CONF-9510318--

An Iternative Algorithm for Correcting Sequencing Errors in DNA Coding Regions*

Ying Xu, Richard J. Mural[1], and E. C. Uberbacher

Informatics Group
Computer Science and Mathematics Division and [1]Biology Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6364

Presentation to be published as full article in *DIMACS Workshop on Gene-Finding and Gene Structure Prediction,* Philadelphia, PA, October 13-14, 1995.

*Research was supported by the Office of Health and Environmental Research, U.S. Department of Energy under contract No. DE-AC05-840R21400 with Lockheed Martin Energy Systems, Inc.

An Iterative Algorithm for Correcting Sequencing Errors in DNA Coding Regions

(Extended Abstract)

Ying Xu, Richard J. Mural† and Edward C. Uberbacher
Informatics Group
Computer Sciences and Mathematics, and †Biology Divisions
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6364

Abstract

Insertion and deletion (*indel*) sequencing errors in DNA coding regions disrupt DNA-to-protein translation frames, and hence make most frame-sensitive coding recognition approaches fail. This paper extends the authors' previous work on indel detection and "correction" algorithms, and presents a more effective algorithm for localizing indels that appear in DNA coding regions and "correcting" the located indels by inserting or deleting DNA bases. The algorithm localizes indels by discovering changes of the preferred translation frames within presumed coding regions, and then "corrects" the indel errors to restore a consistent translation frame within each coding region. An iterative strategy is exploited to repeatedly localize and "correct" indel errors until no more indels can be found. Test results have shown that the algorithm can accurately locate the positions of indels. The technology presented here has proved to be very useful for single pass EST/cDNA or genomic sequences, and is also often beneficial for higher quality sequences from large genomic clones.

1 Introduction

This paper addresses issues related to localization and correction of indel errors in DNA coding regions using statistical methods.

One of the very significant properties of a DNA coding region is that it exhibits a much stronger coding signal, measured using any frame-sensitive coding recognition method, in the translation frame compared to the other frames. In anonymous DNA, we use the *preferred* translation frame to refer to the frame with the highest coding potential. When indels occur in a coding region, its (preferred) translation frame changes within the region. By discovering changes in the preferred translation frames within a coding region, we can detect indels. To localize a coding indel involves (1) accurately locating the transition point of the preferred translation frame and (2) locating the coding regions in an anonymous DNA.

Correction of (located) indels, unlike the localization problem, can only be expected to be partially achieved since some information may have been lost (for example, bases that have been deleted) when an indel error occurred. Hence the basic goal in indel correction is to recover a consistent translation frame within a (presumed) coding region.

We previously designed and implemented a statistical method for localizing and "correcting" indels in DNA coding regions [3]. The algorithm first locates translation frame transition points by dividing a DNA sequence into segments in such a way that adjacent segments have different preferred translation frames and the sum of coding potentials in the preferred translation frames over every segment is maximized, where the coding potential is measured using an in-frame 6mer

preference model. Coding potentials are then measured around each transition point using two fixed-size windows (30 base long), one to the left and one to the right of a transition point. Transition points within poor coding neighborhood are removed from further consideration. The rest of the transition points are considered to be caused by indels. The sequence is "corrected" by adding 1 or 2 neutral bases like "C" at each transition point to make a consistent preferred translation frame.

Though test results on this algorithm have shown to be quite promising, they have also revealed some weakness of the method. As mentioned in [3], the accuracy of the error detection algorithm drops in the following situations: (1) when an indel occurs very close to the boundary of an exon; and (2) when a number of indels appear very close to each other (in this case, the algorithm may only locate a portion of the indels due to the constraint on segment size in the algorithm).

In this paper, we present an improved algorithm which overcomes some of the these problems, and also extends the indel localization/correction on a single strand to both strands of DNA. The improved algorithm uses an iterative scheme to localize and "correct" indels in the following manner. In each iteration, the algorithm, similar to the original one, first locates transition points by discovering changes in the preferred translation frames, and then it, unlike the original algorithm, partitions a sequence into (presumed) coding and non-coding regions based on coding measures. Transition points found within a (presumed) coding region are considered to be indels. These indels are then "corrected" by adding or deleting a base at each transition point to make a consistent translation frame (in each coding region). The algorithm iterates on the partially-corrected DNA sequence until no more indels can be found.

2 Algorithm

2.1 Inframe 6mer preference model

Inframe 6mer preference model is a simple but yet powerful statistical method for coding signal recognition [1]. The model consists of three preference values, $pf_0(X), pf_1(X), pf_2(X)$, for each of the possible 6mers X, which are defined as follows:

$$pf_r(X) = \log \frac{f_r(X)}{f_n(X)}, \qquad \text{for } r = 0, 1, 2, \tag{1}$$

where $f_0(X)$ is the frequency of 6mer X appearing in a coding region and in the correct translation frame, $f_1(X)$ is the frequency of X appearing in a coding region and in the correct translation frame + 1, i.e., the first base of X is the second base of a codon, $f_2(X)$ is the frequency of X appearing in a coding region and in the correct frame + 2, i.e., the first base of X is the last base of a codon, and $f_n(X)$ is the frequency of X appearing in a non-coding region. In our application, all the 6mer frequencies were calculated from a large set of DNA sequences[1].

Let $S = a_1.....a_n$ be a DNA sequence. The preference model calculates the coding potential of a segment $a_k...a_m$ in each of the three possible translation frames, $r = 0, 1, 2$, as follows:

$$pf_r(a_k...a_m) = pf_{(k+5-r)\bmod 3}(a_k...a_{k+5}) + pf_{(k+6-r)\bmod 3}(a_{k+1}...a_{k+6}) + \tag{2}$$

[1]The set contains 450 DNA sequences with 462608 coding bases and 2003642 non-coding bases.

$$pf_{(k+7-r)\text{mod}3}(a_{k+2}...a_{k+7}) + ... + pf_{(m-r)\text{mod}3}(a_{m-5}...a_m),$$

where **mod** is the modulo function. If $r = 2$ gives the highest value on the right-hand side then frame 2 is the *preferred* translation frame of segment $a_k...a_m$.

2.2 Localization of transition points

A "corrupted" coding sequence (with bases deleted or inserted) can be partitioned into segments in such a way that each of the segments has a (possibly) different preferred translation frame and its translation to proteins in that frame is consistent with the original "uncorrupted" sequence (ignore the boundaries between segments temporarily). We call each such boundary point a *transition point*. The problem addressed in this subsection is how to *localize* the transition points.

In this subsection, we do not distinguish coding from non-coding regions while trying to locate the transition points, each of them a potential indel. Two types of information are extracted through solving an optimization problem (optimized to have the highest total coding potential): (1) the locations of transition points, and (2) whether a base should be deleted or inserted at each transition point if an indel was determined to have occurred. We always insert a base "C" if an insertion is needed.

We formulate the transition point localization problem as follows. For a given DNA $S = a_1...a_n$, we want to find a set of locations on S, with the distance between two such points larger than $\mathcal{K}$ ($\mathcal{K} = 30$ in our implementation), so that the base at each such location can be deleted or a new base C can be inserted right in front of it to obtain a "corrected" sequence S^c with one consistent preferred translation frame, and the following objective function is maximized:

$$\max_{r\in\{0,1,2\}} pf_r(S^c) + N\mathcal{P}, \tag{3}$$

where $\mathcal{P}$ is a negative value used as a penalty for each located transition point, and N is the number of transition points found.

In the rest of this subsection, we present a fast dynamic programming algorithm to solve this optimization problem. Now we introduce some notation. $C_0(i,r)$ denotes the value of the maximum solution to the optimization problem (3) over the subsequence $a_1...a_{i+5}$, under the constraints that the translation frame of the subsequence from the last transition point t to position $(i+5)$ is frame r; $C_1(i,r)$ is defined similarly except that the distance between t and $(i+5)$ is at least $\mathcal{K}$. We use X to represent the last 6mer in the "corrected" subsequence $a_1...a_{i-1}$ corresponding to the maximum solution (note that X is necessarily the last 6mer of $a_1...a_{i-1}$ due to "corrections"), and $X(s)$ to represent a 6mer formed by appending the string s to the end of X after deleting the first $size - of(s)$ bases of X. Note that the reason we use $a_1...a_{i+5}$ instead of the natural $a_1...a_i$ is that an insertion or deletion at position i may affect the coding calculation up to position $i+5$.

There are three possible cases at each position i for calculating $C_0(i,r)$: (1) i is not a transition point; (2) i is a transition point and a base should be inserted in front of i; and (3) i is a transition point and it should be deleted. For $r = 0, 1, 2$ and $i \geq 7$, we have the recurrences (4) - (6).

Case 1: When there is no change in the translation frame at position i, we have:

$$C_0(i,r) = C_0(i-6,r) + \sum_{j=0}^{5} pf_{(i+j-r)\text{mod}3}(X(a_i...a_{i+j})), \tag{4}$$

Case 2: When there is a translation frame change at position i, and a base C is inserted in front of position i, we have:

$$\begin{aligned} C_0(i,r) = \;& C_1(i-6,(r+1) \bmod 3) + pf_{(i-(r+1))\text{mod}3}(X(C)) + \\ & \textstyle\sum_{j=0}^{5} pf_{(i+j-r)\text{mod}3}(X(Ca_i...a_{i+j})) + \mathcal{P}, \end{aligned} \tag{5}$$

Case 3: When there is a translation frame change at position i, and the base a_i is deleted, we have:

$$C_0(i,r) = C_1(i-6,(r+2) \bmod 3) + \sum_{j=1}^{5} pf_{(i+j-r)\text{mod}3}(X(a_{i+1}...a_{i+j})) + \mathcal{P}, \tag{6}$$

In the general situation, $C_0(i,r)$ is equal to the highest value of the right hand sides of (4) - (6); and for $r = 0,1,2$ and $i \geq \mathcal{K} - 5$, we have:

$$C_1(i,r) = C_0(i-\mathcal{K}+6,r) + \sum_{j=i-\mathcal{K}+12}^{i+5} pf_{(j-r)\text{mod}3}(a_{j-5}...a_j). \tag{7}$$

The initial values for $C_0()$ and $C_1()$ are defined as follows.

$$C_0(j,r) = pf_{(6-r)\text{mod}3}(a_1...a_6), \text{ and } C_1(i,r) = -\infty, \text{ for } 1 \leq j \leq 6 \text{ and } 1 \leq i \leq \mathcal{K} - 6. \tag{8}$$

Note that by definition,

$$\max_{r\in\{0,1,2\}} C_0(n-5,r) \tag{9}$$

corresponds to the maximum solution to the optimization problem (3). Hence to locate the transition points and to obtain the information on how the sequence S should be "corrected", we only need to calculate $C_0(n-5,r)$, for $r = 0,1,2$, using the recurrences (4) - (8) in the increasing order of i until i reaches $(n-5)$.

2.3 Construction of coding envelops and "correction" of indels

While the inframe 6mer preference model can accurately compute the preferred translation frame of a DNA the Markov model seems to be more convenient for our purpose to locate coding regions. Our coding recognition algorithm calculates the coding potentials at each base using a 5^{th} order non-homogeneous Markov model [2]. For each of the three translation frames, the algorithm calculates a coding score between 0 and 100 (0 for no coding and 100 for the highest coding possibility) within a window of size 60 bases entered at the current base.

Using the predicted coding curves, we want to partition the DNA sequence into (presumed) coding and non-coding segments. The basic idea for the segmentation can be described as follows. We use two parameters $\mathcal{H}_0$ and $\mathcal{H}_1$, with $0 < \mathcal{H}_0 < \mathcal{H}_1 < 100$, to separate coding from non-coding

regions (for example, $\mathcal{H}_0 = 25$ and $\mathcal{H}_1 = 75$). The DNA sequence is partitioned in such a way that the sum of variations of coding measures of each segment from $\mathcal{H}_0$ or $\mathcal{H}_1$ is minimized, i.e., segments are generated so that the coding potentials within each segment are "consistently" closer to $\mathcal{H}_0$ than to $\mathcal{H}_1$, or vice versa. To filter noise, we require that each segment should have at least $\mathcal{L}$ bases.

More specifically, we want to partition the DNA sequence S into segments $S_1, S_2, ...S_p$, with each S_i having at least $\mathcal{L}$ bases, and find an assignment $h(S_i)$ from $\{S_i\}_{i\in[1,p]}$ to $\{\mathcal{H}_0, \mathcal{H}_1\}$, with $h(S_i) \neq h(S_{i+1})$, in such a way that the following function is minimized. We use b_{ij} to represent the j^{th} base of segment S_i and $c(b_{ij})$ to denote the coding measure of b_{ij} (by the Markov model).

$$\sum_i \sum_j (c(b_{ij}) - h(S_i))^2. \tag{10}$$

This minimization problem can be solved by a fast dynamic programming as follows. We need to introduce a few notations first. Let $P_1(i, x)$ denote the minimum value of the minimization problem (10) over the subsequence $a_1...a_i$ under the constraint that the last segment in the minimum solution has assignment $\mathcal{H}_x$, with x being 0 or 1; $P_0(i, x)$ is defined similarly except that the last segment can be smaller than $\mathcal{L}$.

The following recurrences can be proved by an inductive argument on i, $i \geq \mathcal{L}$.

$$P_0(i, x) = \min\{P_0(i-1, x), P_1(i-1, 1-x)\} + (c(a_i) - \mathcal{H}_x)^2, \tag{11}$$

$$P_1(i, x) = P_0(i - \mathcal{L} + 1, x) + \sum_{j=0}^{\mathcal{L}-1} (c(a_{i-j}) - \mathcal{H}_x)^2. \tag{12}$$

The initial values of $P_0()$ and $P_1()$ are defined as follows. For $i < \mathcal{L}$,

$$P_0(i, x) = \sum_{j=1}^{i} (c(a_j) - \mathcal{H}_x)^2, \qquad \text{and} \qquad P_1(i, x) = -\infty. \tag{13}$$

Note that by definition,

$$\min_{x\in\{0,1\}} P_1(n, x) \tag{14}$$

corresponds to the minimum solution to the minimization problem (10). This value can be calculated using the recurrences (11) - (13) for each base a_i from left to right until i reaches n.

Indels are determined as transition points in a presumed coding region, and they are "corrected" by inserting or deleting a base. The algorithm recovers a consistent translation frame by deleting one base at each presumed indel position or inserting a neutral base "C" in front of this position, as described in Section 2.2.

2.4 An iterative strategy

The main goal of exploiting an iterative strategy is to overcome one inherent problem in the single-pass algorithm we have presented in Sections 2.2 and 2.3: predicted indels have to be at least $\mathcal{K}$ bases apart. This constraint is used to prevent short-range fluctuations in the transition

point localization algorithm, but it also intrinsically prevents discovering indels close to each other. Practically, we have also found that by using an iterative strategy, the algorithm improves its indel localization accuracy around exon boundaries.

The basic idea of the iterative strategy can be explained as follows. The algorithm keeps three lists L_1, L_2 and L_3 throughout its execution. L_1 contains all transition points discovered up to the current iteration; L_2 contains all the transition points of L_1 that have been determined not in any coding region; and L_3 contains the newly discovered transition points, points not in L_1 yet, in the current iteration. Initially L_1, L_2 and L_3 are all empty sets. At each iteration, the "corrections" are made on the original DNA sequence S ("corrected" based on the accumulative information up to the current iteration). Based on the partially "corrected" sequence, L_1, L_2 and L_3 are updated.

2.5 Extension to two strands

The indel localization and "correction" algorithm presented in Sections 2.2 – 2.4 applies only to one strand of a DNA. In the most general case, a DNA may contain coding regions in both strands, and hence extension of the algorithm to two strands is necessary.

We have used a simple strategy to extend the algorithm to two strands by simply applying the single-strand algorithm on two strands separately, and then combining the indel predictions on both strands after resolving possible inconsistencies existing in the two versions of the "corrected" sequence. We assume that coding regions on the forward and reverse strands do not overlap.

For the two versions of "corrected" sequence, the algorithm uses GRAIL II [3] to do coding predictions (the system uses not just coding signals but also uses information related to splice junctions, translation starts, gc compositions, exon flanking regions, etc). Then the algorithm, based on the coding prediction on both strands, partitions the two sequences, after a simple alignment, into regions, each of which contains coding regions only on (at most) one strand, using a strand-calling function described in [8].

3 Implementation and Results

We have implemented the indel localization and "correction" algorithm as a front-end subsystem of the GRAIL DNA sequence analysis system [4, 8] to construct a version of GRAIL which is very error tolerant and also intend to use this as a testbed for further development of sequencing error-correction technology. The algorithm is implemented in C programming language on a Sparc 10 workstation under operating system SunOS 4.1.2. As a part of the GRAIL II system (version 1.3), the error correction subsystem can be accessed through an X-based graphical client/server system, called XGRAIL. An executable code of the client is available to the public by anonymous ftp from *arthur.epm.ornl.gov* (128.219.9.76).

Extensive tests have been conducted on the algorithm. We have used a set of 220 DNA sequences containing 222608 coding bases and 1010517 non-coding bases as our test set. To conduct the tests, we have artificially implanted 1% and 2% of indels, respectively, in the coding regions of the DNA sequences, where 1% means that 1 indel was implanted every 100 (coding) bases on average with one exception that no indels are implanted in exons of 50 bases or shorter. The indels are generated using a uniformly-distributed random number generator.

Table I summarizes the performance of the indel localization and correction algorithm. Found indels and falsely found indels are counted as follows. For each actual indel, if there is a predicted

indel within distance less than 30 bases it is counted as a found indel; and for each predicted indel, if there are no actual indels within 30 bases it is counted as a falsely found indel (**FE**). We have used the GRAIL coding prediction as a filter to filter out predicted indels that are within any GRAIL predicted coding regions. The average distance is calculated as the sum of the distance between every found indel and its corresponding actual indel divided by the number of found indels.

Table I: Performance of localization/correction algorithm

Error rate	Total indels	Found indels	Ave. Dist.	FEs	FEs after filter
1.0%	1879	1348 (72%)	8.7 bases	636	361
2.0%	3621	2616 (72%)	12.4 bases	574	290

References

[1] E. C. Uberbacher 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.

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

[3] Y. Xu, R. J. Mural and E. C. Uberbacher (1995), "Correcting Sequencing Errors in DNA Coding Regions Using a Dynamic Programming approach", *CABIOS*, Vol. 11, No. 2, pp 117 - 124, 1995.

[4] Y. Xu, 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.

[5] D. J. States and D. Botstein (1991), "Molecular Sequence Accuracy and The Analysis of Protein Coding Regions", *Proc. Natl. Acad. Sci. USA*, Vol. 88, pp. 5518 - 5522.

[6] J. Posfai and R. J. Roberts (1992), "Finding Errors in DNA Sequences", *Proc. Natl. Acad. Sci. USA*, Vol. 89, pp. 4698 - 4702.

[7] X. Guan and E. C. Uberbacher (1995), "Alignment of DNA and Protein Sequences Containing Frameshift Errors", *CABIOS* in press.

[8] E. C. Uberbacher, Y. Xu and R. J. Mural (1996), "Discovering and Understanding Genes in Human DNA Sequence using GRAIL", *Methods in Enzymology*, 1996. In press.

The following figures show two examples (sequences HUMAPOA4A and HUMFESFP with sizes 3613 and 12263, respectively) of indel predictions. The solid bars in the top represent actual exons. The hollow rectangles in (A) repressent GRAIL II exon predictions on the corrupted sequences. Hash marks in (B) and (C) represent positions of actual and predicted indels, respectively. Hollow rectangles in (D) represent GRAIL exon predcitions on "corrected" sequences.

Figure 1:

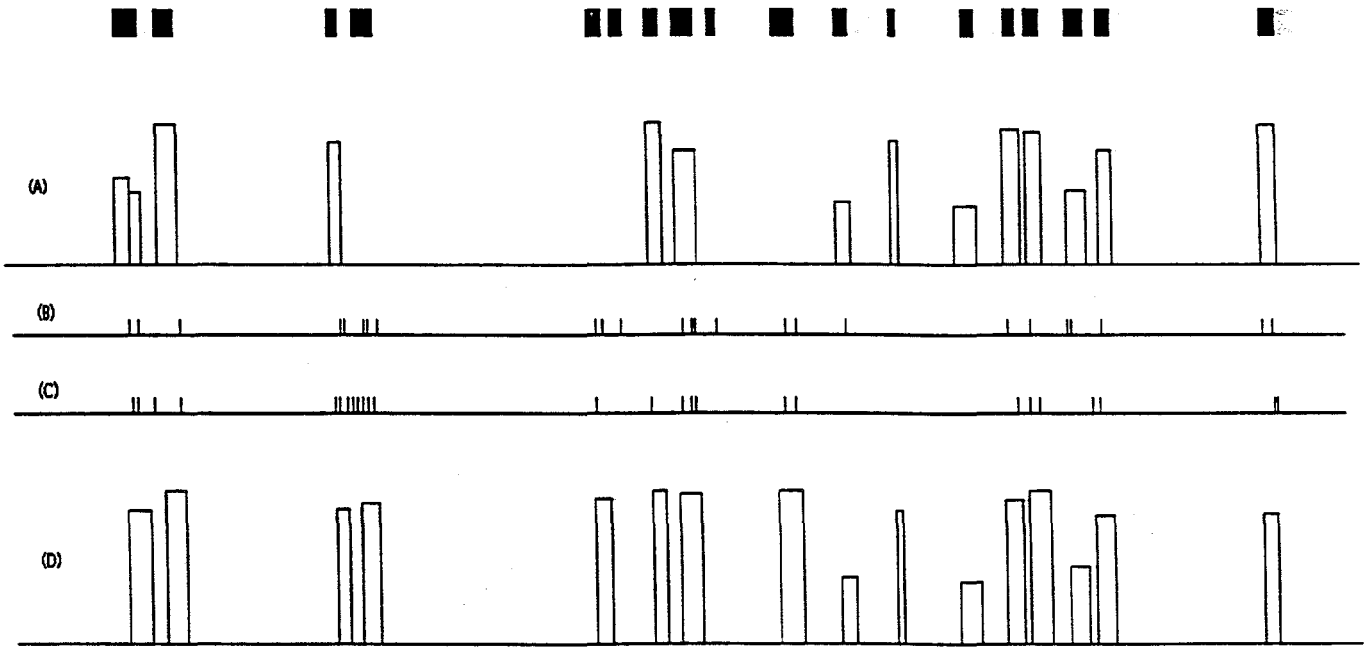

Figure 2: