EuGène: an eukaryotic gene finder that combines several sources of evidence

Thomas Schiex[1], Annick Moisan[1], and Pierre Rouzé[2]

[1] INRA, Toulouse, France,
Thomas.Schiex@toulouse.inra.fr,
http://www-bia.inra.fr/T/schiex
[2] INRA, Gand, Belgique
Pierre.Rouze@gengenp.rug.ac.be

Abstract

In this paper, we describe the basis of EuGène, a gene finder for eucaryotic organisms applied to *Arabidopsis thaliana*. The specificity of EuGène, compared to existing gene finding software, is that EuGène has been designed to combine the output of several information sources, including output of other software or user information. To achieve this, a weighted directed acyclic graph (DAG) is built in such a way that a shortest feasible path in this graph represents the most likely gene structure of the underlying DNA sequence.
The usual simple Bellman linear time shortest path algorithm for DAG has been replaced by a shortest path with constraints algorithm. The constraints express minimum length of introns or intergenic regions. The specificity of the constraints leads to an algorithm which is still linear both in time and space. EuGène effectiveness has been assessed on Araset, a recent dataset of *Arabidopsis thaliana* sequences used to evaluate several existing gene finding software. It appears that, despite its simplicity, EuGène gives results which compare very favorably to existing software. We try to analyze the reasons of these results.

Motivations

It is standard, in a thorough sequence annotation, to take into account several sources of evidence in order to try to precisely locate genes (exons/introns) in eucaryotic sequences. The sources exploited typically include:

– matches against databases (cDNA, ES protein databases…);
– output of splice sites or translation start prediction software;
– more or less sophisticated "integrated" gene finding software, eg. GeneMark.hmm [9].
– experimental evidence or human expertise.

None of these sources of evidence is, alone, sufficient to decide gene locations and the manual integration of all these data is a painful and extremely slow work. The motivation of our work is, as far as possible, to automate this job using

Arabidopsis thaliana as a first test organism. Several integrated gene finders exist that integrate protein or cDNA homologies in their prediction [7, 5, 22]. EuGène is naturally closely related to these tools. It most striking peculiarity is it's parasitic behavior: EuGène has been designed to exploit other tools or information sources, including human expertise or other integrated gene finding programs. This allows to easily select the best available ingredients.

1 Methods

To be able to integrate basic information on a genomic sequence, we have used a simple, general, efficient and yet effective graph-based approach for gene finding that allows to combine several sources of evidence. Rather than directly combining the output of existing gene finding software (as in [10]) we decided to combine the information at the lowest level in order to be able to:

1. maintain the consistency of the prediction;
2. globally assess the impact of each local choice w.r.t. all available evidence.

Given a raw DNA sequence, the basic idea is to consider a directed acyclic graph (DAG) such that all possible consistent gene structures are represented by a path in the graph. The gene structure currently used in EuGène is the simplest reasonable structure, the only signals taken into account being translation initiation sites (`ATG`), translation termination sites (stops) and splice sites. It can handle multiple genes which may be partial on the extremities of the sequence and explores the two strands of the sequence simultaneously as it is now standard in gene finding. Note that merged stops may occur in a prediction. However this can be easiliy deat with using additional tracks. This is not presented here for the sake of clarity.

The DAG used for a simple ad-hoc sequence "`CATGAGGTAGTGA`" is illustrated in figure 1. It is a graph with 13 different tracks that correspond respectively to the 6 forward/reverse coding frames, 6 forward/reverse intronic phases and a so-called intergenic track that covers both true intergenic regions and transcribed untranslated regions (UTR). Each signal occurrence, between two successive nucleotides, generates one or more "switches" between two parallel tracks. Each source-sink path defines a sequence of consistent genes structures. The size of the graph is in $O(n)$ where n is the sequence length.

To choose one path among the $O(13^n)$ paths in this graph, each edge e is weighted by a positive number w_e in such a way that shortest paths in the graph correspond to gene structures that "best respect" the available evidence. A probabilistic interpretation of this model can be given as follows: each edge e has a probability of existence P_e. Under simple independence assumptions, the reliability of a source-sink path is simply defined by the product of all the P_e in the path. A most reliable path is then a shortest path in the graph where $w_e = -\log(P_e)$. The approach is comparable (although not equivalent) to HMM with a non-homogeneous transition matrix between hidden states (our tracks).

Given a directed acyclic graph, the simplest linear time, linear space shortest path algorithm is Bellman's algorithm [1]. It can output an optimal possible gene

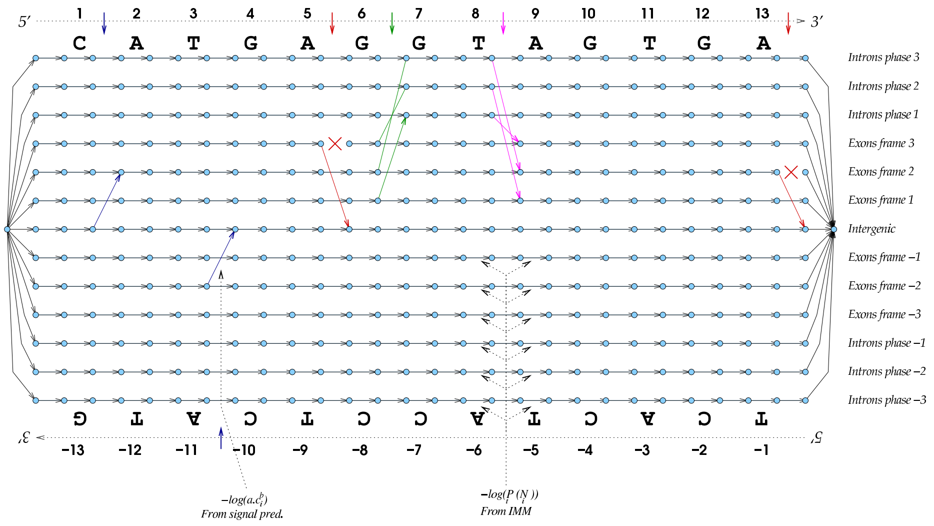

Fig. 1. The DAG explored by EuGène for a simple sequence

structure[1]. Compared to existing published gene finding algorithms, Bellman's algorithm applied to the above DAG is related to several existing gene finding algorithms. Like all recent gene finding algorithms it does in-frame scoring and assembly [21]. More specifically, gene finding algorithms can be classified as segment-based or nucleotide-based:

– Segment-based approaches implicitly or explicitly build an exhaustive list of potential exons and/or introns which they separately score. This is the case for the algorithm presented in [21] or the algorithm presented in [15]. Note that in the worst case, the number of possible signal well-delimited segments grows quadratically in the length of the sequence. Since each potential segment must scored, such segment-based approaches [21] are quadratic in the length of the sequence. The strength of these algorithms is that they allow essentially arbitrary segment scoring functions. To improve expected time complexity, one can exploit the property that exon-segments cannot contains in frame stop codons: the assumption that in frame stop codons occurs following a Poisson process suffices to make the expected number of possible exons linear in the length of the sequence.The algorithm presented in [15] in some sense exploits this property by only taking into account exons scores in the global score: it is worst-case linear in the number of exons and has therefore an expected linear complexity in the length of the sequence (under the Poisson assumption). The essential limitation is that it either ignores any possible intron, UTR or intergenic regions scoring or again becomes quadratic in the length of the sequence.

[1] Linear space is needed to recover the shortest paths found.

– to avoid quadratic complexity, so-called nucleotide-based gene finding algorithms, such as HMM-based algorithms (eg. Viterbi algorithm) do not explicitly score each segment separately but exploit the property that the score of a segment is defined as the sum of its elements. This allows to simultaneously score all segments sharing common sub-segments reducing worst-case time complexity to linear. This is eg. the case for the basic version of Bellman's algorithm: a gene parsing is just a path in the DAG described above and the score of such a gene parsing is simply defined as the sum of the scores of all the edges that make the path. The complexity is linear in space and time but segment scoring must be defined as a combination of the scores of its elements and cannot, for example, arbitrarily depend on the length of the exons as in segment-based approaches. However, it has better worst-case time complexity than segment-based approaches such as [21] and can exploit intron or intergenic region scoring.
  Possible variants of the Viterbi algorithm that take into account arbitrary explicit probability distributions on state lengths do exist [17] and have effectively been used in gene finding [3], but they become worst-case time quadratic. EuGène's algorithm is a sophistication of Bellman's algorithm that try to optimize the compromise between efficiency and generality: it is linear both in time and space but allows to take into account constraints on the minimum length of some gene elements (introns, single exon genes, intergenic regions). Basically, this means that the score of a segment is effectively the sum of the score of its elements unless the segment is too short: in this case the score becomes $-\infty$. The approach is comparable to explicit state duration HMM with uniform duration densities (see [16], pp. 270). Maximum length constraints can also probably be taken into account while keeping linear time and space but we haven't find any practically significant use of such constraints in gene finding.
  The complete algorithm actually used in EuGène allows to specify minimum length constraints for intergenic regions depending on the orientation of the genes that border the region: convergent ($\rightarrow\leftarrow$), divergent ($\leftarrow\rightarrow$) or confluent ($\rightarrow\rightarrow$ or $\leftarrow\leftarrow$) genes. In the sequel, for the sake of simplicity, we only describe the version which takes into account minimum length constraints independently of the orientation of surrounding elements[2].

1.1 Algorithm

Given a nucleic acid sequence of length n, we orient it so that the $5'$ end is on the left and the $3'$ end on the right. Naturally, the reverse strand is oriented in the reverse direction. Nucleotides are numbered both from 1 to n (forward strand) and from -1 to $-n$ (reverse strand). We denote sites by their position and their type. The position of a site is an integer (positive or negative but different from 0) such that site p occurs before nucleotide p. The type of a site can be either

[2] The complete version is not essentially different, remains linear, but requires the introduction of additional tracks in the graph.

(I) for a translation initiation site, (A) for an acceptor site, (D) for a donor site or (T) for a termination site. For any given site, the frame of a site (p, t) is noted $F(p)$ and is defined as $sign(p).(((|p| - 1)\ mod\ 3) + 1)$.

As exemplified in figure 1, we consider that both initiation (`ATG`) and termination sites are part of their exons which means that initiation sites occurs on the left of the `ATG` but termination sites occur on the right of the stop codons. Given a sequence of length n and a list of sites (p, t) defined by their position p and their type t, we can define the directed acyclic graph $G = (V, E)$ explored by EuGène as follows:

- the set of vertices V contains $2 + 13 \times (2n + 2)$ vertices. There is a source vertex s and target vertex t. For any given nucleotide at position $1 \leq i \leq n$ in the sequence, there are 13×2 vertices. The first 13 vertices are called the left vertices of nucleotide i and are denoted by $v^l_{i,j}, -6 \leq j \leq 6$. The 13 remaining ones are the right vertices and are denoted by $v^r_{i,j}, -6 \leq j \leq 6$. Furthermore, there are 13 extra right vertices before the first nucleotide which are denoted by $v^r_{0,j}$ and 13 extra left vertices after the last nucleotide denoted by $v^l_{n+1,j}$. For all these vertices, j is called the track of the vertex. For a given nucleotide, from a modeling point of view, the 13 (right/left) vertices correspond to different states or tracks:
  - the 0 track correspond to intergenic (i.e., UTR and real intergenic) predictions
  - tracks 1 to 3 (resp. -1 to -3) correspond to exons in frame 1 to 3 (resp. -1 to -3)
  - tracks 4 to 6 (resp. -4 to -6) correspond to introns in phase 1 to 3 (resp. -1 to -3). The phase of an intron is the position were the splicing occurs wrt. nucleotides: phase 1 introns splice at the frontier of two codons, phase 2 introns splice before the 2nd base of codons and phase 3 introns splice before the 3rd base of codons.
- the set of edges E contains the following edges:
  - edges that connect the source vertex s to all the $v^r_{0,j}$. These are the so-called "prior" edges.
  - edges that connect each $v^l_{n+1,j}$ to the target vertex t. These are the so-called "posterior" edges.
  - for each nucleotide i, 13 edges connect $v^l_{i,j}$ to $v^r_{i,j}, -6 \leq j \leq 6$. These edges are called "content" edges.
  - for all $0 \leq i \leq n$, 13 edges connect $v^r_{i,j}$ to $v^l_{i+1,j}, -6 \leq j \leq 6$. These edges are called the "no site" edges.
  - for each site (p, t) occurring on the forward strand $(p > 0)$, extra edges are added according to t. These edge are called t-"signal" edges according to the type of the site.
    - $t = (I)$: an edge that connects $v^r_{p-1,0}$ to $v^l_{p,F(p)}$ is added to the graphed. It simply represents the fact that an `ATG` in frame $F(p)$ allows to go from intergenic to exonic in frame $F(p)$.
    - $t = (T)$: an edge that connects $v^r_{p-1,F(p)}$ to $v^l_{p,0}$ is added to the graph. Furthermore, the edge that connects $v^r_{p-1,F(p)}$ to $v^l_{p,F(p)}$ is

deleted. This simply represents the fact that an in frame Stop codon effectively stops the current exon.

  * $t = (D)$: for all $i, 1 \leq i \leq 3$, an edge that connects $v^r_{p-1,i}$ with $v^l_{p,((F(p)-i+3)\ mod\ 3)+4}$ is added. This represents the fact that splicing an exon in frame i using a donor site in frame $F(p)$ leads to an intron in phase $1 + ((F(p) - i + 3)\ mod\ 3)$.
  * $t = (A)$: for all $i, 1 \leq i \leq 3$, an edge that connects $v^r_{p-1,i+3}$ with $v^l_{p,((F(p)+i+1)\ mod\ 3)+1}$ is added. This represents the fact that after an intron in frame i using an acceptor site in frame $F(p)$ leads to an intron in phase $1 + ((F(p) + i + 1)\ mod\ 3)$.

- similar modifications are done for sites occurring on the reverse complement strand.

The graph defined in this way is a directed acyclic graph. Considering weights, we just assume that all edges $e \in E$ are weighted with a weight $w(e) \in [0, +\infty]$. As we mentionned before, in order to identify a shortest path in this graph, the simplest standard algorithm is the so-called Bellman algorithm ([1] or [4], section 25.4).This algorithm is the most simple instance of dynamic programming and is based on the fact that any sub-path of an optimal path must also be optimal. This algorithm associates to each vertex $u \in V$ a variable $SP[u]$ that will *in fine* contain the length of a shortest path from s to u and a variable $\pi[u]$ that will contain the vertex that must be used to reach u in a shortest path. Initially, only $SP[s]$ is known and equal to 0. In order to compute the length of a shortest path from vertex s to a vertex v it suffices to know the length of a shortest path from s to any vertex u that immediately precede v in the graph (the edge (u, v) is in E). The length of a shortest path from s to v is simply the minimum over all u preceding v of the sum of the length of the shortest path from s to u and the weight $w((u, v))$. The parent $\pi[v]$ is then fixed to the the vertex u that minimizes this sum. The graph being acyclic, any topological ordering can be used to apply this simple relaxation procedure iteratively and finally compute $SP[t]$. This computation can be made in a time and space linear in the size of the graph and therefore in the length of the sequence (the linear space is essentially needed fo the $\pi[v]$ data structure, needed to recover the shortest path at the end).

The minimum length constraints we want to take into account can be formalized as follows. A minimum length constraint is defined by a length ℓ and a track number $k, -6 \leq k \leq 6$. A prediction is a path from s to t in the prediction graph. For any path in the prediction graph, we say it violates a constraint $\langle \ell, k \rangle$ if it contains a sub-path $\langle u_0, u_1, \ldots, u_{m-1}, u_m \rangle$ such that:

– $u_0 \neq s$ and has a track different from k,
– $u_m \neq t$ and has a track different from k,
– for all vertices $u_i, 1 \leq i \leq m - 1$, u_i's track is k
– and finally $m - 1 < 2\ell$

Such a sub-path is said to be a violating sub-path, the edge (u_0, u_1) being the opening edge and (u_{m-1}, u_m) being the closing edge. A path is said to be feasible if it does not violate any constraint.

The algorithm exploits the fact that if we consider an optimal feasible path P from s to a given vertex v then either the last edge (u, v) of it connects 2 vertices of the same track or not:

1. in the first case, then obviously, the sub-path that connects s to u is also an optimal feasible path from s to u or else we could use it as a short-cut and improve over P without changing feasibility. We get the usual Bellman's property.
2. in the second case, this is no more true because an optimal feasible path from s to u may finish by a sub-path such that the addition of edge (u, v) will be the closing edge of a sub-path that violates a constraint $\langle \ell, k \rangle$. Since we want to consider only feasible paths, we can conclude that the sub-path of P that connects s to u is either entirely composed of vertices from track k (except s) which means that it cannot contain an opening edge or it must finish by a path $\langle u_1, \ldots, u_{m-1}, u \rangle$ that remains on track k for at least $m = 2\ell$ vertices. Then the sub-path from s to u_1 must be a feasible optimal path from s to u_1 or else we could shortcut. We get an adapted version of Bellman's property.

To exploit these properties, as in Bellman's algorithm, EuGène's algorithm associates two variable to each vertex u in the graph: $SP[u]$ contains the length of the shortest feasible path that goes from s to u, $\pi[u]$ contains a reference to a vertex that must be used to reach u using a shortest feasible path. Initially, only $SP[s]$ is known and equal to 0. Consider a feasible shortest path from s to a given vertex v: it must reach v using an edge (u, v) that connect one of the vertices that immediately precede v in the graph to v. This edge can be either:

– an edge that connect two vertices of the same track (a "no signal" or a "content" edge). The first case of above property applies and the shortest feasible path that reach v through u, noted $SP_u[v]$ has length $SP[u] + w((u, v))$. In this case, we say that u is the u-parent of v.
– an edge that connects two vertices from different tracks ((I), (T), (D) or (A)-signal edges). The second case of the above property applies. Let k be the track of u and $\langle \ell, k \rangle$ the constraint that applies to track k (if no constraint exists, one can use the trivial $\langle 0, k \rangle$ constraint). The shortest feasible path that reach v through u can simply be obtained by going back on track k for 2ℓ vertices, reaching a vertex w on track k (if vertex v is too close to s, we will reach the vertex $w = v^r_{0,k}$ and stop there). Let $d_k[w, u]$ be the length of the path that goes from w to u staying on track k. Since ℓ is bounded, both w and $d_k[w, u]$ can be computed in constant time. The length of a feasible shortest path that reach v from u is then $SP_u[v] = SP[w] + d_k[w, u] + w((u, v))$, which again can be computed in constant time. In this case, we say that w is the u-parent of v.

Consider all the vertices u immediately preceding v. The number of such vertices is bounded by the number of tracks (13). We can compute all $SP_u[v]$ in constant time. Now, we know that $SP[v] = \min_u(SP_u[v])$. Let $u^* = arg\min_u(SP_u[v])$. We then set $\pi[v]$ to the u^*-parent of v.

This elementary constant time procedure can be applied iteratively from vertex s to vertex t using any topological ordering of the vertices. According to the property above, after processing, $SP[t]$ contains the length of the shortest feasible path from s to t and this path can be recovered by back-tracing along the $\pi[]$ variables, starting from t. Since $|V| = O(n)$, the algorithm is time linear in the length of the sequence. The only data-structures used being the $SP[]$ and $\pi[]$, it is also space linear.

1.2 Scoring

The algorithm above applies to any weighting of the graph. It is now time to describe how edges are weighted. The first version of our prototype, called EuGène I, integrates the following sources of information:

– output of five interpolated Markov models (IMM, [18]) for respectively frame 1, 2, 3 exons, introns and intergenic sequences. Given the sequence, the IMMs allow to compute the probability $P_t(N_i)$ that the nucleotide N_i at position i appears on each track t. The corresponding edge is weighted $-\log(P_t(N_i))$ (see Figure 1). These models have been estimated on the AraClean v1.1 dataset [14] using maximum likelihood estimation for all orders from 0 to 8. For each conditional probability distribution, a linear combination of the distributions obtained for orders 0 to 8 is used according to the amount of information available, as in Glimmer [18]).
A difference with the IMM approach used in Glimmer is that the interpolated Markov models used to estimate the probabilities $P_t(N_i)$ are used in such a way that the graph G that represents a sequence and the graph $\bar{G}$ that represents its reverse complementary are equivalent up to a half-turn of the graph. This guarantees that any sequence will be analyzed exactly has its reverse complementary which seems desirable. To do this, the Markov models on the forward strand are estimated supposing that the Markov property holds in one direction. The Markov model on the reverse strand assume that the Markov property holds in the other direction. The same matrix can therefore be used on both forward and reverse strand (after a reverse complementation) which reduces the number of coding models from 6 to 3.
– output of existing signal prediction software. These software typically output a so-called "confidence" $0 \leq c_i \leq 1$ on the fact that a possible signal occurring at position i is used (i.e., the corresponding switch used). This confidence cannot decently be interpreted directly as a probability. We make the assumption that the switch's weights have the parametric form $-\log(a.c_i^b)$ where the constants a and b have to be estimated for each source of evidence (see Figure 1).
These a, b parameters are estimated once the IMM parameters have been estimated. The estimation is done by maximizing the percentage of correct predictions by EuGène on the same learning set (Araclean 1.1). For fixed values of the parameters to optimize, the measure of the percentage of correct predictions can simply be performed by applying EuGène on the whole

Araclean dataset. Since EuGène is linear time, this is relatively efficient. The parameter estimation is then done using a very simple genetic algorithm whose results are refined by locally sampling the parameter space (which has a reasonable number of dimensions). The whole process is brutal and relatively inefficient but is performed only once. More work is needed to see how this estimation process could be made computationally less brutal.

A second version, called EuGène II can use, in conjunction with these basic information, results from cDNA and protein databases search:

– cDNA alignments in conjunction with splice sites are used to modify the graph as follows: matches (resp. gaps) delete intronic (resp. exonic) tracks in the graph. This forbids paths that would be incompatible with the cDNA data. Actually, cDNA/EST hits are found using Sim4 [6].
– similarly, EuGène II can exploit protein matches (found using BlastX). However, one cannot be confident enough in such information to directly use matches/gaps as constraints and we therefore simply modify the $P_t(N_i)$ using a simple pseudo-count scheme.

In practice, the structure and weights of the graph can be directly modified by the user using a very simple language that allows to include information about starts, splice sites, exonic, intronic, intergenic tracks on a per nucleotide basis. The language allows to delete or create "signal" edges in the graph and to modify the weight of any edge (either signal or content edges). To modify weights, we simply rely on a probabilistically inspired pseudo-count scheme. Initially, all edges are not only weighted by a positive weight w but also by a count c. If the user wants to modify the weight of an edge, s/he must mention a weight $w' \in [0, +\infty]$ and a count c'. The corresponding edge will then be weighted $-\log(\frac{\exp(-w).c+exp(-w').c'}{c+c'})$ with count $c + c'$.

2 EuGène in action

In this section we show how EuGène works in practice by applying it to the contig 38 from the Araset dataset [12, 11] which contains two genes with respectively 3 and 13 exons. We first collect information about splice sites and `ATG` by submitting the sequence to NetGene2 [20], SplicePredictor [2] and NetStart [13] using a dedicated Perl script. The combination of information sources is, at this time, our best "similarity-free" known ingredient selection for *A. thaliana*. This automatically builds files containing positions and strengths of "switches" in the graph. Additional information may be provided by the user. For example, a line stating "`start r4 vrai 3.7e-02 nocheck`" states that there is a reverse start at position 4 and the weight of the corresponding edge should be $-\log(0.037)$. The "`nocheck''` indicates that the user does not want EuGène to report at the end if this `ATG` has been effectively used in the prediction.

We then start EuGène and ask for a graphical zoom from nucleotide 3001 to 7000 (region of the second gene according to Araset's annotations). On this sequence, EuGène I locates all exons/introns border of the 2 genes. EuGène outputs images in PNG or GIF format which can directly be used on Web pages. The X-axis is the sequence. The Y-axis represents successive "tracks": reverse introns, frame -3, -2, -1 exons, intergenic sequences (this includes UTR), frame 1, 2, 3 exons and forward introns.

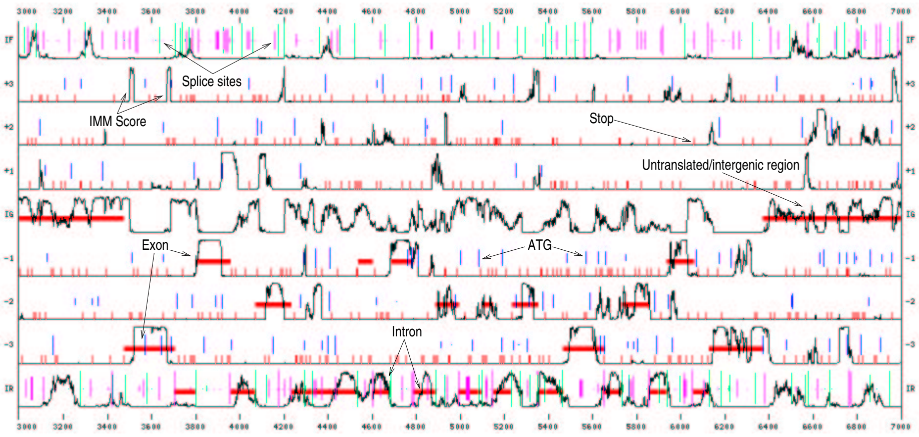

Fig. 2. EuGène I applied to contig 38 of Araset

On each track, the black curve represents the output of the IMM models smoothed over a window of 100 nucleotides and normalized. The large horizontal blocks represent EuGène's prediction. On the exonic tracks alone, small vertical bars represent potential stops and potential starts (`ATG`), the height of the bar being representative of the quality of the `ATG` according to the available evidence. On the intronic tracks, the vertical bars represent donors/acceptors. Again, the height of the bar is representative of the quality of the splice site.

EuGène I is not always as successful, eg. on the sequence of a tRNA synthetase, where EuGène I chooses one wrong splice site for one exon border. However, enough cDNA data exists so that EuGène II is able to unambiguously locate all exons/introns borders. The initial spliced alignment between the genomic sequence and the cDNA is done using `sim4` [6] and then used by EuGène. On the graphic below, this additional information is provided: the intronic tracks show blocks that represent cDNA matches interconnected by thin lines that represent gaps (connecting splice sites). In this case, no protein database information is used.

Considering protein databases, EuGène accepts similarity information from different databases. As it has been said before, each similarity information modify the weights following a pseudo-count scheme. If several databases are used, different counts are associated to each database. These counts are optimized as the a, b parameters for signal sensors. Using SPTR, PIR and TrEMBL as 3 different databases, one can see that the optimized count for SPTR are stronger than for PIR and that the TrEMBL count seems to converge towards 0.

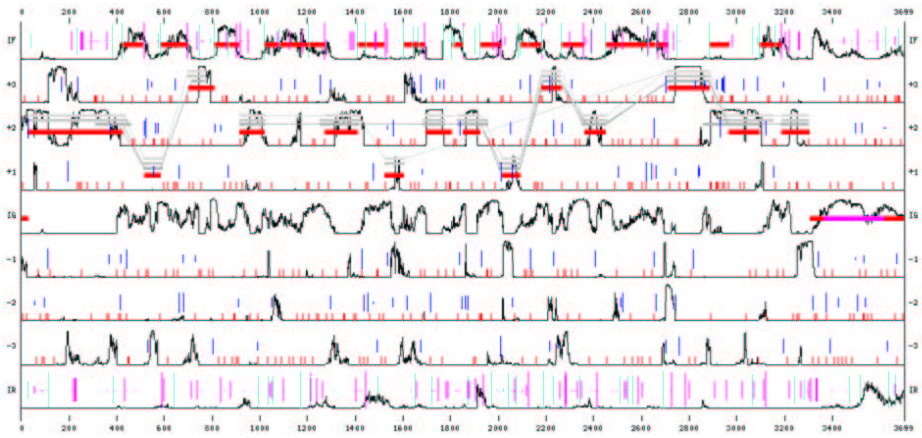

Fig. 3. EuGène II applied to a tRNA synthetase with cDNA data

3 Evaluation

The parameters used in EuGène have been estimated on Araclean [14]. Araclean contains 144 genes with very few context around the genes. This lack of context makes it difficult for EuGène to decide whether an intergenic or intronic mode should be used on the sequences border. Anyway, EuGène I correctly identifies all the exons (including the `ATG`) of 76% of the sequences of Araclean. Taking into account the lack of context, if we inform EuGène I that all predictions must start and end in intergenic mode (i.e., no partial gene is allowed) then the percentage of perfectly recognized genes gets up to more than to 82% (without using any cDNA/EST/protein homology information).

However this test is not very satisfactory. To get a better idea of EuGène performances, we have used another recent dataset. The "Araset" dataset has originally been designed and used to assess several existing gene or signal finding software [11]. The clear winner of this evaluation is GeneMark.hmm [9]. We therefore decided to compare EuGène to GeneMark.hmm on this dataset (which does not share any sequence with the Araclean dataset used for EuGène parameter estimation). The evaluation performed in [11] includes different performance measures, among which we will consider:

- sites prediction quality: for splice sites, the number of predicted, correct, over-predicted and missing sites has been measured.
- exon prediction quality: for all exons, the number of predicted, correct, overlapping, wrong and missing exons has been measured.

– gene level prediction quality: in this case, a gene is considered as correctly predicted if all its exons are correctly predicted. Il all exons are not correctly predicted, the gene can be missing (all exons are missing), partial (some exons are missing or partial), wrong (the predicted gene does not exist – as far as we know), split (a correct gene has been split in two or more predicted genes), fused (correct adjacent genes have been predicted as one gene).
– gene borders prediction: for predicted translation initiation sites (`AUG`) and terminator sites (stops), the number of correct prediction has been measured. For `AUG`, we also give the number of incorrect predictions which are nevertheless in phase with the real `AUG` site.

In each case, the overall performance is abstracted in the two usual *specificity* (S_p) and *sensitivity* (S_n) measures introduced in [19].

$S_n =$ number of correctly predicted items / number of actual items
$S_p =$ number of correctly predicted items / number of predicted items

For a more detailed definition of the measures, we invite the reader to refer to [11, 19]. Note also that the biological truth, considered here as unambiguously provided by Araset's annotations, is not so simple: alternative splicing or translation initiation exists and make these numbers less reliable than one may think *a priori.*

Table 1 presents the results[3] obtained for splice sites. EuGène I is less sensitive but more specific than GeneMark.HMM. Table 2 gives similar numbers for translation initiation (`AUG`) and terminator (stop codons) sites. Here, the results are clearly in favor of EuGène.

For the gene model level we also evaluated EuGène II. We used SPTR as the protein database. We also have a cDNA/EST database built using EMBL, and cleaned from documented partially or alternatively spliced cDNA. However, this database is built entirely automatically and is known to contain data inconsistent with Araset annotations for some of the genes of Araset (these inconsistencies are the results of contamination with genomic DNA and undocumented alternatively or partially spliced cDNA). Although using this cDNA/EST information alone improves the performances of EuGène, the current version of EuGène II tested uses only protein informations from SPTR because of its reliability. A clean cDNA/EST database would improve EuGène II's performances.

Although an in-depth analysis is still needed to be more conclusive, it appears from the measures above that the essential strength of EuGène lies in its ability to correctly identify gene's borders: compared to GeneMark.HMM, EuGène merges or split very few genes and identifies most translation initiation sites or terminator sites correctly. We think that this improvement is caused by two original elements of EuGène: one is the use of a dedicated intergenic Markov model, which is not a background 0^{th} order model and is different from the intronic

[3] Reference [11] gives a number of introns (and therefore acceptors and donors) of 860. However, parsing the reference file distributed with the instances, we only found 859 introns.

Program	predicted	correct	overpredicted	missing	S_n	S_p	predicted	correct	overpredicted	missing	S_n	S_p
GeneMark	981	797	184	63	0.93	0.81	914	772	142	88	0.90	0.84
EuGène I	798	742	56	117	0.86	0.93	801	733	68	126	0.85	0.92

Table 1. Donor and acceptor sites prediction (859 sites each)

Program	predicted	correct	in phase	S_n	S_p	predicted	correct	S_n	S_p
GeneMark	196	129	31	0.77	0.66	173	136	0.81	0.79
EuGène I	190	143	18	0.85	0.75	193	155	0.92	0.80

Table 2. Translation initiation and terminator sites prediction (168 sites each)

Program	predicted	correct	overlap	wrong	missing	S_n	S_p	ratio wrong	split	fused
GeneMark	1104	845	172	87	26	0.82	0.77	0.08	10	4
EuGène I	991	837	102	57	89	0.82	0.84	0.06	1	5

Table 3. Exons prediction

Program	actual	predicted	correct	missing	partial	wrong	split	fused	S_n	S_p
GeneMark	168	208	67	1	100	27	18	12	40%	32 %
EuGène I	168	194	104	1	63	19	8	2	62%	54 %
EuGène II	168	198	125	1	42	22	9	0	74%	63 %

Table 4. Gene level measures

model; the other lies in the fact that, except for the IMM, its parameters have been estimated by maximum of "successful recognition" rather than maximum likelihood. This probably tends to compensate for the weaknesses of the model. Initially, we thought that the good quality of the splice sites predictors used by EuGène (NetGene2 and Splice Predictor) was a major reason for its good performances. Looking to the splice sites prediction ability of EuGène, this is less obvious now although EuGène clearly inherits the excellent specificity of NetGene2.

This report is still quite preliminary and we expect to enhance EuGène's effectiveness in a near future (and apply it to other organisms). Actually, compared to other gene finding algorithms, EuGène is relatively simple: it uses a single Markov model set independently of GC% (although this is probably more important for the human genome than for *Arabidopsis thaliana*), does not take into account signals such as polyA or promoters or does not model UTR. This should leave room for improvements. Another important direction lies in the use of protein and cDNA/EST hits. We think that the integration, at the lowest level, of spliced alignment algorithms in the spirit of [6, 8] could help both to better take into account similarity information and also to filter out most inconsistent data. Other sources of similarity such as conservation of exons between (orphan) genes of the same family should also be integrated (M-F. Sagot, personal communication). We are currently working on these points.

Binaries for EuGène can be requested from Thomas.Schiex@toulouse.inra.fr. One of the weakness of EuGène is naturally its parasitic behavior: to have a fully complete running version of EuGène one must first install NetStart, SplicePredictor and NetGene2 which itself depend on SAM. For light users, a Perl script that submit one sequence to the web interfaces of all these softwares and that collect the results in the adequate file format has been developed. Because EuGène parameter estimation has been done using precise versions of these softwares, the best idea would be *a priori* to install the same version... we recently installed a local version of the above tools on a new machine using NetGene2 2.42 and SAM 3.2. The bad surprise was that NetGene2 did not gave the same output as the Web version which we used up to now. The nice surprise is that without any a, b parameter re-estimation, the gene-level sensibility and specificity are now equal respectively to 64.2% and 89.8% respectively which shows both the good robustness EuGène and the positive side-effect of its parasitic behavior: any improvement in the hosts may yield an improvement in EuGène itself.

Acknowledgements : this work has been supported by a grant from the "Programme prioritaire génome & fonction" of the *Institut national de la recherche agronomique* (INRA). We would also like to thank N. Pavie and C. Mathé (INRA, Gand) for their help with Araset, S. Rombauts (INRA, Gand), B. Lescure (INRA, Toulouse), H. Chiapello and I. Small (INRA, Versailles) for sharing their experience de l'arabette, P. Dehais and S. Aubourg (INRA, Gand) for their feedback and their help in making EuGène more user friendly.

References

[1] R. Bellman, *Dynamic Programming*, Princeton Univ. Press, Princeton, New Jersey, (1957).

[2] V. Brendel and J. Kleffe, (1998), *Prediction of locally optimal splice sites in plant pre-mRNA with application to gene identification in Arabidopsis thaliana genomic DNA*, Nucleic Acids Res., 26, pp. 4749–4757.

[3] B. C and K. S, Apr 1997, *Prediction of complete gene structures in human genomic dna.*, J Mol Biol, 268, pp. 78–94.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*, MIT Press, (1990). ISBN : 0-262-03141-8.

[5] K. D, H. D, R. MG, and E. FH, (1997), *Integrating database homology in a probabilistic gene structure model.*, in Pacific Symp. Biocomputing, pp. 232–44.

[6] L. Florea, G. Hartzell, Z. Zhang, G. Rubin, and W. Miller, Sept. 1998, *A computer program for aligning a cdna sequence with a genomic dna sequence*, Genome Res, 8, pp. 967–974.

[7] R. IB, M. L, and K. NA, Jun 1996, *Gene structure prediction using information on homologous protein sequence.*, Comput. Appl. Biosci., 12, pp. 161–70.

[8] U. J., Z. W., and B. V., (2000), *Optimal spliced alignment of homologous cDNA to a genomic DNA template*, Bioinformatics, 16, pp. 203–211.

[9] A. V. Lukashin and M. Borodovsky, (1998), *GeneMark.hmm: new solutions for gene finding*, Nucleic Acids Res., 26, pp. 1107–1115.

[10] K. Murakami and T. Takagi, (1998), *Gene recognition by combination of several gene-finding programs*, BioInformatics, 14, pp. 665–675.

[11] P. N, R. S, D. P, M. C, R. DV, L. P, and R. P, Nov. 1999, *Evaluation of gene prediction software using a genomic data set: application to arabidopsis thaliana sequences.*, Bioinformatics, 15, pp. 887–99.

[12] N. Pavy, S. Rombauts, P. Déhais, C. Mathé, D. Ramana, P. Leroy, and P. Rouzé, (1999), *Evaluation of gene prediction software using a genomic dataset: application to Arabidopsis thaliana sequences*, in Proc. of 2^d Georgia Tech conference on BioInformatics, Atlanta.

[13] A. Pedersen and H. Nielsen, (1997), *Neural network prediction of translation initiation sites in eukaryotes: prespectives for EST and genome analysis*, in Proc. of ISMB'97, AAAI Press, pp. 226–233.

[14] P. R. P.G. Korning, S.M. Hebsgaard and S. Brunak, (1996), *Cleaning the genbank arabidopsis thaliana data set*, Nucleic Acids Res., 24, pp. 316–320.

[15] G. R, Winter 1998, *Assembling genes from predicted exons in linear time with dynamic programming.*, Journal of Computational Biology, 5, pp. 681–702.

[16] L. Rabiner, (1989), *A tutorial on hidden markov models and selected application in speech recognition*, Proc. IEEE, 77, pp. 257–286.

[17] L. R. Rabiner, (1989), *A tutorial on hidden markov models and selected applications in speech recognition*, Proc. of the IEEE, 77, pp. 257–286.

[18] S. L. Salzberg, A. L. Delcher, S. Kasif, and O. White, (1998), *Microbial gene identification using interpolated Markov models*, Nucleic Acids Res., 26, pp. 544–548.

[19] E. Snyder and G. Stormo, *DNA and protein sequence analysis: a practical approach*, IRL Press, Oxford, (1995), ch. Identifying genes in genomic DNA sequences, pp. 209–224.

[20] N. Tolstrup et al., (1997), *A branch-point consensus from Arabidopsis found by non circular analysis allows for better prediction of acceptor sites*, Nucleic Acids Res., 25, pp. 3159–3163.

[21] T. D. Wu, (1996), *A segment-based dynamic programming algorithm for predicting gene structure*, Journal of Computational Biology, 3, pp. 375–394.

[22] H. X, A. MD, Z. H, and K. AR, Nov 1997, *A tool for analyzing and annotating genomic sequences.*, Genomics, 46, pp. 37–45.