YACOP - Readme file

Maike Tech

Contents

1 General

The metatool YACOP [1] generates a prediction, which is based on the output of existing gene finding programs like Glimmer or Orpheus. In present YACOP supports the combination of Critica105b [5] (with wublast or NCBI blast), Glimmer2.02 [3] or Glimmer2.10 [2] (with RBSfinder [4]), Orpheus [6] (with dps [7]) and ZCurve1.0 [8]. However, it's possible to extend YACOP and integrate additional tools (see section 8, p.8), for its source code is available. The current version of YACOP is realized in Perl (v.5.6.1). Its implemented and tested under RedHat Linux v2.4.18 and Debian Linux v2.6.5.
YACOP expects input in FASTA (starting with script `combinate`) or multiple-FASTA format (starting with script `combinate_multi`). Some of the methods used internally rely on nucleotide, codon, or oligomer frequencies that have to be derived from the input therefore the minimal length of the

input sequence is 50 kB. When started with option `concat`, even shorter contigs can be analyzed, if sufficient fragments of the sequence are available (given as multiple-FASTA).
The starter script `combinate` automatically triggers the gene finding programs and evaluation of their predictions. That prerequisites the correct installation of the tools (Critica, Glimmer etc.). In case of problems concerning the individual gene finding programs, please consult the respective readme files. For ZCurve, Critica and Orpheus it is as well possible to post an output file as to start the program [1]. The paths directing to the tools should be set in the central `ini`-file of YACOP (see section 4, p. 2).

2 System requirements

Most gene prediction tools are implemented in Perl, C or C++. Therefore, these programming languages should be available on your system. If you don't use default paths or default compilers, the tools as well as the sources of YACOP should be edited individually.
Some gene prediction tools (Critica, Orpheus), used by YACOP rely on database searches and statistical analysis, for that it is not recommendable to install i on an normal pc. Unless you would like to use YACOP to combine the output of Glimmer and ZCurve, which are fast and require far less capacity.
The following programming languages have to be available on your system:

1. Perl5.6.1 or higher (should be installed in `/usr/bin/perl`)
2. BioPerl.3 or higher
3. C and C++ (default compiler gcc) needed for Glimmer, Orpheus, dps and Critica

YACOP can be used on multi-processor systems running PBS (Portable Batch System) as well as on single processor systems. In this case, the programs are executed sequentially. For the adaptation of the YACOP source code to different system architectures, see section 7, p. 6.

3 Additional scripts

Some additional conditions have to be fulfilled: The scripts `check_qstat`, `gl2rbs`, `separate` and `toUp` need to be available in your `PATH`. You can either copy the scripts to your `bin` directory or set respective links there. Another possibility is to integrate the whole YACOP home directory into your `PATH`. If you would like to use the scripts manually, have a look at the comments. When you just type the name of a script in the command line, it will give you a line how to use it correctly.

- `gl2rbs` is needed for conversation of Glimmer output to input of RBSfinder.
- `toUp` converts lowercase sequences to Critica-compatible uppercase version. If you start YACOP with `combinate_multi` invocation of `toUp` occurs automatically, unlike during startup of `combinate`, where you should trigger the script manually (if your sequence is composed of lowercase characters). The uppercase sequence is written to the file FASTAfile`_uc` (after usage with `combinate_multi` this file will be removed).
- `check_qstat` is only needed if you use YACOP on multi-processor machines. It checks out current state of the subprocesses. In the script the monitoring utility `qstat` is called, which is part of the OpenPBS.

[1] This is only integrated in the last version of YACOP (since June 2004). The feature to post Critica or Orpheus output to YACOP was added, because of the long runtime of this programs for big genome projects. In the case of ZCurve a Linux version was not available at the time of the first implementation. For older versions the output can be generated under Windows (or Linux) and then be posted manually during invocation to the script `combinate` (not possible for `combinate_multi`).

- `separate` will be invoked in `-concat` mode. It separates the output of `*.tbl` or `.sum_out` to the subsets of the concatenated contigs. The individual results will be written to `~/data/longtime/contigs_tbl/_nr.NAME` or `~/data/`*longtime*`/contigs_sum/_nr.NAME`. The suffix `NAME` corresponds to the contig name. The invocation of `separate` is triggered by `combinate`. By default, only the mode-dependent results in `*.tbl` will be separated. If you like to use `separate` on the whole results (mode- and length independent), which are stored in `.sum_out`, you should comment respective code in.

4 Running YACOP

4.1 Parameters during invocation of Yacop

There are two sets of parameters to be considered. First the required parameters: your sequence in FASTA format, the `ini`-file and a prefix for your organism. **These should be given in exact order** (like in the example). Second the optional parameters (given in `[ ]`): the flags for the tools to use and the flags for special modes (`concat` and `verbose`). Here the order doesn't matter.
The script `combinate` should be called with a statement like:

```
combinate gene.fasta ini-file PREFIX [-gr/-gc -c [critica.out] -o [orpheus.out]
-gb gene.gbk -z [zcurve.pred] -v -concat]
```

(The `[ ]` are only added to denote optional arguments and should be left out in a real call.)
The parameters for `combinate_multi` are alike:

```
combinate_multi multi_fasta ini-file PREFIX [-c -o -gr/-gc -z -gb file.gbk -v
 -concat]
```

Note, tthat you should type '`perl ~/YACOP/combinate ...`' if you didn't add the YACOP home directory to your `PATH` variable.
The filename of the FASTA file must be given with the **absolute path**. The term `PREFIX` is synonym for a prefix naming your organism. It is used to compose names of result-files and may be formatted according to the regular expression `/^R[A-Z]{2,3}/`.
The parameters `-gr`, `-c`, `-z` and `-o` tell YACOP which tools to use. The parameter `-c` activates invocation of Critica, if no Critica output file (normally `*3.cds`) is given (not possible for `combinate_multi`). `-gr` activates Glimmer plus RBSfinder. Instead of `-gr` the parameter `-gc` can be given. This triggers Glimmer with RBSfinder like `-gr`, meanwhile for Glimmer training Critica output is used ([9]). This requires, that Critica should be also started or Critica output should be given (`-c`). `-o` triggers the invocation of Orpheus in combination with dps, if no Orpheus output file is given (not possible for `combinate_multi`). With the parameter `-z` ZCurve will be started, if no ZCurve output file is added (not possible for `combinate_multi`).
The parameter `-gb` must be set together with the path to a GenBank file. This option allows to check the performance of YACOP and the integrated tools, by comparing the annotation deposited at GenBank with the output generated by the programs (see section 6, p. 6.
The Parameter `-v` is for verbose mode (writing information of `ini`-file to `properties.inf`). `-concat` starts YACOP in concatenation-mode (see section 5.1, p. 4).
YACOP copies the resulting output to a directory `~/data/`*longtime* (located at the current home directory), see section 6, p. 5.

Summery of parameters:

`-c` triggers invocation of Critica or posting Critica output, if a file is added (*.cds)

`-gr` triggers invocation of Glimmer/RBSfinder

`-gc` triggers invocation of Glimmer/RBSfinder, with Critica-output as training set ([9]). Requires Critica output! So Critica should be invoked (`-c`) or Critica output should be given (`-c *.cds`)

`-o` triggers invocation of Orpheus or posting Orpheus output, if a file is added

`-z` posting ZCurve output or starting ZCurve, if you only type `-z` without adding a file

`-gb` posting a GenBank file

`-v` printing properties to file (verbose)

`-concat` run `combinate` in concatenation-mode (see section 5.1, p. 4)

4.2 The `ini`-file

The `ini`-file (example is given in `YACOP/GenePred.ini`) contains all parameters required by the tools, the path to their home directories and parameters needed by the Perl program `meta_pred.pl`. Additionally a short description of their function and an example is given. All parameters can be set by using this central repository. For a more detailed description of the individual parameters of the tools, you should have a look at the respective readme-files.
The parameter `mode` defines how YACOP merges the individual predictions and how to select the gene starts. Possible settings are:

- `crit_orp_gl` combines the predictions of Critica with the predictions made by both Glimmer and Orpheus. For the prediction of the startcodons the output of Critica gets the highest priority. If an orf is not predicted by Critica, but by Orpheus and Glimmer, the Orpheus output is favored.
- `crit_zc_gl` outputs Critica predictions with the intersection of ZCurve and Glimmer, Critica starts will be preferred, else-wise the start position predicted by ZCurve is taken into account.
- `zc_crit_gl` combines Critca predictions with the intersection of ZCurve and Glimmer predictions. Starts of ZCurve are favored where possible, else-wise Critica starts are taken.
- `crit_zc_gl_orp` combines the predictions of Critica with the intersection of the predictions of Glimmer, ZCurve and Orpheus. Start coordinates are taken from Critica output, otherwise for predictions not found in Critica output the prediction of ZCurve is taken.
- `zc_crit_gl_orp` combines the predictions of Critica with the intersection of the predictions of Glimmer, ZCurve and Orpheus. For the prediction of startcodons, in this mode the ZCurve output is favored. If a predicted orf is not found in the ZCurve output, the Critica start coordinate is taken.
- `zc_gl` outputs intersection of predictions of ZCurve and Glimmer with ZCurve start.
- `gl` only glimmer output is considered.

In order to implement additional modes, the subroutine `eval_start_mode(HASH)` has to be modified (see section 7, p. 6).
Additionally a parameter `-minlength` can be set in the `ini`-file. This causes the tool to reject all predictions shorter than given number (denoted in BP). This limitation obtains for all predictions except such made by Critica. For ZCurve also the specific call (i. e. the name of the executable) should be set. If Glimmer should be trained with Critica output, it is possible to confine the predicted genes that will be considered in the training by p-Value (`gr_crit_maxpval`) and by length (`gr_crit_minlen`).

5 Input format

5.1 FASTA or multiple-FASTA

The input format should be of type FASTA or multiple-FASTA. The script `combinate` handles the triggering of the individual tools. The script `combinate_multi` splits multiple-FASTA input into several FASTA-files. Each of these temporarily generated files will then be processed by an instance of `combinate`, created by `combinate_multi`. If started in `concat` mode `combinate_multi` concatenates the sequence fragments in given multiple-FASTA file and posts it as one sequence to `combinate`. All predicted coding regions traversing a concatenation site are removed from the result and stored to file `meta_pred.msg` in resp. output directory.
Example for the FASTA format:

```
>AP000398_Buchnera_sp._APS_complete_genome.
TTATCCACAGATTTGTTCTTTACTAATAATAATAGTAATTATTATTTTTTATTTTTTTTATTTTTTT
GAATTTAAACCTTAAAGAAAAGAAAAAGATCTTTTTTTTAAGATATTATGTTTTTAAGATTAACATG
TGTTATCTTGAATAAAATATTAATACTATTTGAATATTTTTAAATTTTTAAAAGGTTTTTATATGTT
```

`combinate` accepts only sequences composed of uppercase characters. This is due to limitations imposed by some of the tools that do not work for lowercase sequences. `combinate_multi` does not have this limitation. The script `toUp` is called before passing sequence files to `combinate`. `toUp` (see section 3, p. 2) converts the input sequence to an uppercase version.
The annotation of a contig should start with > (FASTA convention) and should not exceed one line. Note that some of the tools used for prediction do not accept sequence ids which are too long or contain white spaces. E. g. Critica uses the id to form the names of the output files. If the names contain special characters or are too long this may cause Critica to crash without any error message.

6 Output formats

YACOP stores results in the directory `~/data/`*longtime*. Where *longtime* corresponds to the time of invocation of YACOP in long format. If you are not contend with this, you can change the output path in the script `combinate` (see variable `output_dir`).
Each prediction of YACOP consists of three basic output files:

6.1 `PREFIX.sum_out`

The file `PREFIX.sum_out` contains all predictions made by the tools integrated in YACOP. The file can be regarded as a table. Each line describes one predicted orf. The landmarks used to organize the output are the stopcodons occurring in the sequence. The first column contains the positions of the stopcodons (sorted in ascending order), given by the first nucleotide of the codon. The second column lists the reading frame of the gene (comp). The following columns contain the output of the different tools. Each start coordinate is defined as the first nucleotide of startcodon.
Start positions given for **Glimmer** are those generated by RBSfinder (Glimmer/RBSfinder). RBSfinder alters the initial predicted start of Glimmer by shifting them up or downstream if the results differ. The number, given in column 3 is the offset, RBSfinder has introduced for the start position. In brackets the amount of the shift from the position Glimmer has predicted is denoted. A positive number indicates a downstream shift, a negative correspondingly an upstream shift. The output for **Critica** is the position of the startcodons and a p-value, rating the quality of the prediction. For **Orpheus**, only the predicted start position is given. For **ZCurve** the predicted start will be given with a score (like Critica score, rating the prediction of ZCurve). In evaluation mode, i.e. the name of a **GenBank** file is given during startup, the start position and the annotation as deposited

in the file are added to the output.
Example for a file of type `PREFIX.sum_out` (with all integrated outputs):

```
2081 F     164 ( 33)    197 6.82e-147   164   197 0.18091   197 glucose inhibited divis
2771 R    2924 (  -)      -         -     -     -       -     -
3100 F    2278 (  -)   2278  3.08e-20  2278  2278 0.22193  2278 ATP synthase A chain
3376 F    3139 (  -)   3139  2.27e-07     -  3139 0.25229  3139 ATP synthase C chain
3980 F    3497 (  -)   3497  8.36e-16  3497  3497 0.27178  3497 ATP synthase B chain
```

stop comp | Glimmer/RBSf. | Critica | Orpheus | ZCurve | GenBank

6.2 PREFIX.tbl

The file `PREFIX.tbl` contains a gene table. The genes compiled in this file are extracted from the sets of genes predicted by the individual tools. The arrangement of genes is depending on the mode, which has to be set in the `ini`-file. The default mode (as given in example `ini`-file, see section 4, p. 2) is `crit_zc_gl`. For each predicted coding region, one line of output is generated. The first column contains the id of the gene. The id results from the given PREFIX and a number incremented automatically. The id is formatted according to the regular expression `/^R([A-Z]{2,3})\d{5,6}/`. In addition, the name of the contig and the coordinates of the gene are added.
NOTE: 3'-end is given exclusive stopcodon, 5'-end is inclusive startcodon. An example for the format of PREFIX.tbl:

```
RBU000001 AP000398_Buchnera_sp._APS_complete_genome._197_2080
RBU000002 AP000398_Buchnera_sp._APS_complete_genome._2278_3099
RBU000003 AP000398_Buchnera_sp._APS_complete_genome._3139_3375
RBU000004 AP000398_Buchnera_sp._APS_complete_genome._3497_3979
```

6.3 PREFIX.fasta

The file `PREFIX.fasta` consists of the amino acid sequences (inclusive startcodon) of all predicted genes in `PREFIX.tbl`. Each sequence is annotated with the id according to the entries deposited in the tbl-file.

```
>RBU000001
MFNLRNFDVIVVGAGHAGTEAAMASSRMGCKTLLLTQKISDLGALSCNPAIGGIGKSHLVKEIDAL
GGMMAKAIDYSGIQFRILNSSKGPAVRSTRAQADKILYHETVKKILKKQNNLLILEAEVKDLIFKN
YSVVGVLTQNEINFYSRSVVLAAGTFLGGKIHIGLKSYSAGRIGDKSAIDLSVRLRELSLRVNRLK
TGTPPRIDINTVNFNNLLIQNSDTPVPVFSFMGNVSHHPKQIPCYLTHTNEKTHEIIRKNLDKSPI
YTGFLKGLGPRYCPSIEDKIVRFPDRKSHQVFLEPEGLSSIKVYPNGISTSLPIEVQEQIVASIKG
LEKSKIIRPGYAIEYDFFDPKDLNLTLESKLIKGLFFAGQINGTTGYEEAASQGLLAGLNAALSSK
NTEGWFPRRDQAYLGVLIDDLTTQGTEEPYRMFTSRAEYRLSLREDNADLRLTEIGRKLGLVNDSR
WIRYNQKVLNIQTEMNRLKKNKISPISPDADILKKLYNINLIKEISMSELLKRPQIRYQDLQSLES
FRTGIVDLEAIGQIENEIKYAGYIKRQSEEIERHLKNENTFLSSIYDYNKIRGLSSEVVKKLNDYK
PISIGQASRISGITPAAISILLIHLKKESYKHTL
```

6.4 Other files in the output directory

In the output directory you will find several more files. Most of these are output files of the tools, Critica, Glimmer, ZCurve and Orpheus (see table 'Output files'). You will also find a file `meta_pred.msg`. This file contains error messages and predicted genes, that were removed from the resultset in `concat` mode (because they cross concatenation sites). If you started YACOP in `verbose` mode (option `-v`) the file `properties.inf` will be added. This file contains the parameter composition of this invocation of YACOP (set in the `.ini` file). The files named `sh_c`*longtime*`.sh` and `sh_o`*longtime*`.sh` are the invocation scripts of Critica and Orpheus if YACOP is startet on

multi-processor mashines using PBS. *longtime* denotes the instantiation time of resp. `Nudge` object in long format (set in `*Nudge.pm`, e. g. `CritcaNudge.pm`). In this case the files `sh_*`*longtime*`.o` contain all messages printed to STDOUT and `sh_`*longtime*`.e` all messages printed to STDERR by Critica (`*=c`) or Orpheus (`*=o`).

Table 1: Additional YACOP files

file	description
`meta_pred.msg`	error messages, removed genes in `concat` mode
`properties.inf`	configuration of properties for this run (verbose mode)
`sh_c`*longtime*`.sh`	qsub starterscript for Critica
`sh_o`*longtime*`.sh`	qsub starterscript for Orpheus
`sh_`*longtime*`.o`	STDOUT messages of batch processes (i. e. Critica or Orpheus)
`sh_`*longtime*`.e`	STDERR messages of batch processes (i. e. Critica or Orpheus)

Table 2: Output files of the integrated tools

Tool	script	output file
Glimmer	`long-orfs`	`gl_PREFIX.orfs`, `gl_PREFIX.msg_lo`
	`extract`	`gl_PREFIX.extract`
	`build-icm`	`gl_PREFIX.model`
	`glimmer2`	`gl_PREFIX.out`, `gl_PREFIX.msg_gl`
RBSfinder	`gl2rbs`	`PREFIX_tmp.gl2rbs`
		`gl_PREFIX.rbs_out`, `gl_PREFIX.rbs_msg`
Critica	`blast-contigs`	`PREFIX.blast`
	`make-blastpairs`	`PREFIX.blast.pairs`
	`scanblastpairs`	`PREFIX.triplets`
	`iterate-critica`	`PREFIX`nr.cds (nr ≡ iteration)
Orpheus	`dps`	`orph_PREFIX.dps`
	`orpheus2` (`-wcu`)	`orph_PREFIX.nuc_usage`
	`orpheus2` (`sure` oder `rcu`)	`NAME.orfaln`, `NAME.orfnuc`, `NAME.orfprot`
	`starter`	`orph_PREFIX.weights`

7 Modification of the source code

7.1 Adaptation to multi- or single-processor machines

The current version of YACOP can be startet using PBS (Portable Batch System, see 2, p. 1) for Critica and Orpheus. Glimmer and ZCurve are always started as non-batch-processes. The invocation of the batch processes occures with the command qsub(1B), which is provided in the OpenPBS package. The script `combinate` calls the starter objects (inheriting from `Nudge.pm`, e. g. `CriticaNudge.pm`) to create the scripts for submission via qsub and triggers the invocation of the scripts. The objects extending `Nudge.pm` also provide subroutines to start the subprograms sequentially.
In the source code of `combinate` the line

```
$critica->start_qsub_all();
```

triggers invocation of Critica as batch process. If Critica shall be started as non-batch process, this line should be commented out and the following lines should be commented in:

```
$critica->start_blastcontigs();
$critica->start_makeblastpairs();
$critica->start_scanblastpairs();
$critica->start_iteratecritica();
```

The code adaption for Orpheus looks quite similar.
Running as batch process:

```
$orpheus->start_qsub_all();
```

Running as non-batch process:

```
$orpheus->start_starter(OrpheusNudge->nuc());
$orpheus->start_dps();
$orpheus->start_orpheus2(OrpheusNudge->wcu());
$orpheus->start_orpheus2(OrpheusNudge->sure());
$orpheus->start_starter(OrpheusNudge->weights());
$orpheus->start_orpheus2(OrpheusNudge->rcu());
```

For the usage of qsub, additionally the script `check_qstat` is provided. During invocation of `check_qstat` the name of a qsub-script has to be given as a commandline parameter (this is done automatically by `combinate`). It is used to control the execution of the respective subprocess. If you are not using a bash (borne-again shell) on your system, the scripts `check_qstat` and the variable `Nudge::BASH_CALL` (in `Nudge.pm`) should be modified.

7.2 Adding new output modes

The output modes of YACOP are implemented in `meta_pred.pl`. To add new modes you should declare a global variable (like `$M_CRIT_ZC_GL`) to store the name of the new mode. In the routine `eval_start_mode(HASH)`, you should add an `if`-clause with the code to execute, if your mode is set (i. e. which predicted start should be returned). This routine is called by the routine `quick_results(ResultSet)`, which composes the reduced result object `QuickResults`, containing only start and stop of a predicted gene respective to your mode (used to write the files `PREFIX.tbl` and `PREFIX.fasta`). If the new mode is set in the `ini`-file (with the name you stored in the global variable, see above), it will be automatically set by `combinate` and executed by `meta_pred.pl`.
Additionally you should add the mode to respective `if`-clause in the routine `write_all(DataHandle)`, which is called by `output_all(ResultSet)` and writes all predictions of the tools used in your mode to `PREFIX.sum_out`. If you intend to add the predictions of a new tool to the output of YACOP you should add a new `elsif`-clause to that routine, in which the predictions of your tool (called `MyTool` in the example) will be written to STDOUT like:

```
elsif ($MODE eq $M_ZC_GL_MYTOOL){
    printf "%7s %s  %7s (%3s)  %7s %5s  %5 %s %s\n",
    $val->{$STOP}, $val->{$COMPL}, $val->{GlimmerRBSParser->type},
    $val->{(GlimmerRBSParser->type)."S"}, $val->{ZCurveParser->type},
    $val->{(ZCurveParser->type)."S"}, $val->{(MyToolParser->type)},
    $val->{GenebankParser->type}, $val->{$GBK_ANNO};
}
```

7.3 Altering output format

If you require other formats you should change one of the writing routines or create a new one and add the functioncall to the main program of `meta_pred.pl`. If you do so, note that the object `QuickResult` only contains the predictions respective to current mode and minlength, meanwhile `ResultSet` contains all.
Current output- and related routines in `meta_pred.pl` are:

- `output_all(ResultSet)` calling `write_all(HASH)`
  (writing `PREFIX.sum_out`)

- `write_table(QuickResults)` calling `eval_start_mode(HASH)`
  (writing `PREFIX.tbl`)

- `write_multiple_fasta(QuickResults)` calling `eval_start_mode(HASH)`
  (writing `PREFIX.fasta`)

Additionally, predictions which are discarded in `concat` mode are written to STDERR in routine `trim_results(ResultSet)`. Error messages of `meta_pred.pl` are redirected to the file `meta_pred.msg` in `combinate::start_metapred()`.

8 Integration of other prediction tools

8.1 Invocation of your tool

The invocation of your tool has to be added to the skript `combinate`. You should add a flag to the hash `%arg`, which corresponds to the commandline parameter of `combinate` you chose for your tool. Additionally a variable to hold the starter object of your tool (like `$glimmer_rbs`) should be declared in the script. The parameters needed for your tool (e. g. `home directory` etc.) should be added to the `ini`-file. In the routine `eval_args` the commandline parameters of `combinate` will be evaluated. For the invocation itself you should write an object extending `Nudge.pm` comparable to `GlimmerNudge.pm` and you should add the incarnation of this object with the necessary arguments to `eval_args` like:

```
$glimmer_rbs=GlimmerNudge->new($arg{FASTA_FILE}, $output_dir, $arg{PREFIX})}
```

You also should add the call of respective subroutines to `combinate` like:

```
$glimmer_rbs->start_glimmer();
```

8.2 Handling the output of your tool

For parsing the output of your tool, a parser according to `GlimmerParser.pm` should be implemented and integrated to `meta_pred.pl`. For the invocation of `meta_pred.pl`, the commandline parameter invoking your tool should be set in a variable in `meta_pred.pl` like:

```
my $F_GLIMMER_RBS = 'gr';
```

The invocation of `meta_pred.pl` is triggered by `combinate`. For that you should add the parameter for your tool to the hash `%orf_check_commm`, which is used in the routine `start_metapred()` (or `script_metapred()` (if `meta_pred.pl` is started as batch process) to create the commandline call of `meta_pred.pl` like:

```
$orf_check_comm{GLIMMER_RBS}=" ".$arg{GLIMMER_RBS}." ".$glimmer_rbs->get_resultoutput();
```

In `start_metapred()` you should add the new value of `%orf_check_comm` to the call in `start_metapred()`. You also should add a new output mode to integrate the predictions of your tool (see 7.2, p. 7).

9 Links

Critica1.05 http://geta.life.uiuc.edu/~gary/programs/CRITICA/

wublast http://blast.wustl.edu/

NCBI blast http://www.ncbi.nih.gov/BLAST/

Glimmer2.01 http://www.tigr.org/software/glimmer/

Glimmer2.10 http://www.tigr.org/software/glimmer/

RBSfinder http://www.tigr.org/software/

Orpheus http://pedant.gsf.de/orpheus/

dps ftp://cs.mtu.edu/pub/huang/

ZCurve1.0 http://tubic.tju.edu.cn/Zcurve_B/

Perl5.6.1 http://www.perl.com/

BioPerl1.3 http://bioperl.org/

PBS http://www.openpbs.org/

References

[1] Tech M., Merkl R. 2003. Yacop - Enhanced Gene Prediction Obtained by a Combination of Existing Methodes. *In Silico Biology* in press.

[2] Salzberg S., Delcher A., Kasif S., White O. 1998. Microbial gene identification using interpolated Markov models. *Nucleic Acids Res.* **26**:544-548.

[3] Delcher A., Harmon D., Kasif S., White O., Salzberg S. 1999. Improved microbial gene identification with GLIMMER. *Nucleic Acids Res.* **27**:4636-4641.

[4] Suzek B.E., Ermolaeva M.D., Schreiber M., Salzberg S. 2001. A probabilistic method for identifying startcodons in bacterial genomes. *Bioinformatics* **17**:1123-1130.

[5] Badger J., Olsen G. 1999. CRITICA: Coding Region Identification Tool Invoking Comparative Analysis. *Mol. Biol. Evol.* **16(4)**:512-524.

[6] Frishman D., Mironov A., Mewes H.-W., Gelfand M. 1998. Combining diverse evidence for gene recognition in completely sequenced bacterial genomes. *Nucleic Acids Res.* **26**:2941-2947.

[7] Huang X. 1996. *Micorb. Compar. Genomics* **1**: 281-291.

[8] Guo F.-B., Hou H.-Y., Zhang C.-T. 2000. ZCURVE: a new system for recognizing protein-coding genes in bacterial and archaeal genomes. *Nucleic Acides Res.* **31**: 1780-1789.

[9] A. C. McHardy, A. Groesmann, A. Pühler, F. Meyer 2004. Development of joint application strategies for two microbial gene finders. *Bioinformatics* **20**:1622-1631.