GLIMMER Release Notes
Version 3.01 (Beta)

Arthur L. Delcher

10 October 2005

1 Introduction

This document describes Version 3 of the Glimmer gene-finding software. This version incorporates a nearly complete rewrite of the code, resulting in improvements in both sensitivity and specificity of the predictions.

This is a beta version of the software. Not all features are fully implemented and tested. Users discovering problems or errors are encouraged to report them to `adelcher@umiacs.umd.edu`.

2 What's New in Version 3

Changes have been made in the algorithms to score and select genes in Glimmer3, and also in the options and output formats:

1. In both Glimmer2 and Glimmer3, orfs are scored, and those scoring above the threshold value form the candidate set.

   In Glimmer2, pairwise overlaps between these candidates are examined, and using a series of rules, orfs are eliminated or start sites adjusted. This continues in an iterative fashion until no further changes occur. In many cases, the rules cannot resolve an overlap between two orfs, and both are output in the final list of predictions, which have comment tags indicating this.

   In Glimmer3, a single dynamic-programming, HMM-like algorithm is used to select the highest-scoring orfs and their start sites. This algorithm guarantees that the predictions have no overlaps (longer than the allowed value set by the `-o` option). Thus, there are no longer any comments with the Glimmer3 predictions, and in general there are fewer predictions, reducing the false-positive rate. Out tests indicate that there is no corresponding increase in false negatives for Glimmer3 compared to Glimmer2.

2. Glimmer3 scores orfs in the reverse direction, *i.e.*, 3' to 5'. This improves the accuracy of scores near the start codon of genes because the trailing context of the ICM is in the coding region of the gene (on which it has been trained).

3. The `long-orfs` program now uses an amino-acid distribution model to filter the set of candidate orfs before a subset of sufficiently long, non-overlapping orfs is selected.

4. The `make` system and directory structure has been revised so that the source, object and executable files are now in separate directories.

5. There have been some changes in program parameters, including:

   (a) Program options are now specified *before* required parameters, rather than after. Most options now have a long form in addition to the single letter form.

   (b) `build-icm` uses a parameter to specify the output file for the ICM, instead of sending it to standard output like Glimmer2. This parameter can be "-" to direct output to standard out, if desired.

    (c) `glimmer3` requires a third parameter, which is used as a prefix for its output files.

6. There have been some changes in the format and/or meaning of output values. Specifically:

    (a) GLIMMER3 produces two output files: a file with detailed information about all orfs (similar to the first part of GLIMMER2 output), and a file containing just the final predictions (like the second part of GLIMMER2 output).

    (b) The prediction coordinates in GLIMMER3 now include the stop codon and so will differ from GLIMMER2 values by 3.

    (c) Orfs are now printed with a score, which is 100 times the log odds per base of the in-frame coding score versus the score of the independent, non-coding model. These scores provide a consistent scale to compare scores of different orfs.

    (d) The `-X` option will now report genes extending past the end of a sequence with a coordinate that is either less than or equal to zero, or greater than the sequence length.

7. GLIMMER3 can now process multiple-sequence input files. The outputs for each sequence are preceded by the fasta-header line of the sequence in both the `.detail` and the `.predict` files.

8. Two GLIMMER2 options have been eliminated:

    `-p` Was used to specify acceptable overlaps of genes as a percentage of their lengths. This is problematic since the choice of start site affects gene length.

    `-w` Specified the minimum length of an orf that might be considered a gene based on scores of intersecting orfs. Setting a suitably low score threshold (with the `-t` option) effectively includes these orfs.

3 Installing and Running Glimmer3

GLIMMER software was written for the Linux software environment. The following instructions assume a Linux system. They also work under Mac OSX.

3.1 Installation

To install GLIMMER3, download the compressed tarfile `glimmer300.tar.gz` from the website. Then uncompress the file by typing

```
tar xzf glimmer300.tar.gz
```

A directory named `glimmer3.01` should result. In that directory, is a subdirectory named `src`. Within the `src` subdirectory type

```
make
```

This will compile the Glimmer3 programs and put the executable files in the directory `glimmer3.01/bin`. These files can be copied or moved to whatever directory is convenient to the user.

3.2 Running Glimmer

Running Glimmer is a two-step process. First, a probability model of coding sequences, called an interpolated context model or ICM, must be built. This is done by the program `build-icm` from a set of training sequences. These sequences can be obtained in several ways:

1. From known genes in the genome, *e.g.*, genes identified by homology searches
2. From long, non-overlapping orfs in the genome as produced by the program `long-orfs`.
3. From genes in a highly similar species/strain.

Once the probability model is built, the `glimmer3` program itself is run to analyze the sequences and make gene predictions. `glimmer3` has a number of different options that affect its predictions. One of these (`-b`) provides the program with a position weight matrix (PWM) representing the ribosome binding site for genes and is used to improve the accuracy of start site predictions.

3.3 Useful Scripts

In the `scripts` subdirectory are several scripts that are useful for running Glimmer3:

`g3-from-scratch.csh` is a sample shell script that first uses program `long-orfs` to extract a training set of genes and then runs `glimmer3` on the result. Before using the script, the user will need to modify the paths to the programs/scripts by changing the directories specified in the lines that begin with `set awkpath` and `set glimmerpath`. It may also be desirable to change the `glimmer3` options on the `set glimmeropts` line.

To run the script, say, on the genome sequence in file `genom.seq` and prefix the output files with the tag `run1`, simply type:

```
g3-from-scratch.csh genom.seq run1
```

`g3-from-training.csh` is a sample shell script that uses a given set of gene coordinates to extract a training set and then run `glimmer3`. This script uses the program `elph` (available from TIGR at `www.tigr.org/software/ELPH`) to create a PWM from the region upstream of the start sites in the specified coordinate sets. It also uses the first codons in the training set to estimate the start-codon distribution for the genome. This script also requires the user to modify the path variables.

To run the script on the genome sequence in file `genom.seq`, with file `train.coords` containing the positions of the training sequences in `genom.seq`, and using tag `run3` to prefix the output files, type:

```
    g3-iterated.csh genom.seq train.coords run3
```

`g3-iterated.csh` is a shell script that combines the two preceding scripts. It uses the predictions from the scratch run to create a training set for the second run. The reason for a second run is that the output from the first run will have a more accurate set of start sites than the output from the `long-orfs` program, which automatically uses the most upstream start site. These start sites allow the creation of a PWM for the ribosome binding site and the estimation of start-codon usage in the genome.

To run the script on the genome sequence in file `genom.seq` and prefix the output files with the tag `run3`, type:

```
    g3-iterated.csh genom.seq run3
```

The Awk scripts in the `scripts` directory are called by the above scripts.

4 Sample Run Directory

A directory containing a sample run of Glimmer3 is provided. This directory, named `sample-run` contains the genome sequence for *Treponema pallidum* (file `tpall.1con`) and a list of annotated genes for it (file `tpall.nh`). The files whose names begin `from-scratch` are the result of running the script

```
  g3-from-scratch.csh tpall.1con from-scratch
```

The files whose names begin `from-training` are the result of running the script

```
  g3-from-training.csh tpall.1con tpall.nh from-training
```

Users will need to modify the path directories in these scripts to be able to run them.

5 Notes on the Programs

5.1 `glimmer3` Program

5.1.1 `glimmer3` Options

The invocation for `glimmer3` is:

`glimmer3` [⟨*options*⟩] ⟨*sequence*⟩ ⟨*icm*⟩

where ⟨*sequence*⟩ is the name of the file containing the DNA sequence(s) to be analyzed, ⟨*icm*⟩ is the name of the file containing the ICM model produced by `build-icm`, and ⟨*options*⟩ can be the following:

`-A` ⟨*codon-list*⟩ or `--start_codons` ⟨*codon-list*⟩

Specify start codons as a comma-separated list. Sample format: `-A atg,gtg` Use the `-P` option to specify the relative proportions of use. If `-P` is not used, then the proportions will be equal

`-b` ⟨*filename*⟩ or `--rbs_pwm` ⟨*filename*⟩

Read a position weight matrix (PWM) from ⟨*filename*⟩ to identify the ribosome binding site to help choose start sites.

`-C` ⟨*p*⟩ or `--gc_percent` ⟨*p*⟩

Use ⟨*p*⟩ as the GC percentage of the independent model, *i.e.*, the model of intergenic sequence. Note: ⟨*p*⟩ should be a percentage, *e.g.*, `-C 45.2`

`-E` ⟨*filename*⟩ or `--entropy` ⟨*filename*⟩

Read entropy profiles from ⟨*filename*⟩. The format is one header line, then 20 lines of 3 columns each. The columns are amino acid, positive entropy, and negative entropy, respectively. Rows must be in alphabetical order by amino acid code letter. This currently is not used in the GLIMMER3 prediction algorithm, but is used in the `long-orfs` program.

`-f` or `--first_codon`

Use the first possible codon in an orf as the start codon for initial scoring purposes. Otherwise, the highest-scoring codon will be used. This only affects the start positions in the `.detail` file. The final start predictions in the `.predict` file are always based on the scoring functions.

`-g` ⟨*n*⟩ or `--gene_len` ⟨*n*⟩

Set the minimum gene length to ⟨*n*⟩. This does not include the bases in the stop codon.

`-h` or `--help`

Print the usage message.

`-i` ⟨*filename*⟩ or `--ignore` ⟨*filename*⟩

⟨*filename*⟩ specifies regions of bases that are off limits, so that no bases within that area will be examined. The format for entries in the file is one line per region, with the start and end positions of the region specified as the first two fields on the line. The rest of the line is regarded as comments. Additionally, any line beginning with a `#` is regarded as a comment. *E.g.*, the following file:

```
   1001     1600   Comment here
# The region can be specified high-low as well as low-high
   5600     5601
```

would ignore bases 1001 . . . 1600 and 5001 . . . 5601 in the input sequence. This option should not be used with multi-sequence input files.

`-l` or `--linear`

Assume a linear rather than circular genome, *i.e.*, there will be no "wraparound" genes.

`-L` ⟨*filename*⟩ or `--orfs_coords` ⟨*filename*⟩

⟨*filename*⟩ specifies a list of orfs that should be scored separately, with no overlap rules. *** Not implemented currently ***

`-M` or `--separate_genes`

⟨*sequence-file*⟩ is a multifasta file of separate genes to be scored separately, with no overlap rules *** Not implemented currently ***

`-o` ⟨*n*⟩ or `--max_olaps` ⟨*n*⟩

Set the maximum overlap length to ⟨*n*⟩. Overlaps of this many or fewer bases are allowed between genes. The new dynamic programming algorithm should <u>*never*</u> output genes that overlap by more than this many bases.

`-P` ⟨*number-list*⟩ or `--start_probs` ⟨*number-list*⟩

Specify the probability of different start codons (same number and order as in `-A` option). If no Pg-A option is given, then it should be 3 values for `atg`, `gtg` and `ttg`, in that order. Sample format: `-P 0.6,0.35,0.05`. If `-A` is specified without `-P`, then starts are equally likely (which is very unusual).

`-q` ⟨*n*⟩ or `--ignore_indep_len` ⟨*n*⟩

Set the maximum length orf that can be rejected because of the independent probability score column to $\langle n \rangle - 1$. Without this option, this value is calculated automatically to be the length such that the expected number of orfs this long or longer in a random sequence of a million bases is one.

`-r` or `--no_indep`

Don't use the independent probability score column at all.

`-t \Desc{n}` or `--threshold \Desc{n}`

Set the threshold score for consideration as a gene to ⟨*n*⟩. If the in-frame score $\geq$ ⟨*n*⟩, then the region is given a number and considered a potential gene.

`-X` or `--extend`

Also score orfs that extend off the end of the sequence(s). This option presumes that the sequence(s) is linear and not circular. Reported positions off the end of the sequence are the nearest positions in the correct reading frame. Note that this ignores any partial codons at the ends of a sequence. Suppose, for example, that a sequence is 998bp long and an orf in reading frame +1 starts at position 601 and extends off the end of the sequence. Then the end of that gene/orf will be reported at position 999, as if the stop codon were in positions 997. . . 999. This is true even if the last two characters of the sequence are, say, `cc` and cannot possibly be part of a stop codon.

Any scores associated with orfs that extend past the end of a sequence are computed using only complete codons contained in the sequence.

`-z` ⟨*n*⟩ or `--trans_table` ⟨*n*⟩

Use Genbank translation table number ⟨*n*⟩ to specify stop codons.

`-Z` ⟨*codon-list*⟩ or `--stop_codons` ⟨*codon-list*⟩

Specify stop codons as a comma-separated list. Sample format: `-Z tag,tga,taa`

5.1.2 `glimmer3` Output Formats

`.detail` File

The `.detail` file begins with a list of the parameters used by the program. Here is a sample:

```
Sequence file = msmeg.1con
ICM model file = ms2.icm
Excluded regions file = none
List of orfs file = none
Truncated orfs = false
Circular genome = true
Minimum gene length = 110 bp
Maximum overlap bases = 104
Minimum problem overlap fraction = 10.0%
Input is NOT separate orfs
Threshold score = 70
Minimum gene length for weak scores = max
Use first start codon = false
Start codons = atg,gtg,ttg
Start probs = 0.611,0.332,0.057
Stop codons = taa,tag,tga
```

Following that, for each sequence in the input file the fasta-header line is echoed and followed by a list of orfs that were long enough for `glimmer3` to score. Here is a sample of the beginning of such a section:

```
>gms:3447|cmr:632 chromosome 1 {Mycobacterium smegmatis MC2}
Sequence length = 6988209
GC percentage = 67.4%
Ignore independent score on orfs longer than 1444

           ----- Start -----           --- Length ----  ------------- Scores -------------
 ID  Frame   of Orf  of Gene     Stop   of Orf of Gene     Raw InFrm F1 F2 F3 R1 R2 R3 NC
        -3      173      155       42      129     111    1.15     7  -  1  7  -  -  7 83 1.00
        +2       14       32      355      339     321   -1.92     0  -  0  2  -  -  - 97 1.00
        +3  6988011      336      461      657     123   -2.37     0 43  -  0  -  -  0 56 1.00
        -3      464      392      273      189     117   -9.72     0  0  -  0  -  -  0 99 1.00
        +1      100      364      495      393     129   -1.30     0  0  -  -  -  -  - 99 1.00
        +3      462      468      647      183     177    0.08     2  -  -  2 25  0  - 72 1.00
        -3      671      641      477      192     162    2.14    49  -  - 19 17  0 49 12 1.00
        -3      866      806      672      192     132    3.24     0 99  -  -  0  -  0  0 1.00
        +2     986    1349    1549     561     198    -5.58     0 99  0  0  0  -  0  0 1.00
        -3    1652    1346     867     783     477     4.99     0 99  -  -  0  -  0  0 1.00
0001    +1     496     529    1692    1194    1161    13.60    99 99  -  -  0  -  -  0 1.00
0002    -1    1983    1977     415    1566    1560     7.39    99  -  -  - 99  -  -  0 1.00
        +3    2094    2106    2249     153     141    -1.65     0  - 99  0  -  0  0  0 1.00
0003    -2    2365    2254    1559     804     693     5.05    99  -  -  -  - 99  -  0 1.00
        -1    2415    2385    2269     144     114     3.07     0  0 99  -  0  -  -  0 1.00
        -1    2580    2568    2416     162     150     9.16     0  0 99  -  0  0  0  0 1.00
0004    +2    1670    1721    2614     942     891    14.25    99  - 99  -  -  -  -  0 1.00
        +1    2110    2326    2745     633     417    -0.57     0  0  -  -  -  -  - 99 1.00
        -3    2846    2765    2598     246     165    -8.44     0  -  -  -  - 88  0 11 1.00
```

Below is a description of the columns. All positions are counted from the beginning of the sequence with the first base being position 1.

`ID` An identification number for a potential gene. Only orfs whose in-frame (`InFrm`) score is above the threshold score (set by the `-t` option) have an entry in this column.

`Frame` The reading frame of the orf—positive for forward strand, negative for reverse strand. It is determined by the position of the leftmost base of the stop codon: frame $+1$ if the stop begins in position $1, 4, 7, \ldots$; frame $+2$ if the stop begins in position $2, 5, 8, \ldots$; frame $+3$ if the stop begins in position $3, 5, 9, \ldots$; frame -1 if the stop begins in position $3, 5, 9, \ldots$ (so the left most base is position $1, 4, 7, \ldots$; frame -2 if the stop begins in position $4, 7, 10, \ldots$ frame -3 if the stop begins in position $5, 8, 11, \ldots$. Note that if the gene wraps around the end of the sequence the same rules applied to start position will not yield the same reading frame.

`Start` The positions of the first base of the orf and the first base of the start codon of the gene. Note that the gene start may be different for the same orf in the `.predict` file.

`Stop` Position of the last base of the stop codon.

`Length` Number of bases in the orf and in the gene. It node does *NOT* include the bases of the stop codon.

`Raw` Score This is 100 times the per-base log-odds ratio of the in-frame coding ICM score to the independent (*i.e.*, non-coding) model score. It gives a rough quantification to how well an orf scores that can be compared between any two orfs.

`InFrm` Score The normalized (to the range $0 \ldots 99$) score of the gene in its reading frame. This is just the appropriate-frame value among the next six scores.

Frame Scores The normalized (to the range $0 \ldots 99$) score of the gene in each reading frame. A `-` indicates the presence of a stop codon in that reading frame. The normalization compares only scores without stop codons and the independent (non-coding) `NC` score. If the orf is sufficiently long, *i.e.*, longer than the value stated in "`Ignore independent score on orfs longer than...`", the `NC` score is not used. Thus, for example, the orf with ID 0002 above automatically scores 99 in its reading frame because the other reading frames all have stop codons, and the orf is longer than the minimum ignore length.

`NC` Score	The normalized independent (*i.e.*, non-coding or intergenic) model score. This model is adjusted for the fact that the orf, by definition, has no in-frame stop codons.
Last Column	This is the entropy-distance ratio, and is not currently used in the scoring process.

`.predict` File

This file has the final gene predictions. It's format is the fasta-header line of the sequence followed by one line per gene. Here is a sample of the beginning of such a file:

```
>gms:3447|cmr:632 chromosome 1 {Mycobacterium smegmatis MC2}
orf00001      499     1692  +1    13.14
orf00004     1721     2614  +2    14.20
orf00006     2624     3778  +2    10.35
orf00009     3775     4359  +1     9.34
```

The columns are:

Column 1	The identifier of the predicted gene. The numeric portion matches the number in the `ID` column of the `.detail` file.
Column 2	The start position of the gene.
Column 3	The end position of the gene. This is the last base of the stop codon, *i.e.*, it includes the stop codon.
Column 4	The reading frame.
Column 5	The per-base "raw" score of the gene. This is slightly different from the value in the `.detail` file, because it includes adjustments for the PWM and start-codon frequency.

6 Versions

6.1 Version 3.01

- Eliminated unused functions.
- Eliminated `-p` and `-w` options.
- Implemented the `-X` option allowing orfs extending off the end (of a non-circular) sequence to be scored.

- Changed the width of the PWM in the scripts from 5 to 6.
- Added the `g3-iterated` script to combine running Glimmer3 from scratch and using the output as a training set for a second run.
- Lowered default threshold score (`-t` option) in scripts.