Determining 3′-ended EST fragments

E. Angelou, J. de Jong, B. Rexhepi and A. Sottoriva

June 1, 2007

1 Introduction

The determination of the 3′ end of a GenBank sequence is important in many ways. It is used to construct transcriptional units, to investigate the polyadenylation variability or in probe design for DNA microarrays. Currently, GenBank database lacks consistency in annotation of the sequences in GenBank. It is not usually clear if the sequence is 3′ or 5′ ended. We developed an algorithm that given any GenBank EST determines if it is a 3′ end or not.

2 The parser

The role of the parser is to extract DNA sequences from Genbank files and provide clean strings of nucleotides to the evaluation functions. In our case the parser is strongly dependent on the file format used: the input files have to be in the Genbank format, each of them with N ($N \geq 0$) sequences.

2.1 Genbank file format

The Genbank format cathegorizes the information as records, each record corresponds to a tag which contains several additional information such as the source of the sequence, references, authors and related publications. For each object on a Genbank file the last record is the DNA sequence itself. In our implementation we are not interested on the information stored in the annotations, this data are indeed unclear and in particular they are not coherent with any sort of standard format. We consider therefore the information provided by the annotations untrustworthy, and we just consider the DNA sequence.

2.2 Parsing

The parser extract all the sequences which start with `ORIGIN` and end with `//`. The sequence has then to be cleaned from spaces and row numbers, and is ready to be processed by the scoring subroutines. Furthermore, in our case the files were compressed in `.gzip` format, in order to be able to read them we implemented an on-the-flight uncompression using the *PerlIO::gzip* package. The directory on which are stored the files is given to the parser by command line: the parser will scan all the *.gzip* files with an *est* pattern in the name (to select

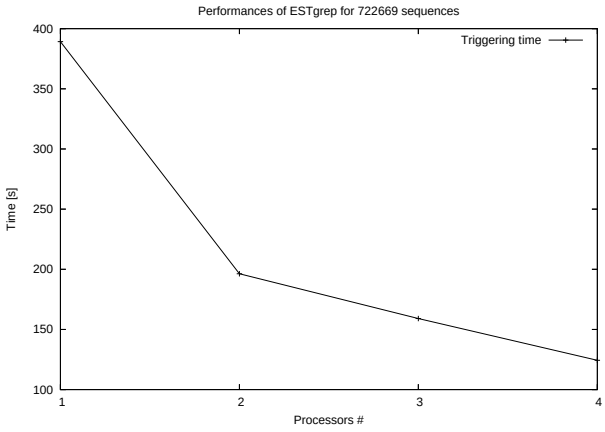
only EST files). More generally, the set of parameters that can be provided via command line is:

- *EST_files_directory*: the directory where the gzipped EST files are stored.
- *number_of_processors*: the number of processors needed (see *Parallelization*).
- *max_files*: the maximum number of files to scan (for testing purpouses).
- *-t threshold*: the desired scoring treshold.
- *-s sig_weight*: the weight applied to the PolyA signal scoring.
- *-p tail_weight*: the weight applied to the PolyA tail scoring.

The last three values permit to the user to tune the response of the program according to a given testset. This allow the user to train the program on the data that he believe to be trustworthy.

2.3 Parallelization

Observing the process of scanning and scoring of the sequences we discover that is strongly scalable. Our scoring algorithm is based on known patterns and statistical evaluation of each nucleotides string, independently from the others. Given the enourmous amount of sequences to scan we implemented a parallelization, both for the scanning and for the scoring processes. If specified, when the user launch the parser, it will spawn in multiple different *processes* using a simple *fork()*. Each of these processes will work separately with a subset of the files present in the directory. Using processor systems, but also with modern multicore processors, the performance are fairly increased and due to the nature of the problem we can have an almost linear speedup.



3 Randomly generating ESTs

The performance in terms of false positive and false negatives of the software were made using complete cDNA sequences chopped into artificial ESTs. We retreived the sequences from the *Unigene* database. The advantage on building a testset from these information are:

- the files are already in genbank format.
- the cDNA sequence if for sure complete and 3'.
- the annotations present in the database (unfortunately not inside the files) tell where are the PolyA signal and the PolyA tail.

The program that converts these sequences into ESTs is the *Chopper*. First, together with the cDNA we provide to the chopper the information (position of signal and tail) present in the pages of the database. The chopper than *chops* the sequence randomly, producing subsequences of circa 450 nucleotides (approximately the size of an EST). It then saves all the output in different genbank files which contain in the name the cutting position and, in the case that the cutting position contained the tail and/or the signal, a further tag is shown in the name of the file. After the chopping process we have a set of artificial EST files where we know which of them contain the PolyA signal, the PolyA tail or both of them. We then apply our classification program checking for false negatives and false positives.

4 Cutting off the poly(A) tail

The function cutOff is responsible for finding poly(A) tails and poly(T) heads and cutting them off.

The function uses the following procedure to look for a poly(A) tail:

- Take a window of 7 nucleotides at the end of the sequence
- If there are more then 6 As in this window then
  - The sequence has got a possible poly(A) tail
  - While there are more than 6 As in this window
    - Shift the window 1 nucleotide back
  - Take the distance from the 5 most A in the last window to the end of the sequence to be the length of the possible poly(A) tail
- Else the sequence has got no poly(A) tail

A similar algorithm is then used to look for a poly(T) head, after which the function compares the lengths of the possible poly(A) tail and possible poly(T) tail, cuts of the longest one, and return a tuple as specified above.

The first version of this function adopted a criterium that was copied directly from (Beaudoing et al.). According to (Beaudoing et al.) the end of a sequence is a poly(A) tail if it contains 12 As in a window of 15 nucleotides.

Consequently they cut it off. Since most of the sequences we observed, contained shorter (but documented) poly(A) tails, we decided to change this criterium: The end of a sequence is a poly(A) tail if it contains 6 As in a window of 7 nucleotides. Furthermore, we decided to shift the window back one nucleotide each time, until there were less than 6 nucleotides in the window. This was not mentioned in the papers but it would allow us to find and cut off the entire poly(A) tail and not just the last part of it.

The eventual function takes as an argument a (reference to a) string containing lowercase characters a, c, g, and t in the case of reverse complement RNA, or lowercase characters a, c, g, u in the case of RNA. An advantage of passing the argument by reference is that it improves performance, e.g. by not having to copy argument values multiple times. The function returns a tuple (x, y) where x can take the values (1) -1 (a poly(T) head was cut off), (2) 0 (nothing was cut off), and (3) 1 (a poly(A) tail was cut off). The value of y is the length of the tail cut off.

5 Generating random sequences

The program generate.pl generates a random sequence of protein, rna, or dna and writes it to a file. It is used to test the algorithm for determining 3 ends on random sequences.

One way of testing whether the program worked correctly was to use random sequences of a, c, g, and u (or t), i.e. random sequences of rna or cdna, as input. Correctly is meant here in a sense that it would not try to find signals where there were not any. These sequences are generated by the program generate.pl. The program defines three arrays of characters representing the letters which rna, dna, or protein sequences may be built up from. Until the specified length has been reached, characters randomly chosen from one of these arrays are added to the result sequence. This sequence is then written to a file as specified above, and used as input for the program ESTgrep that tries to determine the 3 end of a sequence. It performed excellently, and classified all sequences correctly.

6 Basic signal search

Detecting a polyadenylation signal is based on searching for certain hexamers in the EST we are given to classify. The existence of such a signal strongly suggests that the EST we are dealing with is 3'-ended, since it is highly unlikely that such a sensitive signal could be found in abundance in other parts of the mRNA sequence. Therefore, we can assume that finding a polyadenylation signal almost ensures that the fragment under inspection is 3'-ended.

However, there are multiple hexamers that act as polyadenylation signals making it harder to trace them and most importantly score them according to the propability that they indeed are signals and not random occurrences in the mRNA sequence. In *Beaudoing et al*, a comparison of 3'-end EST polyadenylation signals shows that the 2 most common naturally occurring variants are AATAAA and ATTAAA, but they only account for 73.2% of the polyadenylation signals. The remaining 26.8% is composed of less common signals as de-

1. Use the data from the cutOff function, so that if the orientation is known search for the signal in only one end of the signal
2. Search in the last and/or first 50 nucleotides for each signal, in order of decreasing probability of occurrence.
3. Retain the position of the most common hexamer that is found closer to the mean position, as described in Table 2 *Beaudoing et al*.
4. Map these results to a score using a scoring function based on:
   - The probability of the signal occuring.
   - The proximity of the signal to the ideal (mean) position.
5. Finally, before returning the score, check if the signal is in an A-rich area, and therefore propably artifictual. If so, return a zero score.
6. Return a tuple (score, orientation)

Figure 1: Step by step description of signal detection (pseudocode of the actual checkSignal function).

picted on Table 2 of the respective paper, and as it is also suggested in *Sheets et al* by point mutations of the most common occuring hexamer, AATAAA.

Our strategy then is to search the EST sequence for each of the polyadenylation signals described above. We know that the hexamers should occur near the end of the sequence, or alternatively at the beginning if we are dealing with a reverse complemented sequence. Since in the previous step we have already searched for a poly(A) tail (or poly(T) head...) we can use this information to search only one end of the sequence. As we are interested in the strongest signal possible, we search the last 50 nucleotides of the sequence for the hexamers in order of occurrence in nature, so as to pick up the most common signal in the EST. We then have to score the signal. Assuming that the EST is 3'-ended (since there is a polyadenylation signal) we can use the propability of occurrence from Table 2 as a scoring function, thus scoring more for "stronger", more common signals. Since most of the probabilities of the point mutation hexamers derived from AATAAA are not computed, we have decided to score them on a 0.5% frequency of occurance, so as to give them the chance of achieving a meaningful score.

7 Artificial signals and signal position

In order to increase the certainty that the detected hexamer is indeed a polyadenylation signal, we can use two more techniques to eradicate obviously artifictual signals and also use the position of the signal as further proof.

An EST produced randomly could falsely be considered that it contains a polyadenylation signal should its end comprise an A-rich area. In that case, a single nucleotide could cause the search to return a positive signal, since the majority of the hexamers used for detection are composed primarily of adenine

1. Use the data from the cutOff function, so that if the orientation is known search for the signal in only one end of the signal
2. Search in the last and/or first 50 nucleotides for each signal, in order of decreasing probability of occurrence.
3. Retain the position of the most common hexamer that is found closer to the mean position, as described in Table 2 *Beaudoing et al*.
4. Map these results to a score using a scoring function based on:
    - The probability of the signal occuring.
    - The proximity of the signal to the ideal (mean) position.
5. Finally, before returning the score, check if the signal is in an A-rich area, and therefore propably artifictual. If so, return a zero score.
6. Return a tuple (score, orientation)

    Figure 2: Step by step description of the finalScore function(pseudocode of the actual checkSignal function).

nucleotides. A more detailed analysis of this problem and a similar solution can be found in the Results and Discussion section of *Beaudoing and Gautheret*. In our case, we consider a window of 20 nucleotides containing the signal and if the window is composed of A's in a percentage more than 70% we discard the signal as being artifictual and thus no score is returned.

On the other hand, in *Beaudoing et al*, a study of the expected position of a signal can give stronger indication of a polyadenylation site. In the paper, the mean position at which each signal is positioned is computed along with the standard deviation from it, assuming a Gaussian distribution. Using the results derived from Table 2, we consider only the signal closer to the mean position, and then adjust the scoring given for the type of signal as follows:

- If the signal is found one standard deviation away from the mean, the score is multiplied by 2.
- If the signal is found within two standard deviations, we multiply the score by 1.2.
- In case the signal is found away from the mean, it is still considered as an actual signal and scored normally from the signal detection algorithm described above.

The signals for which we have no actual data as to their mean position and st. deviation are given the same values as the most common occurring hexamer. Although not scientifically rigorous, it seemed as the best solution since those sequences are point mutation of this sequence, hence we can assume similar behavior in respect to positioning.

8 Scoring and Training set

Some general remarks have to be made for the scoring of the sequences. First of all, we favored scoring to a black-or-white approach in determining 3′ ESTs since in this case the user can set their own threshold for the detection of a 3′-ended EST sequence. This way, the user can fine-tune the program for maximum discovery or minimum false positives, depending on the problem at hand. Correct scoring however is essential in this process.

Our scoring mechanism weighs two different scores: one provided from the poly(A) tail detection part and one from the polyadenylation signal detection part of the algorithm. The final score for the given sequence is produced by first adding the weighted scores and then multiplying by 2 if both signals are present, so as to further favor sequences that contain both 3′-end characteristics.

Since it is difficult to be absolutely certain that our scoring mechanism can help distinguish between 3′-ended ESTs and 5′ or random sequenced ESTs, we used a self created training set to fine-tune it. The set was created by randomly chopping complete mRNA sequences for which all data as to polyadenylation signals and poly(A) tails were contained in the GenBank annotation, and then tag them so as to make sure that our program correctly identifies 3′-end fragments containing signals and/or a poly(A) tail.

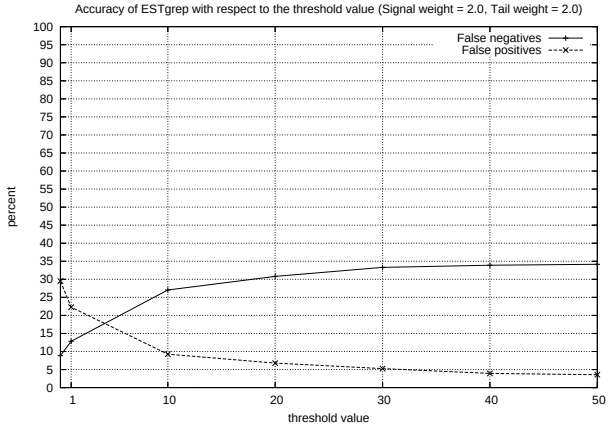
Later testing and re-adjustment using the provided test set has produced the following results:

- Set for maximum detection rate: about 87% of the 3′ EST's are correctly classified, with a penalty of 14% of false positives in random sequences.
- Set for minimum false positives: setting a higher threshhold, yields a drastic decrease in the percentage of false positives (less than 0.5%) at the penalty of correctly detecting around 50% of the 3′-end ESTs.

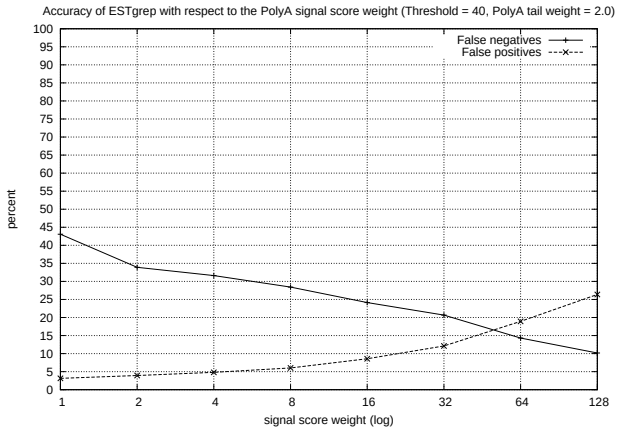
It is obvious that our code is rather conservative, as we felt that a large amount of less significant signals described in the literature would not help the discovery rate significantly but would obscure the results by introducing more false positives.

9 Parameters tuning

ESTgrep permits to tune some parameters such as the threshold value and the weight of the score for the PolyA signal and the PolyA tail directly from the command line. We performed some experiments on the dataset provided to discover the optimal configuration in order to obtain the minimum number of false negatives while keeping the false positives percentage below 5%. First we changed the threshold value keeping the scoring weights fixed:



Afterwards we investigated the accuracy of our software keeping the threshold value fixed and varyiating the signal scoring weight:



It was not necessary to perform experiments also for the polyA tail given it is a pattern relatively less involved in the presence of false positive and false negatives. The results tell us that an optimal paramiter setting could be THRESHOLD = 40 and SIGNAL_WEIGHT = 2.0.

10 Conclusions

The software we developed is entirely based on a pattern-oriented policy. It therefore triggers an 3' ended EST sequence only when one of the possible patterns (one of the possible PolyA signal or a PolyA tail) is found on the sequence, at a certain position. The scoring is then computed according to some statistical information that we have from reference papers. As discussed above, the algorithm doesn't consider any of the possible annotation present on genbank files. We consider those untrustworthy and many times misleading. Despite the program implements a significant level of complexity on triggering EST sequences, it doesn't show very impressive performance. However we think we obtained interesting results, with a quite high triggering rate, maintaining at the same time a false positive rate below the barrier of 5%. Moreover our software performs fairly well in terms of usage of computational power with in addition the possibility of exploiting parallelism.

Reference

- *Alternate Polyadenylation in Human mRNAs: A Large-Scale Analysis by EST Clustering*. Daniel Gautheret, Olivier Poirot, Fabrice Lopez, Stphane Audic, and Jean-Michel Claverie
- *Point mutations in AAUAAA and the poly (A) addition site: effects on the accuracy and efficiency of cleavage and polyadenylation in vitro*. M D Sheets, S C Ogg, and M P Wickens
- *In silico detection of control signals: mRNA 3'-end-processing sequences in diverse species*. Graber JH, Cantor CR, Mohr SC, Smith TF
- *Identification of Alternate Polyadenylation Sites and Analysis of their Tissue Distribution Using EST Data*. Emmanuel Beaudoing, and Daniel Gautheret
- *Patterns of variant polyadenylation signal usage in human genes*. Beaudoing E, Freier S, Wyatt JR, Claverie JM, Gautheret D