ORNL/TM-12821

Engineering Physics and Mathematics Division

Correcting Sequencing Errors in DNA Coding Regions Using a Dynamic Programming Approach

Ying Xu, Richard J. Mural†, and Edward C. Uberbacher

†Biology Division

DATE PUBLISHED - December 1994

Research supported by the Office of Health and Environmental Research,
U.S. Department of Energy, and the
Laboratory Directed Research and Development Programs

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
Managed by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under Contract No. DE-AC05-84OR21400

DISCLAIMER

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

CONTENTS

ABSTRACT

This paper presents an algorithm for detecting and "correcting" sequencing errors that occur in DNA coding regions. The types of sequencing error addressed include insertions and deletions (*indels*) of DNA bases. The goal is to provide a capability which makes single-pass or low-redundancy sequence data more informative, reducing the need for high-redundancy sequencing for gene identification and characterization purposes. This would permit improved sequencing efficiency and reduce genome sequencing costs. The algorithm detects sequencing errors by discovering changes in the statistically preferred reading frame within a putative coding region and then inserts a number of "neutral" bases at a perceived reading frame transition point to make the putative exon candidate frame consistent. We have implemented the algorithm as a front-end subsystem of the GRAIL DNA sequence analysis system (Uberbacher and Mural, 1991; Xu *et al.*, 1994) to construct a version which is very error tolerant and also intend to use this as a testbed for further development of sequencing error-correction technology. Preliminary test results have shown the usefulness of this algorithm and also exhibited some of its weakness, providing possible directions for further improvement. On a test set consisting of 68 Human DNA sequences with 1% randomly generated indels in coding regions, the algorithm detected and corrected 76% of the indels. The average distance between the position of an indel and the predicted one was 9.4 bases. With this subsystem in place, GRAIL correctly predicted 89% of the coding messages with 10% false message on the "corrected" sequences, compared to 69% correctly predicted coding messages and 11% falsely predicted messages on the "corrupted" sequences using standard GRAIL II method (version 1.2). The method uses a dynamic programming algorithm, and runs in time and space linear to the size of the input sequence.

1. Introduction

Locating and characterizing genes in DNA sequence is one of the major tasks of the Human Genome Project. Unfortunately the sequencing strategies which are the most time and cost efficient, such as single-pass sequencing, result in relatively low quality sequence data where errors make the identification of biologically important features difficult using computational systems. The development of methods which can provide accurate descriptions of gene features despite the presence of significant error is a considerable challenge. Potentially such methods could greatly enhance the usefulness of low redundancy DNA sequencing strategies and reduce the cost of sequencing dramatically compared to standard strategies involving 6-10 fold redundancy.

We originally developed a DNA sequence analysis system, GRAIL (Gene Recognition and Analysis Internet Link) (Uberbacher and Mural, 1991; Mural *et al.*, 1992), and have recently upgraded this system to make its coding region prediction (here called *exons*) more accurate (Xu *et al.*, 1994). Unlike the original system, the new version, GRAIL II (version 1.2), predicts discrete coding regions in a DNA sequence instead of a continuous coding probability function. It uses a variable-length window tailored to each discrete exon candidate to evaluate the coding potential of the candidate. A coding region (or exon) candidate is a translationally open region bounded by a putative acceptor splice junction (or a translation start) and a putative donor splice junction (or a stop codon); hence each candidate has a fixed inherent reading frame. The current scheme to evaluate discrete exon candidates uses more genomic context information in the exon discrimination process, including information about translation starts and splice junctions, and also the presence or absence of coding character in regions adjacent to the candidate, than did the original system. This method significantly improves upon the sensitivity and specificity of the original GRAIL system, particularly in identifying short exons.

A drawback of this method (and all other similar methods as well) is that it is sensitive to insertion and deletion (*indel*) DNA sequencing errors since these may change the open reading frame and hence interrupt the reading frame of a coding region. For example, a true exon may not be considered as an exon candidate in GRAIL II if the candidate's reading frame changes due to indels (however part of the exon may be considered as a candidate). On a test set of 68 Human DNA sequences with 1% randomly implanted indels in the coding regions, although GRAIL II predicts 91.7% of the coding exons (predicted regions overlap actual exons to some extent) with a 9% false positive rate, it predicts on average only 69% of the coding messages with 11% false information (see Figure 1). Minimizing this significant corruption of the predicted gene message is the main motivation of this work.

A number of algorithms have been proposed and implemented for detecting and correcting sequencing errors at the level of gel reading, for example using neural networks or maximum likelihood methods (Drury *et al.*, 1992). These algorithms deal with systematic errors of DNA sequencing devices mainly using instrumental information about the sequencing devices. Here we are interested in finding methods, using feature-related information, to detect and "correct" sequencing errors which escape these processes and corrupt important sequence features. In this paper we address the sequencing errors that change the structure of protein coding exons, and in particular,

Figure 1. The solid bars represent the exon positions of sequence HUMACTGA, and the hollow rectangles represent the GRAIL exon predictions. (a) GRAIL exon prediction on the original HUMACTGA. (b) GRAIL exon prediction on the artificially "corrupted" HUMACTGA.

sequencing errors that create changes in the reading frame within a coding region. Therefore our focus here will be on insertions and deletions which destroy the original reading frame, and not on on base substitutions which have virtually no effect on the coding region recognition process.

In GRAIL an exon is recognized along with its *preferred* translational reading frame, the reading frame with the highest coding probability. The algorithm presented in this paper consists of two main steps. It first finds all the transition points of the preferred reading frame along a given DNA sequence, and then considers the subset of transition points in apparent coding regions. To find all the transition points, the algorithm divides a DNA sequence into segments in such a way that two adjacent segments have different preferred reading frames, and the sum of a pre-defined objective function value over every segment is maximized. Each transition point of the preferred reading frames represents a potential indel(s). Most of these will be in non-coding DNA where the preferred reading frame function is not meaningful and transitions might be expected under normal circumstances (without indels). The algorithm then uses coding information from both sides of a transition point to determine if it may be within a coding region. If this seems likely, bases are inserted to recover frame consistency. If the transition is determined to occur in non-coding region no action is taken since a preferred reading frame transition is meaningless in this context. Once frame corrections are made throughout the perceived coding areas of the sequence standard GRAIL II analysis is performed on the "corrected" sequence.

Our test results have shown that the algorithm can detect 76% of indels that have been randomly implanted in the coding regions and make "corrections" to make frame consistency. As a result of the correction, GRAIL II can find 89.3% of the coding message on average with 10% false positive rate, compared to the 69% correctly predicted coding messages with 11% false positive rate when used directly on the corrupted sequences with 1% error rate. On average, the position of a predicted indel is about 9 bases away from the position of the actual indel. The final prediction accuracy of the new algorithm with 1% indel rate in coding regions is nearly as good as the standard GRAIL II system (version 1.2) on perfect sequence data (93% of coding message with 10% false positive).

2. System and Methods

The error correction program is written in the *C* programming language and is implemented on a Sparc II workstation under operating system SunOS 4.1.2. The program will soon be available through the X-based graphical client/server system XGRAIL. An executable code of the XGRAIL client is available to the public by anonymous ftp from arthur.epm.ornl.gov, and can be found in the directory /pub/xgrail/sun.

3. Algorithm

This section presents the dynamic programming algorithm developed to detect and "correct" sequencing errors in DNA coding regions. The types of sequencing errors addressed with this algorithm include insertions and deletions of coding bases. The basis of our error detection algorithm is the discovery of changes of the preferred reading frame within a coding region. Deletions or insertions of multiple of 3 bases will not in general cause any preferred reading frame changes, so

our algorithm will not be able to detect such errors. This is, however, not a serious problem since indels of this magnitude are highly improbable and presumably would not cause a coding region to be missed (by GRAIL II). The algorithm corrects apparent frame changes (whether caused by insertions or deletions) by inserting additional base(s), e.g., it treats an insertion of one base as a deletion of two bases. Correcting all frame changes by insertion avoids deletion (in some situations) of experimentally called bases.

3.1. Statement of the problem

We consider a DNA sequence containing a gene or part of a gene such that coding information is on one strand of the DNA. The algorithm can easily be extended to multigenic regions with coding on both strands, but we do not discuss this here.

Consider a DNA sequence D. Each segment of D has three possible translation frames, called *reading frames*. We use the 6-mer in-frame preference model (Uberbacher and Mural, 1991; Uberbacher *et al.*, 1993) to evaluate the *preference values* (defined below) of a segment in three possible reading frames. The reading frame with the highest preference value is called the *preferred reading frame*.

As the first step of solving the error detection problem we find the transition points between preferred reading frames. We model this problem as the following optimization problem. We want to partition D into segments in such a way that the sum of the preference values of the segments is maximized.

The boundaries of the segments correspond to the transition points of the preferred reading frames. As shown in Figure 2, most of these occur in non-coding regions, and are not relevant to the current purpose. We are interested in transitions that occur in coding regions due to indels.

To prevent short range fluctuations, we also require that each segment should have at least K bases. In our current implementation, $K = 30$. The following figure shows examples of "optimal" partitions of the given DNA sequence when $K = 6$ and $K = 30$. Figure 3(a) shows many short range fluctuations when $K = 6$ while Figure 3(b) is much more stable.

To make this more precise, we give the following formal description of the problem. Let $P_0(X)$, $P_1(X)$ and $P_2(X)$ denote the *preference values* of a 6mer X appearing in a coding region in the correct translation frame + 0, + 1, and + 2 versus appearing in non-coding regions, respectively. Consider a DNA sequence $D = a_0a_1...a_n$. We define the *preference value* of the segment $a_j...a_k$ of D, $0 \leq j \leq k-5 \leq n$, in reading frame $r = 0, 1, 2$ to be the following:

$$P_r(a_j...a_k) = \sum_{i=j}^{k-5} P_{(3+i-r) \bmod 3}(a_ia_{i+1}..a_{i+5}).$$

We call r the preferred reading frame of $a_j...a_k$ if $P_r(a_j...a_k)$ has the highest value among $P_0(a_j...a_k)$, $P_1(a_j...a_k)$, $P_2(a_j...a_k)$.

We want to partition D into segments $D_1, D_2, ..., D_m$, $1 \leq m < n$, such that the following objective function is maximized

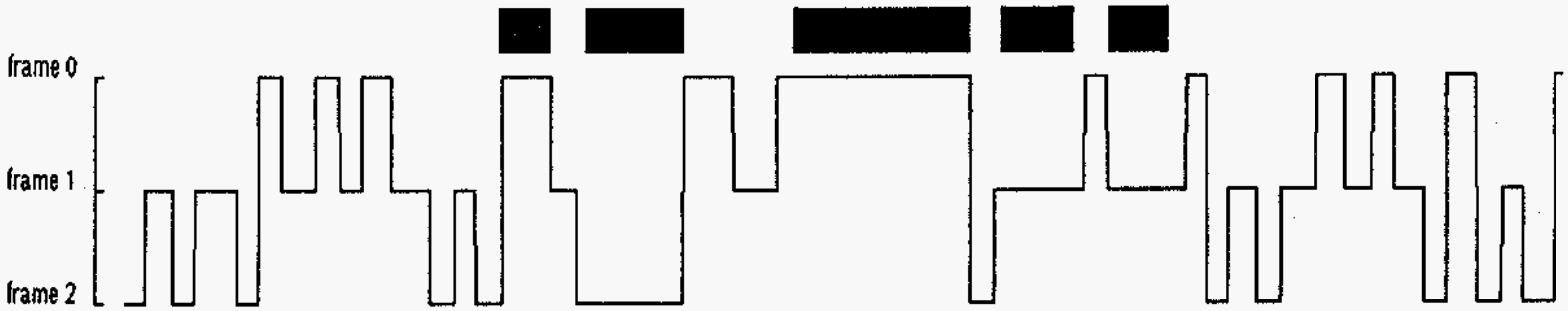

Figure 2. The solid bars represent the exon positions of HUMACTGA. The connected line represents preferred reading frame changes along the sequence, in which each horizontal portion represents the perferred frame in that region and a vertical portion represents a frame change.

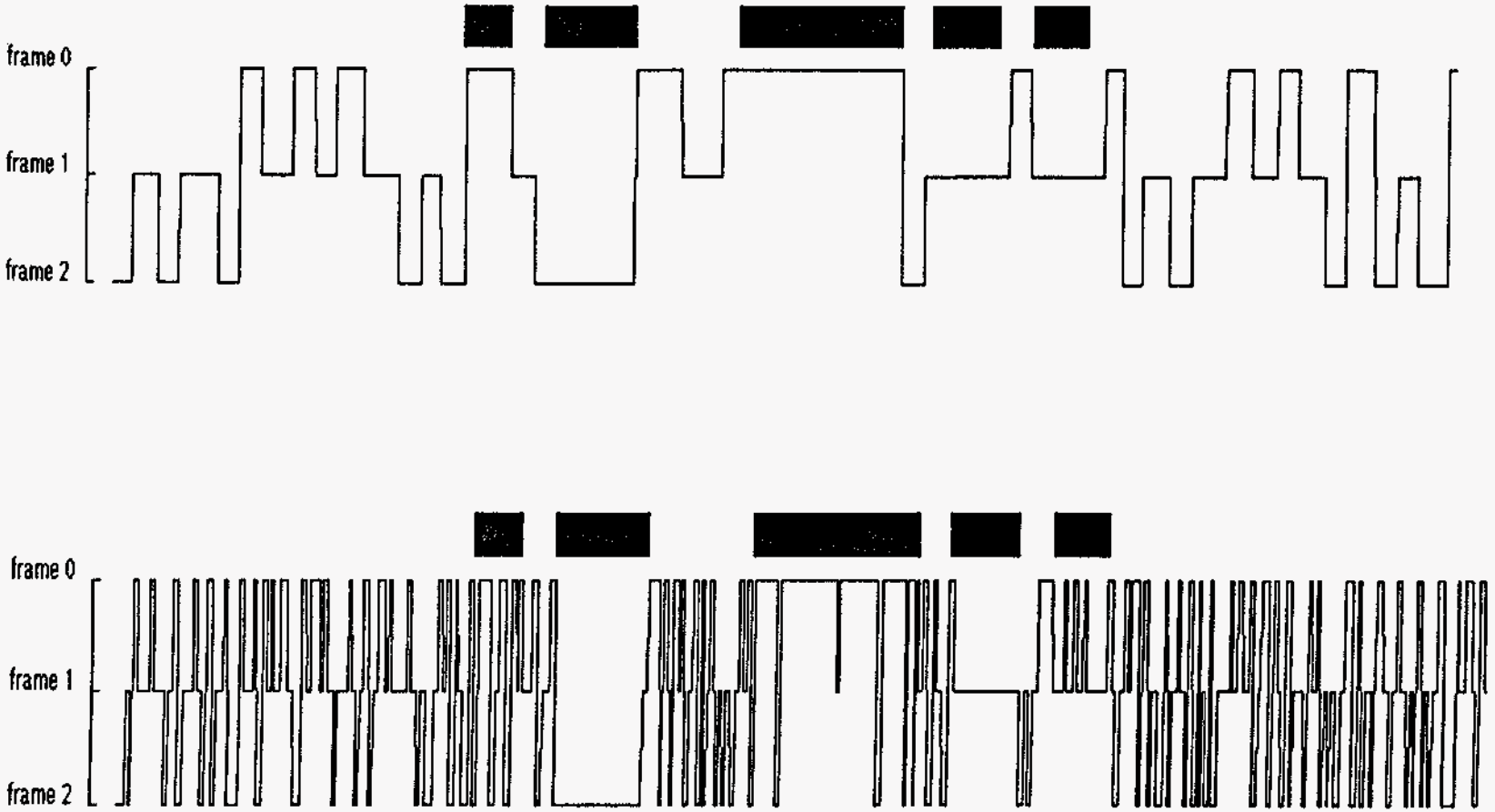

Figure 3. The solid bars represent actual exons of HUMACTGA and the connected line represents the preferred frame changes along the sequence HUMACTGA. (a) Preferred reading frame changes when $K = 30$. (b) Preferred reading changes when $K = 6$.

$$\sum_{i=1}^{m} P_{r(i)}(D_i),$$

under the constraint that each D_i is at least K bases long and no two adjacent segments have the same preferred reading frame, where $r(i)$ denotes the preferred reading frame of segment D_i. Note that a single segment may contain in-frame stop codons.

3.2. Dynamic programming algorithm

This subsection presents a dynamic programming algorithm to solve the optimization problem defined in the previous subsection. We first consider solving a simpler case of the problem by ignoring the constraint that each segment has to be at least K bases long.

The algorithm scans through a DNA sequence from left to right carrying with it partial solutions for the best way (maximum objective function) to partition the sequence into segments of different preferred reading frames. Since there are three reading frames possible throughout, three partial solutions are retained by the algorithm, one for each reading frame at a given point (base). These represent the best way to partition the sequence into preferred reading frame segments up to the current point and ending in each of the three frames. A useful property of this problem is that partial solutions at the current point can be constructed from partial solutions at the previous points. Keeping three best partial solutions at each point guarantees that sufficient information is present to construct best partial solutions at later points. At the next step we wish to extend the partial solutions to incorporate the next base. There are three possible ways to construct each new partial solution. For example, the new partial solution in reading frame 1 could be constructed by extending the previous solution in frame 1, or by extending the partial solutions in frame 0 or 2 and invoking a change in reading frame. Of these three choices the one with the highest score will be saved as the best partial solution for frame 1. The same process is used for frames 0 and 2. This process is repeated until the end of the sequence is reached. At the end of the sequence, the overall best solution of the three frames gives the final solution to the above defined optimization problem.

Formally, for each base a_i of a DNA sequence $D = a_0a_1...a_n$, $i \geq 5$, we define $B(i,r)$ to be the optimal value of the objective function defined above on sequence $a_0...a_i$ under the constraint that the last segment of the corresponding partition of $a_0...a_i$ is in reading frame r. By definition, the highest value of $B(n,0)$, $B(n,1)$ and $B(n,2)$ corresponds to the optimal solution to the above defined objective function.

By the above discussion, we have shown the following equations, based on which a simple dynamic programming algorithm can be used to find the optimal solution.

$$B(i,r) = \max \left\{ \begin{array}{l} B(i-1,r) + P_{(i+1-r) \bmod 3}(a_{i-5}...a_i), \\ B(i-6,(r+1) \bmod 3) + P_{(i+1-r) \bmod 3}(a_{i-5}...a_i), \\ B(i-6,(r+2) \bmod 3) + P_{(i+1-r) \bmod 3}(a_{i-5}...a_i) \end{array} \right\}, i \geq 6$$

and $B(5,r) = P_{(3-r) \bmod 3}(a_0 \ldots a_5)$, where $i \geq 6$ comes from the fact that we are using 6-mer models.

Having these equations, we can use a simple dynamic programming algorithm to calculate $B(n,r)$ values. We calculate $B(i,r)$ values for each i from left to right until $i = n$.

Now we consider the original problem. In the above solution, the preferred reading frame may change every few bases and cause short range fluctuation. By putting a lower bound on the shortest segment we can prevent this from happening. To add this constraint to the above method we need to guarantee that reading frame changes can happen at least K bases apart. Let $C(i,r,0)$ denote the optimal value of the objective function on sequence $a_0 \ldots a_i$ under the constraint that all segments in the corresponding partition have at least K bases except possibly the current one (the right-most one in the current partial solution), and the current segment is in reading frame r; and let $C(i,r,1)$ denote the optimal value of the objective function on sequence $a_0 \ldots a_i$ under the same condition plus the condition that the current segment is also at least K bases long. By using similar argument as above we have the following equations, based on which a simple dynamic programming algorithm given below is used to find the optimal solution.

$$C(i,r,0) = \max \left\{ \begin{array}{l} C(i-1,r,0) + P_{(i+4-r) \bmod 3}(a_{i-5} \ldots a_i), \\ C(i-6,(r+1) \bmod 3,1) + P_{(i+4-r) \bmod 3}(a_{i-5} \ldots a_i), \\ C(i-6,(r+2) \bmod 3,1) + P_{(i+4-r) \bmod 3}(a_{i-5} \ldots a_i) \end{array} \right\},$$

$$C(i,r,1) = C(i-K,r,0) + \textstyle\sum_{j=i-K}^{i-5} P_{(j+3-r) \bmod 3}(a_j \ldots a_{j+5}),$$

for $6 \leq i \leq n$, and $C(5,r,0) = P_{(3-r) \bmod 3}(a_0 \ldots a_5)$ and $C(5,r,1) = -\infty$. By definition,

$$\max_{r \in \{0,1,2\}, i \in \{0,1\}} \{C(n,r,i)\}$$

gives the optimal solution of the objective function.

To find the optimal solution, we calculate $C()$ values from left to right. During calculation, we keep a table to record the $C()$ values for the last K positions. Also to recover the transition points, we need to keep a table to record all the pointers pointing to the corresponding previous table entry. We use a four dimensional table $T[K,3,2,2]$ to keep both the values and pointers, where each integer represents the size of the corresponding dimension of the table. By tracing back where the reading frame changes occur in the optimal solution, we can obtain all the transition points.

This algorithm can be implemented in $O(Kn)$ time using $O(n)$ space. In our current implementation, $K = 30$.

An alternative but just as effective way to model the problem is not to require the minimum size of each segment, but to introduce a penalty factor for each reading frame change (to prevent short range fluctuations). Note that the choice of the penalty factor $\mathcal{P}$ indirectly determines the typical minimum size of each segment. In our current implementation, we have chosen the penalty factor $\mathcal{P}$ so that on average, the minimum segment size will be roughly about 30 bases. By doing

so the transition points corresponds to an optimal solution $\max\{B(n,0), B(n,1), B(n,2)\}$ defined by the the following equation.

$$B(i,r) = \max \left\{ \begin{array}{l} B(i-1,r) + P_{(i+4-r) \bmod 3}(a_{i-5} \ldots a_i), \\ B(i-6,(r+1) \bmod 3) + P_{(i+4-r) \bmod 3}(a_{i-5} \ldots a_i) - \mathcal{P}, \\ B(i-6,(r+2) \bmod 3) + P_{(i+4-r) \bmod 3}(a_{i-5} \ldots a_i) - \mathcal{P} \end{array} \right\}, i \geq 6$$

and $B(5,r) = P_{(3-r) \bmod 3}(a_0 \ldots a_5)$.

This problem can be solved in $O(n)$ time, which is slightly better than the first algorithm, and $O(n)$ space.

The performance of the two algorithms is similar. Based on the 68 test sequences, the first algorithm finds the indel positions a little more accurately ($\pm$ few bases).

3.3. Error detection and correction

The basic assumption of the algorithm is that if two adjacent regions show strong coding signal in two different reading frames then an indel(s) has occurred. Many preferred reading frame transition points found by the above algorithm occur in a non-coding region, and these transition points are simply the results of "randomness" of the non-coding DNA. We need to filter out these points from further consideration.

We use a 30-base long window to measure the the coding potential of the two 30-base regions on each side of a transition point. If both regions show strong coding signal in different reading frames we consider the transition point an occurrence of an indel. In our current implementation, we have used a 5^{th} order non-homogeneous Markov chain model (Uberbacher *et al.*, 1993; Borodovsky *et al.*, 1986) in evaluating coding potential of the adjacent regions of each transition point.

Consider one such point. Let L and R be the two 30-base long segments on the left and right sides of the point, and the preferred reading frames of the two segments be $r(L)$ and $r(R)$, respectively. If $r(L) = (r(R)-1) \bmod 3$, we consider a deletion has occurred at the transition point and we insert a base "C", which is considered to be a "neutral" base (cannot create a stop codon in all 3 possible frames), there. If $r(L) = (r(R)+1)$, we consider an insertion or two deletions have occurred. Since our goal is to make one consistent preferred reading frame we treat one insertion and two deletions the same. In this case we insert two "C's".

Figure 4 shows three examples of error findings by our error detection algorithm.

4. Implementation

The algorithm involves a number of parameters, the choices of which can affect performance. These include the minimum size of a segment or the penalty factor which indirectly determines the minimum size of a segment, the size of the window for evaluating coding potential around each transition point, the threshold for determining if a region has strong enough coding character to be called a coding region.

The choice of minimum segment size is a compromise between two considerations. Shorter segment size would make the system more sensitive to closely spaced indels. However the size of

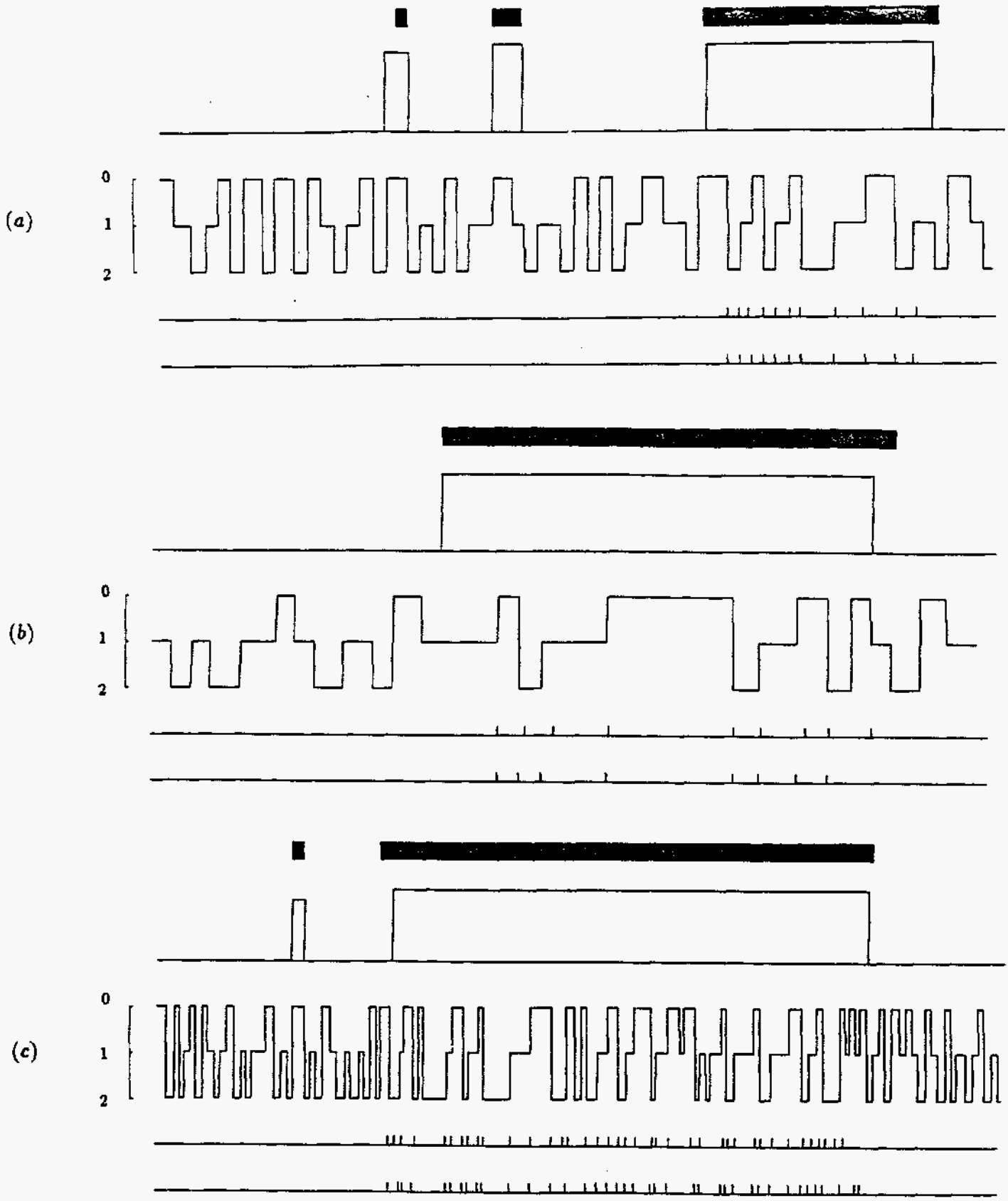

Figure 4. Examples of GRAIL predictions on the "corrected" sequences. The bars represent actual exons, and the hollow rectangles are GRAIL's exon prediction; Hash marks on the two horizontal lines represent the positions of indels (upper line) and predicted indels (lower line), respectively. (a) HUMAPOA4A. (b) HUMBAR. (c) HUMTRHYAL.

a segment should be at least as big as the size of the evaluation window (for coding evaluation) since otherwise a evaluation window may cover more than one segment. To have a statistically reliable coding evaluation in the regions adjacent to a potential indel, the size of a evaluation window and hence the size of a segment cannot be too small. In our current implementation we set the minimum segment size and the evaluation window size both to 30 bases. As a result this 30 base segment limitation, the vast majority of indels missed in the current system are less than 30 bases from an exon edge. Also when the error rates are very high (2% or more) and indels occur very closely spaced, the current system sometimes corrects double indels with a single predicted transition point.

In choosing the threshold for coding scores, our experience is that if we set the threshold too low we will include some transition points from non-coding regions, and hence create possible false coding regions or merge coding regions that are close to each other but in different preferred reading frames. We have taken a conservative approach, setting the threshold fairly high so only very few transition points in non-coding regions are included. By doing so, the algorithm may miss some transition points in coding regions which show (based on our coding recognition algorithm – the 5^{th} order non-homogeneous Markov chain model) weak coding signals or are very close to the edges of exons.

To refine our estimates of the parameters we analyzed a set of 10 genes (independent of the test set) while varying the minimum segment length, the coding evaluation window size, and the coding threshold, for the case of 1% randomly implanted indels. The segment length and evaluation window sizes were varied from 20 to 50 bases in steps of 5. At each combination of minimum segment length and evaluation window size the coding threshold was evaluated in increments of 0.01 (out of 1.0). The performance was relatively insensitive to the changes in the segment length and window size in the range of 30 to 45 bases, although as might be expected, the 30 base length for both parameters gave better results on the test data with a higher (2%) error rate, where indels tend to be closer together. The test results in the next section were obtained with the following parameters: 30 bases as the minimum segment length and the coding evaluation window size, and coding threshold of 0.9 (out of 1.0).

We have implemented both algorithms presented in the previous section. The test results shown in the next section are based on the implementations of the first algorithm.

5. Results and Discussion

To evaluate the sensitivity and specificity of our algorithm and how they are related to different sequencing error rates, we conducted three tests on the same 68 Human DNA sequences with different (randomly generated) error rates. In the three tests, we randomly implant 0.5%, 1.0% and 2.0% indels in the coding regions of the 68 sequences[1], where 1.0% error rate means that on

[1]hsckbg hsg6pdgen hsgstpig hsmpog humalatp humaccybb humactga humadag humagal humaldcg humalifa humalpha humalppd humant1 humant2x humapexn humapoa4a humapoe4 humaprt humatp1a2 humatpgg humbar humbmyh7 humcapg humcavii5 humcel humcs3 humcspb humctla1a humcyc1a humcyp2d6 humcypiie humdes humdkerb humef1a humerpa humfesfp humgamgloa humgapdhg humgck humghn humhsd3ba humhsp90b humibp3 humkerep humkertra humkrt1x humlyl1b hummis humnmyca humodc1a humorahbbe hump45c1 humpaia humpci

average for every 100 coding bases, there is one indel.

The indels are implanted randomly according to a random number generator. The only constraint is that no indels will be placed in a coding exon of 50 bases or shorter.

Though on the "corrupted" DNA sequences, GRAIL II (version 1.2) can locate almost as many exons as on the non-corrupted ones it finds less coding message in terms of coding bases. Figure 5 shows a typical GRAIL II prediction on a corrupted DNA sequence versus its prediction on the "corrected" one.

Tables I, II and III summarize the performance of the error-detection algorithm combined with GRAIL II on the three data sets. Table I shows how well the algorithm alone can detect the indels on sequences with 0.5%, 1% and 2% indels, respectively. The vast majority of incorrectly found errors are "eliminated" during the subsequent GRAIL II step; even with frame changes these are not considered to be within viable exon candidates.

Table I: Performance of error-detection algorithm prior to GRAIL II

Error rate	Total errors	Found errors	Incorrectly found errors	Ave. Dist.
0.5%	372	292 (78%)	122 (29%)	9.7 bases
1.0%	774	587 (76%)	126 (18%)	9.4 bases
2.0%	1339	856 (64%)	110 (11%)	13.4 bases

As we can see from this table, the performance of the error-detection algorithm is similar for error rates of 0.5% and 1%, but it drops significantly when the error rate goes up to 2%. The reason for this, we believe, is that when a number of indels occur in a region very close to each other (say 20 bases apart) our coding recognition algorithm is not sensitive enough to recognize each of the small coding regions resulting from the indels. Hence, the error-detection algorithm fails to detect the errors or only detects a portion of the errors close together.

Tables II and III summarize the performances of GRAIL II (version 1.2) on the "corrupted" sequences versus on the "corrected" sequences. The 68 test DNA sequences have a total of 508 exons consisting of 92437 coding bases.

Table II: Performance of GRAIL II on corrupted data

Error rate	Corrupted Sequences			
	TP(exons)	TP(bases)	FP(exons)	FP(bases)
0.0%	487 (96%)	87578 (95%)	46 (8.6%)	7710 (8.0%)
0.5%	476 (94%)	72838 (79%)	46 (8.8%)	7955 (10.0%)
1.0%	464 (91%)	63616 (69%)	47 (9.2%)	8251 (11.5%)
2.0%	443 (87%)	54097 (59%)	47 (9.6%)	8512 (13.6%)

Table III: Performance of GRAIL II on "corrected data"

humpgammg humpim1a humplpsp humpnmta humpomc humprca humpsap humrash humtbb5 humthb humtkra humtrhyal humvpnp

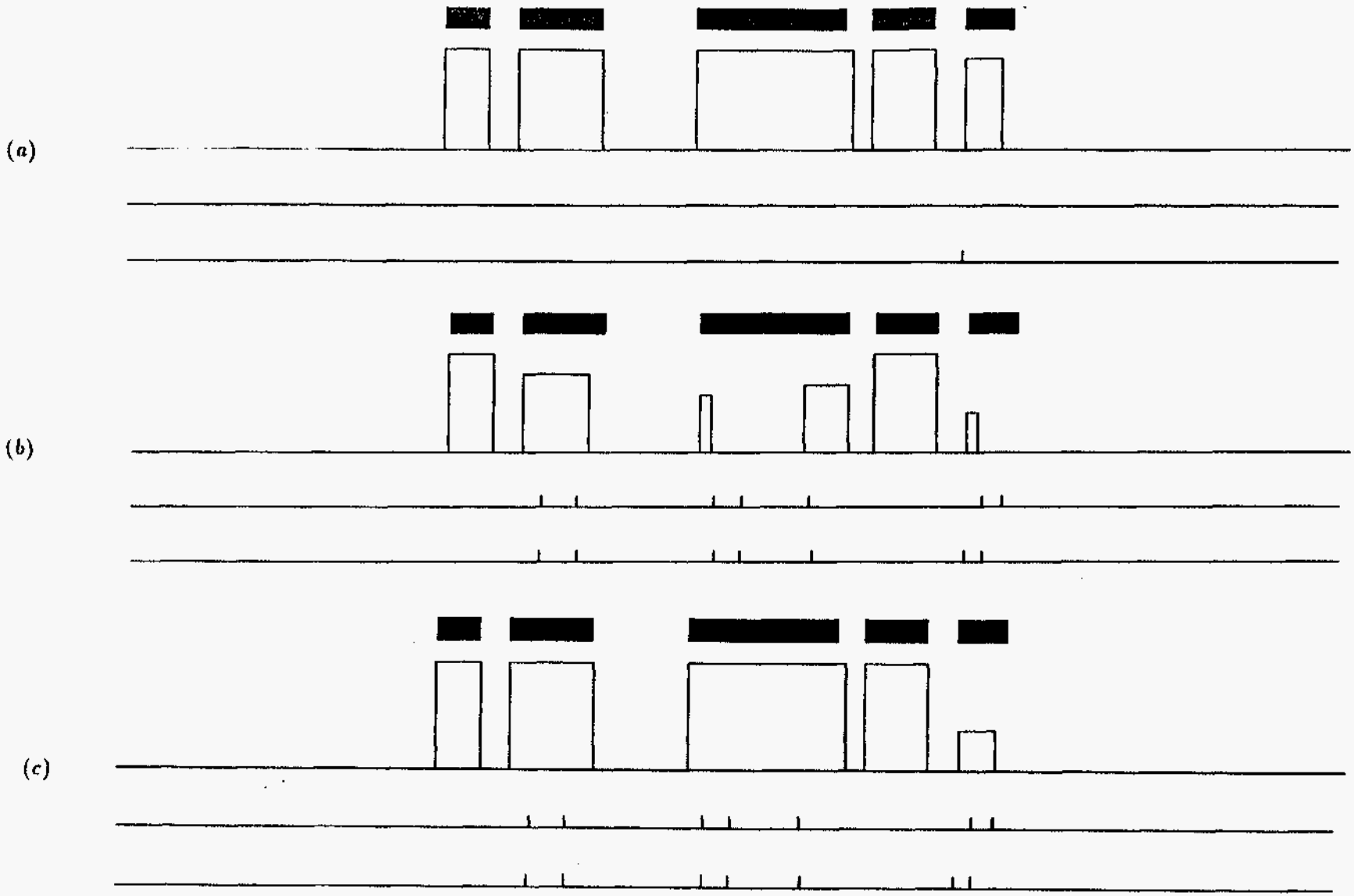

Figure 5. GRAIL predictions on the original HUMACTGA, its artificially "corrupted" version and the "corrected" version. The hash marks on the two horizontal lines represent the positions of indels (upper line) and predicted indels (lower line), respectively. (a) Prediction on the original HUMACTGA. (b) Prediction on the "corrupted" HUMACTGA. (c) Prediction on the "corrected" HUMACTGA.

Error rate	Corrected Sequences			
	TP(exons)	TP(bases)	FP(exons)	FP(bases)
0.0%	487 (96%)	87198 (94%)	49 (9.1%)	10268(10.5%)
0.5%	480 (94.5%)	85166 (92%)	49 (9.2%)	10087 (10.6%)
1.0%	476 (94%)	82526 (89%)	49 (9.3%)	9983 (10.8%)
2.0%	458 (90%)	73754 (80%)	49 (9.6%)	9968 (11.9%)

TP: The number of true positives;

FP: The number of false positives.

From Tables II and III, we can see that the correction algorithm, as a front-end subsystem, has improved the performance of GRAIL II significantly on corrupted DNA sequences. We can also see that the technique can, as expected, generate some false information if not used carefully. A few non-coding regions have been converted into coding regions and predicted as exons by GRAIL II as a result of insertions of new bases. By combining the outputs of GRAIL II on the original (possibly corrupted) sequence and the outputs on the "corrected" sequence it may be possible to further distinguish the originally missed exons due to sequence corruption from the false exons.

Most of the newly generated false coding bases are from overly extended predicted "coding" regions. Two possible situations can cause the error detection algorithm to make this type of mistake. The first one is when two exons, in two different reading frames, are fairly close to each other. In this type of situation, the two exons may be merged due to the insertion of new bases and creation of a new open reading frame that contains both exons. The second one is when one of the edges of an exon falls very close to the boundary of its open reading frame and the region on the other side of the boundary shows some coding potential. In this type of situation, the edge of the exon may become extended by GRAIL II's prediction.

We also tested the error correction algorithm on a number of pseudogenes to evaluate whether detection of pseudogenes was enhanced and whether frame-shifts would be detected. Generally few additional pseudogene "coding" regions were detected compared to standard GRAIL versions. In the vast majority of cases, the algorithm described the extent of pseudogene "coding" regions more accurately than standard GRAIL, and successfully corrected frame-shifts to provide good recovery of what had been the coding message.

In conclusion, a new algorithm for detecting and "correcting" sequencing errors has been proposed and implemented. The test results indicate that the algorithm can restore to a significant extent the value of a corrupted sequence for coding region and coding message prediction. This first step demonstrates the potential of this approach in the quest for more cost-effective and efficient genome sequencing and analysis methods.

6. Acknowledgments

This research was supported by the Office of Health and Environmental Research, United States Department of Energy, under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

References

Borodovsky, M., Yu. Sprizhitskii, E. Golovanov and A. Aleksandov (1986) "Statistical Patterns in the Primary Structures of Functional Regions in E. Coli.", *Molekulyainaya Biologiya*, 20, pp. 1390 - 1398.

Drury, H. A., D. G. Polittle, J. M. Ollinger, P. Green and L. J. Thomas Jr. (1992) "Maximum-likelihood Estimation of Restriction-fragment Mobilities from 1-D Electrophoretic Agaroes Gels", *SPIE Proceedings*, Vol. 1660, pp. 564 - 575.

Mural, R. J., J. R. Einstein, X. Guan, R. C. Mann and E. C. Uberbacher (1992), "An Artificial Intelligence Approach to DNA Sequence Feature Recognition" *Trend in Biotechnology*, 10, pp. 66 - 69.

Uberbacher, E. C., J. R. Einstein, X. Guan, and R. J. Mural (1993) "Gene recognition and assembly in the GRAIL system: progress and challenges", *Proceedings of The 2nd International Conference on Bioinformatics, Supercomputing and Complex Genome Analysis*, H. A. Lim, J. W. Fickett, C. R. Cantor and R. J. Robbins, Eds, World Scientific. pp. 465 - 476.

Uberbacher, E. C. and R. J. Mural (1991) "Locating protein-coding regions in human DNA sequences by a multiple sensors-neural network approach", *Proc. Natl. Acad. Sci. USA*, Vol. 88, pp. 11261 - 11265.

Xu, Y., R. J. Mural, M. Shah, and E. C. Uberbacher (1994) Recognizing exons in genomic sequence using GRAIL II, *Genetic Engineering: Principles and Methods*, Jane Setlow, Ed., Plenum Press, Vol. 16, pp. 241- 253, June 1994.

INTERNAL DISTRIBUTION

1. M. Beckerman
2. J. R. Einstein
3. C. W. Glover
4. W. C. Grimmell
5. X. Guan
6. J. P. Jones
7. H. E. Knee
8-12. R. C. Mann
13. S. Matis
14-18. R. J. Mural
19. E. M. Oblow
20. C. E. Oliver
21. S. Petrov
22. N. S. V. Rao
23. D. E. Reichle
24. D. B. Reister
25. M. B. Shah
26. R. F. Sincovec
27-31. E. C. Uberbacher
32. R. C. Ward
33. X. Ying
34. EPMD Reports Office
35-36. Laboratory Records Department
37. Laboratory Records, ORNL-RC
38. Document Reference Library
39. Central Research Library
40. ORNL Patent Office

EXTERNAL DISTRIBUTION

41. Office of Assistant Manager for Energy Research and Development, Oak Ridge Operations, U.S. Department of Energy, P.O. Box 2008, Oak Ridge, TN 37831
42. Jim Decker, Director, Office of Energy Research, Dept. of Energy, Washington, DC 20585
43. David Smith, Health Effects & Life Sciences Research Division, Office of Health & Environmental Research, Dept. of Energy, Washington, DC 20585
44. B. J. Barnhart, Health Effects and Life Sciences Research Division, Office of Health and Environmental Research, Dept. of Energy, Washington, DC 20585
43. John Wooley, Health Effects and Life Sciences Research Division, Office of Health and Environmental Research, Dept. of Energy, Washington, DC 20585
44. Marvin Stodolsky, U.S. Department of Energy, Office of Health & Environmental Res., ER-72 GTN, Washington, DC 94720
45. Jay Snoddy, U.S. Department of Energy, Office of Health & Environmental Res., ER-72 GTN, Washington, DC 94720
46. Michelle Broido, U.S. Department of Energy, Office of Health & Environmental Res., ER-72 GTN, Washington, DC 94720
47. Robert Robbins, U.S. Department of Energy, Office of Health & Environmental Res., ER-72 GTN, Washington, DC 94720
48. Mark S. Boguski, National Institutes of Health NCBI/NLM, 8600 Rockville Pike, Bethesda, MD 20894
49. M. D. Zorn, Lawrence Berkeley Laboratory, MS 50B-3216, 1 Cyclotron Road, Berkeley, CA 94720
50. J. Fickett, Los Alamos National Laboratory, P.O. Box 1663, Los Alamos, NM 87545
51. Christian Burks, Los Alamos National Laboratory, Theorectical Biol. & Biophysics Grp., T-10, MS K710, Los Alamos, NM 87545

52. A. Jamie Cuticchia, John Hopkins Hosp./Welch Med. Library, 1830 East Monument Street, Baltimore, MD 21205-2100
53. E. Branscomb, Human Genome Center, Biomedical Sciences Division, Lawrence Livermore National Laboratory, Livermore, CA 94550
54. A. Lapedes, Center for Human Genome Studies, Los Alamos National Laboratory, P.O. Box 1663, Los Alamos, NM 87545
55. Radoje Drmanac, Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, IL 60439
56. Chris Fields, The Institute For Genomic Research, 932 Clopper Road, Gaithersburg, MD 20878
57-58. Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831
59. Raymond F. Gesteland, University of Utah, Howard Hughes Medical Institute, 6160 Eccles Genetics Building, Salt Lake City, UT 84112
60. Steven Henikoff, Fred Hutchison Cancer Research Ctr., HHMI-FHCRC, 1124 Columbia Street, Seattle, WA 98104
61. Tim Hunkapiller, University of Washington School of Medicine, Department of Molecular Biotechnology, FJ-20, Seattle, WA 98195
62. Leroy Hood, University of Washington School of Medicine, Department of Molecular Biotechnology, FJ-20, Seattle, WA 98195
63. Thomas G. Marr, Cold Spring Harbor Laboratory, P.O. Box 100, Cold Spring Harbor, NY 11724
64. Sylvia J. Spengler, Lawrence Berkeley Laboratory, Human Genome Center, MS 1-213, 1 Cyclotron Road, Berkeley, CA 94720
65. Jude Shavlik, Computer Science, University of Wisconsin, 1210 W. Dayton Street, Madison, WI 53706
66. Lawrence Hunter, Lister Hill Center, National Library of Medicine, Bethesda, MD 20892.
67. Prof. Roger W. Brockett, Harvard University, Pierce Hall, 29 Oxford Street, Cambridge, MA 02138
68. Prof. Donald J. Dudziak, Dept. of Nuclear Engineering, 110B Burlington Engineering Labs, North Carolina State University, Raleigh, NC 27695-7909
69. Dr. James E. Leiss, Rt. 2, Box 142C, Broadway, VA 22815
70. Prof. Neville Moray, Dept. of Mechanical and Industrial Eng., University of Illinois, 1206 West Green Street, Urbana, IL 61801
71. Prof. Mary F. Wheeler, Department of Mathematics, Rice University, P.O. Box 1892, Houston, TX 77251