TWOBOX SOFTWARE: DESCRIPTION AND USER'S GUIDE

Table of contents

1. General

The program TwoBox v3.17 has been developed for finding an aggregate (a *system*, or *signal*) of similar sites in the set of given sequences composed of the fixed alphabet. One site (or none) is chosen from each sequence, and those sites should be as similar as possible to each other. The program primarily tries to find such site in every sequence, but it also may reject few sequences if such system is better in terms of the functional used (see Section 2).

The sought-for site can consist of a single box (ie., contiguous segment of the sequence), or of two boxes separated by some number of characters (a *linker*). The length of a linker is fixed or varies within the given interval. Each box length may be specified independently of other.

The program also allows user to search for a signal in presence of known information about conserved position(s) within a box. Such data may be submitted in the form of a *motif* for a box (or boxes). Detailed information about the program arguments and input data see in Sections **Error! Reference source not found.–Error! Reference source not found.**.

TwoBox is the elaboration of our previous heuristic algorithm [1-3] for finding one-box signal by global optimization of predefined functional of the signal quality. The result is a quasi-optimal solution with the greatest value of the functional over all local maxima reached during the search, which is limited by internal criteria or/and duration or/and number of the algorithm steps. Output data of the program is described in Section 6.

The algorithm was extended to the two-box case in a straightforward manner. Recall that the original algorithm widely uses the basic operation of trying all possible sites of the candidate box along the whole sequence. The operation examines $m-l+1$ sites, where m is the sequence length, and l is the box length. If the sought-for site consists of two boxes with the lengths l', l'' separated by a linker of exactly d characters, the similar search is equivalent to finding a box with the length $l=l'+l''$ in a sequence of $m-d$ characters, which involves $(m-d)-l+1$ variants. This search can be treated as a single-box case for the sequence produced from original one by an implicit mapping. If the linker length varies in the interval $[d_{\min}, d_{\max}]$, the overall number of variants to examine will be

$$\sum_{d=d_{\min}}^{d_{\max}} \left[(m-d)-l+1\right] = (m-l+1)\cdot D - \bar{d}\cdot D,$$

where $\bar{d} = \frac{1}{2}\left(d_{\min} + d_{\max}\right)$ is the average length of the linker, and $D = d_{\max} - d_{\min} + 1$ is the uncertainty of the length. Disregarding the latter term, the presence of the linker length uncertainty D is computationally equivalent to the extension of a sequence by a factor of D as compared to the fixed length linker. Such extension of the sequence would be the result of implicit mapping embedded into the algorithm.

As mentioned in [1, 3], the computational complexity of the single-box algorithm is estimated as squared number of the sequences and cubed average length of the sequence. Therefore, two-box extension of the algorithm will have the same complexity if the linker length is known exactly, and the complexity grows as D^3 if uncertainty D of the length exists.

Due to high computational complexity of the algorithm (that rapidly grows as the uncertainty of the linker length increase), the program TwoBox from the very beginning was intended for parallel cluster with intra-processor communications via MPI v.1.1 or later [4, 5]. Any number of processors will do; the program can bring into play all CPU available, and calculation time will decrease approximately s-1 times, where s is the number of CPUs [2]. At least two logical processors are required, so the program is capable of working on typical dual-core PC.

The program is provided as an executable for x86 platform. It is intended for a cluster consisting of several PCs with Windows, interconnected by TCP/IP LAN. The MPI environment is established with use of the free software MPICH2 v.1.2 (by Argonne National Laboratory). This product (or later version of it) have to be installed in all computers of the cluster.

TwoBox uses a command line interface; it has to be run from a command processor of the operating system. The programming language is C, the compiler is Microsoft Visual Studio 2005 Service Pack 1, name of the executable – twobox.exe. Target CPU is Intel 32-bit. The operating systems tested were Microsoft Windows XP Service Pack 3 and Microsoft Windows Server 2003 Service Pack 2. Using TwoBox on other processors and/or operating systems is certainly possible, but may require to re-compile the program or to carry out additional testing.

Software developer: Dr. Lev Rubanov, leading scientist, IITP RAS (Kharkevich Institute).

Contact e-mail: rubanov@iitp.ru

2. Functionals of the system quality

The algorithm uses two quality functionals; both are based on Hamming distance: we use the total number of equal corresponding letters of two candidate sites as a likeness measure. Let n source sequences are given, each one consisting of not greater than m letters. The algorithm aims to choose a site in each sequence (maybe except few sequences) such that the entire system of sites maximizes the overall quality. The site may be either a box w of the length l (single-box case) or two boxes w', w'' with the respective lengths l', l'' separated by a linker of the length $d \in \left[d_{\min}, d_{\max}\right]$ (two-box case). The method of quality computation is as follows.

In single-box case, the quality of the site w_k chosen in the kth sequence, relative to the whole system, is calculated as

$$q_k = \sum_{\substack{i=1 \\ i\neq k}}^{n} \left(l - H(w_i, w_k)\right), \qquad (1)$$

where $H(w_i, w_k)$ is Hamming distance between the boxes w_i, w_k (the number of equal letters at corresponding positions in these boxes.

In two-box case, the quality of the site, i.e. pair of boxes w'_k, w''_k, relative to the whole system of such pairs, is calculated by the formula

$$q_k = \sum_{\substack{i=1 \\ i \neq k}}^{n} \left(l' + l'' - H(w'_i, w'_k) - H(w''_i, w''_k) - P(d_i) - P(d_k) \right), \tag{2}$$

where function $P(d)$ determines the penalty for a deviation of the linker length from the expected value between $d_{\min}$ and $d_{\max}$. The specific function is the algorithm parameter, in particular, it can be null function.

Eqs. (1, 2) are applied in absence of *a priori* information about specific positions in the box or boxes. If we know letters that must (or should) be at certain positions of the box, the algorithm allows to specify this information in the form of sought-for *motif*. The motif is a string of the same length as the box; each character is either a letter from the sequence alphabet (upper and lower case letters are processed differently) or a dummy character (dot/asterisk/question mark) which means no information about this letter. One or two motifs may be used for two-box sites.

In single-box motif case, the site quality Eq. (1) transforms into the formula

$$q_k = \sum_{\substack{i=1 \\ i \neq k}}^{n} \left(l - H(w_i, w_k) + \sum_{j=1}^{l} (\delta_{kj} - \Delta_{kj}) \right), \tag{3}$$

where $\delta_{kj} = \delta$ if *j*th letter of the site chosen from *k*th sequence is equal to *j*th letter of the motif (irrespectively of its case), otherwise, $\delta_{kj} = 0$; $\Delta_{kj} = \Delta$ if *j*th letter of the site chosen from *k*th sequence is *not* equal to *j*th letter of the motif (specified in uppercase), otherwise, $\Delta_{kj} = 0$. In other words, the site quality is added a prize δ for each motif-matching letter and is subtracted a penalty Δ for each letter mismatching the uppercase motif letter. Constants δ and Δ are the algorithm parameters.

In two-box motif case, the site quality Eq. (2) transforms into the formula

$$q_k = \sum_{\substack{i=1 \\ i \neq k}}^{n} \left(l' + l'' - H(w'_i, w'_k) - H(w''_i, w''_k) - P(d_i) - P(d_k) + \sum_{j=1}^{l'} (\delta_{kj} - \Delta_{kj}) + \sum_{j=1}^{l''} (\delta_{kj} - \Delta_{kj}) \right), \tag{4}$$

where the first sum in parentheses is calculated over w'_k box, the second sum – over w''_k box, and notations δ_{kj}, Δ_{kj} are the same as in Eq. (3). One of the sums vanishes if a motif is given only for one box. Finally, if the result of calculation by Eq. (2)–(4) is a negative amount, the algorithm deems null quality of the site.

Once the quality of all sites has been calculated by Eq. (1)–(4), the primary functional of the whole system quality is defined by equation

$$Q_1 = \frac{1}{P-1} \sum_{k=1}^{n} q_k \to \max, \tag{5}$$

where P is the total number of nonempty sites chosen from all sequences (*capacity* of the system).

The secondary functional of the whole system quality is defined as

$$Q_2 = \frac{1}{P(P-1)} \sum_{k=1}^{n} q_k \to \max. \tag{6}$$

The functional Q_1 characterizes the average quality of the system sites, and the functional Q_2 describes the average likeness of the sites. In the extreme case of equal sites, the peaks of both functionals coincides. In other cases, the algorithm primarily uses Q_1 for optimization, taking Q_2 into account if Q_1 gives equal values.

As optional alternative for a motif, the algorithm provides functional Q_1 in modified form aimed at searching more conserved sites. The following functional is used in single-box case:

$$Q_1 = \frac{1}{P-1} \sum_{k=1}^{n} q_k + \sum_{j=1}^{l} c_j. \tag{7}$$

In two-box case we use the formula

$$Q_1 = \frac{1}{P-1}\sum_{k=1}^{n} q_k + \sum_{j=1}^{l'} c_j + \sum_{j=1}^{l''} c_j \,. \qquad (8)$$

Values c_j in Eqs. (7, 8) are produced by the function $c(r)$ that decreases monotonically as the system becomes less conserved at jth position of the sites. In other words, each position with equal letter in all sites is awarded the maximum prize, and this prize decreases if different letter occurs in one or more sequence. Specific function $c(r)$ is the algorithm parameter. Current version of the TwoBox uses a stepwise function with three values: for totally conserved position, for conserved position with given small number of exceptions allowed, and null value otherwise. We accentuate that the quality functional in the form Eq. (7, 8) may be applied only if sites are searched without use of motifs.

3. Input data

The program TwoBox gets input data through the command line arguments. The command line has the following format (optional arguments are bracketed):

```
TWOBOX [options] infile [outfile]
```

- `options` allow a user to change default mode or parameter of the algorithm; each option must start from hyphen (-) or slash (/) character; several options must be separated by blanks;
- `infile` must be a name of the input data file (with optional absolute or relative path) which contains one or several sets of source sequences in FASTA format. If no path is given, the program will seek for the file in current directory;
- optional argument `outfile` specifies a name of the file for the program results. If the argument is omitted, the program will use the input data file name with additional extension ".out". Like `infile`, this argument may include a path; otherwise, the current directory is used.

In accordance with Windows rules, the command line arguments are case insensitive. If the file/path name contains blanks, the corresponding argument should be enclosed in double quotes.

Formats of the input data are described in Sections 4, **Error! Reference source not found.**. Brief help on program arguments and options can be obtained by the command `TWOBOX -h` or `TWOBOX -?` (`TWOBOX /h` and `TWOBOX /?` produce the same result).

4. Command line options

The program TwoBox accepts the following options (disregarding a letter case):

`-a` If specified, the output file will be opened in append mode (may be helpful for scripting); otherwise, the existing file will be overwritten.

`-acid` If specified, the source sequences use the alphabet of amino acids (default is the nucleotide alphabet. See Section **Error! Reference source not found.** for detail.

`-ba`<number> Value c_j of the function $c(r)$ for totally conserved position (see Section 2); default value is 3.

`-bc`<number> Value c_j of the function $c(r)$ for conserved position with exceptions (see Section 2); default value is 1.

`-bx`<number> Number of maximum allowed exceptions to consider a position of the site as conserved (see Section 2); default value is 1.

`-c`<number> This option controls data output to *stdout* from root branch of the program: `-c0` = less information, `-c1` = standard, `-c2` = more data. Helpful for monitoring.

`-d`<number> This option controls diagnostic data output from a secondary branch of the program to the corresponding result file. Once the program terminates, these file are either removed, or appended to the root branch result file, or remain in place, depending on `-z` option (see below). No diagnostics is included by default, and found best solution(s) will be the only output of the program. This parameter can be helpful for the program analysis and debug.

`-f` If specified, the program writes the entire sequence of found solutions into a file in CSV format to import them in the spreadsheet application (eg. Excel).

`-g`<number> This option controls the aggregation of signal quality values within a *Q*-list [2]:
`-g1` (or `-gs`) – all values are summed (default mode),
`-g2` (or `-gm`) – the best quality is chosen.

`-h` Show help on command line arguments of the program.

`-i`<number> Maximum number of the algorithm iterations to perform within a *Q*-list until better value of the system quality functional is reached. The number is used in criterion 1 of the algorithm termination. The value of 5 is used as default, but one should increase it to e.g. (0.5…2)·*n* for *n*>16.

`-j`<mode> This option determines how to consider the site novelty in the algorithm termination:
`-j0` (or `-ji`) – do not take the novelty factor into account for the
`-j1` (or `-ja`) – consider the site novelty in the criterion 1 check,
`-j2` (or `-jr`) – reset a counter if a new site is found (a threshold for that counter is set by option `-i`).

`-l`<number> The length *l* of single box (or *l′* of first box in two-box mode) of the sought-for sites (default 18).

`-ll`<number> The length *l″* of second box of the sought-for sites in two-box mode (default 0, and the program operates in single-box node).

`-m′`string`′` The motif of single (or first) box of the sought-for sites (none by default). The string may consist of letters from the sequence alphabet (case sensitive) and dummy characters – dot, asterisk, question mark (see the motif character interpretation in Section 2). The string length must be equal to the value of `-l` option.

`-mm′`string`′` The motif of second box of the sought-for sites (none by default). The string is similar to that of `-m` option, but its length must be equal to the value of `-ll` option.

`-mb`<number> Value of prize δ in formulas Eqs. (3, 4); default is 1.

`-mp`<number> Value of penalty Δ in formulas Eqs. (3, 4); default is 5.

`-n`<number> Maximum length of the sequence name to print in results. By default, no truncation is applied.

`-p`<number> Minimum number of the sequence couples to use as tree edges at assembly phase of the algorithm. This parameter is used in the criterion 2 of the algorithm termination. If used in the current version, this parameter should be specified as `-p0` which means using of all $n(n-1)/2$ couples.

`-q`<number> This option determines the maximum allowed number of permutations in *Q*-list, in order to make the criterion 1 more stringent. Default value `-q0` is interpreted as absence of such limitation.

`-r`<number> The number of best signals output in the result file (3 by default). Due to this option, the user can choose the most biologically sound solution from found by the algorithm.

`-smin`<number> Minimum allowed length $d_{\min}$ of the linker between the boxes in two-box case (default is 16).

`-smax`<number> Maximum allowed length $d_{\max}$ of the linker between the boxes in two-box case (default is 22).

`-spen`″string″ This option allows user to specify the linker length penalty function *P*(*d*) (see Section 2 for detail). The double quoted string contains values of the function separated by commas. First value corresponds to the length $d = d_{\min}$, last value – to $d = d_{\max}$. The string must contain $D = d_{\max} - d_{\min} + 1$ values; string "8,0,0,2,4,6,8" is used by default.

`-t`<number> The minimum permitted likeness of two sites, i.e., the number of corresponding positions that mismatch. By default, in the single-box mode the algorithm uses the value *l*/2, and the value (*l*′+*l*″)/2 is used in the two-box mode.

`-u`<number> The limit number of the algorithm iterations to perform (i.e., the number of permutations to check from totality of $n!$). Default limit is 10000.

`-v`<number> The number of *Q*-list items generated at the time, to increase the algorithm stability (by default, the bunch consists of 3 permutations).

`-w`<number> Maximum duration of the algorithm work (the number of minutes). The default value `-w0` means no time limit.

`-x`<number> This option sets how the algorithm will calculate the likeness of candidate sites:

`-x0`: calculating “on the fly” whenever needed. This mode of operation would be helpful in insufficient memory conditions, because it is still more effective than use of likeness matrix partly stored in the page file;

`-x1`: the algorithm starts with the computation of triangle pairwise likeness matrix for all candidate sites. Then it uses the matrix during whole operation. This default mode makes the algorithm 3–4 times faster, but the matrix occupies approximately $m^2 \cdot n^2$ bytes (where *n* is the *mapped* sequence length, see Section 1). For instance, the algorithm needs at least 1 gigabyte memory per processor, in order to process 30 sequences with the mapped length of 1000.

`-y`<number> This option specifies the minimum allowed multiplicity of sequence couples coverage. Such parameter of criterion 2 determines the length of *P*-list [2]. By default, $\log_2 n$ will be used as the value (if *n* is not a power of two, the closest greater power-of-two number is used).

`-z`<number> This option allows user to choose a disposition of the result files of secondary branches on completion of the algorithm:
`-z0` (or `-zr`) – remove those output files;
`-z1` (or `-za`) – append those files to the main result file, then remove the files themselves;
`-z2` (or `-zs`) – result files of secondary branches remain in place.

5. Input file

The name of input data file must be specified by argument `infile` in the command line. This file contains a set of source sequences in FASTA format (with some exceptions). Each sequence is represented by exactly two lines (internal line breaks are prohibited, but the line length is not limited):

- first line starts with character ‘>’ followed by the sequence name;
- second line is the sequence itself, i.e., a case insensitive string of the fixed alphabet. Default is

the nucleotide alphabet {A, C, T, G, U}, where letters T and U are deemed equal[1]. In order to use the amino acid alphabet {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}, the user have to specify `-acid` option in the command line. The sequence may include gap characters (hyphen, underscore or equals sign), but blanks and tabs are not allowed.

The program TwoBox supports processing of multiple data sets in single run, provided that other options have the same values. One set of sequences should be separated from another set by at least one blank line. No comments are allowed in the file.

The program checks that input data file conforms to the above format, and outputs a message in case of any error.

6. Output data

6.1 Return code

Main function of the program returns a value which can be a sum of signals listed in Table 1. Such return code may be helpful in automated processing with use of scripts.

Table 1. TwoBox return codes

Code	Symbol	Description
0	OK	Normal completion of the program
1	ERARG	Error in command line arguments
2	ERDATA	Error in the input data file
4	ERIO	Input/output error
8	EODATA	End of the input data file
16	ERMEM	Not enough memory or input data error
32	HELP	Help was requested
64	EDEBUG	(Reserved)
128	ERINT	Internal error of the algorithm
256	ERMPI	MPI error

6.2 Result file

The program outputs the table of found signals to this file. Each signal is represented by the sought-for site position in each sequence. The number of signals to include in the table is determined by `-r` option in the command line (3 by default). The signals are given in descending order of value of the quality functional Eq. (5) or Eqs. (7, 8); if the values are equal, in descending order of average likeness functional Eq. (6) value. The file is in usual text format (see Fig. 1).

Each table starts with a heading line which describes that solution in general. For example, first line in Fig. 1 contains the following information:

- `Best quality result` – the solution is better with regard to the quality functional Eq. (5, 7, 8). The wording will be "`Best avg proximity result`" if the sulution is better with regard to the likeness functional Eq. (6);
- `#1` – number of the solution in descending order of signal quality;
- `[18/22]` – serial number of the algorithm iteration, where this solution was found, and overall number of iterations performed (iterations are numbered starting from 0);
- `7` – capacity of the system (see Section 2);
- `59.33` – value of the functional Eq. (5), (7) or (8);
- `8.47` – value of the functional Eq. (6).

[1] Extensions of the nucleotide alphabet are not supported by current version of the program.

```
Best quality result #1 [18/22] (power: 7, quality: 59.33, avg proximity: 8.47)
  1:XX|PH0858[0142] aaaaaatagaaa (52,  8.66)
  2:XX|PH0859[0006] aaaaagtattaa (53,  8.83)
  3:XX|PH1075[0058] aaaatctaccat (42,  7.00)
  4:XX|PH1086[0176] aaaatatataaa (55,  9.16)
  5:XX|PH1087[0062] aaaaattattaa (53,  8.83)
  6:XX|PH1091[0005] aaaagatattcc (46,  7.66)
  7:XX|PH1093[0167] aaaagatataaa (55,  9.16)

Best quality result #2 [19/22] (power: 7, quality: 59.00, avg proximity: 8.42)
  1:XX|PH0858[0142] aaaaaatagaaa (52,  8.66)
  2:XX|PH0859[0006] aaaaagtattaa (52,  8.66)
  3:XX|PH1075[0056] ataaaatctacc (41,  6.83)
  4:XX|PH1086[0176] aaaatatataaa (54,  9.00)
  5:XX|PH1087[0062] aaaaattattaa (52,  8.66)
  6:XX|PH1091[0005] aaaagatattcc (48,  8.00)
  7:XX|PH1093[0167] aaaagatataaa (55,  9.16)

Best quality result #3 [9/22] (power: 7, quality: 58.66, avg proximity: 8.38)
  1:XX|PH0858[0142] aaaaaatagaaa (51,  8.50)
  2:XX|PH0859[0006] aaaaagtattaa (53,  8.83)
  3:XX|PH1075[0144] taaagtttctaa (40,  6.66)
  4:XX|PH1086[0176] aaaatatataaa (53,  8.83)
  5:XX|PH1087[0062] aaaaattattaa (54,  9.00)
  6:XX|PH1091[0005] aaaagatattcc (46,  7.66)
  7:XX|PH1093[0167] aaaagatataaa (55,  9.16)
```

Fig. 1. Example of the result file in single-box mode.

Then one line per source sequence follow. Items of the line have the following meaning (we use the second line in Fig. 1 as an example):

- `1` – serial number of the source sequence in whole set;
- `XX|PH0858` – name of the sequence;
- `[0142]` – position (starting from 1) where 5'-edge of sought-for site was found in the sequence;
- `aaaaaatagaaa` – a box representing the sought-for site;
- `52` – quality of the site according Eq. (1) or (2);
- `8.66` – average likeness of the site relative to other sites of the system according Eq. (4).

If such line contains only number and name of the sequence, it means that no site was chosen from the sequence in this solution (i.e., the system capacity $P < n$).

Example of the result file in two-box mode is given in Fig. 2. The only difference from above format is that two positions are shown in brackets (5'-edges of two boxes), and the site is presented as those boxes separated by the linker length.

The described information will always appear in the result file. Depending on chosen diagnostic mode (see option `-d` in Section 4), the file may also contain various messages, intermediate solutions, likeness matrix, etc. Some of those messages are described in the next section.

6.3 Output to console

The program TwoBox displays messages on the console during its operation. Specific contents depends on `-c` parameter specifed in command line. An example of default console log is shown in Fig. 3. Most of possible messages are described below.

Number of parallel tasks: 1+kk

This message informs that the program works in parallel on kk+1 processors (the root branch and kk secondary branches).

```
Best quality result #1 [1/54] (power: 12, quality: 189.63, avg proximity: 15.80)
  1:Citrus sinensis psbI          [0007;0031] ttgatg-18-tataaa (164, 14.90)
  2:Aethionema cordifolium psbI   [0007;0030] ttggta-17-tttggt (143, 13.00)
  3:Aethionema grandiflorum psbI  [0007;0030] ttggta-17-tttggt (143, 13.00)
  4:Arabidopsis thaliana psbI     [0007;0031] ttggta-18-tataaa (184, 16.72)
  5:Arabis hirsuta psbI           [0007;0031] ttggta-18-tataaa (184, 16.72)
  6:Barbarea verna psbI           [0007;0031] ttggta-18-tataaa (184, 16.72)
  7:Capsella bursa-pastoris psbI  [0007;0031] ttggta-18-tataaa (184, 16.72)
  8:Crucihimalaya wallichii psbI  [0006;0030] tttgta-18-tataaa (164, 14.90)
  9:Draba nemorosa psbI           [0007;0031] ttggta-18-tataaa (184, 16.72)
 10:Lepidium virginicum psbI      [0007;0031] ttggta-18-tataaa (184, 16.72)
 11:Lobularia maritima psbI       [0007;0031] ttggta-18-tataaa (184, 16.72)
 12:Nasturtium officinale psbI    [0007;0031] ttggta-18-tataaa (184, 16.72)

Best quality result #2 [34/54] (power: 12, quality: 188.00, avg proximity: 15.66)
  1:Citrus sinensis psbI          [0007;0031] ttgatg-18-tataaa (165, 15.00)
  2:Aethionema cordifolium psbI   [0007;0030] ttggta-17-tttggt (142, 12.90)
  3:Aethionema grandiflorum psbI  [0007;0030] ttggta-17-tttggt (142, 12.90)
  4:Arabidopsis thaliana psbI     [0007;0031] ttggta-18-tataaa (183, 16.63)
  5:Arabis hirsuta psbI           [0007;0031] ttggta-18-tataaa (183, 16.63)
  6:Barbarea verna psbI           [0007;0031] ttggta-18-tataaa (183, 16.63)
  7:Capsella bursa-pastoris psbI  [0007;0031] ttggta-18-tataaa (183, 16.63)
  8:Crucihimalaya wallichii psbI  [0007;0030] ttgtat-17-tataaa (155, 14.09)
  9:Draba nemorosa psbI           [0007;0031] ttggta-18-tataaa (183, 16.63)
 10:Lepidium virginicum psbI      [0007;0031] ttggta-18-tataaa (183, 16.63)
 11:Lobularia maritima psbI       [0007;0031] ttggta-18-tataaa (183, 16.63)
 12:Nasturtium officinale psbI    [0007;0031] ttggta-18-tataaa (183, 16.63)
```

Fig. 2. Example of the result file in two-box mode.

Job accepted (n=xx m<=yy l=zz)

This confirmation appears after the next portion of data was read from the input file. Here, xx is a number of source sequences; yy is the maximum length of a sequence; zz is the length of sough-for sites (a sum of two box lengths in two-box case).

ttt s) Signal from [Tkk] for Pii(nn) (pow=xx q=yy avprx=zz new=mm)

This message appears after each iteration of the algorithm that was executed for a permutation from the *P*-list. Here, ttt is elapsed time in seconds (hereinafter, "elapsed" means time since start of current data portion processing); kk is the number of secondary branch which found this signal; ii is sequential number of the permutation in the *P*-list; nn is the iteration number (starting from 0); xx is capacity of the signal; yy is quality of the signal; zz is average likeness of sites; mm is the number of sites found in the first time.

ttt s) Signal from [Tkk] for Qii,jj(nn) (pow=xx q=yy avprx=zz new=mm)

This message appears after each iteration of the algorithm that was executed for a permutation from a *Q*-list. Here, ttt is elapsed time in seconds; kk is the number of secondary branch which found this signal; ii is the number of that *Q*-list; jj is sequential number of the permutation in that *Q*-list; nn is the iteration number; xx is capacity of the signal; yy is quality of the signal; zz is average likeness of sites; mm is the number of sites found in the first time.

Terminated due to P-list end

The algorithm terminated because *P*-list has finished and all *Q*-lists have been processed.

Terminated due to iteration limit

The algorithm terminated because maximum number of iterations were done.

Terminated due to time limit

The algorithm terminated because specified time has been exhausted.

```
(C) 2009, Laboratory of Mathematical Methods and Models in Bioinformatics
Institute for Information Transmission Problems, Russian Academy of Sciences
Program for specific Two Box Search (multiple CPU, v.3.17)  L.Rubanov, 2009

Number of parallel tasks: 1+4
Job accepted (n=7  m<=200  l=12)
6 s) Signal from [T3] for P2(2) (pow=7  q=55.66  avprx=7.95  new=7)
6 s) Signal from [T1] for P0(0) (pow=7  q=56.00  avprx=8.00  new=3)
6 s) Signal from [T2] for P1(1) (pow=7  q=57.00  avprx=8.14  new=2)
6 s) Signal from [T4] for P3(3) (pow=7  q=56.00  avprx=8.00  new=2)
12 s) Signal from [T1] for Q0,0(5) (pow=7  q=57.33  avprx=8.19  new=5)
12 s) Signal from [T2] for Q1,0(6) (pow=7  q=54.66  avprx=7.80  new=2)
12 s) Signal from [T3] for Q2,0(4) (pow=7  q=57.66  avprx=8.23  new=7)
12 s) Signal from [T4] for Q3,0(7) (pow=7  q=56.00  avprx=8.00  new=3)
18 s) Signal from [T2] for Q1,1(9) (pow=7  q=57.00  avprx=8.14  new=0)
18 s) Signal from [T1] for Q0,1(8) (pow=7  q=57.00  avprx=8.14  new=2)
18 s) Signal from [T4] for Q3,1(11) (pow=7  q=55.66  avprx=7.95  new=1)
18 s) Signal from [T3] for Q2,1(10) (pow=7  q=56.00  avprx=8.00  new=6)
24 s) Signal from [T2] for Q1,2(12) (pow=7  q=54.00  avprx=7.71  new=1)
24 s) Signal from [T1] for Q0,2(13) (pow=7  q=55.33  avprx=7.90  new=1)
24 s) Signal from [T4] for Q3,2(14) (pow=7  q=56.00  avprx=8.00  new=0)
25 s) Signal from [T3] for Q2,2(15) (pow=7  q=58.66  avprx=8.38  new=2)
30 s) Signal from [T2] for P4(16) (pow=7  q=59.00  avprx=8.42  new=7)
31 s) Signal from [T4] for P5(18) (pow=7  q=54.66  avprx=7.80  new=1)
31 s) Signal from [T1] for Q0,3(17) (pow=7  q=57.33  avprx=8.19  new=0)
32 s) Signal from [T3] for Q2,3(19) (pow=7  q=58.66  avprx=8.38  new=0)
36 s) Signal from [T4] for Q5,0(21) (pow=7  q=56.33  avprx=8.04  new=0)
37 s) Signal from [T2] for Q4,0(20) (pow=7  q=59.33  avprx=8.47  new=1)
37 s) Signal from [T1] for Q0,4(22) (pow=7  q=53.66  avprx=7.66  new=7)
38 s) Signal from [T3] for Q2,4(23) (pow=7  q=54.66  avprx=7.80  new=5)
43 s) Signal from [T4] for P6(24) (pow=7  q=59.00  avprx=8.42  new=1)
43 s) Signal from [T2] for Q4,1(25) (pow=7  q=57.00  avprx=8.14  new=2)
44 s) Signal from [T1] for Q0,5(26) (pow=7  q=57.33  avprx=8.19  new=0)
44 s) Signal from [T3] for Q2,5(27) (pow=7  q=59.33  avprx=8.47  new=1)
47 s) Signal from [T4] for Q6,0(28) (pow=7  q=59.33  avprx=8.47  new=2)
47 s) Signal from [T2] for Q4,2(29) (pow=7  q=55.66  avprx=7.95  new=2)
Terminated due to P-list end
Best quality result #1 [20/30] (power: 7, quality: 59.33, avg proximity: 8.47)
Best quality result #2 [16/30] (power: 7, quality: 59.00, avg proximity: 8.42)
Best quality result #3 [24/30] (power: 7, quality: 59.00, avg proximity: 8.42)
RC: OK  (time: 47 sec)

Global RC: OK  (time: 47 sec)

Thank you for using the program.
```

Fig. 3. Example of the program output to console.

Terminated due to unknown reason

The algorithm terminated because of some reason which may be explained by the program return code or other messages.

Best quality result #kk [mm/nn] (power: xx, quality: yy, avg proximity: zz)

General characteristics of finally chosen best solution: kk is a number of the solution in descending order of signal quality; mm is serial number of the algorithm iteration, where this solution was found; nn is overall number of iterations performed; xx is capacity of the system; yy is the signal quality; zz is average likeness of the signal sites.

Best avg proximity result #kk [mm/nn] (power: xx, quality: yy, avg proximity: zz)

Similar to the previous message; only a reason of choice differs.

RC: xx (time: yy sec)

This message summarizes results of processing for current data set. Here, xx is a symbol of return code (see Table 1); yy is the elapsed time in seconds.

Global RC: xx (time: yy sec)

Similar to the previous message, but appears after all input data portions have been processed. Here, yy is overall working time since program start.

P-list consists of xx permutations

This message informs that the *P*-list consisting of xx permutations was built based on criterion 2.

Generate xx permutation(s) for Qyy list

This message informs that the *Q*-list with number yy that consists of xx permutations was built based on criterion 1.

ttt s) Send to task [Tkk] permutation Pii(nn): x1 x2 ... xn

This message appears when the algorithm transfers a permutation from *P*-list to the free secondary branch for processing.

ttt s) Send to task [Tkk] permutation Qii,jj(nn): x1 x2 ... xn

Similar to the previous message, but the permutation is taken from *Q*-list. Here, ttt is elapsed time in seconds; kk is a number of the secondary branch; ii is a number of the *Q*-list; jj is a number of the permutation in the *Q*-list; nn is the current iteration number; x1 x2 ... xn is transferred permutation of the sequence numbers.

Stop P-list processing; task [Tkk] free

Informs that entire *P*-list has been processed, and secondary branch with number kk became free.

Stop Qxx list (length nn); task [Tkk] free

Informs about closing the *Q*-list with number xx after processing of nn permutations, because criterion 1 is met. As a result, the secondary branch with number kk becomes free.

Recurrent permutations rejected: xx

During the algorithm operation, xx permutations recur and were rejected.

The algorithm needs at least 2 tasks!

The program was run on single processor and cannot proceed.

Error xx in a secondary task

The error with code xx occurred in a secondary branch of the program (see Table 1 for error codes).

Error xx creating result

The error with code xx occurred during selection of the best result (see Table 1 for error codes).

Invalid argument "xxx"

Erroneous argument xxx was specified in the command line.

[Tkk]: Cannot open input data file "xxx"

This message informs about an error which occurred in the secondary branch with number kk, when the program was opening the input data file named xxx (the file does not exist or already used by other program or operating system).

[Tkk]: Cannot open output data file "xxx"

This message informs about an error which occurred in the secondary branch with number kk, when the program was opening or creating output file named xxx for the program results.

Cannot open history file "xxx"

The program cannot open the history file with name xxx to write the chronological list of found solutions.

6.4 History file

Unlike the result file which contains only several best solutions (depending on `-r` option in the command line), the history file contains entire list of found solutions in chronological order. This file will be written if the program TwoBox is run with the command line option `-f`; it may be helpful in analysis of the algorithm behavior. The file has CSV (comma-separated values) format to facilitate import to a spreadsheet application, such as Excel. Each line contains the data for one signal and uses Tab character as field delimiter; character fields are enclosed in double quotes. The order and contents of the fields are shown in Table 2. The very first line of the file contains field headings rather than values. These headings are given in the first column of the table.

Таблица 2.

Type, "heading"	**Description**	**Notes**
Numeric, "#"	Sequential iteration number of the algorithm	Such number is assigned at the permutation generation, therefore, the file may be unordered by this field, due to different duration of parallel branches of the program
Numeric, "P"	Permutation number in the *P*-list or the number of *Q*-list, depending on the next field	The numeration starts from 0
Numeric, "Q"	Permutation number in the *Q*-list with number from the previous field, or -1 to indicate that the permutation belongs to the *P*-list	The numeration starts from 0
Numeric, "Pwr"	Signal capacity (the number of source sequences from where the sites were selected)	
Numeric, "New"	Number of the sites that appear in the first time in this signal	
Numeric, "Qt"	The signal quality as per Eq. (5) or (7) or (8)	
Numeric, "QtAvg"	Average likeness of the signal sites as per Eq. (6)	
Numeric, "W1"	Position of the box in first source sequence (starting from 1). In two-box mode, this field contains two comma separated numbers which are the positions of two boxes	This quadruple field repeats *n* times (*n* is the number of sequences). Index 1 in the headings change correspondingly up to *n*.
String, "name of the sequence"	Sought-for site (name of the sequence is shown as the column heading). In two-box mode, the two sites are shown in the format like in Fig. 2	
Numeric, "Qw1"	The site quality according to Eqs. (1)–(4)	
Numeric, "QwA1"	Average likeness of the site with regards to other sites of this signal	
Numeric, "p1"	A number between 1 and *n*	These *n* fields represent a permutation of source sequence numbers that led to this signal
...		
Numeric, "p*n*"	A number between 1 and *n*	

Thus, each line contains 5*n*+7 fields, where *n* is the number of sequences in the source data set.

7. Software distribution package

The software distribution package is a compressed file twobox.zip which contains the files listed in Table 3.

Table 3

File name	Description
twoboxXXX_YY.pdf	This document in different languages (XXX is version number, YY is country code)
twobox.exe	Executable module for x86 Windows architecture with mpich2 v1.2 installed
twobox.exe.manifest	System description of the executable module
vcredist_x86.exe	C++ redistributable libraries by Microsoft. If a computer has not Microsoft Visual Studio 2005 Service Pack 1 installed, you should run once this executable in that PC
*.txt	Examples of TwoBox input file with a set of sequences. The files named *_MA.txt contain the same sequences in multiple alignment form for the sake of clearness
test.bat	An example of the script to run the program with test input data
*.out	Typical output files resulted from the script execution

Additional files may also be available from the software download page:

mpich2_XXX_x86.zip — To run the program TwoBox on the PC without MPICH2 product installed, user can take advantage of this file contents (see Section 8 for detail).

source.zip — If available, this file contains source code of the program along with configuration files needed to compile it in Microsoft Visual Studio 2005 Service Pack 1.

8. Installation and checking

1. On each PC to be used with the program TwoBox, create a folder e.g. d:\twobox, and extract the entire contents of distribution package into that folder.
2. If Microsoft Visual Studio 2005 Service Pack 1 was not installed on the PC earlier, run vcredist_x86.exe from the created folder and follow instructions displayed on screen.
3. Install MPICH2 version 1.2 or later if it was not done earlier. The product can be downloaded for free from the Argonne National Laboratory Web site at

   http://www.mcs.anl.gov/research/projects/mpich2/downloads/index.php?s=downloads

   Please take in mind that you need a version for Windows 32-bit (IA32) platform.

   Note: If you would not install MPICH2 package for some reason, you may download its libraries in abovementioned file mpich2_120_x86.zip and extract the file contents into the program folder created at step 1. Then run a command smpd -install from that folder before running TwoBox. However, we cannot guarantee successful operation and full functionality of the program in such case.
4. Run Windows command processor (cmd.exe) and switch to the program folder:

   ```
   d:
   cd d:\twobox
   ```

For better convenience, we suggest to create a shortcut for the command processor on desktop, then substitute the working directory in that shortcut properties with d:\twobox. After that you will not need to type the above commands each time you use the program.

5. To check the program operation, enter the command `mpiexec -n 2 twobox –h` . The help information on program arguments will output to the command processor window. Depending on Windows security and firewall settings, you can see some warnings or requests to confirm running the program and allow it to access the network. We recommend to add TwoBox program to the firewall exception list to avoid those warnings in future.
6. To verify the program operation using the script, enter the command `test` . The program will run on this PC in four-way mode despite actual number of CPUs and cores (you should not worry about this, because mpich2 libraries will make all necessary emulation). However, sample output files from the distribution package will be overwritten so you should rename or copy them somewhere prior to enter the command. Depending on CPU number and performance, the entire script processing can take 30-40 minutes or more.
7. Once the script complete, compare new result files (*.out) with those from the distribution. Due to different performance and various random factors, the files may not be identical, but the results have to be similar.
8. To use the program for real experiments, you should specify the desired cluster configuration as described in the MPICH2 documentation.

9. How to use TwoBox effectively

User should consider computational complexity of the task. Full exhaustion for n sequences of the length m involves of the order of m^n operations, therefore, it is NP-problem. Our quasi-optimal algorithm is of polynomial complexity which is m^3n^2 for single-box sites and $m^3n^2(d_{max}–d_{min})^3$ for two-box sites, where (d_{min}, d_{max}) is the interval of intra-box linker length variation allowed.

Therefore, we recommend to set the variation interval as narrow as possible, using the program options `-smin`, `-smax`. Remember to modify penalty function (`-spen` option) if you change the default interval.

Memory limitation is not so important if you have at least 1 GB RAM per CPU. In general, the largest block the program allocates is a similarity matrix of $m^2n^2(d_{max}–d_{min})^2$ bytes. However, in case of longer sequences you can experience a sharp growth of disk activity due to swapping. In such situation, we recommend to use option `-x0` for memory/speed tradeoff.

The following technical limits are set in the current program version:

- number of sequences in the input file: $2 \le n \le 256$;
- length of the sequence: $m < 4096$;
- length of the sought-for box(es): $l < 32$ (single-box), $l' + l'' < 32$ (two-box).

These technical limits may be easily modified if necessary.

Each but one parallel branch of the program make the corresponding CPU 100% loaded. So we suggest to specify the following number of branches to run on a PC, when planning the cluster configuration:

- the number of cores plus 1 – for multiple core CPU (3 for dual-core, 5 for quad-core, etc.)
- 3 for single core CPU with Hyper-Threading enabled
- 2 for single core CPU without Hyper-Threading (or disabled)

Specifying greater number of branches is allowed, but normally not result in performance gain.

10. References

1. Danilova L.V., Gorbunov K.U., Gelfand M.S., Lyubetsky V.A. Algorithm of regulatory signal recognition in DNA sequences (2). *Molecular Biology (Moscow)*, 2001, V. 35, No. 6, p. 841-848.
2. S.N. Istomina, L.I. Rubanov. Parallel algorithm of regulatory signal search in bacterial genomes. *Information Processes*, 2002, V. 2, No 1, p. 85-90 (in Russian) http://www.jip.ru/2002/Isto.pdf
3. L.V. Danilova, V.A. Lyubetsky, M.S. Gelfand. An algorithm for identification of regulatory signals in unaligned DNA sequences, its testing and parallel implementation. *In Silico Biology*, 2003, V. 3, No. 1,2, p. 33-47. http://www.bioinfo.de/isb/2003/03/0004/
4. MPI: A Message-Passing Interface Standard. *Message Passing Interface Forum*, Version 1.1: June 1995. Knoxville, University of Tennessee, 1995.
5. MPI-2: Extensions to the Message-Passing Interface. *Message Passing Interface Forum*, July 18, 1997. Knoxville, University of Tennessee, 1997.