can cause problems with the peak spacing calibration due to the low-selectivity of the initial peak finding process. Small peaks due to noise and large ones due to real fragments are initially included in the peak list. If there are too many peaks due to noise, the spacing calibration curve can be thrown off. This indicates the importance of adequately filtering the data beforehand. Because the bandpass filtering process is straightforward and fast, it is the first choice for eliminating this problem. However, two further additions have been made to compensate for cases where filtering does not adequately eliminate spurious peaks.

Standard least squares techniques allow that an error value be placed on each point in the data set for which a curve is being fit (16). In performing the least squares fit here, an error term is placed on each data point Δx_i . In every iteration after the first, its value is taken as the inverse of the confidence value for the peak x_i obtained in the prior iteration. For the first iteration, this confidence is set to be proportional to the height. Because the least squares algorithm weights points with smaller error terms more heavily in curve fitting, this results in more likely (stronger) peaks having more weight in determining the spacing curve. The second compensation for spurious peaks is the utilization of second-order spacing information (higher orders can be included

Figure 7. The base calling algorithm applied to data from three instruments sequencing the M13mp19 vector. The region shown in each is approximately from fragment 83 to 165 in each of the following: a. Ultrathin slab gel system, conditions: 4% polyacrylamide, 7.5M Urea, 40°C constant temperature, 3000V constant, 0.5 sec. sampling period. b. Capillary gel electrophoresis system, conditions: 6% polyacrylamide, 8.3M Urea, 40°C constant temperature, 30kV constant, 0.5 sec. sampling period. c. ABI system, conditions: 6% polyacrylamide, 7.8M Urea, 30W constant power, 6 sec. sampling period.

as well, if necessary). Second order spacing is the spacing between a given peak (i) and the second peak following it ($i+2$). Including this information helps in the case where there are many spurious peaks. The calibration step is then performed using both first and second order spacing values to arrive at a reasonable estimate of the real underlying spacing curve. Once the spacing calibration curve is obtained for the current iteration, spacing confidences are assigned by

$$C_i^{\text{spacing}}(x) = 1 - \frac{\sqrt{c_{i-1}(\Delta x_{i-1} - \Delta x^{\text{fit}}(x))^2 + c_{i+1}(\Delta x_i - \Delta x^{\text{fit}}(x))^2}}{\sqrt{2}\Delta x^{\text{fit}}(x)} \quad (6)$$

In this equation, x is the location of the peak i , Δx_{i-1} and Δx_{i+1} are the spacings between this peak and the nearest neighbor on the left or right, c_{i-1} and c_{i+1} are the confidences of the peaks to the left and right, respectively, and $\Delta x^{\text{fit}}(x)$ is the target spacing for this location according to the calibration polynomial. This equation penalizes peaks for how far they fall away from the expected spacing, utilizing the confidences of the adjacent peaks so that low confidence neighbors do not penalize for incorrect spacing as much as high confidence neighbors.

It has been observed that different adjacent combinations of the four types of end-terminated fragments result in different spacing values between the fragments during separation (19). Though these variations are small and do not significantly affect this algorithm, it is possible to modify the spacing module to recognize the variations in spacing due to adjacent base combinations and to modify its confidence judgements accordingly. This would likely provide a further improvement in the accuracy of the algorithm.

Peak-height module. The peak-height module operates in a similar manner. The purpose of this module is to judge peaks based upon their height as well as the amount of overlap they have with signal occurring in the other channels. For calibration, this module determines, for each of the four channels, a polynomial which represents the peak height variations as a function of x . This compensates for weak signals in a particular channel due to reaction or detection problems, signal or reaction strength variations within a channel during the run, and differences in scaling between channels due to different fluorescent properties of the dyes. To increase the likelihood of calibrating only to real, fragment-based peaks, it is possible to utilize the second order maxima and minima for this process (20). Second order maxima are the points at which the list of maxima reaches a maximum.

Table 1. Base calling parameters.

Parameter	Default	Description
High Order	2.00	Integer indicating highest order of peak spacing information for the spacing calibration.
Height Weight	1.00	The weight to place on the input of the height module.
Spacing Weight	1.00	The weight to place on the input of the spacing module.
Width Weight	1.00	The weight to place on the input of the width module.
Quit Confidence	0.90	The average confidence level at which to cease iteration.
Maximum Iterations	10.00	The maximum number of iterations allowable before stopping.
Throwout Confidence	0.15	During iteration, the confidence below which base calls are discarded.
Final Out	0.45	After completion of iteration, the minimum confidence for bases to be included in the finished base list.
Beginning Width Tol.	1.70	At the beginning of the run, how wide a peak must be compared to width calibration before insertion of extra base calls.
End Width Tolerance	2.00	At the end of the run, how wide a peak must be compared to width calibration before insertion of extra base calls.
Spacing Tolerance	2.10	How big a space between fragments can be compared to the spacing before base insertion.

Except for the width and spacing tolerances for base insertion, the default values shown were used for all data reported. The width and spacing tolerances depend more closely on the length of the data set and the resolution, needing to be fine tuned for specific conditions to achieve optimal performance.

Once a representative height curve is found for each channel, height confidences are assigned by the equation

$$c_i^{size}(x) = \frac{s_i(x)s_i(x)/t_i(x)}{t_i(x) \sum_{j=1}^N \frac{s_j(x)}{t_j(x)}} \quad (7)$$

where N is the number of channels of data (typically four), i is the channel in which the peak being judged occurs, x is the location of the peak $s_i(x)$, is the peak signal height, and $t_i(x)$ is the expected height for a peak at that location in channel i from the calibration. This equation gives the highest confidence value to a large peak that occurs without any significant signal overlap with signal in other channels. The term $s_i(x)/t_i(x)$ is squared because if this appeared only as a first order term and there was no signal present in any of the other channels at this location, a confidence value of one would result no matter what the peak height. The second $s_i(x)/t_i(x)$ term effectively judges the peak's relative height independently of the amount of signal present at that location in the other channels. One of the two factors can be seen as a 'relative' peak height judgement (relative to the signal in the other channels), and the second as an 'absolute' peak height judgement.

Peak width module. The third module utilizes half-height peak widths for calibration and confidence assignment. The width of fragment peaks versus time follows a continuous curve for any sequencing run, generally increasing with time. The calibration step consists of fitting a polynomial to the half-height peak-width versus x. The width of all peaks is judged by a half-height algorithm which calculates the width of the peak at the height half way between the maximum and the minimum on each side of the center of the peak. Using the minimum and maximum as reference points causes the width calibration to be sensitive to resolution, which is desirable. It penalizes peaks with poor resolution by assigning a decreased width measurement, while assigning a larger width value to peaks that are closer to baseline separated.

Large peaks consisting of several fragments that are not individually distinguishable might affect the calibration, causing it to be weighted towards peak width values that are too large. To compensate for this, the average peak width is determined from the width measurements of all the peaks, and those which are more than 1.5 times the average are counted as two or more peaks, with the measured width being divided among them. The calculated width measurements are then entered into an array, and a curve is fit for width (w) versus data-point number (x). After calibration, confidences for peaks based on width are assigned by

$$c_i^{width}(x) = \frac{w_i^{actual}(x)}{w^{expected}(x)} \quad (8)$$

where i is the channel in which the peak being judged occurs, and x is the location of the peak along the data stream, $w_i^{actual}(x)$ is the calculated width of peak i, and $w^{expected}(x)$ is the expected width of peak i as determined by the calibration polynomial. For cases where $w_i^{actual}(x) > w^{expected}(x)$, one is returned.

Producing the finished sequence. After the iterative portion of the algorithm is completed, a final step must be performed to insert bases in appropriate locations where no base was called because of low signal strength or other problems. Peaks often need to be inserted late in the run where the decreased resolution produces large peaks that actually consist of several fragments in a row. To determine how many individual fragments a given peak represents, the calibration curve from the width object is used, and appropriate bases are inserted above a user settable threshold for maximum width tolerance.

In the case of gaps due to GC compressions or areas where base calling could not initially be determined with reasonable confidence, the calibration from the spacing object is used to determine where bases or 'no-calls' must be inserted to avoid gaps in the sequence. Once again, there is a user settable parameter that gives a tolerance threshold above which bases are inserted. For example, if this threshold is set at 1.8, then the algorithm will insert a base at a particular location if the gap between successive bases is more than 1.8 times the expected spacing at that location.

After this step, a final base labelling is produced, along with a confidence value for each base and an average confidence level for the entire sequence. This sequence can then be used in fragment assembly or for other purposes.

Setting parameters. The base calling algorithm requires that the user set several parameters for operation (Table 1). There is a standard set of parameters that works well for most data presented to the program so far. These parameters can be further optimized to increase performance on data from any particular instrument. For the experiments performed and reported here, a set of parameters was determined that worked well for data from the ABI instrument. This set of parameters was used for all ABI experiments reported. Sets of parameters were also determined which work well for the ultrathin slab gel system and the capillary system. Though these are relatively simple parameters conceptually, it is important that they are chosen well for the best performance of the algorithm. In practice, it is easy to find a set that works for data collected under given experimental

conditions. These parameters then do not need to be adjusted unless the experimental conditions change.

RESULTS

Data from three sources has been utilized to test the base calling algorithm. Figure 7 a–c shows base called data from three different systems respectively: our current ultrathin slab gel instrument (10), our capillary electrophoresis system (11), and an ABI 373A commercial DNA sequencing instrument, respectively. In each of the panels, the same section of sequence data obtained from the sequencing sequencing vector in M13mp19. Though this is the same fragment section, there are a number of things that vary such as: the number of points sampled, the time-scale, the resolution, and the sensitivity. This illustrates the similar performance of the algorithm on data that varies in source and quality.

To quantitatively measure the performance of this system, we analyzed nine sequencing reactions that were separated on an ABI 373A, and compared the performance of the commercial base-calling algorithm to ours. For this comparison, the single stranded phage vector M13mp19 was utilized in each reaction. The separations were performed and data collected utilizing the commercial instrument. All pre-processing was performed by the ABI software so that the same trace files were produced for base-calling by both our program and the ABI program. Base calling was performed with a single set of parameters for our program, and with the standard settings for the ABI software.

The resulting base sequences were compared to the published M13mp19 sequence in GenBank using the 'bestfit' routine in the Genetics Computer Group Sequence Analysis Software Package [Genetics Computer Group, Madison, WI]. This routine uses the Smith and Waterman local homology algorithm (21) to find the regions of best homology between two sequences. One metric given by this routine to judge how well two sequences match is called the *quality ratio*. This is given by,

$$\text{Qual Ratio} = \frac{\text{Matches} - (0.9 \times \text{Mismatches} + \text{Gap count} + \text{Sum of gap lengths})}{\text{Sequence length}} \quad (9)$$

Each corresponding fragment sequence produced by the ABI software and our own algorithm were compared for fit to the published sequence. Figure 8 shows the average of these comparisons over the nine samples for alignments from 100 to 500 bases in length. Due to decreased resolution and other factors, data quality had deteriorated significantly in all runs by the 500 base mark. Though on average our algorithm performed better for all lengths of fragment aligned, it is interesting to note where the difference is more pronounced.

In the alignment to 100 bases, both algorithms had a lowered quality ratio due to anomalies such as GC compressions and mobility shifts that occurred in this region. However, because the overall data quality was high excepting these anomalies, the modules in our algorithm were able to make very good calibrations in this region, providing for more accurate base calling. As the graph shows, alignment quality climbs for both programs to a maximum somewhere between 200 and 300 bases, and then decreases rapidly. This is where our algorithm drops in performance to a level similar to the ABI software. This is

Figure 8. Sequence alignment accuracy comparison between base calls from our algorithm and the commercial ABI software, averaged for eight samples. Alignment accuracy is shown for comparisons from 100 up to 500 base calls from the primer. The experimental conditions were: 6% polyacrylamide, 7.8M Urea, 30 Watts constant power.

Figure 9. A graph of the correlation between alignment accuracy (quality ratio) of sequences produced by the program and mean confidence values. The data is from eighteen samples run on the HUGE system under the same conditions as in figure 7.

due to the degradation in the quality and consistency of the data in this region, making it more difficult to perform effective calibrations for features such as peak spacing and width. Though the algorithm continues to perform reasonably in comparison to the commercial package, it illustrates that the greatest strength of this algorithm is in regions where the data quality allows for accurate calibrations of the modules.

Another experiment was performed to evaluate our pre-processing software and examine the correlation between average confidence values and quality ratio. For this purpose, 18 samples were analyzed that were separated by HUGE on four different gels. Data was collected from each gel for a maximum of 70 minutes of electrophoresis. From the image file, lanes were chosen, and each sample lane was pre-processed identically. First, the raw data from the beginning of the electrophoresis run to

the primer peak was deleted. Next, the data was noise filtered by applying the Fast Fourier Transform filter, multi-component analysis was performed, mobility shift correction applied and the data was normalized. All of the parameters for the pre-processing steps were the same for all samples. For comparing base calling accuracies each trace file was edited further, such that each file contained the identical section of the M13mp19 DNA sequence (from base 81 to base 458). The processed trace file was then base called using standard settings for the feature-detecting modules. The sequence of each file was compared to the correct sequence using the GCG software package described above. The quality ratio was plotted versus mean confidence for each sample (Figure 9). Quality ratios ranged from 0.611 to 0.964 and mean confidences ranged from 0.644 to 0.798. Figure 9 shows that data with mean confidences of 0.74 and higher yield base calls with quality ratios of approximately 0.9 or higher. The relationship between quality ratio and mean confidence is approximately linear from 0.74 to 0.65. Thus, the average confidence for a particular data set is a good indicator of the accuracy of the base called data.

DISCUSSION

An important part of the base calling process is the pre-processing. These steps directly influence the data quality passed to the algorithm, and thus directly affect performance. The steps presented here for pre-processing are not unique; there are numerous other possibilities for signal conditioning, particularly in the areas of normalization and noise filtering.

One aspect of the pre-processing that may need further work is the dye-mobility shift correction. It has been observed in some data that the mobility-shift correction is different in the region of the first 40–50 bases from the primer than for the rest of the separation. This may be due to an effect such as a differing separation mechanism for shorter fragments during electrophoresis. This problem can be addressed by removing the first 40 bases of data, however, this decreases the amount of useable sequence produced in each run. It may be desirable to experimentally determine the function representing these mobility offsets so they can be better modelled and compensated for.

After pre-processing, trace files are produced which can widely range in quality due to the sequencing method used, the conditions, the fragment being sequenced, and also the way in which pre-processing was performed. To indicate how these variations have affected base calling, final confidence values are returned on each base label. These values range from near zero to one. As explained earlier, these confidence values are the geometric mean of the input of the judgement modules incorporated in the program. Because the current modules examine peak spacing, height, and width, the confidence values for each base are directly related to these features of each peak where a base label has been placed. This means that small, highly overlapped, or closely spaced peaks will receive lower confidence values. Regular, consistently spaced peaks with little signal overlap in other channels will receive the highest confidence values. These confidence values can potentially be used in fragment alignment algorithms to provide more information regarding the expected accuracy of each base, particularly in regions where there are disagreements or gaps during alignment. Additionally, areas of low-confidence can be used to pinpoint areas where problems occurred with some part of the experiment and thus need further attention.

As discussed, an average confidence of all the bases called for each trace file is calculated. This average is affected by several things, including a) the minimum confidence threshold set by the user, b) the signal strength, and c) resolution. The setting of the minimum confidence threshold for including a base call has several effects. If decreased, then more low confidence bases are included in the list, and a decreased average results. Increasing the parameter increases the average until a point is reached where the threshold is so high that many bases are discarded. At this point, the program attempts to compensate for these regions of missing base calls by inserting bases according to the spacing calibration, and these inserted bases may have lower confidence levels depending on where they are inserted.

Another factor to be considered is the algorithm's speed. All tests were performed on a 33 MHz 68040 based workstation. For data obtained on the commercial ABI instrument as reported above, base calling takes 7–12 seconds which includes peak finding, iteration, and base insertion in gap areas. For high quality data, the average confidence typically converges to a stable value within 2–3 iterations, each of which takes less than one second. For lower quality data where there are resolution or sensitivity problems, convergence does not occur as rapidly, and iteration is often terminated at the user settable maximum iteration count. Though the resulting sequence is often useable (and comparable in quality to the resulting sequence from the ABI system), the lack of convergence can indicate that there is some problem with the trace file input to the algorithm.

Conclusions and future work

As mentioned, there are a number of parameters necessary for the user of the program to choose in order to achieve the best performance for data from a particular instrument or set of experimental conditions. Choosing the parameters allows the operator flexibility in experimental conditions or instrument configurations. However, adjusting the parameters requires knowledge and some experimentation with different combinations of settings. It is desirable that this process be automated for users who do not have the background to effectively choose parameters, or for those who may be changing parameters quite often because of changing experimental conditions. We hope to continue towards this automation at the various steps of pre-processing and for the base calling algorithm.

It is anticipated that the same algorithm and programs written here could be used on data from other sequencing techniques, possibly including Mass Spectrometry (22) and other new approaches to sequence analysis. The only requirement would be that the data is in a similar four-channel, peak-by-peak format. Data from alternative sequencing technologies would be likely to have similar requirements for peak recognition and base calling.

Finally, we plan to soon modify one of the existing fragment alignment algorithms to incorporate confidences, and then to perform comparisons on the accuracy of the resulting consensus sequences with those obtained from unaltered alignment algorithms. We expect an improvement in the alignment accuracies due to the extra information available on each base-call in comparison to purely statistical consensus techniques. This may help especially where there are disagreements or gaps encountered in aligning fragments. Additionally, a confidence value maybe returned for each base in the finished, assembled data. These confidence values would be available to assist researchers using the sequence information.

Algorithm availability

We have developed the algorithm as part of a program for data collection and analysis that is still expanding. The program was written in Objective-C and runs on NEXTSTEP based computers (NeXT hardware or Intel-486 based personal computers running NEXTSTEP). Source and executable code for the current version are available via FTP. For information via electronic mail, send a message to: giddings@whitewater.chem.wisc.edu.

ACKNOWLEDGEMENTS

We would like to thank John Luckey and Xinchung Tong for data, and Doug Kolner for sequencing reaction mixtures. This work was supported by NIH Human Genome grant # HG00321 and DOE Human Genome grant # DE-FG02-90ER61026.

REFERENCES

1. Hunkapiller, T., Kaiser, R.J., Koop, B.F., and Hood, L. (1991) *Science*, **254**, 59–67.
2. Sanders, J.Z., MacKellar, S.L., Dodd, C.T., Otto, B.J., Heiner, C., Hood, L.E., and Smith, L.M. (1990) *Structure & Methods*, Volume 1: Human Genome Initiative & DNA Recombination, Eds. R.H. and M.H. Sarma, Adenine Press, 89–102.
3. Elder, J.K., Green, D.K., and Southern, E.M. (1986) *Nucleic Acids Res.*, **14**, 417–424.
4. West, J. (1988) *Nucleic Acids Res.*, **16**, 1847–1856.
5. NeXT Publications (1993) *NeXTSTEP Object-Oriented Programming and the Objective C Language*. Addison-Wesley Publishing Company, New York, USA.
6. Hayersroth, B. (1985) *Artificial Intelligence*, **26**:3, 291–321.
7. Waterman, M.S. (1984), *Journal of Mathematical Biology*, **46**, 473–500.
8. 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) *Nature*, **321**, 674–679.
9. Brumley, R.L. Jr., and Smith, L.M. (1991) *Nucleic Acids Res.*, **19**, 4121–4126.
10. Kostichka, A. J., Marchbanks, M. F., Brumley, R. L., Drossman, H., and Smith, L. M. (1992) *Bio/Technology*, **10**, 78–81.
11. Luckey, J.A., Drossman, H., Kostichka, A.J., Mead, D.A., Dcunha, J., Norris, T.B., and Smith, L.M. (1990) *Nucleic Acids Res.*, **18**:15, 4417–4421.
12. Karger, A.E., Harris, J.M., and Gesteland, R.F. (1991) *Nucleic Acids Res.*, **19**, 4955–4962.
13. Smith, L.M., Kaiser, R.J., Sanders, J.Z., and Hood, L.E. (1987) In: *Methods In Enzymology*, Vol. 155, pp. 260–301.
14. Erickson, C.L., Lysaght, M.J., and Callis, J.B. (1992) *Analytical Chemistry*, **64**, 1155–1163.
15. Janssens, F., and Francois, J.P. (1991) *Analytical Chemistry*, **63**, 320–331.
16. Press, W.H., Flannery, B.P., Saul, A.T., and Vetterling, W.T. (1986) *Numerical Recipes The Art Of Scientific Computing*. Cambridge University Press, Cambridge, England, 386–429.
17. Smith, L.M., Fung, S., Hunkapiller, M.W., Hunkapiller, T.J., and Hood, L.E. (1985) *Nucleic Acids Res.*, **13**, 2399–2412.
18. Fujii H., and Kashiwagi, K. (1992) *Electrophoresis*, **13**, 500–505
19. Bowling, J.M., Bruner, K.L., Cmarik, J.L., and Tibbetts, C. (1991) *Nucleic Acids Res.*, **19**, 3089–3097.
20. Watzig, H. (1992) *Chromatographia*, **33**:5–6, 218–224.
21. Smith and Waterman (1981) *Advances in Applied Mathematics* **2**, 482–489.
22. Parr, G.R., Fitzgerald, M.C., and Smith, L.M. (1992) *Rapid Commun. Mass Spectrom.*, **6**:369–372.