Gnomon – NCBI eukaryotic gene prediction tool

A. Souvorov [*], Y. Kapustin, B. Kiryutin, V. Chetvernin, T. Tatusova, and D. Lipman

National Center for Biotechnology Information, Bethesda MD

February 25, 2010

[*]To whom correspondence should be addressed.

1 Methods

NCBI gene prediction is a combination of homology searching with ab initio modeling. The use of ab initio is threefold: a) we use ab initio scores for evaluating the alignments and locating the optimal CDS in the alignments, b) in the case when we have a partial alignment we extend this alignment using the ab initio prediction, and c) when there is no experimental information we make an ab initio model. This process produces the gene models that could be classified as completely supported, partially supported or not supported at all. The general philosophy behind this process is that we strongly prefer to use experimental information whenever it is available.

Before we start a genome annotation we collect several data sets. First we collect all available cDNA for the studied organism and sometimes the cDNA for closely related organisms. Then we generate a Target protein set and a Search protein set. The former is a collection of the proteins that we believe should be found on the genome. Usually it includes all known proteins for the studied organism and several sets of known proteins for well-studied genomes. The latter set is a much wider collection of eukaryotic proteins. We try to align on the genome all proteins from the Target Protein Set. The proteins from the Search Protein Set are aligned only if they are similar enough to predicted models, in which case these additional alignments help in refining the models. In addition to the sequences for the homology search we create an organism specific parameter set which is used for evaluation of the ab initio scores.

The chart of the data flow is shown in the Figure.1. There are several programs that are involved in the process of the gene prediction. We use Compart which analyzes the Blast [1] hits and finds compartments which are the approximate positions of the target sequences on the genome. This program is designed to recognize gene duplications. Compart step is done separately for cDNA and proteins sets. For each compartment we make a spliced alignment using Splign for cDNA compartments and ProSplign for protein compartments. The alignments are fed into Chainer which combines partial alignments into hopefully full length or at least longer chains. Finally, Gnomon decides if the chains are full length models and extends the chains if needed. This procedure is run twice. For the first round we use the cDNA and Target protein alignments. All predicted first round models are compared with the proteins from the Search protein set and the proteins found to be good matches are aligned on the genome using ProSplign.

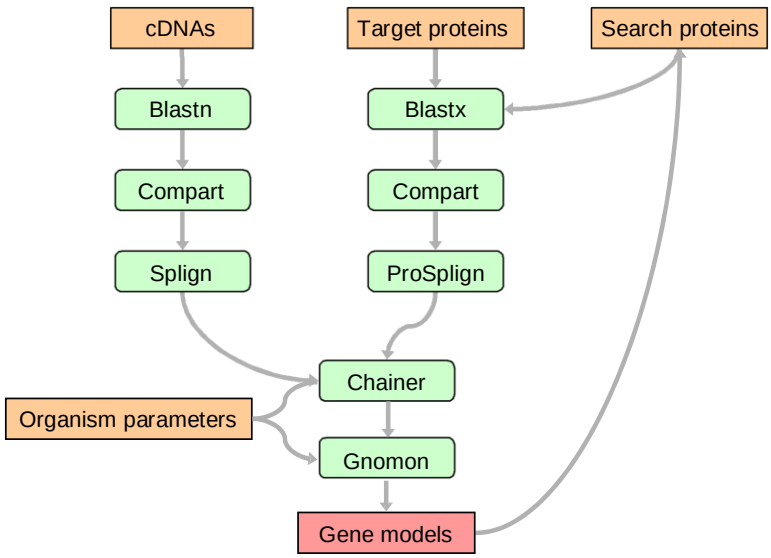

Figure 1: Available cDNAs and the Target proteins are used to build the first round predictions. These models are compared with the proteins from much broader Search protein set. Good matches are added to the support for the second round predictions. Compart finds approximate positions of the target sequences on the genome taking into account possible gene duplications. Splign and ProSplign are used to build spliced alignments. Chainer combines partial alignments into longer models. Gnomon extends partial models and creates the final annotation.

These additional alignments and all the above cDNA and Target protein alignments are used for the final round of predictions.

For each genome being annotated at NCBI we run the Gnomon procedure shown in the Figure.1. All obtained gene models are shown as Ab initio models map of NCBI Map Viewer[1]. Although the protein and RNA sequences of the models are not loaded into GeneBank, they could be searched using the genome specific Blast tool.

Our final annotation is a combination of the best placements of RefSeq mRNA alignments and completely or partially supported Gnomon predictions. If they overlap the RefSeq alignments supersede the Gnomon models. The Gnomon models with frame shifts or premature stops are usually shown as pseudo genes.

1.1 Gnomon

Gnomon is a gene prediction HMM-based program. The core algorithm is based on Genscan [4] which uses a 3-periodic fifth-order Hidden Markov Model for the coding propensity score and incorporates descriptions of the basic transcriptional, translational and splicing signals, as well as length distributions and compositional features of exons, introns and intergenic regions. The most important distinction of Gnomon from Genscan and other ab initio prediction programs is its ability to conform to the supplied alignments and extend and complement them when necessary.

Mathematically, an HMM-based ab initio prediction is a search in the gene configuration space for the gene which provides the maximal score. Sometimes due to limitations of the underlying biological model the optimal solution may be different from the real gene model we are looking for. The number of these artifacts could be considerably lowered if in gene making we take into account the independent experimental evidence which we may have in the form of alignments. If in the search space we exclude all configurations that are not compatible with the available alignments, then the optimization process in this collapsed space will yield a gene configuration which is possibly suboptimal from the ab initio point of view but exactly follows the experimental information we have. As can be seen in Fig.2 partial alignments could be extended and possibly connected by this procedure. UTR, if they present in the alignment, are also included in the gene model.

[1] http://www.ncbi.nlm.nih.gov/mapview/

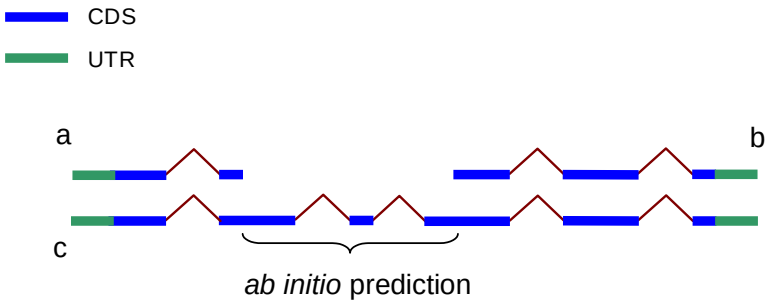

Figure 2: Gnomon searches for the optimal model in a collapsed configuration space which includes only configurations compatible with the alignments. In the process the partial alignments are extended and possibly connected. Although UTRs are not scored by Gnomon they are included in the the gene model when they are available from the alignment. In this example the final model c include two partial alignments a and b and complements them by ab initio prediction.

For the genes for which we don't yet have any experimental information Gnomon will create conventional ab initio predictions.

Following the Genscan logic Gnomon recognizes as HMM states coding exons and introns on both strands and intergenic sequences. Translational and splice signals are described using WMM [14] and WAM [16] models. A 12 bp WMM model, beginning 6 bp prior to the initiation codon, is used for the translation initiation signal [10]. A 6 bp first order WAM model starting at the stop codon is used for the translation termination signal. The donor splice signal is described by a 9 bp second order WAM model, and the acceptor splice signal is described by a 43 bp second order WAM model. Both donor and acceptor models include 3 bp of the coding exon. Coding portions of exons are modeled using an inhomogeneous 3-periodic fifth-order Markov model [3]. The noncoding states are modeled using a homogeneous fifth-order Markov model.

Lengths distribution of the states is usually peaked (or even multi-modal) near relatively short length and very smooth for longer lengths. For example, from Fig.3 we can see that human introns have a peak near 100 bp which is much more pronounced for the genes located in the areas of a high C+G

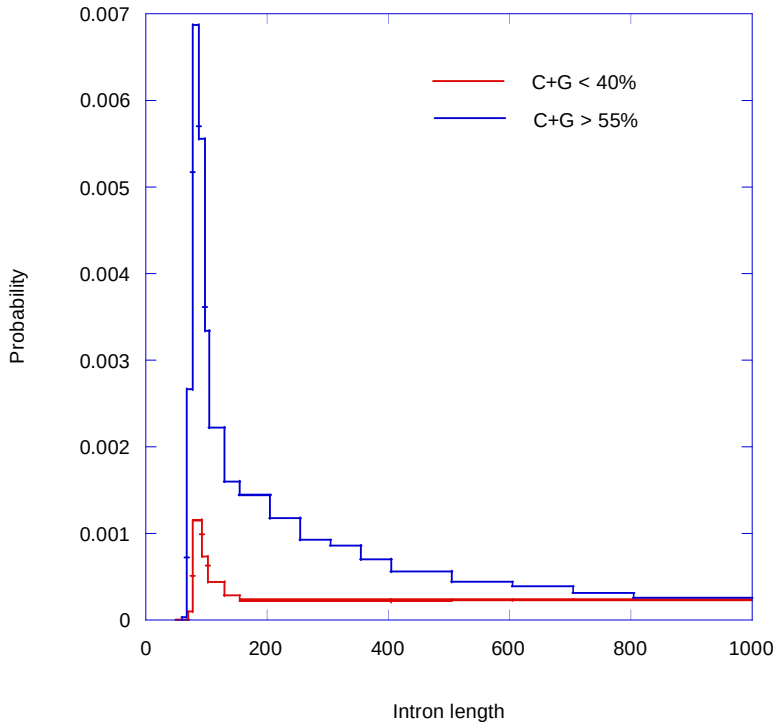

Figure 3: In human, as in all other organisms we annotated, the distribution of intron lengths has a sharp peak near 100 bp and a shoulder with a very smooth fall. In the areas of the genome with high C+G content the fraction of the introns in the peak area is much higher than in the areas with low C+G content.

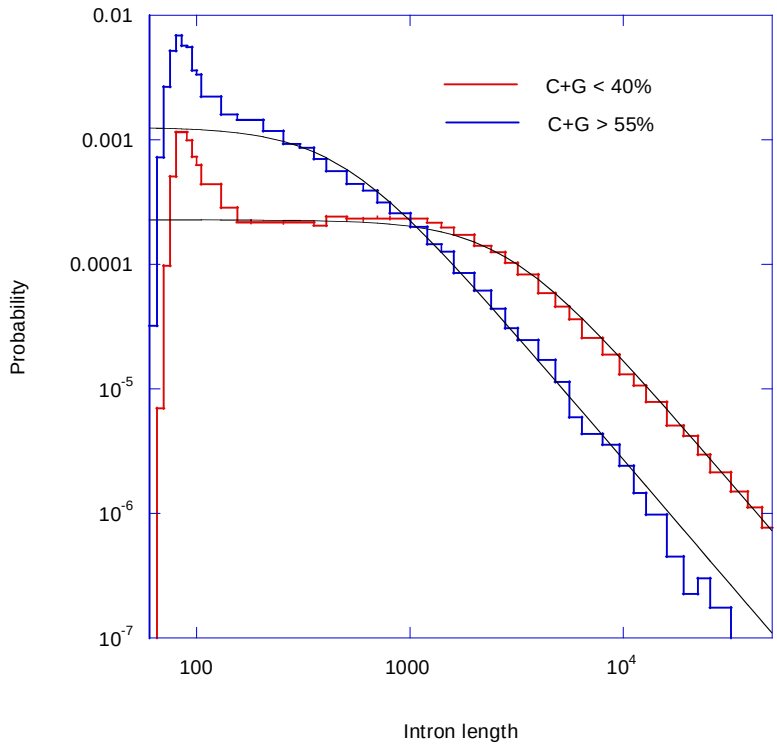

Figure 4: The length distribution of the human introns in log-arrhythmic coordinates illustrates that in the low C+G content areas the short intron peak is much lower while the shoulder of the distribution is much longer. The long intron distribution is very well approximated by the formula (1).

In Fig.4 is shown that the peak is followed by a long shoulder which is much longer in areas of low C+G content. Quite interestingly, in all genomes we studied the shoulder is very well approximated by a simple formula

$$F = \frac{A}{L^2 + l^2} \quad (1)$$

Where l is the intron length and A and L are constants. We found that all other states behave similarly, but in some cases the shoulder part of the distribution is very small.

For each new genome the described above parameters are obtained using cDNA and protein alignments with full CDS. Currently we have parameters for the following organisms: Apis mellifera, Arabidopsis thailana, Bos taurus, Caenorhabditis elegans, Canis familiaris, Danio rerio, Drosophila melanogaster, Homo sapiens, Mus musculus, Oryza sativa, Poplar, Strongy- locentrotus purpuratus, Tribolium castaneum.

1.2 Chainer

Many of the spliced alignments we obtain using Splign and ProSplign are partial, either because the aligned sequences are partial or in the case of the protein alignments because only conserved portions of the protein could be aligned well enough. Usually for each putative gene we end up having many fragments that complement each other. Analysis and assembling of these partial alignments provides longer gene models and additional information about alternative splicing isoforms [15, 9, and 11].

When number of fragments is large the number of possible assembled models for each putative gene can grow astronomically. To deal with this phenomena methods that produce a minimal set of assembled models that accounts for all available alignments have been developed [7, 5]. These methods are used to assemble cDNA (EST and mRNA) alignments. Our approach is to use all alignments including protein alignments. In addition to the exon-intron structure, the protein alignments provide information about the coding region including the frame. When we chain partial protein alignments with cDNA alignments we should avoid creating transcripts which cannot be used as legitimate gene models. For example, transcripts in which it is not possible to find a proper start codon or transcripts with two or more coding regions that could not be read through should not be created.

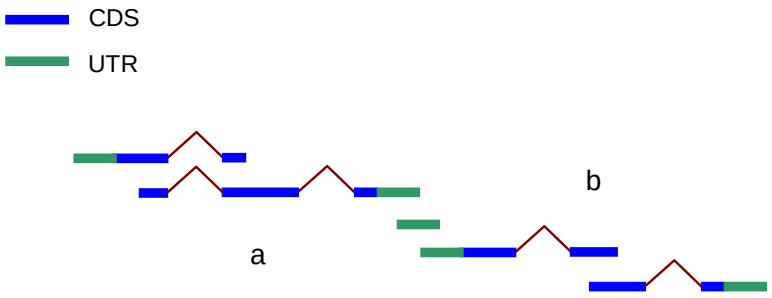

Figure 5: After we have determined the UTR/CDS nature of each alignment we can assemble them taking into account not only exon-intron structure compatibility but also the compatibility of the reading frames. For example, alignments with coding regions that cannot be connected (group a and group b in the picture) are not chained together even if they are otherwise compatible. This reduces the possibility of creating chimeric transcripts.

For our purposes we have adopted and modified the Maximal Transcript Alignment algorithm described in [7]. This algorithm provides a very efficient way of alignments assembling based on their exon structure compatibility. The protein alignments add a new dimension to the compatibility issue. Be- fore we start our assembly process we analyze all cDNA alignments, and for each one we find the best scoring CDS. For scores we use 3-periodic fifth-order Markov model for the coding propensity score and Weight Matrix Method models for the splice signals and translation initiation and termination signals. These are the same scores as we use with Gnomon (see section 1.1), our ab initio prediction tool. All cDNA with the CDS scoring above a certain threshold are marked as coding, and all others are marked as UTR. Some of the CDS may be incomplete meaning that they don't have a translation initiation or termination signal. All protein alignments are also scored the same way and if they don't satisfy the threshold criterion for a valid CDS they are removed.

After we have determined the UTR/CDS nature of each alignment we can assemble them taking into account not only exon-intron structure compatibility but also the compatibility of the reading frames. Two coding alignments

are connected only if they both have open and compatible CDS. An UTR gets connected to coding alignments only if there are necessary translation initiation or termination signals. There are no restrictions on the extension of a 5' UTR other than the exon-intron structure compatibility. To avoid creation of artificial NMD candidates [6], we don't connect a multi-exon 3' UTR. After the compatibility of the alignments is established the rest of the process is performed according to [7]. Not only does this method incorporate the protein alignments into the assembly process, it also in many cases eliminates creation of chimeric models which connect two separate genes (see Fig.5).

For each assembled model we find the best scoring CDS as was described above for individual alignments. If a model includes a protein alignment, its CDS should contain it. If the protein is not on our list of potentially partial proteins and its 5' end is fully aligned and points on an appropriate start codon on the genome, we make this start sticky. It means that the CDS could be extended beyond this point only if we have cDNA alignments that extend to an alternative start codon and the extension will increase the score at least by 10%. Extended or not, the start is marked as a confirmed start and Gnomon won't be allowed to extend it any further.

After the assembly process we may have more than one model per gene locus. From all these models we select the model with the best scoring CDS. If the best model has a complete CDS which includes the translation initiation and termination signals it is combined with other complete models, and this group forms an alternatively spliced gene. If the best model is not complete, all other models are discarded, and the model is directed to Gnomon for extension by ab initio prediction.

1.3 cDNA alignments

Splign is a tool for aligning spliced cDNA sequences against their genomic counterparts. In spirit of Est_genome [12], the program produces accurate spliced alignments via solving an optimization problem formulated specifically to account for splice signals and introns.

The formulation discriminates between the most frequent (GT/AG), less frequent (GC/AG, AT/AC) and arbitrary donor/acceptor sites. A version of the algorithm designed for the use with lower-quality genomes discriminates further by assuming possible substitutions in consensus splices and giving stronger penalties to less preserved splice signals. A lower limit on the length

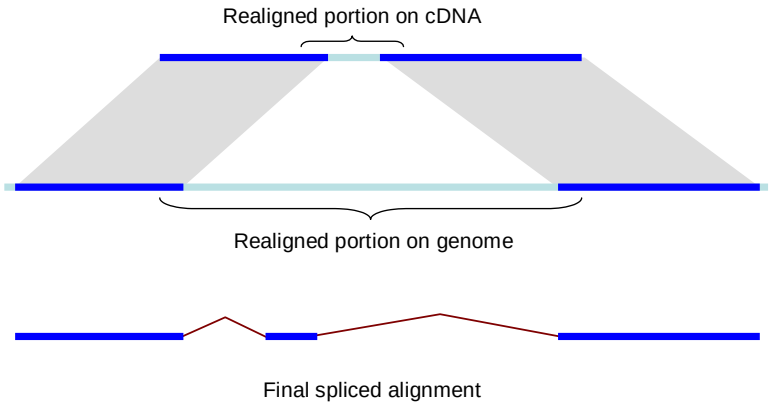

Figure 6: Splign uses the high identity portion of the hits to seed the global alignment which substantially reduces the running time.

of introns is explicitly imposed while the upper limit is implied from the compartmentalization step where no two overly distant local alignments are placed to the same compartment.

Since solving the global sequence alignment problem has the complexity proportional to the product of lengths of the sequences, the hits arranged into compartments on the prior step are used to limit search span on the genomic sequence and to split the dynamic programming matrix into smaller blocks by seeding the global alignment (see Figure.6). Both measures are critical in making the tool feasible for large-scale application.

For each compartment, its genomic search space is expanded by the amount increasing with the length of query cDNA ends not covered by local alignments. This allows detecting the end exons if they are missed by the local alignment tool for reasons such as the length shorter than the word size or being a part of a masked region. Each hit may correspond to an exon, a part of an exon, or even a number of exons. Therefore, it is important to be conservative when using local alignments for alignment seeding. Within each compartment, parts of alignments that overlap on the query are dropped. From the remaining alignments, their longest perfectly matching diagonals are extracted, and their cores are used to seed the global alignment.

While technically any local alignment tool can be used with Splign, our tool of choice in the context of Gnomon is Megablast [17] for same-species

alignments and discontinuous Megablast for cross-species alignments. The latter proved particularly important for aligning sequences from more divergent species where tools relying on perfect oligomer matching would require use of prohibitively small oligomers.

Hits comprising compartments allow to determine whether the query and the subject sequence align in the same strand, but they give no clue on what exactly strand each sequence has. Most mRNA sequences have natural biological order and positive strand can be assumed when aligning them. On the contrary, EST sequences are usually not oriented, so both the original sequence and its reverse complimentary have to be aligned and the strand is determined by comparing the resulting alignments.

Splign is written in C++ using NCBI C++ Toolkit which makes the application portable across several platforms. The source code and the precompiled executables for major platforms are available for download[2].

1.4 Protein alignments

Protein alignments are made by ProSplign. Similarly to Splign, it is a global protein to genome alignment tool which produces accurate spliced alignments. The Blastx and Compart steps precede the ProSplign step. ProSplign is designed to find alignments of even distantly related proteins with lower identity level. In these cases the Blastx hits do not give reliable information about seeds, so ProSplign doesn't use seeds and instead aligns the protein against a slightly extended genomic region identified by Compart as the compartment. Some poorly conserved parts of distantly related proteins simply may not have corresponding counterparts on the genome. ProSplign employs a global approach and aligns these parts anyway. As a result, well aligned regions and poorly aligned regions may alternate in the final alignment. As shown in Figure.7, the post processing step throws away the poorly aligned regions which correspond to not conserved portions of the protein.

ProSplign uses the regular protein Blastp scoring system. In addition to regular gaps ProSplign introduces two additional types of gaps. The frame shifts, genomic insertions or deletions which length is not a multiple of three, have a substantial additional penalty. The introns are treated similarly to the regular gaps but with a small extension cost. Special effort is taken to score properly the spliced amino acids. Since ProSplign is a global alignment

[2] http://www.ncbi.nlm.nih.gov/sutils/splign/

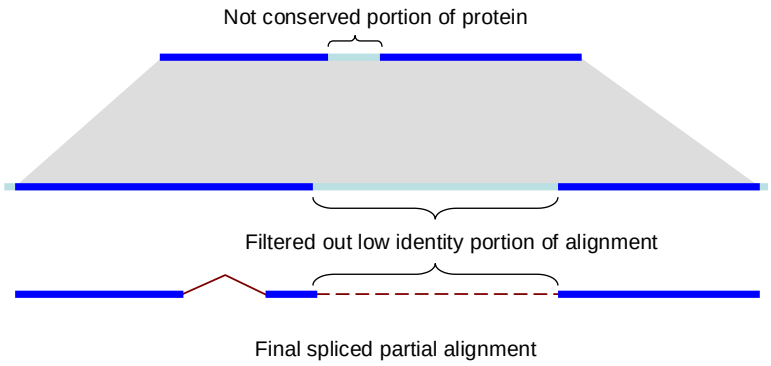

Figure 7: ProSplign globally aligns the protein against a slightly extended genomic region identified by Compart as the compartment. Very low identity portions of the alignment which corresponds to poorly conserved portions of the protein are filtered out in a post processing step. These missing parts of the alignment (shown as a dashed line) will be reconstructed later either by Chainer using other alignments or by Gnomon ab initio prediction.

program, the computational complexity of the algorithm and, even more importantly, the memory use are proportional to the product of the lengths of the involved sequences. Some genes with long introns may cover over a million bases on the genome. To address the memory issue for such cases an extra intron finding stage is introduced. It is well known that any global alignment algorithm needs memory for the backtracking information. At the introns finding stage ProSplign keeps track only of introns. After the intron (and exon) coordinates on the genomic sequence are known, ProSplign realigns the extracted from the genome transcript against the protein at a very little additional computational cost.

1.4.1 Scoring system

ProSplign scores the target protein sequence against translation of the genomic sequence which is made on the fly. Figure.8 shows an alignment ex- ample and its elements which are scored. A translation may change its frame because of an insertion or deletion or it may jump over an intron which some- times results in a spliced amino acid. By default ProSplign uses Blossum62 [8] scoring system with default Blastp gap penalties ($P_{open\ for}$ gap opening and P_{ext} for gap extension). For protein to DNA alignments it is easier to measure the gap length in nucleotide bases. Since one amino acid corresponds to three nucleotide bases, one nucleotide extension cost is set to $P_{ext}/3$. The total penalty of a gap of length of l base pairs is defined as

$$P_{gap} = P_{open} + \frac{lP_{ext}}{3} \qquad (2)$$

This penalty is applied to all gaps, insertions or deletions, which length is a multiple of three. Gaps which length is not a multiple of three result in a frame shift and have a much higher opening penalty $P_{fs\ open}$. The extension penalty for the frame shifts is the same as for the regular gaps.

The introns are scored as a special type of gaps with a very small extension cost I_{ext} and an opening cost I_{open} which is different between the most frequent (GT/AG), less frequent (GC/AG, AT/AC) and arbitrary splice sites. The protein to genomic alignment scores are codon oriented while the intron location is not. In fact, an intron could split a codon. If this happens ProSplign still scores two parts of the codon as a unit. An intron may also appear inside a gap. In this case the gaps on the both sides of the intron are treated as one gap with a combined length. This allows introducing only

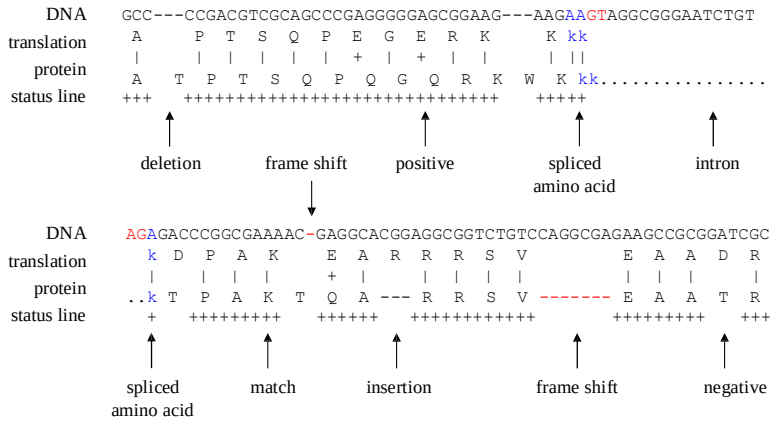

Figure 8: This figure shows basic elements of a protein alignment. Protein sequence is scored against the translation of the genomic sequence. Gap length is counted in nucleotide bases. Frame shifts, gaps which length is not multiple of three, cause the translation to change the frame. The translation jumps over an intron. One nucleotide base gap extension cost is one third of regular one amino acid extension cost. The frame shifts are penalized much more than the regular gaps. Alignment positions with positive score are marked with a plus sign in the status line. The status line is used during the post processing step.

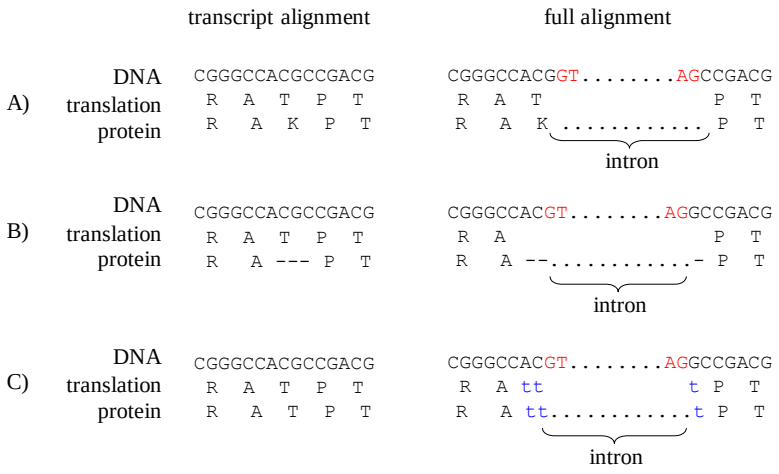

Figure 9: Each example is a fragment of an alignment with one intron. The alignment at the right is the full alignment including the intron. For simplicity the nucleotides inside the intron are not shown except for the splice sites which are shown in red. The alignment at the left is the alignment of the protein and the spliced transcript extracted from the genome. The idea behind the ProSplign intron scoring is to insure that the program creates identical alignments when presented with a genomic sequence and the corresponding transcript sequence. The case A) is the simplest case in which the intron is located exactly between two codons. The total score in this case is the Blossum62 score (which exactly corresponds to the score of the transcript variant of the alignment) minus the intron penalty $I_{open} + lI_{ext}$ where l is the intron length. In the case B) the intron happens to be inside a gap. In this case the score includes the Blossum62 component, a penalty for a three base pairs or one amino acid long gap which is $P_{open} + P_{ext}$, and the above intron penalty. The first two components are the same for the transcript variant of the alignment. It is important to mention that in this case, even with an intron in between, ProSplign recognizes that this gap doesn't introduce a frame shift in the translation. In the case C) the intron splits a Threonine codon. But still, this amino acid is fully accounted for in the scoring.

one opening penalty and make a correct decision if this gap is a regular gap or a frame shift. In other words, the score of an alignment with introns could be thought of as a combination of the score of the intronless alignment of the protein and the transcript extracted from the genome and penalties for the introns. This feature is very useful for the memory optimization described below. Intron scoring is illustrated in Figure.9.

1.4.2 Algorithm details

A classical Needleman Wunsch type [13] global alignment algorithm for aligning of a genomic sequence of a length L_{gen} and a protein of a length L_{prot} has to calculate a set of optimal scores and backtracking data in each of the $L_{gen} \times L_{prot}$ nodes. To reconstruct the alignment the backtracking information should be stored for each node. With some eukaryotic proteins being several thousand amino acids long and spanning about million bases on the genomic sequence the memory allocation of such scale becomes unpractical. In all these cases the bulk of the involved genomic sequence is located in introns, and if we knew the introns positions we could have aligned the protein against the much shorter transcript extracted from the genome. Following this idea ProSplign carries out the alignment in two steps. First, it aligns the protein and the full length genomic sequence. During this step it keeps track only of the optimal scores and the intron structure of the alignment. After the intron structure is known, ProSplign extracts much shorter transcript and runs the algorithm with full-fledged tracking but without additional intron related memory and computation overhead.

In addition to the best score, the optimal scores for a gapped alignment include two best scores with a gap in one or another sequence. In the case of ProSplign we effectively have several different types of gaps which are regular gaps, frame shifts, and introns with four different splice cites (GT/AG, GC/AG, AT/AC and arbitrary) for all of which we need the additional scores.

Introns can be located at any position relative to a codon. Like for ordinary gaps we need one additional score for introns located exactly between two codons. If the codon is split in its first position then the upstream exon includes a nucleotide which will affect the final score differently depending on the other two nucleotides on the other end of the intron. Following the usual dynamic programming rules we have to maintain five additional scores for this situation (one for each letter A, C, G, T and N found in the last base of the upstream exon). It seems that the same logics dictate that we have

to add another twenty five scores for the introns which split the codon in the second position. In fact, because we know the amino acid we are aligning against the split codon, we still can maintain only five additional scores – one for each possible nucleotide on the other end of the intron. Consequently, we need eleven additional scores for each type of introns. This analysis doesn't take into account the introns that are located inside gaps as in Figure.9B. Proper accounting for these introns demands some more optimal scores which will be described in a different publication.

ProSplign keeps two L_{gen} long rows of optimal scores which is enough for running the optimization algorithm. Instead of filling a $L_{gen} \times L_{prot}$ matrix with the backtracking information ProSplign keeps chains of introns that are optimal for each of the L_{gen} nodes of the above two rows (see Figure.10). The intron representation in a chain consists of the beginning coordinate of the intron, the intron length and a pointer to the previous intron. In the worst case scenario the memory used will be proportional to the product of L_{gen} and the number of introns. In practice it may be much less than that because many intron chains for close nodes are identical or have identical sub chains which could be reused. For this purpose ProSplign maintains a memory pool of unused introns. Any time a new intron is included in a chain it is allocated from the pool. If an intron is not used by any chain any more the memory is returned to the pool.

1.4.3 Post processing

Not all parts of a protein are conserved well enough to provide a reliable alignment. In fact, some parts may not correspond to anything on the genome. Still, the global alignment algorithm will align the whole protein rendering a very low identity alignment for the non-conserved portions of the protein. These unreliable and often misleading pieces of the alignment are filtered out during the post processing step.

For this purpose the post processing evaluates alignment positions. If an amino acid is aligned against a codon (including codons split by an intron), and the Blossum62 score of this combination is positive then all three corresponding alignment positions are considered to be positive. All cases of mismatches with a negative score, partial matches (an amino acid aligned against one or two bases) and all gaps other than introns are considered as negatives. Any portion of the alignment could be evaluated according to the fraction of the positive alignment positions it contains. Alignment positions

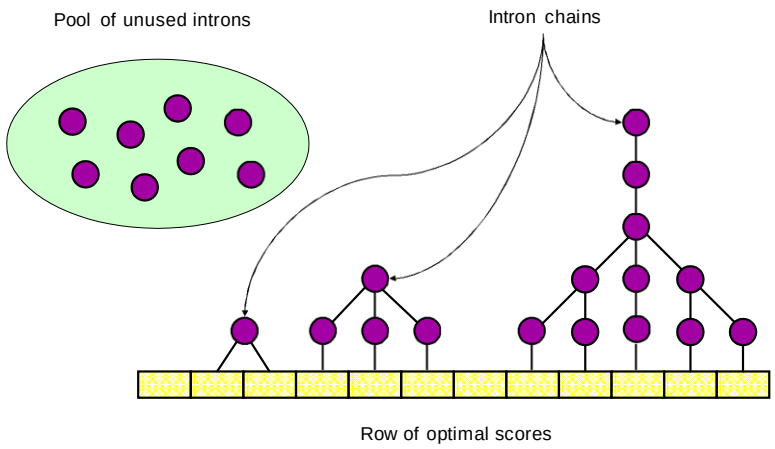

Figure 10: To run the optimization algorithm ProSplign keeps two last rows of optimal scores which are L_{gen} long. Instead of maintaining the full set of the backtracking information which takes a proportional to $L_{gen} \times L_{prot}$ amount of memory ProSplign retains for each node of the last two rows only chains of the optimal introns. In this case the memory used is proportional to the product of L_{gen} and the number of introns. The memory usage can be further optimized because many introns in different chains are identical. When a new intron is found it is allocated from a pool of unused introns and when this intron is not included in any chain any more it is returned back to the pool. After the optimization is complete and the intron locations are known, ProSplign realigns the extracted from the genome transcript against the protein at a very little additional computational cost.

corresponding to introns is ignored for this calculation. The positive alignment positions are shown as plus symbols on the status line of Figure.8. ProSplign keeps only parts of the alignment that satisfy the following rules:

1. The total fraction of the positive alignment positions in a retained part must be not less than Ktotal.

2. Any flanking stretch of a retained part must have a fraction of positive elements which is not less than Kflank . In particular, it means that there are no flanking negative elements.

3. There is no stretch of a retained part that is longer than L and has a fraction of positive elements which is less than Kmin.

4. Any retained part must be longer than Lmin.

An alignment may have more than one retained part. The gene structure located outside of these parts are determined later using other evidence or ab initio.

1.5 Finding compartments

Both Splign and ProSplign are global alignment tools, and computationally it is not feasible to use them without finding rough placements of the target sequences on the genome. Usually, the Blast hits give a starting point which is good enough for this purpose. Since very often there is more than one location on the contig where a target sequence could be aligned, the Blast hits should be analyzed to give locations for each copy.

Let's say that two hits are compatible if they follow the natural flow of the target sequence. For the alignments on the positive strand the relative position of the hits should be the same on both the target sequence and the genome, and it should be the opposite for the alignments on the negative strand. Compatible hits may overlap but none of them should be totally contained within the other. This definition of the compatibility is transitive.

A sequence of compatible hits h forms a compartment. The Compart finds all non-overlapping compact compartments on the genome for a given target sequence using maximal coverage algorithm. Each compartment c is assigned a coverage which is a measure of how well it represents the target sequence

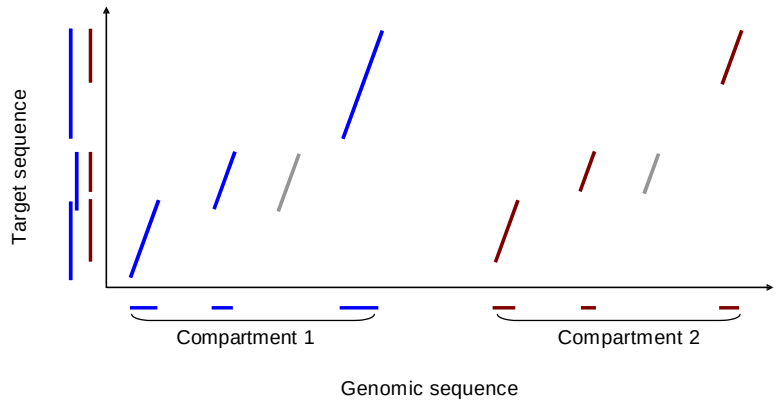

Figure 11: When more than one copy of the gene is present, the maximal coverage algorithm tries to find a set of compact compartments on the genome each of which is a putative gene location. We use a special additional compartment penalty to prevent algorithm from starting a new compartment each time it finds a duplicated exon (grey color in the picture).

$$\Phi^c = \sum_h w^h L^h_{eff} \qquad (3)$$

In this equation Lh_{eff} is the effective length of the hit h on the target sequence. Usually it is simply the hit length, but if the hit overlaps with a neighbor hit its effective length is decreased by a half of the overlap. We have two choices for the weight w^h. When the weight equals the identity of the hit the coverage (3) is the number of matches. We use this choice with the cDNA alignments for which most useful hits are of very high identity. The other choice is a constant weight equal 1. In this case the coverage (3) is simply the target sequence length covered by the hits. We usually use it with the protein alignments.

When there is more than one compartment, the target sequence is covered multiple times, and to a certain extent finding all compartments is equivalent to maximization of the total coverage. This is not true when we deal with exon duplication events as opposed to gene duplication events (see Figure.11). In these cases the additional hits should be ignored rather than turned into additional compartments. Since usually in these exon duplication

events only a relatively small portion of the gene is duplicated we introduce a penalty P_{new} for an additional compartment. This penalty ensures that a new compartment is created only if there is enough gene material for it. The value of this parameter is usually 25%–40% of the target sequence length. So our maximal coverage algorithm finds the compartments configuration which maximizes the following total coverage

$$\Phi = \sum_{c} (\Phi^c - P_{new}) \qquad (4)$$

The process of optimization is performed very effectively using the dynamic programming algorithm [2]. First, all hits are sorted into ascending order by their beginning positions along the genomic sequence. For each hit, possibilities are evaluated of using it to extend one of already opened compartments, or to start a new compartment. The possibilities are assessed using (4) and the best variant is stored along with the pointer to the prior hit upon which the variant is based. After that, the hit with the highest value of (4) is selected and the backtracking is carried out to reveal the optimal hit chain. All the hits not included in this chain are ignored. Each hit in this chain which is not compatible with the previous hit indicates the start of a new compartment. We loosely use the term compartment as either a set of selected hits or simply as the region on the genome where these hits are located.

References

[1] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. Journal of Molecular Biology, 215(3):403–410, 1990.

[2] Richard Bellman. Dynamic Programming. Princeton University Press, 1957.

[3] M. Borodovsky and J. McIninch. GenMark: Parallel gene recognition for both DNA strands. Journal of Computational Chemistry, 17:123–134, 1993.

[4] Chris Burge and Samuel Karlin. Prediction of complete gene structures in human genomic DNA. Journal of Molecular Biology, 268(1):78–94, 1997.

[5] Eduardo Eyras, Mario Caccamo, Val Curwen, and Michele Clamp. EST-Genes: alternative splicing from ESTs in Ensembl. Genome Research, 14(5):976–987, May 2004.

[6] P. A. Frischmeyer and H. C. Dietz. Nonsense-mediated mRNA decay in health and disease. Human Molecular Genetics, 8(10):1893–1900, 1999.

[7] Brian J Haas, Arthur L Delcher, Stephen M Mount, Jennifer R Wortman, Roger K Smith, Linda I Hannick, Rama Maiti, Catherine M Ronning, Douglas B Rusch, Christopher D Town, Steven L Salzberg, and Owen White. Improving the Arabidopsis genome annotation using maximal transcript alignment assemblies. Nucleic Acids Research, 31(19):5654–5666, Oct 2003.

[8] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. Proceedings of the National Academy of Sciences of the United States of America, 89(22):10915–10919, Nov 1992.

[9] Z. Kan, E. C. Rouchka, W. R. Gish, and D. J. States. Gene structure prediction and alternative splicing analysis using genomically aligned ESTs. Genome Research, 11(5):889–900, May 2001.

[10] M. Kozak. Compilation and analysis of sequences upstream from the translational start site in eukaryotic mRNAs. Nucleic Acids Research, 12(2):857–872, Jan 1984.

[11] Barmak Modrek and Christopher Lee. A genomic view of alternative splicing. Nature Genetics, 30(1):13–19, Jan 2002.

[12] Richard Mott. EST_GENOME: A program to align spliced DNA sequences to unspliced genomic DNA. Computer applications in the biosciences : CABIOS, 13:477–478, 1997.

[13] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology, 48(3):443–453, 1970.

[14] R. Staden. Computer methods to locate signals in nucleic acid sequences. Nucleic Acids Research, 12(1 Pt 2):505–519, Jan 1984.

[15] T. G. Wolfsberg and D. Landsman. A comparison of expressed sequence tags (ESTs) to human genomic sequences. Nucleic Acids Research, 25(8):1626–1632, Apr 1997.

[16] M. Q. Zhang and T. G. Marr. A weight array method for splicing signal analysis. Computer applications in the biosciences : CABIOS, 9(5):499–509, Oct 1993.

[17] Z. Zhang, S. Schwartz, L. Wagner, and W. Miller. A greedy algorithm for aligning DNA sequences. Journal of Computational Biology, 7(1-2):203–214, 2000.