Genome-wide Compilation of mRNA Polyadenylation Signals in *Arabidopsis* Using Advance Searching Technologies

Johnny C. H. Loke, *Miami University*, **Dave Strenski**, *Cray, Inc.*, **Eric Stahlberg**, *Ohio Supercomputer Center*, **P. Christopher Wood** *and* **Q. Quinn Li**, *Miami University*,

ABSTRACT: *New computing technologies have recently become available to search and rapidly analyse large numbers of DNA sequences for critical information. Two technologies are of particular interest, general high-throughput sequencing libraries available on Cray systems such as the Cray SV1 and field-programmable gate arrays (FPGA) available as add-ons to such systems as the Sun Fire 6800. We have applied these technologies for the study of mRNA polyadenylation signals in the model plant* Arabidopsis*. In doing so, we have not only revealed many highly consensus signals in a predicted location, but also found an additional polyadenylation signal that were not previously identified by experimental approaches. Our results show the power of the technologies in bioinformatic studies.*

1. Introduction

Fifty years ago in 1953, two biochemists namely, Watson and Crick, discovered a double-helix DNA structure believed to contain genetic information coded in residue bases called 'nucleotides'. This genetic information is transcribed into messenger molecules called, messenger RNA (mRNA), which are "decoded" (translated) into proteins—this being the "central dogma" of genetics. Polyadenylation of mRNA is a crucial step during the maturation of the linear eukaryotic mRNA; thus it is essential in gene expression (Li and Hunt 1997; Zhao et al., 1999). Polyadenylation is a process by which a poly(A) tract is added onto the 3' end of a precursor-mRNA (Fig. 1) after capping and splicing, to ensure its functionality, such as translatability, stability and translocation to cytoplasm.

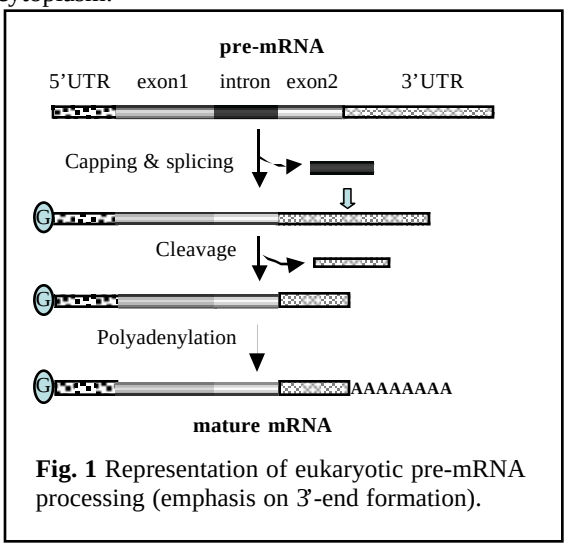


Fig. 1 Representation of eukaryotic pre-mRNA processing (emphasis on 3'-end formation).

Recent studies have shown that the regulation of mRNA polyadenylation is related to many cellular conditions, e.g. cancer progression, cellular immunity to pathogen attack, and differential tissue development (Chen et al., 1999; Kleiman and Manley, 2001; Kashiwabara et al., 2002). We are currently studying the regulatory sequence elements (*cis*-elements) residing in the mRNA which are recognized by a protein complex in the cells, and which direct the polyadenylation processing [where to cleavage and add a poly(A) tract]. In general, regarding polyadenylation signals, there are now more contradictions to the previously defined signals, in which less consensus signals become common (MacDonald and Redondo 2002). In plants, there is very limited information about these *cis*-elements; as such it becomes an obstacle for accurate gene annotation in many genome projects.

The availability of the vast genomic and cDNA (complementary DNA to mRNA) sequences through large-scale genome sequencing projects has provided us with many valuable databases that can be used in the search of polyadenylation signals. However, the overwhelming volume of the genome sequence data becomes a challenge for traditional biologists, who generally do not utilize advanced computational tools. The newly emerging field of bioinformatics is a representation of the marriage of the best technologies from biological and computational sciences. The bioinformatics approach complements and enhances traditional biochemical methods by examining a much larger number of genes (Graber et al., 1999). Further analysis can then be confirmed and accuracy of results verified by using a molecular approach. At present, the major principles of gene annotations are based on the identification of functional RNA and coding sequences (*Arabidopsis* Genome Initiative, 2000).

With a better understanding of the *cis*-elements, it can enhance and improve gene predictions that ultimately improve accuracy of gene annotation (Graber et al., 2002). The bioinformatics study of polyadenylation signals will open up a new frontier in gene annotation technology by predicting the ends of the genes, which will be another criterion to categorize genes.

In this paper, we report on efforts to characterize nucleotide regions of significance to polyadenylation. To this end, the Cray SV1 bioinformatics library has been exploited to look for the nucleotide patterns that are significantly over-represented in the region with predicated polyadenylation signals based on the results of traditional approaches (mutagenesis). The results of the analysis of these patterns will be tested experimentally for the significance of the patterns in polyadenylation, and also will aid gene annotation by predicting potential polyadenylation sites that indicate the ends of regular mRNA.

In addition to the Cray SV1, we had hoped to use the TimeLogic DeCypher system, but it has yet to play a role in the search for polyadenylation signals. The necessary precursor to create alignments for the 16,000 sequences either en mass or in selected groups has not produced results. The computational demand for the en masse alignment was very high, with the quality of the resulting consensus questionable. Subdividing the sequences into smaller sets through filtering and categorization proved so far ineffective at appreciably reducing the number of sequences to be aligned, or in providing an effective categorization that did not compromise the search for the unknown polyadenylation signals. Consequently, a reliable Hidden Markov Model for polyadenylation signal has not yet been created to utilize the high throughput TimeLogic DeCypher system in the large-scale search effort.

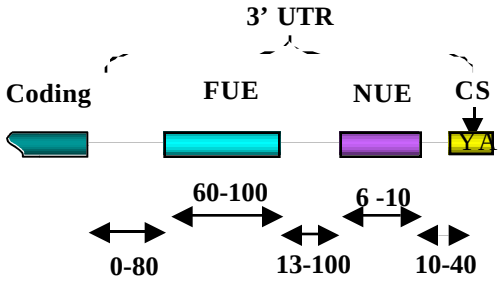
2. Motivation for Studying Model Plant *Arabidopsis*

Arabidopsis thaliana, commonly known as Thale Cress, is very suitable model for genomic studies for the reasons of its short life cycle (about 6 weeks), ease of genetic transformation, small genome size (only 125 million nucleotide base pairs, compare to human with 3 billion base pairs), and the availability of its complete genomic sequences. It has been recognized as one of the best genetic models in biological studies for the past 15 years. Due to the commonality of the genetic code and homology of the majority of the genes in eukaryotic organisms, the results from the studies of such model systems can be used in other systems. Under the similar token, our results with *Arabidopsis* can be applied to studies focusing on other important agricultural crops, such as soybeans, tomatoes, rice, corn, etc.

3. The Working Model Used for the Study

A working model (Figure 2) illustrates the arrangement of plant polyadenylation *cis*-elements, based on the genetic studies done in plant systems (Rothnie 1996; Li and Hunt 1997). It shows possible positions in which to search for the polyadenylation signals.

Figure 2. The working model for signals of plant mRNA polyadenylation. 3'UTR, 3' untranslated region; Coding, the part of the sequence coding for protein information; FUE, far upstream element; NUE, near upstream element; CS, the cleavage site [or the poly(A) site]. YA, Y represent either nucleotide C or U (or T in DNA), cleavage occurs between the nucleotide Y and A. The numbers (in nucleotides) are the relative position or size of the elements.



4. *Arabidopsis* Data File

The DNA sequences collected from the 3' UTRs of the full-length cDNA was kindly provided by Dr. Brian Haas from The Institute for Genome Research, Rockville, Maryland. The database, in FASTA format with header and sequences in text form, contains 16,255 such sequences. The sequences range in length from 1 character to 3118 characters long, with an average sequence length of 231.

5. Cray SV1ex and Bioinformatics Library

Several years ago, programmers within Cray realized that the special hardware features that have been part of the product for years would be useful in solving gene sequencing types of problems. This inspired developers within Cray to build a bioinformatics specific library to make these hardware features more available to programmers. The name of this new library is the Cray Bioinformatics Library (CBL) and is originally targeted for the Cray SV1ex platform. The hardware features accessible from the CBL include: popcnt (count of number of one bits in a word), leadz (count the number of leading zero before the first one bit), vector shift (treat the entire vector register, 64 words or 4096 bits as one register and shift it left or right), bit matrix multiply (multiply two 64 by 64 bit matrices in a bit-wise sense, AND the rows and columns, and XOR the result).

The CBL includes routines for: reading sequences from FASTA files into a continuous memory array, routines for compressing and uncompressing this data with 2, 4, 5, or 6 bit compression, routines for searching and sorting this data, routines for storing and retrieving data from the secondary storage device (SSD), and routines for copying data, with bit start and stop location, along with other memory management routines.

6. Programming techniques for NUE search

The first step in any program is to read in the data. For this project the data comes in the form of a FASTA file. A FASTA file can contain one or multiple sequences and is a readable ASCII file. Each sequence in the file has two parts, the sequence title and the sequence itself. The title line starts with the character ">" at the beginning, and end with a carriage return (\n). The next part of the FASTA file contains multiple lines that are the sequence itself. The end of the sequence is identified by the end-of-file character or ">" which marks the beginning of the next sequence. The data for this program is simply read by using the cb_read_fasta routine. This routine moves the data from the file to memory and arranges the data such that all the header records are place in one array and all the sequence data is place in another array with the sequence data spaced out on multiples of 4 word boundaries. The routine returns arrays containing information about the number of sequences read and the length of these sequences, in characters. The routine also returns the starting address for each sequence.

Typically the next step would be to compress all the sequence data. Since the sequence data contain only 4 unique characters {A, C, G, T} only 2 bits are needed to store this information. Reviewing the ASCII character set for {A, C, G, T} in binary format we see the following.

```
A = hex: 41 = binary: 0100 0001
C = hex: 43 = binary: 0100 0011
G = hex: 47 = binary: 0100 0111
T = hex: 54 = binary: 0101 0100

a = hex: 61 = binary: 0110 0001
c = hex: 63 = binary: 0110 0011
g = hex: 67 = binary: 0110 0111
t = hex: 74 = binary: 0111 0100
```

Note that the second and third columns (from the right) contain a unique 2 digit code for the characters. Namely: A=00, C=01, G=11, T=10. Also note that this 2 digit code also works for the lower case characters a=00, c=01, g=11, t=10, thus the 2 bit compress works irregardless of whether the input data is in upper or lower case.

For our initial search for the most common pattern in the NUE region we chose not to compress the entire file, since we were only looking at a small, 6 to 10 character field out of an average sequence length of 231 characters. Instead, the program extracts small parts of the sequence from the NUE region and compresses them.

To find the most common pattern in the NUE region we first have to look at the maximum number of possible patterns that could be generated with the 4 characters {A, C, G, T} in the maximum 10 character field. This number is 4^{10} or a little more then a million possible combinations. This number is a reasonable size, so the program allocates an array called a pattern_frequency, 4^{10} elements long. To start the search, the program loops over the 5 different pattern lengths, 6 through 10, performing a new search for each length. The NUE region was initially defined as starting from 10 to 40 characters upstream from the cleavage site, but since the program runs in just a few seconds on a single SV1ex processor the search was expanded to locations 1 through 50. The program starts with the pattern length of 6 and location 1. It then copies the data from the sequence data structure from the first sequence into a variable called nue_pattern. This copy is done with the CBL routine cb_copy_bits, because of its easy mechanism for copying random characters from one array to another, even if they cross word boundaries. The variable nue_pattern is then sent to the CBL routine cb_compress, which performs the 2 bit compression on it and writes it to the variable nue_pattern_compressed. By shifting this compressed bit pattern to the right by word length (64 bits) minus the length of the NUE pattern (12 bits), the result is an integer between 0 and 4095. (The number 4095 or 4^6-1 is the maximum number of 4 character combinations for a pattern of 6 characters.) This integer is then used as an index into the pattern_frequency array and the pattern associated with that index is incremented by one. The program then repeats the inner loop and the next pattern is copied from the sequence data structure for the same sequence, starting at location 2, using the same length of 6 characters. The pattern is again compressed and shifted to form a new index that is used to increment the pattern_frequency array associated with that pattern. This continues until all 50 starting locations have been completed for the first sequence. The program then moves to the second sequence in the list and does the same operations for the 50 starting locations of that sequence, incrementing the pattern_frequency array.

When all the sequences have been searched at all 50 starting locations for the 6 character NUE, the program then loops through the pattern_frequency array to find the pattern with the largest count, or the most common pattern for the patterns with 6 characters within the NUE region from locations 1 through 50. The program also continues the sorting and prints out all the patterns that have a pattern_frequency count greater then zero, along with the count, and the minimum and maximum locations where it was found within all the sequences.

The program then zeros out the pattern_frequency array, increments the pattern length from 6 to 7, and starts the whole process over. Again, at the end of each trip through the outer loop, the program prints out all the non-zero patterns and their counts for that pattern length, sorted from largest to smallest.

While running the program, it was discovered that additional code needed to be added to check for very short sequences. While analysing the sequences with a different program, it was discovered that some of the sequences were a single character long, along with many sequences being less than 50 characters long. Therefore, code was added to ensure that the pattern starting location plus the pattern length was less than or equal to the sequence length. Otherwise it would skip that sequence.

Another problem arose while examining the results. Several of the sequences contained a long stretch of a single character or double character pattern repeated. For example, the pattern TTTTTT was found to be the most common pattern. This pattern was located 3394 times out of the 16255 sequences. The trouble with the results is that long stretches of T's get counted multiple times. Consider a stretch of 10 T's. The pattern TTTTTT (6 T's) is found at the first location (far right). Then the code moves one character to the left and finds the same pattern again and increments the counter again. The program moves to the left one more character and finds the pattern again. So a pattern of 6 T's will get counted 5 times in a sequence of 10 T's. This biases the results toward these long single and double character repeats.

To fix the problem, additional code was added to check whether this pattern has already been found in this sequence. In this way, every pattern is counted only once per sequence. With the modified code the most common pattern is AATAAA, with 2021 occurrence in the 16255 sequences.

Since the original problem involves knowing neither the most common pattern nor it's location within the NUE region, the program needed to give more information about the location of the most common pattern. This was accomplished by expanding the pattern_frequency variable from a one-dimension array into a two-dimensional matrix. The length is still 4^{10} elements long, but now has a second dimension of 50 elements wide. Thus, in addition to accumulating the simple pattern frequency count, we can also count patterns based on their location within the sequences.

7. NUE Search Results and Performance

As explained in section 6, the code took on several different variations, based on whether we wanted to count the pattern once or multiple times per sequence and whether we wanted the results on a per pattern or pattern/location basis. Listed below are the top 10 most common patterns found in the sequences. The number to the right of the pattern is the number of times it was found. Presented are the results for both multiple matches per sequence and single matches per sequence.

6 Character Pattern Length

Multi-count		Single-count	
TTTTTT	3394	AATAAA	2021
AATAAA	2242	TTTGTT	2021
TTTGTT	2224	TTTTGT	1993
TTTTGT	2122	TTGTTT	1859
TTGTTT	2025	TGTTTT	1806
AAAAAA	2006	ATTTTT	1795
ATATAT	1945	TTTCTT	1720
TGTTTT	1937	TTTTAT	1695
ATTTTT	1897	TATTTT	1662
TTTCTT	1858	TTTTTT	1629

7 Character Pattern Length

Multi-count		Single-count	
TTTTTTT	1717	TTTGTTT	972
TTTGTTT	1043	AATAAAA	945
AATAAAA	988	TTTTGTT	940
AAAAAAA	982	TTTTTTT	819
TTTTGTT	980	TTGTTTT	796
TATATAT	951	TTTTCTT	767
ATATATA	895	TTTCTTT	758
AAATAAA	835	AAATAAA	757
TTGTTTT	832	TTTTTGT	724
TTTCTTT	803	TTTATTT	716

8 Character Pattern Length

Multi-count		Single-count	
TTTTTTTT	898	TTTTGTTT	436
ATATATAT	492	TTTGTTTT	429
TATATATA	479	TTTTTTTT	406
AAAAAAAA	475	TTTTTCTT	357
TTTTGTTT	459	AAATAAAA	347
TTTGTTTT	447	TTTTCTTT	341
AAATAAAA	363	TTTTTGTT	337
TTTTTCTT	361	ATTTTTTT	326
TTTTCTTT	349	TTTCTTTT	322
TTTTTGTT	338	TAATAAAA	319

9 Character Pattern Length

Multi-count		Single-count	
TTTTTTTTT	499	TTTTTTTTT	234
TATATATAT	290	TTTTGTTTT	194
ATATATATA	274	TATATATAT	177
AAAAAAAAA	229	TTTGTTTTT	168
TTTTGTTTT	206	TTTTTCTTT	162
TTTGTTTTT	169	ATATATATA	161
TTTTTCTTT	164	TTTTCTTTT	155
ATAAATAAA	158	ATTTTTTTT	154
TTTTCTTTT	158	TTTTTGTTT	154
TTTTTGTTT	155	ATAAATAAA	146

10 Character Pattern Length

Multi-count		Single-count	
TTTTTTTTTT	273	TTTTTTTTTT	123
ATATATATAT	178	ATATATATAT	93
TATATATATA	171	TATATATATA	90
AAAAAAAAAA	90	ATTTTTTTTT	82
AATAAATAAA	83	TTTTTTTTTG	81
ATTTTTTTTT	82	TTTTGTTTTT	79
TTTTTTTTTG	81	TTTTTTTCTT	78
TTTTGTTTTT	79	AATAAATAAA	76
TTTTTTTCTT	78	GTTTTTTTTT	74
TTTTTCTTTT	74	TTTTTCTTTT	73

The code is very efficient to run. The time the program spends in the different sections of the code is as follows: reading the FASTA file takes about a tenth of a second, scanning all sequences for all 50 locations and filling the pattern_frequency array takes about six seconds; and scanning the pattern_frequency array for the maximum value also take about a tenth of a second. Writing the output can take the most amount of time, depending on what output is being written.

The results for the two-dimensional matrix of results, pattern frequency verses location is shown on the following graphs, Figures 3 and 4. Figure 3 shows what the model expected. Several patterns which are more common, (having a much higher frequency count) in the NUE region. What was not expected was the occurrence of an additional set of patterns that show up nearer to the cleavage site, Figure 4.

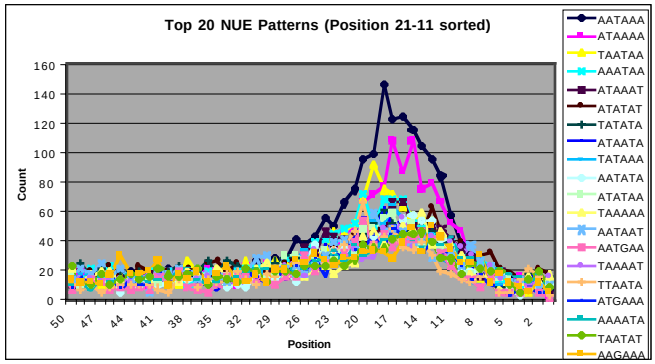


Figure 3. Top 20 NUE patterns in the position of 11-21. This is the area we expected to locate the NUE patterns in the 3'UTR. The patterns are listed on the right.

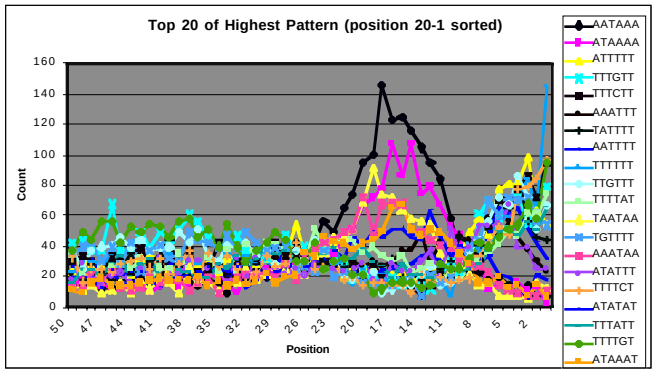


Figure 4. Top 20 NUE patterns sorted according to counts position 20-1. A set of new patterns was found that was unexpected according to our model (Fig.2) in the position 1-10.

8. Possible Programming Techniques for FUE

The next part of this research project is to search the FUE region for the most common pattern. The first difficulty with the FUE search is the length of pattern. As stated in section 6, the maximum possible combination for 4 characters in a 10 character wide field is 4^{10} or 1,048,576. With the FUE region being 60 to 100 characters wide the maximum number of combinations shoots up to 4^{100} or 1.6E60. Since the SV1ex has an addressable memory space of two billion words (due to the 32 address word size), the largest width for all possible combinations are 15 characters, with 4^{15} having 1,073,741,824 possible combinations. This limit could be stretched a bit further packing multiple counts per word, but it would only yield a maximum character width of 4^{16} or 4 billion combinations.

The solution will be to move from a pure exhaustive search to a semi-exhaustive search. The plan is to perform the FUE search using a 15 character FUE length. Once the most common 15 character FUE is found, the counters can be zeroed out and the search performed again, only this time looking for 30 character wide combinations with only the first 15 characters changing. Having finished this second search the counter will be zeroed again and another search performed looking for 45 character patterns with the first 15 character changing and the last 30 characters fixed. Using this bootstrapping approach, we hope to find the most common 60 to 100 character pattern in the FUE region.

From our experiences with the NUE search where each search or filling of the pattern_frequency array only took seconds to fill, this approach for the FUE search should still take on the order of minutes to complete on a single processor.

A final step for this project is to combine the results from the NUE and FUE searches. This phase of the research has not yet been outlined, and will be the topic of another discussion.

9. Conclusion

We are able to use the CBL to compile the polyadenylation signals using Cray SV1ex platform. The success of the search reveals the NUE patterns that are in the expected range of the location, judging by traditional molecular models. Moreover, the computational approach enables us to discover the previously unknown consensus of polyadenylation signals located in the cleavage site between positions 1-11. The authenticity of the signals remains to be tested by experimental approaches.

Acknowledgments

We would like to thank Cray, Inc. for the use of the SV1ex located in Chippewa Falls, Wisconsin, and Ohio Supercomputer Center for use of their SV1 located in Columbus. We would like to thank Dr. Brian Haas (TIGR) for the use of their *Arabidopsis* downstream database.

About the Authors

Johnny Loke is a graduate student in the Department of Botany at Miami University located in Oxford, Ohio. Johnny can be reached at the Department of Botany. Miami

University, 316 Pearson Hall, Oxford, OH 45056, (513) 529-4209, lokecj@muohio.edu. Dr. Q. Quinn Li is an assistant professor in the Department of Botany at the Miami University. Quinn can be reached at the Department of Botany. Miami University, 316 Pearson Hall, Oxford, OH 45056, (513) 529-4256, liq@muohio.edu. P. Chris Wood is the manager of the Center for Functional Genomics and Bioinfortics, Miami University, Oxford, Ohio 45056, (513)-529-4280, woodc@muohio.edu. Dr. Eric Stahlberg is a senior systems manager at the Ohio Supercomputer Center and an adjunct faculty in computer science and biomedical informatics with emphasis on computer algorithms. Eric can be reached at OSC, 1224 Kinnear Road, Columbus, OH 43212, (614) 292-2696, eas@osc.edu. Dave Strenski works for Cray Inc., and is located at a customer site in Michigan. He can be reached at Cray Inc., 7077 Fieldcrest Road, Suite 202, Brighton, Michigan, 48116, (313) 317-4438, stren@cray.com.

References

Arabidopsis Genome Initiative, 2000. Analysis of the genome sequence of the flowering plant Arabidopsis thaliana. Nature 408, 796-815.

Chen, Z., Li, Y., Krug, R.M., 1999. Influenza A virus NS1 protein targets poly(A)-binding protein II of the cellular 3'-end processing machinery. EMBO J 18, 2273-83.

Graber, J.H., Cantor, C.R., Mohr, S.C., Smith, T.F., 1999b. Genomic detection of new yeast pre-mRNA 3'-end-processing signals. Nucleic Acids Res 27, 888-94.

Graber, J.H., McAllister, G.D., Smith, T.F., 2002. Probabilistic prediction of Saccharomyces cerevisiae mRNA 3'-processing sites. Nucleic Acids Res 30, 1851-8.

Kashiwabara, S., Noguchi, J., Zhuang, T., Ohmura, K., Honda, A., Sugiura, S., Miyamoto, K., Takahashi, S., Inoue, K., Ogura, A., Baba, T., 2002. Regulation of spermatogenesis by testis-specific, cytoplasmic poly(A) polymerase TPAP. Science 298, 1999-2002.

Kleiman, F.E., Manley, J.L., 2001. The BARD1-CstF-50 interaction links mRNA 3' end formation to DNA damage and tumor suppression. Cell 104, 743-53.

Li, Q.Q., Hunt, A.G., 1997. The polyadenylation of RNA in plants. Plant Physiology 115, 321-325.

MacDonald, C.C., Redondo, J.L., 2002. Reexamining the polyadenylation signal: were we wrong about AAUAAA? Mol Cell Endocrinol 190, 1-8.

Rothnie, H.M., 1996. Plant mRNA 3'-end formation. Plant Mol Biol 32, 43-61.

Zhao, J., Hyman, L., Moore, C., 1999. Formation of mRNA 3' ends in eukaryotes: mechanism, regulation, and interrelationships with other steps in mRNA synthesis. Mol Biol Rev 63, 405-45.