FRAMED

A gene and sequencing errors finder for prokaryotic and matured eukaryotic sequences

Thomas Schiex
tschiex@toulouse.inra.fr
www-bia.inra.fr/T/schiex

January 2003

Introduction

FRAMED is gene prediction program for prokaryotic and matured eukarytic sequences, It has been used and designed to locate genes in GC-rich genomes (eg. *Ralstonia solanacerum* is 67% GC-rich genome). These genomes raise specific issues: because of the GC-richness, in a coding region almost all codons end with a `g` or a `c`. Since all STOP codons start with a `t`, this means that on the reverse strand, no STOP codon will occur: any true coding open reading frame will automagically induce another (likely non coding) open reading frame on the reverse strand. Actually, the (non coding) mirror ORF will often be longer than the true coding ORF.

Most prokaryotic gene finders assume that STOP codons appear following a Poisson process which means that long ORF are a statistically significant indicator of a gene in the phase of the ORF. This is not rue for GC-rich genomes: an ORF may be the mirror of a gene or a gene itself, and many usual prokaryotic gene finders will be generate many false positives on GC-rich genomes. This is not true for FRAMED.

However, FRAMED is not limited to GC-rich genomes. It works fine with other genomes and tests on *Bacillus subtilis* genome have shown a nucleotide based sensitivity and specificity of 98%, a gene sensitivity of 95% and a gene specificity of 98%. Considering the choice of the start codon, it is identical to the annotations for around 75% of the genes in most cases. People must beware of this last result: in most annotation protocols,the choice of the START codon when no experimental evidence exists is to take the fisrt START in the ORF. Results are much better on our GC-righ organisms but they have been annotated using FRAMED. So this is neither surprising nor a good test of FRAMED.

Finally, FRAMED is able to deal with noisy sequences in all possible aspects : it can handle frameshifts but it is also able to deal with non finalized sequences containing ambiguous nucleic acids codes (such as eg. `N`). This is especially useful to deal with EST (or clusters of EST) or more generally unfinished sequences. On such ambiguous sequences, FRAMED can start or stop a gene on eg. `ATN`/`TGN` respectively. Coding statistics can also take into account such degenerated information. For best results, a specific program for estimating interpolated Markov models on the extended alphabet `A,T,G,C,N` from known (non noisy) sequences is available from the author. Otherwise, FRAMED can either approximate the coding statistics very simply by arbitrarily removing ambiguity or using an averaged probability model.

1 Installing and testing

At this time, FRAMED is only distributed as binaries for Unix operating systems (Linux and Solaris). It is also available on web sites at http://atg/bioinfo/FrameD/FD and http://www.toulouse.inra.fr/FrameD/cgi-bin/FD.

In order to test your distribution, available in a `tar.gz` file, your must first uncompress it using the following command:

```
zcat FrameD.tar.gz | tar xvf -
```

This will create a `FrameD` directory with everything needed to use it. To test that your distribution is OK, type the following line:

```
./FrameD++ -b example.blast
           -e -gtest example.fasta
```

This calls FRAMED, tells him to display blast information available in the file `example.blast` (`-b example.blast`), to compute frameshift expectation (`-e`), to produce a graphical output in a file whose basename will be `test` (the file will be called `test.000.png`) for the sequence in the file `example.fasta`.

Then look to `test.000.png` (this is a PNG image, you can visualize it using Netscape, Explorer or any modern image visualizer) and compare it with the file `orig.000.png` which I produced here. If they are identical, then FRAMED should work fine.

Beyond the binary and the test files, the distribution also contains:

- file `default.mat`: contains a link to the interpolated Markov models for *Ralstonia solanacaerum*

built here. More matrix files for many other organisms are included. You can build more on your own using FRAMED's web site.

- file `FrameD.par` is a parameter file which is absolutely needed by FRAMED. If you want to know the meaning of the contents of this file, please see section 5. This is important in practice since modification of this file allows to tune eg. the frameshift detection sensitivity.
- files `stack.dat` and `loop.dat`: contain the energetic parameters for computing the RBS (Shine-Delgarno) hybridization energy. Taken and adapted from Turner's website. FRAMED use a simplified RNA binding model which ignores stabilizing loops (eg. tetraloops). The algorithm is essentially a free end-gap global alignement algorithm that takes into account bulges, loops and helices.
- directories `cgi-bin`, `Images_CSS` and `Help` should help in setting up a web site for FrameD

If you are interested in another organism than those provided then you must either download another interpolated Markov model from FRAMED web site or build your own interpolated Markov model from the web site or as described in section 6.

1.1 Installing a Web site with FRAMED

Almost all the material needed to make FRAMED available on a web site is included in the distribution. If you want to install a Web site with FRAMED, you must have Perl installed along with the `CGI` package.

In order to install the web site, you must copy the complete FRAMED installation directory under the web server root directory in a directory called `FrameD`. You must allow the execution of cgi-bin scripts in the `cgi-bin` directory. You must make the `tmp` directory writable by the user that will execute the cgi-bin scripts (FRAMED images are stored in this directory). To clean up the images, you must modify the system's `crontab` and insert the `FrameDW3.cron` file in it (you must correct paths in this file). It will remove, every hour, all files in the `tmp` directory that have been created more than 15' ago. You must also have a `tmpmat` directory that will hold temporary matrices (writable by the web server) and a cron process to clean this directory (by default, only 15 days old files are deleted).

If you build your own matrices for your specific organism and want to make them permanent, you will have to modify the arrays `@orga` and `@matrices` in `cgi-bin/FD`.

It is highly likely that you will also have to modify paths in the two scripts in `cgi-bin`. Please, leave the references to FRAMED publication and author as is.

2 Analyzing FRAMED graphical output

FRAMED is able to generate graphical files in PNG (Portable Network Graphic) format. This format is recognized by most web browsers. This makes it easy to distribute and share FRAMED's output.

This section assumes that all graphical elements of FRAMED have been activated (the default as far as the `-e` flag is used). The output of FRAMED lies on 7 rows. From top to bottom: 3 forward coding phases, non coding, 3 reverse coding phases.

The graphical output of FRAMED is composed of several elements that can be switched on or off as you want (see section 4). The following description assumes that all the possible graphical elements are enabled which is the default when the `-e` flag is used. See figure 1 for example.

FRAMED's graphical output represents the sequence on the X axis. The Y axis is composed of 7 layers. From top to bottom:

- the 3 top layers correspond to the direct strand of the sequence being analyzed in the 3 possible frames (1, 2 and 3 indicated on the left of the image).
- the middle layer (denoted IG on the left of the image) correspond to non coding (intergenic) parts.
- the 3 lower layers correspond to the reverse strand, analyzed in the three possible frames (-1, -2 and -3 indicated on the left of the image).

On all the layers, the following elements appear:

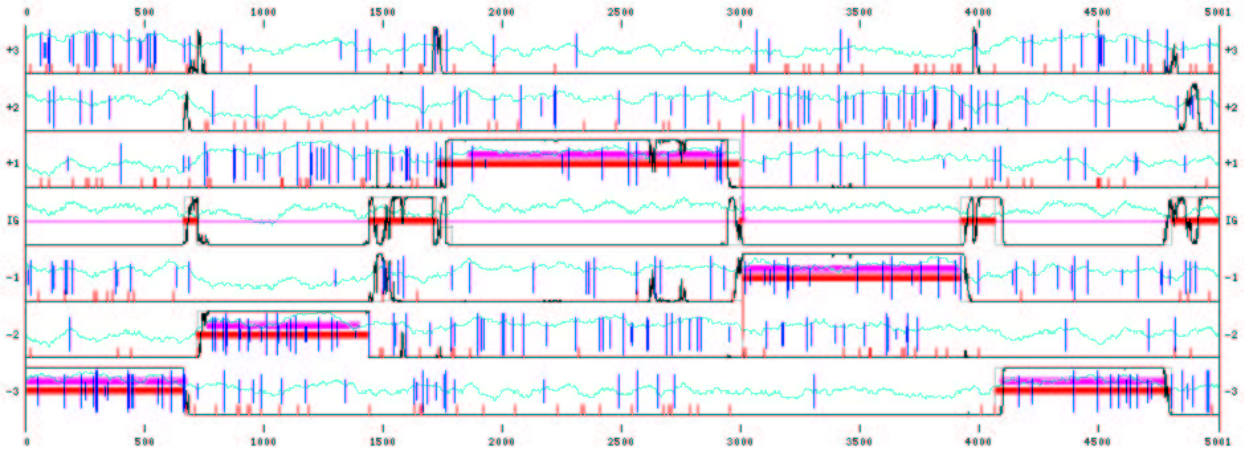

Figure 1: FRAMED graphical output for the `example.fasta` sequence

- the thin blask line represent the coding/non coding potential according to the interpolated Markov models (coding) or to a 0th order Markov model (non coding). On the figure 1, one can see for example that the sequence seems coding on the forward strand on frame 1 from base 1700 roughly to 3000.
- a thick red line indicates FRAMED prediction. A frameshift is indicated by a thin red lines that connects two thick red lines on two different phases. Eg, in figure 1, the previous "coding region" is analyzed as a gene that starts at position 1717 and ends at 2994 on frame 1. A thin red line is visible around 3000on the reverse strand where a spurious frameshift between frame -1 and -2 is detected due to a limitation of FRAMED which is analyzed later in this document (see 8).
- if flag `-e` is used, a thin gray line indicates the mean behaviour of FRAMEDIt may show that FRAMED "hesitates" between several possibilities. For example in figure 1, FRAMED hesitates between several STARTs for the gene on frame 1. This is visible on the intergenic layer: a gray line above 0 appears on the non coding region indicating that FRAMED thinks it is possible that a later START may be used. But the first (at 1717) is preferred.

There are other elements that appear only on the "coding" layers (layers 1,2,3,-1,-2 and -3):

- the blue vertical lines represent STARTS. The higher the bar, the better the detected RBS is.
- the small red vertical lines represent STOPs.
- if the `-b` or `-B` flag is used, blastx hits (or any other information provided in the file indicated after the `-b` flag) appear as magenta thick lines.
- a thing irregular green line indicate the GC3% (percentage of GC of the 3rd position of the codons). in a sliding window. This is computed on a window size specified by the `-w` flag (default is 97 nucleotides)

Finally, on the non coding (intergenic) layer row:

- a thin irregular green line indicates the local GC% (0 corresponds to 25%, 1.0 to 75%). This is computed on a window size specified by the `-w` flag (default is 97 nucleotides)
- if `-e` is used a thin magenta line indicates the frameshift expectation. Insertions are indicated by an upward curve, deletions by a downward curve. In figure 1, one can see that a magenta peak occurs around 3000 (where a spurious frameshift is detected). The peak is upward meaning that an insertion has been detected.

3 Analyzing FRAMED textual output

FRAMED prints text on two different streams. the error stream (stderr) is used for all informational messages, the standard stream is used for all prediction results. So you can save just the prediction results by redirecting the stdout to a file using "¿" on a Unix machine.

FRAMEDhas different versions of prediction output: short and detailed The default is the short one.

- **Short**: the start, end and phase of each gene repdicted is written. Partial genes (on the border of the sequence) are indicated by the use of > or <. Frameshifts are indicated by a vertical bar (|) that separates the sequence on the left and on the right of the frameshift.

  When a frameshift is detected in a gene, FrameD also report as a detected ORF the sequence that begins on the same START as the gene but ignoring the framehift (the ORF stops at the first STOP in frame with the start). This can be controlled using the `-R` flag.

- **Detailed**: this is obtained using the `-pd` flag. The output is different if `-e` is used or not. In both cases, beware: one line is printed per nucleotide in the sequence.

  if `-e` is NOT used: 17 columns are printed:

  – 1: nucleotide number (starting at 1)

  – 2: actual nucleotide (both strand are given)

  – 3: first prediction. 0 means non coding, 1,2,3,-1,-2,-3 are used for gene prediction.

  – 4: second prediction (for overlapping genes). Same as above.

  – 5 to 11: coding/non coding potential in respectively phase 1,2,3,-1,-2,-3 and non coding.

  – 12, 13: STOP occurrence on both strands. The phase of the STOP is given.

  – 14, 15: START occurrence on forward strand. Col. 14 indicates the START phase, next column indicates the quality of the RBS. The higher the better.

  – 16, 17: idem for reverse strand.

  if `-e` is used, 16 columns are printed:

  – 1: nucleotide number (starting at 1)

  – 2: first prediction. 0 means non coding, 1,2,3,-1,-2,-3 are used for gene prediction.

  – 3: second prediction (for overlapping genes). Same as above.

  – 4 to 16: confidence of FRAMED on the fact that the sequence is in the corresponding state (actually this is a probability in a probabilistic model):

    * 4-6: coding phase 1,2,3
    * 7-9: coding phase -1,-2,-3
    * 10: non coding
    * 11: insertion frameshift
    * 12: deletion frameshift
    * 13-14: START used (forward/reverse)
    * 15-16: STOP used (forward/reverse)

4 Controlling FRAMED behavior: the flags

FRAMED behaviour is controlled by several flags. Here is a list of all possible flags with their associated semantics:

d : must be used for analyzing sequences from organisms with a non standard codon table where the `TGA` codon codes for Tryptophane rather than being a Stop codon.

h : gives minimal help on FRAMED flags.

a : must be followed by the filename (possibly with a path) of the parameter file (default `FrameD.par`). Note that you may also set the environment variable `FRAMED_DIR` to specify a directory. This directory path indicates where the various files needed for FRAMED execution lies (`.mat` interpolated Markov model file, `FrameD.par`,`loop.dat`, `stack.dat`...).

i : followed by a floating point number. This number is user to penalize the non coding score from the 0th order Markov chain model. This option increases

FRAMED sensitivity. It is advised to desactivate FRAMED frameshift detection when this option is used (simply use the flag `-f9999`) or else false positive frameshifts will likely appear.

e : compute expectations of frameshifts and predicted states.

E : used to perform eukaryotic EST analysis: ATG is the only START codon, no RBS is sought, genes do not overlap (See `-O`). Priors probabilities on the coding non coding states are modified: the a priori probability that a sequence is coding is lowered.

m : must be followed by the filename of a binary interpolated markov model. The `default.mat` distributed with FRAMED is for *Sinorhizobium meliloti*. You may find models for other organisms on FRAMED web site or you may build your own models (see 6).

n : must be followed by 1, 2 or 3. Indicates the way the score are normalized accross the 7 possibles states (phase 1,2,3,-1,-2,-3 and non coding):

- 1: no normalization
- 2: normalize accross all states
- 3: normalize each coding phase w.r.t. to the non coding score only.

Default is 2. Does not affect prediction, only text/graphical output.

O : followed by a floating point number. Controls overlapping gene ability. Some organisms tend to have many overlapping genes. FRAMED does not like gene overlapping by default. If you want to limit or increase its ability to detect overlapping genes (on the same strand) use `-O` with a floating number between `0.0` and `1.0`. `1.0` makes it easy for FRAMED to detect overlapping genes while `0.0` will makes it difficult. Default value is `0.5`.

w : followed by the size of the smoothing window for all the scores. Default is $w = 48$ which leads to a window of size $2 \times w + 1 = 97$.

f : followed by a floating point number which indicates the opposite of the a priori log of probability of a frameshift. Default is given in `FrameD.par` (around -14 usually). Lowering this number increases frameshift detection sensitivity but lowers the specificity.

C : corrects frameshits. A new file with suffix `.cor` is generated that tries to correct frameshift by inserting 2 `N` when an insertion is detected and inserting one '`N`' when a deletion is detected. If `sdtin` was used for input, `stdout` is used for output.

t : translates to amino acid. A new file with suffix `.tra` is generated that contains the amino acid translation of all ORFS identified. The sequence must be frameshift free. If `sdtin` was used for input, `stdout` is used for output.

b : followed by a filename that typically gives results about BLASTX hits on a protein database. Each line has 5 fileds that correspond to:

`<Hit start> <Hit end> <Score> <E> <Phase>`.

See file `example.blast`.

B : followed by a filename that gives NCBI BLASTX output formatted using the "tabulated output" (-m8) format. Pay attention, this option exists only in recent releases of NCBI-BlastX.

S : overrides the default value for the "Blast pseudo count" given in the parameter file.

g : ask for graphical PNG output. This may be followed by a base filename which will be completed by the number of the figure + `.png` extension. If no filename is given, the sequence filename (trimmed from its suffix) is used instead.

s : followed by a number between 0 and 255 (default 255). Each bit in the binary representation of the number indicates if a given element must (or not) appear on the graphical output:

1 predictions

2 interpolated Markov models normalized scores (smoothed on the `-w` window)

4 Stops

8 Starts

16 Frameshift expectation

32 Mean (expected) behavior

64 local GC percent (smoothed on the `-w` window)

128 local GC3 percent (smoothed on the `-w` window)

u : followed by the number of the 1st nuc. which will be plotted on graphical output (allows for zoom'in). Default is 1. This number is also inserted in the graphical output filename.

v : followed by the number of the last nuc. which will be plotted on graphical output (allows for zoom'in). Default is seq. length. This number is also inserted in the graphical output filename.

c : followed by the number of overlapping nucleotides between 2 successives PNG images. Default is heuristically determined based on resolution and number of nuc. per image.

l : followed by the number of nucleotides that will appear on a single image. The default value is $\min(6000, \text{length to visualize})$. The length to visualize is computed from the value given to `-u` and `-v` (default is all sequence)

x : followed by the horizontal resolution for the PNG images. Default is 900.

y : followed by the vertical resolution for the PNG images. Default is 300.

p : textual output format. May be d (detailed) or s (short).

R : when a frameshifted gene is predicted, FrameD textual output will contain both the prediction of the frameshifted gene and the prediction of a gene with the same START codon as the frameshifted gene and a STOP that corresponds to the first in-frame STOP. This ORF may be used in an annotation protocol that does not support frameshifts. The flag removes this additional prediction.

r : followed by a short sequence that gives the Shine-Delgarno sequence for your organism. Default is *E. coli* motif `attcctcca`.

o : followed by an integer. Gives an offset on the position of the sequence. Modifies only the nuc. numbering.

5 Contents of the `FrameD.par` file

The file contains several lines that control FRAMED behaviour. The content of this file is crucila for FRAMED to work correctly. Do not modify it lightly. Here is an explanation of the contents, line by line:

- 1: Key that restrict the binary to the machine (I use this to know how many copies of FRAMED are used). If you need another key for another machine, just ASK. We can also build key that restrict FRAMED to a class C subnet instead of a machine.

- 2: Default FrameShift penalty. A negative number that tells how likely a frameshift is. Values from -10 to -18 are typically used for *Sinorhizobium meliloti*. EST analysis usually requires smaller absolue value. This should be tuned for your organism and the quality of your sequences.

- 3 and 4: Start parameters α and β. Typically, you should not need to modify these two lines. These parameters are used to control how the RBS quality affect the probability of use of a given START codon. If E is the hybridization energy of the RBS, the following probability is used:

$$\frac{\alpha}{1 + \beta \cdot \exp(E/RT)}$$

  So, α gives a basic probability for a START to be used. For β, the higher it is, the stronger the influence of the RBS quality on the use of the corresponding START. These parameters have been estimated on *Bacillus subtilis* genome using an optimization procedure. START probability is also affected by their type: `atg` is the preferred START codon, then `gtg` and `ttg`.

- 5: Stop penalty. If you lower this, you should increase the ability of FRAMED to detect small (or poorly detected) genes at the cost of more false positive. Values around 3 or 4 are fine usually. This parameter has been estimated on *Bacillus subtilis* genome using an optimization procedure.
- 6: Blast pseudocount. Tells FRAMED how much confidence you have in the blast information. If you want FRAMED to draw blast hits in the graphical output but don't want to take them into account for frameshift prediction/annotation then use a small value like `1e-20`. This is what is done here for *Sinorhizobium meliloti*. Else value 2.0 to 4.0 seems to give good results with SwissProt. This depends heavily on the quality of the protein database used and the blastx score used (true score with `-b` and bitscore with `-B`).

6 Building your own interpolated Markov models

FRAMED uses 3-periodic interpolated Markov models (IMM) to caracterize the contents of coding region. IMMs have been introduced in the prokaryotic software Glimmer 1.0. Such models capture non uniformities in codon usage, amino acids, and also possible correlations between successive codons and amino acids. The `default.mat` file distributed with FRAMED has been estimated on *Sinorhizobium meliloti*. If you want to build your own model for your own organism, you must contact the authors of Glimmer (/www.tigr.org/softlab/glimmer/glimmer.html) and ask for a license and the sources for Glimmer release 1.03 or 1.04 (if you already have an older version, it's ok too). Don't use later version (eg. Glimmer 2.0) because they abandonned interpolated Markov Models for another type of model that FrameD does not use.

1. go in the Glimmer directory (`cd Glimmer` typically)
2. make the binary `build-imm` (eg. `g++ build-imm.c -O -o build-imm` if you use the GNU c++ compiler, else replace `g++` by the name of your favorite C++ compiler).
3. copy the executable in the FrameD directory

Now, to build a model for your organism, first build an IMM model using the `build-imm` binary of Glimmer (see Glimmer documentation), then apply the `glimmer2framed` executable provided with FrameD on the `.delta.context` file produced by `build-imm`. This will produce a `.mat` file that can be used instead of `default.mat`. One advantage of interpolated Markov models is their ability to locally adapt the order of the Markov model estimated to the amount of sequences available. So in practice, the amount of sequences needed to estimate a reliable model is not as crucial as for traditional fixed order Markov models:interpolated Markov models automaticallly adjust the maximum order used according to the available data for each conditional probability distribution. A minimum of 5 to 10 kbases of CDS is a good idea and more is better.

A specific program for estimating extended IMM that can later handle noisy sequences very cleanly is available from the author (tschiex@toulouse.inra.fr). It is well adapted to handle unfinished sequences such as EST data. It is provided as a binary that takes as input a FASTA file of CDS and outputs a (huge) binary matrix file for FrameD.

7 Underlying methods

FrameD gene model is based on a directed acyclic graph designed in such a way that any path in this graph corresponds to a possible gene structure, A simplified view of this graph for a small sequence is represented in Figure 2. It contains 7 parallel tracks that correspond to possible predictions coding regions in each coding phase and non coding regions. START and STOP codons allows to change from a track to another. In practice, to model overlapping genes, FrameD uses a more complex graph with 13 tracks, the 6 additional tracks corresponding to regions where genes overlap on the same strand. The graph model does not model gene overlapping on opposite strands (See section 8). Frameshifts in the sequence are further repsented by additional edges that connect coding phases from the same strand.

To choose a prediction among all possible predictions, the graph is weighted by log-probabilities. 3-periodic in-

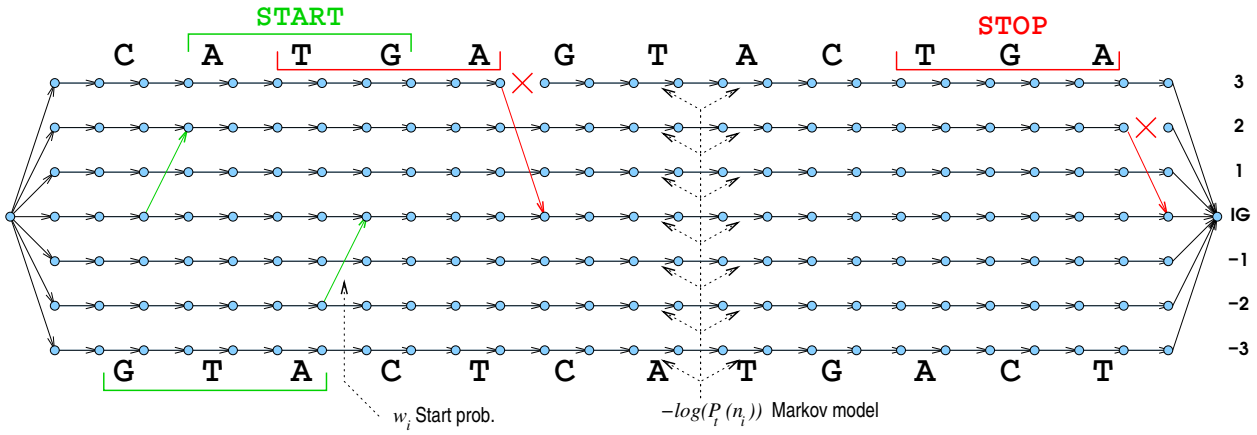

Figure 2: A simplified view of the graph used by FrameD for CATGAGGTACTGA

terpolated Markov models (introduced in Glimmer [1]) are used to estimate the probability that a nucleotide appears in a given coding/non coding track. For signals such as eg. START, if p is the probability that the START is used then the edge that corresponds to the use of the START receives weight $\log(p)$ while the edge that ignores the START receives weight $\log(1-p)$. To estimate the probability, a simple thermodynamic model is used to compute the hybridization energy between a short sequence before the START codon and an RBS motif. This energy is computed using the elementary energies available in [2], ignoring eg. tetraloops. The probability p is derived from the energy E as follows:

$$\frac{\alpha}{1+\beta\cdot\exp(E/RT)}$$

The parameters α and β have been estimated on *Bacillus subtilis* genome using an optimization procedure based on a genetic algorithm. Similarly STOP and frameshift edges receive a constant weight. To find a prediction of optimal score, a simple Belmann dynamic programming algorithm is used. To compute the mean prediction and expected frameshifts, a "Forward-Backward"-like algorithm [3] is used.

For blastX analysis, we we define the mean (bit) score $\bar{s}$ of a hit as the (bit) score divided by the hit length. For each nucleotide in the sequence, the strongest mean (bit) score is used to enhance the coding probability of the nucleotide according to the coding model using a pseudo-count approach: if p is the probability of the nucleotide in a coding region given by the Markov model, the probability used is $p' = \frac{p+\alpha.\bar{s}}{1+\alpha.\bar{s}}$ where α is a user-defined parameter (BlastX pseudo-count).

In order to deal with noisy sequence, this model has been further extended as follows: first, the interpolated Markov models have been extended over the alphabet `A,T,G,C,N`. The probability that a `N` appears after any context is equal to 1.0. The probability that a given nucleotide appears after a given k-mer is estimated from a learning set composed of coding sequences. To estimate these probabilities, we compute the number of times each nucleotide `A, T, G, C` appears after any occurrence of the k-mer (where `N` are considered as wildcards) and derive the probability of appearance of each nucleotide as usual [1].

When such a noise resistant model is not available, FrameD can transparently extend a simple interpolated Markov model over the alphabet `A,T,G,C` to the alphabet `A,T,G,C, N`: the probability that a given nucleotide appears after a given k-mer containing one `N` is simply computed as the mean of the 4 probabilities obtained by replacing `N` by `A,T,G,C` successively. When more `N` appear in the k-mer, the same process is used inductively. This process is identical to the process used in

ESTScan [4] but is less rigorous than a true estimation as described above.

8 Limitations

Do not forget that FRAMED has weaknesses. One of the most important one is that FRAMED is UNABLE (due to the underlying probabilistic model) to predict overlapping genes when the two genes are on different strands[1]. Although this is rare, in the case it happens, FRAMED will tend to explain the situation by a frameshift (because it considers the overlapping of opposite direction genes as impossible). The example sequence has been chosen to make you aware of this limitation (see figure 1). On this sequence, the textual prediction of FRAMED is

```
   Start    End  Ph.

>      1    661 -3
     714   1439 -2
    1717   2994  1
    3000   3004 -2   |
|   3005   3924 -1
    2962   3924 -1
    4064   4807 -3
```

You see that FrameD predicts a gene on phase 1 from `1717` to `2994`. There is another gene on phase -1 that overlaps this gene. Rather than predicting an overlap, FRAMED predicts a frameshift that stops the gene on the reverse strand (gene from `3924` to `2962`) earlier by using a STOP on another phase (-2). This kind of behavior is rare but typical and easy to detect. This restriction may be removed in future versions. Recently, a postprocessor has been added to FRAMED to enable a correct prediction even in this case: when a framehift is detected, FRAMED also predicts an ORF from the start of the frameshifted gene (here `3924`) to the first in phase STOP (here `2962` which corresponds to the correct gene in this case). Thos postprocessing can be deactoivated using the `-R` flag.

[1] FRAMED has no problem to detect overlapping genes on the same strand. Eg. the usual `atga` sequence which is often used to start a gene on the `atg` and stop another on the `tga` is perfectly handled by FRAMED.

There are other weaknesses obviously. One is that FRAMED may miss genes if their coding style is quite different from the mean coding style. One possible way to recover these genes is to use the `-i` flag and/or to lower the Stop penalty in the `FrameD.par` file to eg. `1.0`. But be cautious about the results... false positive may appear. Actually, the visualization of the STOP codons and the coding/non/coding potential of the sequence in the graphical output allows to detect apparently non coding non mirror long orfs: such orfs usually have a "grassy" coding potential, with many peaks, up and downs, which is usually a good indicator of non standard coding statistics (possible horizontally transferred genes).

Acknowledgments FRAMED owes much to GLIMMER. FrameD relies for the moment on an extended version of the Glimmer estimation code for building interpolated Markov Model (see section 6).

References

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

[2] Serra, M., Turner, D., and Freier, S. (1995) Predicting thermodynamic properties of RNA. *Meth. Enzymol.* **259**, 243–261.

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

[4] Iseli, C., Jongeneel, C., and Bucher, P. (1999) Estscan: a program for detecting, evaluating, and reconstructing potential coding regions in est sequences. In Proc Int Conf Intell Syst Mol Biol. pp. 138–48.