EVOLUTION MODEL OF REGULATORY SIGNAL WITH SECONDARY STRUCTURE

1. General information

Conceptual description of the model is provided in detail in the paper [1] and briefly below in Section 6. The model (v.1.0.7) has been implemented as a command line utility for 32-bit Windows environment. Format of the command line is as follows:

```
anneal [options] infile treefile [outfile [webfile]]
```

The command line syntax is case insensitive. Arguments are separated by at least one space; if an argument (such as file name) contains spaces, it must be enclosed in double quotes. If present, each option must start from dash (`-`) or slash (`/`) character, and be separated from other options by at least one space. Most options may be specified in any position of the command line and in any order. However, the order of the file names in the command line is fixed.

The list of available options, their syntax, meaning and default values are provided in Section 3. Since all options have default values, the model can run without any option specified in the command line. The minimum required arguments are the names of two input data files, `infile` and `treefile`.

The argument `infile` specifies a name of the text file containing a set of nucleotide sequences in FASTA format (some extension of the format is allowed, see Section 4.1 for details). The name is arbitrary, it also may include a path. There are several examples of such file in the distribution archive: pr1.txt, pr2.txt, ex1.txt, ex2.txt, ex3.txt.

The argument `treefile` specifies a name of the text file containing a phylogenetic tree in NEXUS format (which is similar to Newick parentheses format, but allows for the edge length to be specified after a node name, using a colon as separator). Some restrictions applied, see Section 4.2 for details. Examples of the tree file in the distribution archive are pr1.tre, pr2.tre, pr3.tre. (Those trees can be displayed using e.g. TreeView program by Roderic Page, v.1.6.6).

The two input files must be consistent: the same names of sequences and of the tree nodes shall be used in both files; otherwise, the program terminates with an error message.

The optional argument `outfile` specifies a name of the text file where the program results and/or working log will be written (see Section 5 for details). If this argument omitted, the program will use the `infile` name with additional extension `.out` instead.

The optional argument `webfile` specifies a name of the HTML file where the current model configuration is presented in obvious form. If the argument omitted, the program will use the `outfile` name instead, with substituted (or appended) extension `.htm`. More details are given in Section 5.

Once being started through a command line, the program works until a termination criterion is satisfied, or endless. The user can monitor its work in the command window log (similar to the `outfile` contents, but in short form), or by viewing output files in an external application (text editor, web browser, etc.) And the user can stop the program at any time by pressing keys Ctrl+C or Ctrl-Break, or just closing the command processor window. Other run-time options are described in Section 6.6.

User should take into account that the program loads a computer very much; each instance of the program completely occupies one processor core. We strongly recommend to run not more than one instance of the program per processor.

2. Installation of the program

The computer shall satisfy with certain hardware and software requirements:

- 32-bit or 64-bit CPU, compatible with Intel x86 architecture
- Operating clock 1 GHz (the more, the better)
- RAM size 256 MB
- Free disk size 20 MB
- Operating system Windows NT/2000/XP/2003/Vista

Reliable hardware, adequate cooling and uninterruptible power supply are essential for obtaining good results.

The program distribution does not include an installer. To install the program, do the following.

1. Download the current version of anneal*xxx*.zip archive from our Web site at http://lab6.iitp.ru/anneal
2. Extract the archive contents into a directory on the local hard disk, e.g. D:\Anneal.
3. Create a shortcut on Windows desktop for command line processor:
    a. Right click at free area of the desktop and select Create > Shortcut
    b. Create Shortcut window appears; type cmd in the field provided, and click Next.
    c. In the next window, type desired name for the shortcut and click Finish.
    d. Right click the created shortcut and select Properties.
    e. On the Shortcut tab, type in the Working Directory field a name of the directory where the archive was extracted (D:\Anneal), then click OK.
4. Double click the created shortcut. The command line processor window opens.
5. Type anneal in the command line and press Enter.
6. The program will output its help on arguments. Scroll up the window to check program version number in the first line. Installation complete successfully.
7. If you get a message that "anneal" is not an executable file, then incorrect name of directory was specified at the step 3e.
8. If you get a message that the system cannot execute the program, your system does not have necessary Microsoft Visual Studio run-time library version installed. Run the file vcredist_x86.exe included in the distribution archive, and follow the instructions.

To uninstall the program, just delete the created directory and shortcut.

3. Command line options

Given below is a list of available command line options of the program in alphabetical order. Some options are not described because they are not intended for the end user. Default value for each option is shown in brackets. Though shown as capital letters, all options and their values are case insensitive.

`-A` [none] If the command line includes this option, the output file (with default name or explicitly specified by `outfile` argument) will be opened in append mode, i.e., new data will be written after the data already existing in the file. Otherwise, the file will be re-written (previous data are lost).

`-AA`*xxx*[,*xxx*...] [none] This option is used in leader peptide (Lp) mode, where all regulatory domain sequences include a leader peptide gene. The option specifies amino acid which is the regulatory one in the case study. Use common abbreviations instead of *xxx* characters: `ALA ARG ASN ASP CYS GLN GLU GLY HIS ILE LEU LYS MET PHE PRO SER TRP THR TYR VAL`. If multiple amino acids are regulatory (such as in case of *ilv* operons), list them using comma as separator. See more details about the Lp mode in Section 6.2.

`-AE`*n* [-AE1] Bonus a_e for equal letters in two sequences alignment. See Section 6.1, Eq. (1).

`-AT`*n* [-AT-0.8] Penalty a_t for different letters in two sequences alignment, in case of transition (purine to purine or pyrimidine to pyrimidine). See Section 6.1, Eq. (1).

`-AV`*n* [-AV-1.2] Penalty a_v for different letters in two sequences alignment, in case of transversion (purine to pyrimidine or vice versa). See Section 6.1, Eq. (1).

`-AD`*n* [-AD-2] Penalty a_d for single gap (or first gap in a series) in two sequences alignment. See Section 6.1, Eq. (1).

`-AG`*n* [-AG-1] Penalty a_g for second and subsequent gaps of the gap series in two sequences alignment. See Section 6.1, Eq. (1).

`-C`*n* [-C0.01] The value of coefficient C which controls a growth of cooling parameter β in simulated annealing procedure. See Section 6.6, Eq. (16).

`-CH`*n* [-CH0.2] Additional bonus a_c for equal letters in pair-wise alignment that occur at the same position of both sequences. See Section 6.2, Eq. (8).

`-D`*n* [-D100] Transition-to-indel ratio D. See Section 6.5.

`-E`*n* [-E10] The value of parameter κ which determines a significance of indel events as compared to substitutions. See Section 6.2, Eqs. (3, 4).

`-F`*n* [-F1000] This parameter sets a frequency of the configuration summary output, both on the console and into output file. By default, the program prints this summary information every 1000 iterations. Use -F1 to print summary after each pass over entire tree (however, it can significantly decrease the performance).

`-GA`*n* [-GA0.25] Relative frequency g_A of the nucleotide A. See Section 6.3.

`-GC`*n* [-GC0.25] Relative frequency g_C of the nucleotide C. See Section 6.3.

`-GG`*n* [-GG0.25] Relative frequency g_G of the nucleotide G. See Section 6.3.

`-GT`*n* [-GT0.25] Relative frequency g_T of the nucleotide T. See Section 6.3.

`-GR`*n* [-GR77] This option allows user to control the configuration data output in HTML form. If zero value specified, no graphic data will output. Other values are interpreted as a sum of the following options:

- 1: Print initial configuration into the file *webfile*0000.htm, where *webfile* is the name specified by argument `webfile` in the command line (or by default).
- 2: Print each configuration with better value of the functional H into the file *webfile*NNNN.htm, where NNNN is serial number of the minimum found. (These files will be linked to each other by means of Initial/Previous/Next/Best links.)
- 4: Print only the best configuration found to the moment, into the file *webfile*.htm.
- 8: Include sequences for leaves into the configuration to print (otherwise, only sequences for internal nodes are printed).
- 16: Use short form to display helices (both anti-terminator and terminator in one line); otherwise each helix occupies separate line. (Not implemented yet).
- 32: Print only best paths from each leaf instead of entire configuration.
- 64: Print multiple alignment along a path; otherwise, each path includes unaligned sequences.

`-H` Display help on program arguments.

`-I`*n* [-I0] Limit number of iterations to perform. Zero value means no limit.

`-J`*n* [-J0] Value of exponent g in H_3 dependency of the edge length. See Section 6.2, Eq. (9).

`-L`*n* [-L0.25] Value of coefficient λ to agree H_1 and H_3 terms of the functional *H*. See Section 6.2, Eq. (2).

`-M`*n* [-M8] This option allows user to control the configuration details output in text form into the file specified by command line argument `outfile`. See also `-O` option. If zero value specified, no configuration data will output. Other values are interpreted as a sum of the following options:

1: Print current state of each sequence.
2: Print initial state of each sequence.
4: Print alignment of each sequence with its initial state.
8: Print alignment of each sequence with a sequence in the parent node.
16: Print matrix **A** for the edge which leads to each node.

`-MINH`*n* [-MINH9] Minimum number of pairs in a helix, $h_{\min}$. See Section 6.4.

`-MINL`*n* [-MINL3] Minimum length $l_{\min}$ of the helix loop. See Section 6.4.

`-MAXL`*n* [-MAXL40] Maximum length $l_{\max}$ of the helix loop. Zero value means no limit. See Section 6.4.

`-MAXB`*n* [-MAXB2] Maximum length $b_{\max}$ of a bulge in the helix shoulder. See Section 6.4.

`-MAXI`*n* [-MAXI2] Maximum length $i_{\max}$ of an internal loop in either shoulder of the helix. See Section 6.4.

`-MINP`*n* [-MINP2] Minimum length $p_{\min}$ of contiguous segment of the helix in case of bulges/internal loops. See Section 6.4.

`-MR`*n* [-MR12] In Lp mode, maximum number *m* of regulatory codons to encourage. See Section 6.2, Eq. (10).

`-MU`*n* [-MU5] In Lp mode, the value of coefficient μ in H_4 term of the functional *H*. See Section 6.2, Eq. (10).

`-N`*n* [-N1000] The number of terms in Taylor series for exp(**R***t*). See Section 6.3, Eq. (13).

`-O`*n* [-O98] This option allows user to control the configuration output in text form into the file specified by command line argument `outfile`. See also `-M` option. If zero value specified, no configuration data will output. Other values are interpreted as a sum of the following options:

1: Print summary for each configuration (slow).
2: Print configuration summary with a frequency specified by `-F` option.
4: Print each configuration (very slow).
8: Print configurations with a frequency specified by `-F` option.
16: Print each configuration with better value of the functional *H*.
32: Print only the best configuration found (after a termination criteria is satisfied or the program is terminated manually).
64: Print best path from each leaf along with a configuration.

`-P`*n* [-P1.5] The value of exponent *p* which controls a growth of cooling parameter β in simulated annealing procedure. If negative value is specified, the program uses another form of the dependency. See Section 6.6, Eqs. (16, 17).

`-Q`*n* [-Q1] The value of exponent *q* in H_1 term of the functional *H*. See Section 6.2, Eqs. (3, 4).

`-R`*n* [-R5] Transition-to-transversion ratio *R*. See Section 6.3.

`-S`*n* [-S18] Tree nodes polling order during each iteration, i.e. the sequence of elementary steps (see Sections 6.5, 6.6). Specified value is interpreted as a sum of the following options:

1: Tree leaves are processed first.
2: Tree nodes are processed in a descending order with regard to length of the path from a node to the root.
4: Tree nodes are processed by levels, from farthermost to closest to the root.
8: Use reverse order for options 1, 2, 4.
16: Do not process leaves, i.e., extant sequences.
32: Take node name alphabetical order into account for options 1, 2, 4.

`-T`*n* [-T600] Limit of computation time in seconds. Zero value means no limit.

`-U` [none] When this option is specified, the output log printed in text file `outfile`, will contain detailed information on each evolution event (slows the program down very much).

`-V`*n* [-V0.01] Rate of transversions. See Section 6.3 for details.

`-W`*n* [-W14] Program working mode, which is determined as a sum of the following options:

1: Use the "standard" matrix **R** which includes only -3/4 and +1/4 values; otherwise, the matrix Eq. (11) is used. See section 6.3.
2: Use only fixed secondary structure for extant sequences if present in input file. Otherwise, the secondary structure specified will be preferred, but not excluding other structure to be selected. See Sections 4.1, 6.4 for details.
4: Generate random initial sequence for internal node if not given in input file. Length of the sequence equals average sequence length over the leaves.
8: When selecting a set of secondary structures, remain only specified number of helices from each cluster. See Section 6.4 for details.
16: Use functional H_3 in the form (7), i.e., computed along paths. Otherwise, H_3 in the form (5-6) is applied.

`-X`*n* [-X0] Threshold X value for the functional H_3. See Section 6.2, Eqs. (6,7).

`-YMHC`*n* [-YMHC1] Number y_r of representative helices selected from a cluster. See Section 6.4.

`-YOVL`*n* [-YOVL5] Minimum overlap $y_{\min}$ of anti-terminator and terminator from a pair. See Section 6.4.

`-YTLP`*n* [-YTLP20] Maximum permitted loop length $y_{\max}$ for a terminator. See Section 6.4.

`-YTOL`*n* [-YTOL30] Maximum percentage G_t of energy difference to consider two helices equivalent for joining in a cluster. See Section 6.4 for details.

`-YTUG`*n* [-YTUG6] Maximum gap $g_{\max}$ between a terminator and U-run. See Section 6.4.

`-YUGP`*n* [-YUGP3] Maximum number u_{gap} of non-U letters between U's within U-run. See Section 6.4.

`-YURA`*n* [-YURA3] Minimum number $u_{\min}$ of U letters within U-run. See Section 6.4.

`-YUTL`*n* [-YUTL20] Maximum distance u_{end} from the end of U-run to the end of sequence. See Section 6.4.

`-Z`*n* [none] Seed value for random number generator. This option can be useful for reproducing interesting results, fixing bugs, etc. If not specified, the program behavior is randomized.

4. Input data

4.1 Sequence file

This text file contains a set of input sequences in FASTA format. The file must include at least sequences for all leaves of the tree given. If sequences are present for internal nodes, the program uses them for initial configuration. Otherwise, empty or random sequences for internal nodes are used depending on `-Wn` option. Sequences in the file may occur in any order. Empty lines are ignored.

Each sequence must be represented by exactly two lines of the file. First line is started from character '>' followed by the sequence name, which cannot be longer than 64 characters (extra characters will be ignored). The name cannot include blanks, tabs or percent characters, other characters are allowed. Note that sequence names are case-sensitive. Generally, we recommend to use short names.

Second line contains the entire sequence in case-insensitive alphabet <A,C,G,T> (letter U is automatically converted into T). The sequence may contain characters '-', '_' or '=' for gaps; they are ignored. All other characters (including space) within a sequence produce an error. Maximum length of the sequence is 1023 nucleotides. Examples of proper sequence files in the distribution archive are pr1.txt, pr2.txt.

Additional option (not provided for in FASTA format) allows to specify the secondary structure for a sequence if known. It can be done in the line with a sequence name, after a percent character. The general format of the secondary structure data is as follows:

`%` [anti-terminator data] `0` [terminator data]

Both anti-terminator and terminator data are optional and use the same format:

A B C D [E F [G H]]

where positive integers A, B, C, D are start and end positions of helix shoulders in the sequence, in 5' to 3' order. Optional negative integers E, F indicate start and end positions of a bulge in either shoulder of the helix (E=F if it is one nucleotide bulge). In case of internal loop, i.e., two-side bulge, negative integers G, H are also used. In all cases, the numbers are separated from each other by at least one space, tab or comma character. Examples of sequence files using this extension, are ex1.txt, ex2.txt, ex3.txt in the distribution archive.

4.2 Tree file

This file specifies a phylogenetic tree for the sequences in the first input file. The tree must be a binary rooted one. Parentheses format NEXUS is used to represent a tree; total length of the tree in text representation must not exceed 4096 characters. The tree may be split into multiple lines in any positions; spaces, tabs and line breaks are ignored. Names of internal nodes are not required; the program assign them automatically if omitted.

Length of the tree edge are specified after a name of the daughter node, using a colon as separator. All lengths must be integer numbers! If a length is not given, the program assumes it equal 1.

Both sequence file and tree file must meet two requirements:
- name of each leaf in the tree file coincides with a sequence name in the sequence file, and
- each sequence name in the sequence file coincides with a node name in the tree file.

Examples of proper tree files in the distribution archive are pr1.tre, pr2.tre, pr3.tre.

5. Output results

Output results of the program are written in two files: text file and HTML file. Names of these files are specified, respectively, by arguments `outfile` and `webfile` in the command line, or assigned by default. The text file can be viewed with a text editor e.g. Notepad (in which case we recommend to switch Word Wrap off), and the HTML file is intended for Web browser.
Note: For HTML file to display correctly, there must be a style file config.css in the working directory. The distribution archive includes this file as well as its version config_.css for grey-scale display and print.

The contents of both output files is greatly dependent on run-time options used, in particular, `-A`, `-F`, `-GR`, `-M`, `-O` options (see Section 3). The primary purpose of the text file is to represent a dynamics of the functional minimization process, and of the HTML file – to reveal details of the current configuration in obvious form. However, one can make the configuration to be printed in the text file, as well as monitor the optimization dynamics by looking through the entire chain of HTML files in browser, of course.

Several examples of the output files are included in the distribution archive. Below we describe the contents of two files, ex3.log and ex3.htm, which we obtained in Example 3 (see Section 7).

5.1 Text file description (by example ex3.log)

The below description refers to line numbers of the file (the numbers are displayed in status bar if the file is viewed in Notepad program with word wrapping switched off).

Line 2: The command line used to run the program.
5-7: The program parameter values (see definitions of options in Section 3).
8: Computed probabilities P_s, P_i, P_d, p, q of evolution events (see Section 6.5).
9: Bonus/penalty values for alignment (see Section 6.1).
10: Matrix **R** (see Section 6.3), all elements are multiplied by 1000.
11-23: Warning about randomly generated initial state of ancestral sequences.
24: Matrix $e^{\mathbf{R}}$ (see Section 6.3), all elements are multiplied by 1000.
26-35: These messages can appear if known secondary structure was given for leaves, but some helix of the structure does not meet the specified conditions for helix build automation.
36: Value of the functional H for initial configuration.
37-38: Alignment of the sequences for node N11 and its parent node. First line of such pair contains (left to right):
– name of the node (N11);
– aligned sequence enclosed in single quotes;
– length of the alignment (L=134);
– name of the parent node (N10);
– H term value for this edge (Hpar=7.275e+002);
– H_1 and H_3 term values for this edge (7.275e+002, 0.00);
– length of the edge (Edge=1);
– number of helices selected for secondary structure (hel=3);
– those helices themselves, e.g. (62-72,85-96)L10e64 means that helix shoulders are from position 62 to 72 and from 85 to 96, helix includes 10 nucleotide pairs, and its energy equals -6.4;
– leader peptide data, e.g. Lpept={s:28 r:55,58 t:61} means that start codon is at position 28, regulatory codons are at positions 55 and 58, and stop codon is at position 61 (all positions are always in the same reading frame);
– matrix $\mathbf{A}=\ln(\exp\{\mathbf{R}t\})$ for this edge (see Section 6.3).
Second line of this pair contains only aligned sequence for the parent node.

39-89: Similar line pairs for other edges of the tree. The only exception is root node (line 61) which has no parent, and, therefore, no second line.

90: Values of the functional terms for this configuration (P3 stands for term H_3 in the form (7), see Section 6.2).

92-93: Here the information about paths is present; since the initial configuration here does not contain any path, this section of the configuration will be described below for the final configuration.

96-2423: These lines represent the optimization dynamics. Each line contains (left to right):
– Elapsed time in seconds;
– Iteration number;
– Value of cooling parameter β (see Section 6.6);
– Minimum and maximum values of the functional H, encountered since previous timestamp line (in this case – during last 1000 iterations);
– Minimum and maximum change of the functional (the difference in (15)) at elementary steps performed during this interval;
– Absolute minimum of the functional reached to the moment;
– Component values of that absolute minimum (P3 stands for H_3 as per (7)).

2424: A reason of the program termination (here, the user has it stopped).

2426-2480: Final optimal configuration in the form similar to the initial configuration.

2483-2496: Set of optimal paths from leaves to root, over similar secondary structures in internal nodes. In this example, such path exists for each leaf. A path occupies one line in the file; such line contains a list of nodes along the path, each node is accompanied with a secondary structure in the corresponding sequence. The secondary structure includes two helices – anti-terminator and terminator, and each helix is represented like in the configuration described above, thus e.g. N08:(58-67,93-102)L10;(86-101,108-123)L16. In the end of this line, a sum of H_3 values over edges of the path is given, e.g. H3=-263.7.

This example contains only initial and final configurations. Depending on options specified, the text file can contain also each configuration which is better than all previous ones, or even include all configurations in turn. However, the format of representation is the same.

5.2 HTML file description (by example ex3.htm)

By default, the program creates two html files: one for initial configuration (with a name *webfile*0000.htm, where *webfile* is the name specified in `webfile` argument of the command line or assigned by default), and one for current configuration (with a name *webfile*.htm) which is modified during work. However, the user has an option to store also intermediate configurations, in which case multiple html files will be connected in a chain with links. Presentation format of the configuration is the same in all cases.

Unlike the text file, HTML file is difficult to refer, so we describe it in general terms and/or by reproducing certain data. In the beginning of the file, values of the program parameters are displayed similar to the text file in Section 5.1. Then a line appears in the form like

#3454: Htotal=1118.8 Iter=2282984 Time=317417 ...

It means that the file represents a configuration in 3454-th local minimum, where the functional H equals 1118.8. This minimum was found in 2282984-th iteration, and 317417 sec elapsed since the program start.

After that, the sequences for each node are shown along with secondary structure and leader peptide, if any. A line starting with a name of the node (e.g. N02) contains: name of the parent node (N01), value of the functional H and its terms H_1,H_3 for the edge from parent to this node (Hpar=72.3(166.86,-378.35)), length of the sequence (L=133), number of helices in the secondary structure (hel=3), start position of U-run and its length (Ura=125(8)), number of

terminators in the secondary structure (T=1), number of anti-terminators (AT=2), and number of anti-terminator&terminator pairs (Pair=2).

Each sequence is repeated several times depending on the secondary structure. Each terminator and each anti-terminator occupies separate line; anti-terminators are highlighted in green, and terminators in magenta (positions corresponding to loops and bulges are not highlighted). If no proper secondary structure was found, but helices exist, they are highlighted in turquoise. If the sequence contains proper leader peptide gene, it shares a line with first terminator. The highlight colors, used for leader peptide, are: turquoise for start codons, blue for regulatory codons, and red for stop codons. If regulatory codons of multiple amino acids occur, they are shown with different color of letters (in this case, white for threonine and yellow for isoleucine).

To the right of the sequence, positions and length of the helix is shown (e.g. (58-70,98-110)L12) and its energy (G=-14.3). Negative zero value of energy means that the helix was not built by the program, but given as part of secondary structure in source data.

After the last node, the line appears in the form like

Overall H=1118.764 H1=3487.906 H3=-3516.570 λ=0.25 H4=-298.000 μ=5

where values of the functional and its terms are shown as well as parameters λ, μ.

The next part of the HTML file contains the best paths over similar secondary structures from each leaf to the root. Each continuous block corresponds to a path, and each line of the block represents the sequence in a node along the path, sequentially from the root to the leaf. In this case, the sequences are shown aligned over the whole path (that is optional). Highlight colors have the same meaning like in previous part, but the area where anti-terminator and terminator overlap, is highlighted in brown.

Shown to the right of each sequence are: its name, secondary structure and H_3 value for corresponding edge (for the leaf, a total H_3 over whole path is shown).
The last line contains a total H_3 value for all paths.

6. Explanatory notes

This section contains selected equations and formulas implemented in the program, to help in better understanding of its options. Further information user can find in the paper [1].

6.1 Pair-wise alignment of the sequences

The pair-wise alignment of two sequences s, s' assigned to ends of the tree edge in a current configuration, is found by dynamic programming procedure, using a similarity function ("score") in the form:

$$\varphi(s,s') = N_e \cdot a_e + N_t \cdot a_t + N_v \cdot a_v + \sum_k \left(a_d + a_g \cdot (l_k - 1) \right) \to \max, \qquad (1)$$

where N_e is a number of positions with equal letters in both aligned sequences; N_t is a number of positions where a transition takes place, i.e. purine is substituted with another purine or pyrimidine with another pyrimidine; N_v is a number of positions where a transversion takes place, i.e. purine is substituted with pyrimidine or vice versa; sum over k is taken for all contiguous nonexpansible domains with length $l_k \geq 1$ within the alignment, such that at each position of the domain either sequence contains a gap; a_e, a_t, a_v, a_d, and a_g are the program parameters. By default, $a_e = 1$, $a_t = -0.8$, $a_v = -1.2$, $a_d = -2$, $a_g = -1$.

6.2 Functional to minimize

The program aims to minimizing an energy-like functional of Gibbs type. For each configuration σ, which is a mapping of set of all nodes of given phylogenetic tree G into the set of all

sequences over 4-letters alphabet (i.e., the configuration assigns a sequence to each node), this functional in general form can be written as consisting of four terms:

$$H(\sigma) = H(\sigma,\theta) = H_1(\sigma) + H_2(\sigma,\theta) + \lambda H_3(\sigma) + H_4(\sigma) \to \min, \quad (2)$$

where θ is a given data – the set of sequences in leaves, one sequence at each leave; $H_1(\sigma)$ reflects an energy of pair interaction along each edge of G in σ ; $H_2(\sigma,\theta)$ reflects an influence of present-day data of the tree leaves; $H_3(\sigma)$ reflects a requirement of the secondary structure conservatism at the ends of each edge and over entire paths in the tree; $H_4(\sigma)$ reflects a requirement of a leader peptide gene presence in each sequence.

Specifically, it is assumed in the current version of the program, that sequences for the leaves are not change, i.e. $\sigma \equiv \theta$ in leaves, therefore, $H_2(\sigma,\theta)$=const, so we shall not consider that term. Other terms are described below.

$$H_1(\sigma) = -\sum_j \left(\ln \prod_{i=1}^{n_j}{}' \left(e^{\gamma_i t_j \mathbf{R}} \right) \left(\overline{\sigma}_{ji}, \overline{\sigma}'_{ji} \right) - \kappa \cdot \sum_m \left(l_{j,m} + 1 \right)^q \right), \quad (3)$$

where the outer sum is taken over all edges j of the tree G; $\overline{\sigma}_j$ and $\overline{\sigma}'_j$ are, respectively, the aligned sequences in the beginning (closer to the root) and end of j-th edge; n_j is the length of that alignment. The product Π' is taken over those positions of the alignment, where both sequences contains nucleotide letters, $\mathbf{R}$ is a fixed matrix of nucleotide substitution rates, t_j is a length of j-th edge. In the current program version, we assume that evolution rate at each position of the sequence is a constant, $\gamma_i = 1$.

The inner sum in (3) is taken over all domains m in the pair alignment that contain gaps in either sequence; $l_{j,m}$ is the length of m-th domain at j-th edge, and parameters κ, q set a significance of indel events as compared to substitutions (by default, $\kappa = 10$, $q = 1$).

To speed up modeling, the matrix-valued exponent in (3) is computed and logarithmed one time for each edge of the tree G. Thus, we use (3) in a more simple form:

$$H_1(\sigma) = -\sum_j \left(\sum_{i=1}^{n_j}{}' \mathbf{A}_j \left(\overline{\sigma}_{ji}, \overline{\sigma}'_{ji} \right) - \kappa \cdot \sum_m \left(l_{j,m} + 1 \right)^q \right), \quad (4)$$

where matrix $\mathbf{A}_j(\cdot,\cdot)$ contains the pre-computed values for j-th edge. The matrix-valued exponent is computed by summation of a series, so we assume t_j being integer numbers.

The third term in (2) can be defined as

$$H_3(\sigma) = H_3(\sigma,h) = -\sum_{j \in V} \Phi(h_j, h'_j), \quad (5)$$

where $h = < h_j, h'_j >$, and $h_j = \{h_{jm}\}$, $h'_j = \{h'_{jk}\}$ are two sets of helices with sufficiently low energy that are found in two sequences σ_j and σ'_j, assigned to the ends of j-th edge. The non-local interaction potential Φ reflects a conservatism of the secondary structure along the tree edges; it depends on a type of the secondary structure of interest.

The current version of the model deals with classical attenuation regulation, so RNA secondary structure contains at least a pair of mutually exclusive helices: terminator $T = (t_{m1}, t_{m2})$ with shoulders t_{m1}, t_{m2} and anti-terminator $A = (a_{m1}, a_{m2})$ with shoulders a_{m1}, a_{m2}, whereas m runs over a set of all such pairs. Thus, the potential Φ is defined as

$$\Phi(h_j, h'_j) = \frac{1}{n_{mk}} \sum_{m,k} \left[\varphi(t_{m1}, t'_{k1}) + \varphi(t_{m2}, t'_{k2}) + \varphi(a_{m1}, a'_{k1}) + \varphi(a_{m2}, a'_{k2}) \right]^{X_+}, \quad (6)$$

where $[u]^{X+} = u$ for $u > X$, otherwise, $[u]^{X+} = 0$; X is a fixed threshold; function φ was defined by (1), and n_{mk} is the number of nonzero items of the sum over m,k. In other words, to compute (6), all possible anti-terminator&terminator pairs are selected from the helix sets h_j, h'_j; then each pair is scored by independent alignment of corresponding anti-terminator and terminator shoulders. For those pairs which have a similarity score above the threshold X, the average score is calculated. By default, the threshold value X=0 is applied.

In addition, another form of H_3 term of the functional is implemented in the program. It is even more non-local, and is calculated along entire paths in the tree, rather than independent edges. We define it in the form

$$H_3(\sigma) = -\sum_{k \in V_1} \max_{p_k} \sum_{m \in p_k} \left[\varphi(t_{m1}, t_{m'1}) + \varphi(t_{m2}, t_{m'2}) + \varphi(a_{m1}, a_{m'1}) + \varphi(a_{m2}, a_{m'2}) \right]^{X_+}, \qquad (7)$$

where the outer sum is taken over all leaves of the tree; p_k is a path from the k-th leaf towards the root. That path is composed of fixed anti-terminator&terminator pairs selected from sequences, which assigned to each node m along the path (note that only one pair is chosen from each sequence).

Despite corresponding shoulders of all helices in (6, 7) are aligned independently of their arrangement within a sequence, we provide for an option to encourage the conservatism of helix location. For that purpose, when using (1) to calculate (6) or (7), we modify a_e:

$$a'_e = a_e + a_c \qquad (8)$$

for each alignment position, where equal letters occur at the same position in both sequences.

One more option, implemented in the program, is taking the edge length t into account, when computing the potential Φ. Such modification is available in both variants (5-6) and (7), we use here a simple dependency

$$U(\Phi) = t^g \cdot \Phi, \qquad (9)$$

where g is a parameter of the model (by default, $g = 0$).

Last term $H_4(\sigma)$ in (2) is considered only in Lp mode, when sequences are assumed to contain a leader peptide gene. The following conditions applied:

– there is a stop codon in the sequence between its start and the rightmost center of anti-terminator loops;
– there is a start codon to the left of the stop codon, with regard to the reading frame;
– there are regulatory codons of the specified amino acid(s) between start and stop codons, also with regard to the reading frame.

If some of the above conditions are not met, we consider $H_4(\sigma) = +\infty$, otherwise

$$H_4(\sigma) = \begin{cases} -\mu \cdot r & \text{for } r \le m \\ -\mu \cdot m & \text{for } r > m \end{cases}, \qquad (10)$$

where r is the number of regulatory codons, μ and m are the model parameters (by default, μ = 5 and m = 12).

6.3 Selection of the matrix R

In order to compute the matrices $\mathbf{A}_j(\cdot,\cdot)$ in (4) for all edges of the tree G, we need the matrix **R** of nucleotide substitution rates that is selected in accordance with a model of substitution. Specifically, the program uses the matrix **R** in the form [2]:

$$\mathbf{R} = \begin{array}{c|cccc} & \text{A} & \text{C} & \text{G} & \text{T} \\ \hline \text{A} & * & \nu g_{\text{C}} & \left(\frac{\delta}{g_{\text{R}}}+\nu\right) g_{\text{G}} & \nu g_{\text{T}} \\ \text{C} & \nu g_{\text{A}} & * & \nu g_{\text{G}} & \left(\frac{\delta}{g_{\text{Y}}}+\nu\right) g_{\text{T}} \\ \text{G} & \left(\frac{\delta}{g_{\text{R}}}+\nu\right) g_{\text{A}} & \nu g_{\text{C}} & * & \nu g_{\text{T}} \\ \text{T} & \nu g_{\text{A}} & \left(\frac{\delta}{g_{\text{Y}}}+\nu\right) g_{\text{C}} & \nu g_{\text{G}} & * \end{array}, \tag{11}$$

where ν is a rate of transversions, and g_{A}, g_{C}, g_{G}, g_{T} are relative frequencies of nucleotides ($g_{\text{A}} + g_{\text{C}} + g_{\text{G}} + g_{\text{T}} = 1$), $g_{\text{R}} = g_{\text{A}} + g_{\text{G}}$, $g_{\text{Y}} = g_{\text{C}} + g_{\text{T}}$. (By default, $\nu = 0.01$ and $g_{\text{A}} = g_{\text{C}} = g_{\text{G}} = g_{\text{T}} = 0.25$). Value of δ is computed depending on so called transition-to-transversion ratio R which is the program parameter (by default, $R = 5$), as follows:

$$\delta = \nu \cdot \frac{R\left(g_{\text{A}} + g_{\text{G}}\right)\left(g_{\text{C}} + g_{\text{T}}\right) - \left(g_{\text{A}} g_{\text{G}} + g_{\text{C}} g_{\text{T}}\right)}{\dfrac{g_{\text{A}} g_{\text{G}}}{g_{\text{A}} + g_{\text{G}}} + \dfrac{g_{\text{C}} g_{\text{T}}}{g_{\text{C}} + g_{\text{T}}}}. \tag{12}$$

Diagonal elements (*) of the matrix **R** are negative numbers, zero's complementing a sum along each row. For default parameter values, the matrix is

$$\mathbf{R} = \begin{pmatrix} -0.03 & 0.0025 & 0.025 & 0.0025 \\ 0.0025 & -0.03 & 0.0025 & 0.025 \\ 0.025 & 0.0025 & -0.03 & 0.0025 \\ 0.0025 & 0.025 & 0.0025 & -0.03 \end{pmatrix}$$

Another option is also provided, when the program works with the standard matrix **R**, often used in phylogeny; the diagonal elements of this matrix are equal to –0.75, and other elements are equal to 0.25. (One can also obtain such matrix in our general model by setting $\nu = 1$, $R = 0$).

As mentioned in Section 6.3, we compute $\exp(\mathbf{R}t)$ in (3) by summation of Taylor series; more exactly, at first we calculate

$$e^{\mathbf{R}} \approx \sum_{n=0}^{N} \frac{\mathbf{R}^n}{n!}, \tag{13}$$

where the number N of terms in the series is a parameter of the program (by default, $N = 1000$). Then we calculate $e^{\mathbf{R}t_j} = \left(e^{\mathbf{R}}\right)^{t_j}$ for each edge j of the tree, assuming that t_j is integer number.

6.4 Selection of the set of secondary structures

To compute the functional H_3 for the configuration σ, we have to select the set $\{h_{jm}\}$ of significant helices for the sequence assigned to j-th node in that configuration. This set should be as small as possible, to decrease the number of enumerated pairs of anti-terminator and terminator helices, and, of course, should only comprise the helices which are either terminator or anti-terminator in some pair. The set is built as follows.

In general, we only consider helices built up from the nucleotide pairs G-C, A-T and G-T, provided that the following conditions are satisfied:

– the helix consists of at least $h_{\min}$ nucleotide pairs;

– the number of nucleotides in the helix loop is between l_{min} and l_{max};
– the helix has not more than one bulge or internal loop (two-side bulge), with maximum length of b_{max} or i_{max} (the latter in each shoulder), respectively;
– in case of a bulge or internal loop, either contiguous segment of the helix must consist of at least p_{min} nucleotide pairs;
– the helix is extended in both directions as much as possible;
– if multiple variants of helices are possible for a fixed loop, we always choose the minimum energy helix.

These limits are the program parameters, their default values are: $h_{min} = 9$, $l_{min} = 3$, $l_{max} = 40$, $b_{max} = 2$, $i_{max} = 2$, $p_{min} = 2$.

From the whole set of helices limited in such a way, we first choose putative terminators $\{t_{jm}\}$. Here we apply two additional requirements:
– the terminator loop shall not be greater than y_{max} ($y_{max} = 20$ by default);
– right (3') end of the terminator shall be close enough to a U-run segment of the sequence (polyuracil); specifically, a gap between the terminator end and U-run start shall not be greater than g_{max} ($g_{max} = 6$ by default). (Note that the gap length may be negative in case of overlapping terminator and U-run.)

Above notion of U-run is strictly defined by setting up three parameters:
– u_{min} is the minimum allowed number of U nucleotides in the U-run;
– u_{gap} is the maximum allowed number of non-U nucleotides between neighbor U letters within the U-run;
– u_{end} is the maximum allowed distance from the end of U-run to the end of sequence.

These parameters have default values $u_{min} = 3$, $u_{gap} = 3$, $u_{end} = 20$.

Not every terminator satisfying with the above conditions is chosen. Instead, we chose only specified number of representatives from each cluster of (approximately) equivalent terminators based on their energy and location in the sequence. Two putative terminators are considered equivalent if their corresponding shoulders are overlapped at least at a half length, and their energy difference percentage is less than G_t (by default, $G_t = 30\%$). Then we chose y_r terminators with minimum energy from each cluster built in such a way (by default, $y_r = 1$). Thus, the initial set of putative terminators is selected.

Then we select suitable anti-terminators, which are alternative for selected terminators. The following condition is implemented in the program: 3'-end of anti-terminator must overlap 5'-end of terminator with at least y_{min} nucleotides (by default, $y_{min} = 6$). The putative anti-terminators are clusterized, and cluster representatives are chosen similar to terminators.

Finally, we discard terminators that have no alternative antiterminator in the given sequence. Thus, we have built a set of anti-terminator&terminator pairs to use in (5-6) or (7).

Our experiments with the program show that in most cases the above procedure leads to the selection of proper secondary structure, known for an extant sequence. However, the program also has an option to specify these structures explicitly in input data (see Section 4 for details).

6.5 Modeling of evolution events

An iteration of modeling consists of elementary steps in accordance with fixed order of internal nodes of the tree. Each step involves a change of the current configuration only in one sequence corresponding to k-th internal node. The following changes are possible: substitution of the nucleotide in some position of the sequence, insertion or deletion of at least one nucleotide.

Specifically, a random position is selected equally probable in the sequence; then the type of event is chosen. Probabilities of substitution (P_s), insertion (P_i) and deletion (P_d) are calculated from two program parameters: transition-to-transversion ratio R and transition-to-indel ratio D:

$$P_i = P_d = \frac{R}{2D(R+1)},\ P_s = 1 - \frac{R}{D(R+1)}. \tag{14}$$

By default, $D = 100$, and these probabilities are $P_s \approx 0.992$, $P_i = P_d \approx 0.004$.

If a substitution is chosen, we do it in accordance with a substitution probability matrix:

	A	C	G	T
A	0	q	p	q
C	q	0	q	p
G	p	q	0	q
T	q	p	q	0

where $p = \frac{R}{R+1}$, $q = \frac{1}{2(R+1)}$ are, respectively, the conditional probabilities of the transition and transversion on condition that a substitution takes place. By default, p=5/6, q=1/12.

The insertion in the selected position is made as follows. A length l = 1,...,32 is chosen with the probability 2^{-l}, and inserted word will consist of l independent equally probable letters. The deletion is made in similar way, but the selected position is considered as a center of the segment to delete.

6.6 Optimization of the functional by simulated annealing

As a result of each elementary step execution, we have a new configuration $\tilde{\sigma}$ which differs from the last configuration $\sigma(n) = \sigma$ only in k-th sequence, where an evolution event occurred. This configuration $\tilde{\sigma}$ is accepted, i.e., $\sigma(n+1) = \tilde{\sigma}$, with probability

$$q(\sigma, \tilde{\sigma}) = \exp\left\{-\beta\left[H(\tilde{\sigma}) - H(\sigma)\right]^+\right\}, \tag{15}$$

where $[u]^+ = u$ for $u \geq 0$, and $[u]^+ = 0$ for $u < 0$. The new configuration is rejected, and the last one remains, i.e., $\sigma(n+1) = \sigma$, with probability $1 - q(\sigma, \tilde{\sigma})$.

Thus, a sequence of configurations appears:

$$\sigma(0) \Rightarrow \sigma(1) \Rightarrow \ldots \Rightarrow \sigma(n) \Rightarrow \ldots$$

If cooling parameter β in (15) grows slowly enough, this sequence approaches stochastically to an absolute minimum of the functional H.

In our implementation, β is changing under the law

$$\beta_n = C \cdot \left(\ln(n+1)\right)^p, \tag{16}$$

where n is a number of the iteration (i.e., one-pass execution of the elementary steps for each internal node of the tree); C and p are the program parameters. By default, $C = 0.01$, $p = 1.5$.

As an alternative, the user may apply another law:

$$\beta_n = C \cdot \left(n+1\right)^{-p}, \tag{17}$$

if negative p value is specified.

There is no way to predict how many iterations will be necessary to reach the absolute minimum of the functional H, as well as to determine whether the local minimum found is really the absolute one. From our experience, several million iterations are usually enough. Depending on the tree size and input sequence lengths, it can take from several hours to several days of

continuous computation. The job duration can be limited by time or iteration number, using the corresponding options of the program.

One simple heuristic criterion has been implemented in the program: if a local minimum was reached at n-th iteration (assuming n>300000), and subsequent n iterations did not lead to a better minimum, the program terminates.

The user can monitor the program performance in the command window log, and view output files in an external application – text editor, web browser, etc. He/she can control the program by manual changing the parameter p in (16):
– each press of '+' key increases value of p by 5% (to speed up cooling);
– each press of '–' key decreases value of p by 5% (to slow down cooling).

The user also can stop the program at any time by pressing keys Ctrl+C or Ctrl-Break, or just closing the command processor window.

7. Examples of the program work

Examples in this section are not intended to be the control tests, and are provided for illustration only. Due to stochastic nature of the model, it is impossible to guarantee that the same result will be obtained.

Example 1: Classic attenuation regulation of threonine biosynthesis in γ-proteobacteria

Input file of sequences, ex1.txt, contains 14 regulatory sites taken from [3]. These sequences were assigned to the leaves of the conventional species tree with 27 nodes. Phylogenetic length of each tree edge was rounded to the nearest integer. Thus we obtained the tree file pr1.tre. Both files use the same names for sequences and for leaves; those are the abbreviated names of species, namely:

EC	*Escherichia coli*
TY	*Salmonella typhi*
KP	*Klebsiella pneumoniae*
EO	*Erwinia carotovora*
YP	*Yersinia pestis*
HI	*Haemophylus influenzae*
VK	*Pasterella multocida*
AB	*Actinobacillus actinomycetemcomitans*
PQ	*Mannheimia haemolytica*
VC	*Vibrio cholerae*
VV	*Vibrio vulnificus*
VP	*Vibrio parahaemolyticus*
SON	*Shewanella oneidensis*
XCA	*Xanthomonas campestris.*

The program was run through the following command line:

```
anneal ex1.txt pr1.tre -l.2 -ch0 -t0 -maxb4 -ymhc2
```

After 7,713,935 iterations which took about 214 hours (on 3 GHz Pentium 4 PC), we got a minimal configuration presented in the file ex1.htm. For that configuration, the functional values were H = 1154, H_1 = 1352, H_3 = –989 (λ=0.2). The conserved secondary structure was reconstructed in all ancestral sequences; one can trace it from each leaf up to the root (corresponding paths are shown in the output files). This also allows the program to propose a multiple alignment of the extant sequences, shown in the end of html file; such alignment is induced by multiple alignments along the paths.

Example 2: Classic attenuation regulation of leucine biosynthesis in γ-proteobacteria

In this example, the input files are ex2.txt and pr2.tre. The species tree is a part of the tree in Example 1; it comprises 23 nodes, of which 12 are the leaves. The experiment was carried out for λ=0.25 with weakened restrictions regarding secondary structures. Result configuration had to be presented in unaligned format, so we used the command line

```
anneal ex2.txt pr2.tre -l.25 -gr13 -maxl70 -minh6 -maxb1 -t0
```

After 1,368,295 iterations during 105 hours, we got a minimal configuration presented in the file ex2.htm. For that configuration, the functional values were $H = 1684$, $H_1 = 1796$, $H_3 = -310$. The conserved secondary structure was reconstructed in all ancestral sequences; one can trace it from each leaf up to the root (corresponding paths are shown in the output file).

Example 3: Program work in leader peptide mode

This example demonstrates the program operation in Lp mode. Here we used the same operons that in Example 1, but this time the sequences were taken starting from the start codon of the leader peptide gene. (The only exception was PQ sequence, for which we could not find a complete Lp gene; so we used AS – *Actinobacillus succinogenes* – sequence instead). The corresponding input files are ex3.txt and pr3.tre.

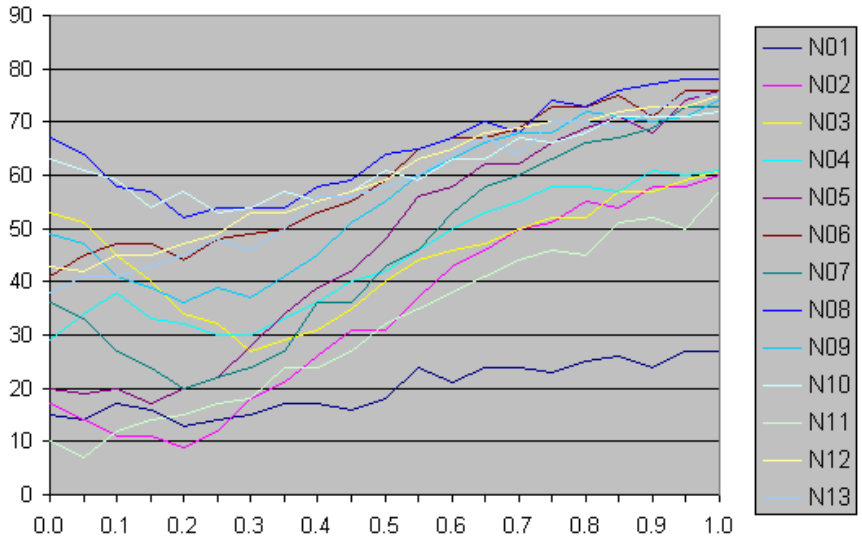
The program was run through the command line

```
anneal ex3.txt pr3.tre -maxb4 -t0 -j1.5 ex3.log -aaThr,Ile -maxi3
```

where the Lp mode was switched on by specifying the names of regulatory amino acids (threonine and isoleucine).

After 2,282,984 iterations during approximately 88 hours, we got the results presented in output files ex3.log and ex3.htm which were described in Section 5. The minimum configuration is characterized by the functional values $H = 1119$, $H_1 = 3488$, $H_3 = -3517$, $H_4 = -298$ ($\lambda = 0.25$, $\mu = 5$). In this configuration, plausible leader peptide genes were reconstructed in all ancestral sequences as well as conservative secondary structures consisting of mutually exclusive anti-terminator and terminator helices.

This allowed us to investigate those sequences with regard to classic attenuation regulation, using our model of such regulation [4, 5]. We got the following dependencies of termination probability (which is inverse to gene expression) vs. amino acid concentration (see figure). One can see that classic attenuation regulation does exist in all reconstructed ancestral sequences. However, the regulation performance, in the average, is worse than for extant sequences, and the root (N01 node) is perhaps the worst case.



References

1. V.A. Lyubetsky, E.A. Zhizhina, L.I. Rubanov. Gibbs Field Approach for Evolutionary Analysis of Regulatory Signal of Gene Expression // *Problems of Information Transmission (Problemy Peredachi Informatsii)*, 2008, Vol. 44, No. 4, pp. 333-351.
2. M. Nei, S. Kumar. *Molecular evolution and phylogenetics.* Oxford University Press, 2000.
3. A.G. Vitreschak, E.V. Lyubetskaya, M.A. Shirshin, M.S. Gelfand, V.A. Lyubetsky. Attenuation regulation of amino acid biosynthetic operons in proteobacteria: comparative genomics analysis // *FEMS Microbiology Letters*, 2004, Vol. 234, No. 2, pp. 357-370.
4. Lyubetsky V., Pirogov S., Rubanov L., Seliverstov A. Modeling Classic Attenuation Regulation of Gene Expression in Bacteria // *Journal of Bioinformatics and Computational Biology*, 2007, Vol. 5, No. 1, pp. 155-180.
5. L. Rubanov and V. Lyubetsky. RNAmodel Web Server: Modeling Classic Attenuation in Bacteria // *In Silico Biology*, 2007, Vol. 7, No. 3, pp. 285-308.