Computational Approaches to Gene Finding

by

Eric Banks

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2000

Author .....
Department of Electrical Engineering and Computer Science
May 2000

Certified by .....
Bonnie Berger
Samuel A. Goldblith Associate Professor
Applied Mathematics
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses
Department of Electrical Engineering and Computer Science

DISCLAIMER

Page has been ommitted due to a pagination error by the author.

Reference page is listed as p.35 but is labeled p.37. No pages are actually missing.

Acknowledgements

First and foremost, my greatest debt of gratitude goes out to Bonnie Berger, my thesis advisor and employer for the past two and a half years. Of course, the thanks I owe her stem from my research: she allowed me the freedom to work on any project that I wanted, always had enough real work for me, and put her complete trust in me to get the job done. But, there's so much more for which she deserves my undying gratitude: from the time off she allowed me in the middle of the academic year to grow as a person and not just as a student, to the real-life advice she offered whenever we discussed anything about my future plans (don't worry, Bonnie, for now I'm living in Boston...). The list is endless, so I'll just end with: Thanks.

None of my research would have been possible without the help of several other graduate students working for Bonnie. In particular, to Lior Pachter I owe my first job in Computational Biology. He took a chance on a green recruit with a big red mark on his past record (yes, I had the lowest score in the entire class in Lior's freshman calculus class. And I *still* continued to skip class!) and showed me a love for the subject. Working with/for him and Serafim Batzoglou was a tremendous experience, teaching me not only about biology but also the ins and outs of computer science. Last, and certainly not least, my work with Alex Coventry is actually the basis of the work in this thesis: his advice and assistance were crucial to the whole project. I especially owe him a big debt of gratitude for his help in the final few days before the thesis was due: Thanks, Alex.

Finally, a special thanks goes out to those here at MIT who took care of me and made sure that my work got done (Alan Mizrahi, Ian Schechter, Michael Kaminsky) and also to those who took care of me and made sure that it didn't get done until the last minute (Uriel Schafer, Michal Mirvis, Josh Kutin, Harry Pell). Of course, there are others, so I'll just thank the MIT ortho minyan in general: nothing can stop us.

Computational Approaches to Gene Finding

by

Eric Banks

Submitted to the Department of Electrical Engineering and Computer Science on June 2000, in partial fulfillment of the requirements for the degree of Master of Engineering

Abstract

With the onset of The Human Genome Project, large databases have been filled with accurate, but unannotated, genomic sequences. Without the ability to determine the exact locations of the actual genes within these sequences, however, these databases are useless. Among the many methods used in verifying genomic regions is the search for homologous regions between an unannotated target sequence and a previously annotated sequence from a different species. This paper explores the use of homology as part of an overall gene finding program and determines its ability to aid in annotating gene regions.

Thesis Supervisor: Bonnie Berger
Title: Samuel A. Goldblith Associate Professor of Applied Mathematics

Table Of Contents

List Of Tables

Chapter 1

Gene Recognition

1.1 Background

A strand of DNA is simply a linked list of molecules, called nucleotides or bases. The alphabet for DNA is very restricted: there are only four possible nucleotides (A=adenine, C=cytosine, G=guanine, T=thymine), and a combination of them comprises the entire DNA strand. In the transcription process, which occurs within each cell, these nucleotides are translated into parallel bases on an mRNA strand. Eventually, the mRNA itself is translated into useful proteins: groups of three consecutive bases on the strand are coded into other molecules, called amino acids, where the actual amino acid formed depends on the three bases used to encode it. The amino acids are all bonded together to form one long protein chain. Not all the nucleotides, however, are translated: whole sections of the mRNA strand, called introns, are skipped or "spliced out." One or more of the translated regions, called exons, comprise a gene, which can now be roughly defined as a sequence of nucleotides necessary for the synthesis of a functional protein molecule.

With the onset of The Human Genome Project, more and more strands of DNA from various species are being collected and recorded every day. Although some pieces of these strands have been annotated in a laboratory such that we know exactly where their exonic and intronic regions lie (and, consequently, the proteins they will ultimately encode), the vast majority of them have not. Without these annotations, a gene sequence provides us no useful information: it is merely a list of seemingly random nucleotides. If

one could, however, determine the locations of the genes on a given sequence, the possibilities of what can be gained are unfathomable. Unfortunately, the annotation methods that can be employed in laboratories are terribly slow and the amount of time it takes to mark a single sequence fragment is unacceptable, let alone an entire genome. Therefore, much faster computational methods are necessary to complete this process by using the previously recorded annotations to predict the locations of the various genes on the unannotated sequences.

1.2 The Challenges

The prediction process is no simple task. Several factors have contributed to the difficulty of the problem, which, when compounded with the dearth of accurate laboratory annotations, has drastically impaired the accuracy of most prediction programs.

1.2.1 Frames

Recall that amino acids are formed from groups of three nucleotides or codons. Looking at a random base, though, one cannot be sure of its placement within the triplet, called its frame. Furthermore, it is very possible that when an intron breaks up a gene into several exonic regions, it does so in the middle of a frame. Therefore, even if one could determine the exact starting base of a given exonic region, one cannot assume that the base is the first of the three bases in its codon. Depending on the frame in which an exon or gene is translated, the resulting proteins can be vastly different.

1.2.2 Exons vs. Introns

Not much information is known about the differences in the relationships between bases in exonic regions and those in intronic regions. Were the relationships more apparent one could use simple probabilistic computations for the predictions; as it stands now, though, more complex calculations must be employed.

1.2.3 Size

The gene sequences themselves are huge; therefore, one cannot employ just any algorithm for searching gene databases or for annotating sequence fragments. Care must be taken to ensure that efficient algorithms are developed to run a gene recognition program on the gene databases.

Chapter 2

Research

2.1 Previous Work

Several programs which have been developed to detect similarities between different sequences are often manually applied to gene annotation. Probably the most widely used is a search tool called BLAST (S. F. Altschul et al. 1990). It uses alignment algorithms to compare protein or DNA sequence queries to protein or DNA sequence databases. FLASH (I. Rigoutsos and A. Califano, 1994) is a similar concept which uses hash tables to keep track of sequence matches.

Many other computational methods have been developed to perform direct gene annotation. A broad categorization of such programs designates the majority of them as statistically based. Several are fashioned on hidden Markov models, including GENIE (D. Kulp et al. 1996), VEIL (J. Henderson et al. 1997), and GENSCAN (C. Burge and S. Karlin, 1997). GENSCAN differs from other gene finding algorithms of its kind in that it allows for partial genes as well as complete genes and for the occurrence of multiple genes in a single sequence, on either or both DNA strands. GRAIL (Y. Xu et al. 1994) is an alternate approach with uses neural networks for gene prediction, while FGENEH (V. V. Solovyev et al. 1995) employs various statistical methods for finding exons.

A second major categorization of gene recognition programs includes homology-based methods. These will be discussed further in the next chapter.

2.2 Berger's Gene Recognition Group at MIT

Our research group has been designing a large program to battle the problem of gene recognition both efficiently and accurately. The program consists of two main parts: the scoring functions and the parsing mechanism.

2.2.1 Scoring Functions

There are certain signals within a sequence that can be used as flags for the possible existence of an exonic region. For example, the first exonic region of a pre-translated protein will always begin with an ATG basepair triplet. Given a random ATG triplet within a sequence, we determine its probability of being a true exon start. These calculations are performed by the ATG scoring function, which gives a particular triplet a score based on its probability.

Other examples of interesting signals are the splice sites on either end of an exon (C. Burge and S. Karlin, 1997). In the vast majority of sequences the same two nucleotides occur at the boundaries between exons and introns. The beginning of any intronic region will most probably be a GT pair, and the same region will end with an AG. For each of these possible signals, we have a function to assign a probabilistic score as to whether or not it is a true splice site or a pseudo site.

2.2.2 Parsing Mechanism

Once all the scores have been collected, the program uses them to run a dynamic program-

ming algorithm which attempts to make the optimal parse of the gene sequence so as to maximize total scores. Normal dynamic programming algorithms applied to this process usually take time $\Theta(n^2)$, which is again unacceptable given the huge amount of data necessary to be parsed. One method we employ for speeding up the parsing is the use of a special linear time algorithm. While that does drastically improve the running time of the program, there is still room for improvement. Other methods are still necessary to make this program a realistic venture.

2.3 Thesis Objective and Contributions

This thesis examines the use of homological indications to reduce the length of the segments being parsed. Homology is a likeness in structure and function between parts of different organisms due to evolutionary differentiation from the same or a corresponding part of a remote ancestor. We compare annotated gene sequences of one species to unannotated sequences of another species and look to find considerable similarities in DNA or protein structure. Given a region that displays such a similarity, we can assume that its function is the same as its corresponding region in the annotated sequence. This region, then, no longer needs to be parsed by the program because we assume that it lies within an exonic region. Furthermore, we can use this homology information to extend our knowledge of the region in which a particular match lies. Besides the obvious decrease in runtime of the recognition program, the homology information then may also assist in making the scores themselves more accurate.

Chapter 3

Homology

What we refer to as "homology" is a powerful, new approach to gene recognition by using cross-species sequence comparison, by simultaneously analyzing homologous loci from two related species (S. Batzoglou et al. 2000). The homologous regions, or homologs, may be either *orthologs* or *paralogs*. These terms are used as defined in (Fitch, 1970): orthologs are genes that have the same function in various species and that have arisen by speciation; paralogs are other members of multigene families.

3.1 Related Work

Other homology based approaches have been developed to identify the exonic coding regions of a gene. PROCRUSTES (M. S. Gelfand et al. 1996) is a good example of a successful gene finder. It is based on a spliced alignment algorithm which explores all possible exon assemblies and finds the multi-exon structure with the best fit to a related protein. Unlike other existing methods, PROCRUSTES successfully recognizes genes with short exons as well as complicated genes with more than 20 exons.

3.2 The Dictionary [1]

A central component of our gene annotation approach is the fragment-matching problem.

That is, given a gene sequence and a database of previously annotated sequences, we would like to find all the matches of length above some threshold between the target and the database. This is a classic string-matching problem, and there are linear-time algorithms for it. The problem with such an approach, though, is that the sizes of the databases we are interested in matching against preclude the possibility of real time computation. Rather than searching through the libraries for each target sequence, or worse, for each potential matching region, we make only a single pass through them. To accomplish this task, we perform some precomputation on the databases: given a minimum match length, or cutoff, we assemble dictionaries of all possible subsequences in a given library and in which sequences those matches occur. The sacrifice made by requiring a large space to store these dictionaries is easily offset by the tremendous gain in never again having to search through the libraries.

3.2.1 Dictionary Construction

Formally, a dictionary is based on a plain sequence file, consisting only of accession codes, which identify a particular sequence within the database, and corresponding sequences of strings. A dictionary hit is a match between some segment in the target sequence and a matching sequence in the database. Sequences and tuples are indexed by integers for the purpose of lookups in the dictionary. The dictionary is organized into six components that collectively enable each of the following operations to be performed in constant time:

- Find a sequence given its integer number.
- List all the sequences that contain a given tuple.

1. The bulk of this subsection appears in L. Pachter, S. Batzoglou, V. I. Spitkovsky, E. Banks, E. S. Lander, D. J. Kleitman, and B. Berger. A dictionary based approach for gene annotation. *Journal of Computational Biology*, no. 3-4 (Fall-Winter): 419-430, 1999.

- Find the accession code of a sequence from its integer number.

Also, the accession code for a sequence can be used to find the sequence number in $\Theta(\log n)$ time using a binary search.

3.2.2 Using the Dictionary to Find Matches

For a given hit, the dictionary returns the following information:

- The position in the target sequence where the match begins.
- The position in the matching database sequence where the match begins.
- The length of the match.
- The accession number of the matching sequence in the database.

The position in the matching sequence is important in determining whether nonadjacent hits in the target sequence correspond to consecutive segments in the matching sequence (thus indicating the presence of an intron). Returned hits correspond to the longest subsequences in the target sequence that completely match segments of the matching dictionary sequence.

3.2.3 Other Applications

The dictionary approach described here lends itself to a number of other applications used in gene prediction, which can further improve the parsing program:

- *Repeat Masking:* Dictionaries can be built from known repeat databases and used for rapidly finding repeat segments in genes. The method is especially useful for exon prediction, where it is advantageous not only to mask complete repeats, but to mask segments that do not occur in exons as well.

- *Different tuple patterns*: The construction of the dictionary can be based on arbitrary tuple patterns and does not need to be restricted to consecutive tuples.

- *Pseudogenes*: Reverse transcribed genes that lack introns are often pitfalls for gene recognition programs. The identification of neighboring exons in inconsistent frames with no room for an intron immediately suggests the presence of a pseudogene. This can be easily checked and automated using the dictionary.

3.3 Classes

The Homology superclass is divided up into several subclasses, each reflecting a different aspect of the homological signals in a gene sequence. Any given sequence is passed to each of the Homology subclasses, where matches to other sequences are determined.

3.3.1 DNA Homology

As its name suggests, this subclass searches libraries of annotated DNA sequences for long contiguous matches to a given target sequence. It uses the sequence dictionaries previously created to perform fast lookups on the data. The dictionary cutoff value we currently use allows for a minimum match length of 11, which seems to cull most of the spurious repeats.

3.3.2 Protein Homology

Recall that proteins are comprised of amino acids which are created from nucleotide triplets, called codons, in the translation process. Most of the amino acids, though, can be formed by several different codons. For example, the amino acid Alanine is constructed from any of the following basepair triplets: GC*A*, GC*C*, GC*G*, or GC*T*. Therefore, it is possible that even if spot mutations occur within an exonic region on a gene sequence, the

resulting proteins formed remain unchanged because the same amino acids are produced by the new codons. Consequently, it is very likely that specific basepairs differ in the DNA homologies of two different species, while the corresponding proteins still match exactly. To account for this phenomenon, we perform a second homology search: this time on proteins. Because we know the exact exonic regions of annotated database sequences, we can easily translate them into proteins and make new protein dictionaries. We then translate a given target DNA sequence into its corresponding protein for each of its three possible frames and perform dictionary lookups on them. Our current cutoff value for the protein dictionaries is 4, which corresponds to the value of 12 basepairs in the original DNA dictionaries.

Given that the Protein Homology appears to be more accurate than DNA Homology simply because of possible spot mutations that may have occurred, one may wonder as to why we even need the DNA Homology subclass. Although current annotations are *generally* accurate, not all are *thoroughly* accurate. Unfortunately, if the annotations have missed by even a single basepair, the entire frame of the following exonic regions is thrown off and the protein we produce is nothing like the actual one. Furthermore, we may want to run our program on a dictionary of *unannotated* sequences for which we have no protein information. Homologous matches in such a run would provide us with valuable information for the sequences of *both* species. Therefore, we use both the DNA and the protein homologies to help offset the inaccuracies and inadequacies of each individual subclass.

3.3.3 Reverse Protein Homology

This subclass is almost identical to the Protein Homology class, but it accounts for a small problem with the dictionary method for proteins. DNA is itself double stranded, and both of these strands ultimately transcribed to mRNA and then translated to proteins. The only difference in the whole process is the direction in which a strand is read. Unfortunately, sequences created from both possible strands are placed in databases and we have no way of knowing which is which. We therefore have a complementary protein class which first reverses the target sequence and then performs the dictionary lookups, to account for the possibility that the match lies in the reverse direction.

An alternative solution could have been to form dictionaries of *both* the original database sequences and their reverse complements, but that method throws away useful information. Because it is virtually impossible for a sequence to form coding regions in both its forward and reverse directions, it is helpful to know the direction in which the match occurred. Using our current method, this information is gleaned simply by looking at which subclass has detected the match in the dictionaries.

3.3.4 Repeat Homology

Various methods have been employed for determining large contiguous matches within exonic regions of two different gene sequences, thereby reducing the size of the sequence that needs to be parsed by the gene prediction program. It would also be beneficial, however, to consider reducing the size of the *intronic* regions. In general, that is a difficult task because the bases in the intronic regions seem to be, for the most part, random, and determining patterns in them is nearly impossible. Some sections of the introns, though, are definitely not random: certain viruses seem to have made a lasting impression on genomic

sequences and have left a pattern of repeated subsequences within them. We use programs such as RepeatMasker (A. F. A. Smit and P. Green, 1998) to cull these repeated regions. RepeatMasker is a program that screens DNA sequences for interspersed repeats known to exist in mammalian genomes and which are almost certainly intronic. Again, there is no need to pass these regions to the parsing routine since they will ultimately be spliced out in the translation process.

3.4 Results

The Homology classes could not yet be tested within the framework of the entire dynamic programming algorithm because the work on the gene finder is as of yet incomplete. Instead, we built several dictionaries and then ran the Homology code by itself on these dictionaries for sequences for which we already have accurate annotations. We then examined the annotations produced by the Homology code and compared them to the true annotations to determine their accuracy. As we will address in the next chapter, the dictionaries were built using sequences that were first fragmented into 500 basepair subsequences.

We obtained primate DNA sequences from GenBank's (http://www.ncbi.nlm.nih.gov/Genbank/index.html) gbpri database. The target sequences actually used were repeat masked versions of the GenBank sequences. Several random sequences were pulled from this collection and run against two different dictionaries. The sequences used all contained both exons and introns, some with multiple genes.

The first dictionary used was that of the fugu (http://fugu.hgmp.mrc.ac.uk/), or puffer fish. Its genes are smaller (because of reduced intron sizes) than those of mammals and the dictionaries created are reasonable in size, perfect for a shared workspace. Also, because its genes are smaller, fugu is one of the few species that has been completely sequenced at this point. Still, it has been demonstrated that, even though it is smaller, there are approximately the same number of genes in the puffer fish genome as there are in the genomes of mammals (S. Brenner et al. 1993). All sequences used to build this dictionary were whole genes, with both intronic and exonic regions.

The second dictionary used was comprised of mouse genome sequence fragments. They were taken from the mouse gene which is orthologous to the human gene sequences being used for the target sequences (S. Batzoglou et al. 2000). These fragments by no means comprise an entire mouse genome and, consequently, we expect there to be fewer dictionary hits than with the fugu dictionary.

It is also useful to compare the false-positive rate for exons to the proportion of the sequences that actually are exonic. If the homology code does no better than chance, then we would expect the numbers to be about the same. The total exon length of the sequences run against the fugu dictionary was 101,570 basepairs, while the total gene sequence length was 2,091,193 bases, giving a 4.63% exon rate. The sequences run against the mouse dictionary had a total exon length of 596,675 out of a total of 12,546,254 basepairs, giving a 4.54% exon rate.

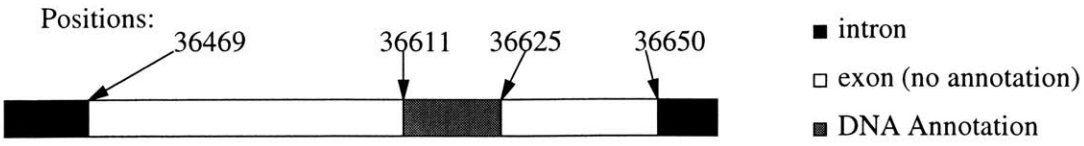

Figure 3: Example of Homology Annotation

Table 3 summarizes the results obtained after running the Homology code on our target sequences for both of the two dictionaries. Probably the most significant detail in the results is that the number of false-positive hits is relatively high; the program performs only slightly better than chance. As these preliminary results would show, the gene recognition program will be unable to assume that annotations produced by the Homology class are definitely exonic. Although this fact in a sense defeats our original purpose for creating the Homology class, the homology work still does, however, provide another useful score that we can add to the scoring function section of the program if necessary. In the next chapter we address methods for improving the performance of the Homology code.

	True-positives	False-positives	Accuracy	% Exon in Sequences
Fugu	5036	35665	12.37%	4.63%
Mouse	346	2879	10.73%	4.54%

Table 3: Homology run on fugu and mouse dictionaries.

3.5 Discussion

The dictionary approach has a number of advantages over standard similarity search techniques. Despite the unprecedented success of alignment algorithms in biology, the algorithms are all handicapped by the problem that short matching segments can be difficult to find in certain cases. There are a myriad of parameters that go into an alignment and each has to be tweaked for each individual target sequence. An advantage of our dictionary method is that when performing a database search, *all* of the exact segment matches in a sequence are rapidly detected. Furthermore, the entire process of database search and exon prediction is automated, and the prediction is based on many good sequences (and their fragments) simultaneously. And exons that have partial or no matches in a database sometimes can still be accurately predicted using the scoring function information. Also, the method takes full advantage of the presence of both long *and* short dictionary hits.

Chapter 4

Alignment: Extending the Matches

4.1 Justification

The dictionary is extremely useful in determining long continuous matches within homologous sequences. But we cannot assume that the entire protein region common to both species is exactly the same. The proteins may play similar but not the same roles in their respective species, and, although much of the protein will match, parts of it will not. The dictionary is deficient in dealing with this issue: it will find only the matches long enough to pass the cutoff requirements, but will completely miss the smaller matches surrounding it. Finding these smaller matches plays a vital role in increasing the size of the match region, which is necessary to reduce the size of the unknown regions within our gene sequence. Hopefully, with alignment capabilities added to the Homology class, we can significantly decrease the number of false-positive annotations produced in the dictionary runs.

After the dictionary matches are determined, a sequence alignment is performed on both ends of each match. We use global sequence alignment techniques (Needleman and Wunsch, 1970) to determine these smaller substring matches between the two given sequences. The scoring schemes used are capable of finer distinctions than exact matches. The particular scoring function used by this program gives a small positive number for a match and a small negative number for either a mismatch or gap. Experimental error in

DNA sequencing favors a mismatch (basepair in-place change mutation) over an insertion or a deletion, so gaps are penalized more than mismatches.

4.2 Design Issues

A simple global alignment algorithm, although easy to implement, runs in $\Theta(mn)$ time and requires $\Theta(mn)$ space, where m and n are the sizes of the respective sequences. These constraints are inadequate when dealing with extremely large sequences. Instead of using a more complex algorithm, however, the sequences themselves were made smaller: before the sequence dictionaries were constructed, the sequences were first fragmented. The reduced size of the dictionary sequences containing regions matching our target sequence enables us to use the simpler algorithm.

The fragmentation is not flawless and can be damaging to our results: if the sequence is fragmented within a homologous region, the match may no longer span a large enough single region to be picked up by a dictionary search. This issue can be avoided, however, by introducing redundancy to the dictionary. The fragmentation does still pose another problem though. RepeatMasker is programmed to find only a small number of known viral repeats; there are still many intronic repeats not culled by the repeat masking process. We assume, however, that these repeats have not been preserved completely between different species because mutations in them are inconsequential to the genomic makeup of the species. Although smaller sections of the repeats may still overlap, there is only a minimal chance that longer matches exist between two different species. Using a large cutoff length for the smallest allowed dictionary matches will ensure that we are not fooled by the small

repeat overlaps. By reducing the size of the dictionary sequences, though, we are in turn forced to reduce the cutoff value for dictionary matches. Consequently, the smaller cutoff point will cause the program to find matches that it would have otherwise ignored with the larger value. Because we would like to assume that all dictionary matches are exonic in origin, the number of false positives picked up by the program is likely to increase significantly. Even with these considerations, we have decided to maintain the smaller dictionary sequence lengths, for an entirely unrelated reason. In order to expedite the gene sequencing process, laboratories have been using shotgun sequencing techniques. The sequences produced by shotgun sequencing are fragmented and are placed as such into the gene sequence databases. In order to ensure that our program can handle such fragmentation, we have decided to fragment all our current sequences also.

Generally, the alignment is determined by walking through an alignment matrix from the right ends of the sequences towards the left. We can assume that at any point at which the matrix scores the alignment as zero or less, the resulting alignment is no longer significant enough for it to be considered a homologous area between the two sequences, even though it may still be the optimal alignment. Because we are interested in determining only the alignment matches close to the dictionary matches, this process works well for aligning the subsequences on the left sides of the matching regions. This method, however, does pose a problem for aligning those on the *right* sides of the matching regions, which should be walked from left to right. Instead of designing two different alignment procedures and conventions, we simply reverse the subsequences to the right of the matches, determine their alignment, and then reverse them back.

Starting gaps in the alignment can be completely ignored because it is not necessary to look past the end of one of the sequences for matching regions. Because the two sequences are almost certainly different sizes, allowing starting gaps would drastically reduce the scores in the alignment matrix. Trailing gaps, however, must be taken into consideration because they translate to insertions or deletions in one of the sequences within the match site.

4.3 Initial Problems

The results of the first run of the program after the alignment capability was added were disheartening. The regions annotated as alignment matches covered virtually the entire target sequence! The problem soon became apparent: other dictionary matches interfered with the results of the alignment of a single match. In general, the alignment scoring function is willing to sacrifice a short-term loss for a large match (and gain in score) further along the sequence. Consider a single match sequence which contains two regions marked by the dictionary as being long matches to our target sequence. When performing the alignment on the right-most match, for example, alignment on its left side produces exaggerated results. The alignment code sees the left-most dictionary match further down, and its match score is so large that it is willing to insert gaps in the alignment in order to reach the other match.

A simple solution to this problem is the use of a "hits" array. Each Homology object was given a bit vector which represented the target sequence and which was the same length as the target sequence. All dictionary matches were determined first, and, before

any alignment was performed, the hits array was marked: for each dictionary match, the positions within the bit vector corresponding to the area of the match on the target sequence were all flipped on. The alignment code, in turn, used the array to align only those subsequences falling between the end of a particular dictionary match and the next closest dictionary match. In this manner, other matches will not affect the alignment of a particular match.

Interestingly, the results of this new alignment were no different from the earlier ones. Apparently, matches from other Homology runs were also interfering with one another. For example, a Repeat match, although ignored by the dictionary, will magnify the scores of the alignment matrix when it falls between two dictionary matches from the same Homology type.

The solution in this case was to make the hits array a single global variable to the Homology class. Now, all dictionary matches are determined first for all Homology classes before any alignment is performed. The bit vector can then account for matches in all Homology classes and eliminate overlaps between the matches and the alignment.

4.4 Results

Table 4a summarizes the results obtained after running the Homology code, with alignment added, on the same sequences for both of the two dictionaries. We assume here that only homology matches with corresponding alignment matches are exonic. The added functionality of alignment helps the program perform slightly better than with the Homol-

ogy class by itself.

	True-positives	False-positives	Accuracy	% Exon in Sequences
Fugu	2913	14796	16.45%	4.63%
Mouse	257	1120	18.66%	4.54%

Table 4a: Homology with Alignment run on fugu and mouse dictionaries.

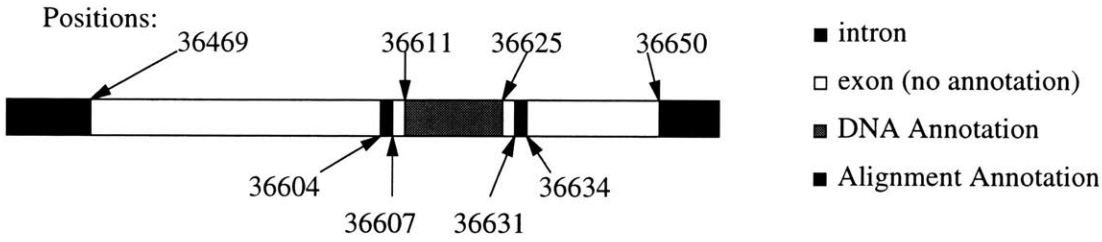

Figure 4: Alignment added to the Homology

One interesting fact, though, stands out in the new data: many of the alignment matches occur within the Protein and ReverseProtein classes in which we had set no minimum threshold for the lengths of matches. Consequently, most of the supposedly "strong" alignment matches were only one residue in length and were most probably just a product of statistical probability. To account for this fact, we performed another run over our data, this time with a minimum cutoff length of 2 residues for a Protein or ReverseProtein alignment match. The results, given in Table 4b, are much more satisfying. The false-positive

rate has been cut drastically and we finally begin to see the results we were predicting when we first set out on this project.

	True-positives	False-positives	Accuracy	% Exon in Sequences
Fugu	991	1255	44.12%	4.63%
Mouse	124	109	53.22%	4.54%

Table 4b: Threshold added to Alignment on fugu and mouse dictionaries.

Chapter 5

Summary

Using only the combination of homologous matches between the shotgunned gene sequences of two different species and a strong peripheral alignment around those matches, we have been able to determine with fairly high accuracy the locations of several exons within random target sequences. Furthermore, with still relatively good accuracy, the program predicts the location of still more exonic regions; each of these latter annotations can serve as another significant score to be employed in the scoring function section of the prediction program. This work in itself is an impressive endeavor, and in the context of the dynamic programming algorithm it will be the foundation on which the rest of the gene prediction program is based.

References

A. F. A. Smit and P. Green, RepeatMasker http://ftp.genome.washington.edu/cgi-bin/RepeatMasker, 1998

C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology* 268:78-94, 1997.

D. Kulp, D. Haussler, M. G. Reese, and F. H Eeckman. A generalized Hidden Markov Model for the recognition of human genes in DNA. *Proceedings of the 4th Conference on Intelligent Systems in Molecular Biology*, 134-142, 1996.

I. Rigoutsos and A. Califano. Searching in parallel for similar strings. *Computational Science and Engineering*, 1(2):60-75, 1994.

J. Henderson,y S. Salzberg, and K. H. Fasman. Finding genes in DNA with a hidden markov model. *Journal of Computational Biology*, 4(2):127-141, 1997.

L. Pachter, S. Batzoglou, V. I. Spitkovsky, E. Banks, E. S. Lander, D. J. Kleitman, and B. Berger. A dictionary based approach for gene annotation. *Journal of Computational Biology*, no. 3-4 (Fall-Winter): 419-430, 1999.

M. S. Gelfand, A. A. Mironov, and P. A. Pevzner. Gene recognition via spliced sequence alignment. *Proceedings of the National Academy of Science, USA*, 93:9061-9066, 1996.

S. F. Altschul, S. F. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403-410, 1990.

S. Batzoglou, L. Pachter, J. Mesirov, B. Berger, and E. S. Lander. Human and mouse gene structure: comparative analysis and application to exon prediction. *Genome Research*, 2000, in press.

S. Brenner, G. Elgar, R. Sandford, A. Macrae, B. Venkatesh, S. Aparicio. Characterization of the pufferfish (Fugu) genome as a compact model vertebrate genome. *Nature*, 366: 265-268, 1993.

V. V. Solovyev, A. A. Salamov, and C. B. Lawrence. Identification of human gene structure using linear discriminant functions and dynamic programming. *Proc. 3rd Conference on Intelligent Systems in Molecular Biology*, pages 367-375, 1995.

Y. Xu, R. J. Mural, M. Shah, and E. C. Uberbacher. Recognizing exons in genomic sequences using GRAIL II. In Setlow, J., editor, *Genetic Engineering: Principles and Methods*, Volume 16. Plenum Press, New York, 1994.