Spaln Ver. 2.2.2 (May, 9, 2016)

Mapping and Alignment of cDNA or Protein Sequence Set onto Genomic Sequence

Osamu Gotoh

Artificial Intelligence Research Center
National Institute of Advanced Industrial Science and Technology (AIST)
AIST Tokyo Waterfront Bio-IT Research Building
2-4-7 Aomi, Koto-ku, Tokyo, 135-0064, Japan
Tel: +81-3-3599-8040, Fax: +81-3-3599-8080
E-mail: o.gotoh@aist.go.jp

Department of Intelligence Science and Technology
Graduate School of Informatics, Kyoto University
Yoshida Honmachi, Sakyo-ku, Kyoto, Kyoto 606-8501, Japan
Tel: +81-75-753-9109, Fax: +81-75-753-9110
E-mail: o.gotoh@i.kyoto-u.ac.jp

Overview

Spaln (space-efficient spliced alignment) is a stand-alone program that maps and aligns a set of cDNA or protein sequences onto a whole genomic sequence in a single job. From Version 1.4, **spaln** supports a combination of protein sequence database and a given genomic segment. From Version 2.2, **spaln** also performs rapid similarity search and (semi-)global alignment of a set of protein sequences again a protein sequence database. **Spaln** adopts multi-phase heuristics that makes it possible to perform the job on a conventional personal computer running under Unix/Linux with limited memory. The program is written in C++ and distributed as source code and also as executables for a few platforms. Unless binaries are not provided, users must compile the program on their own system. Although the program has been tested only on a Linux operating system, it is likely to be portable to most Unix systems with little or no modifications. The accessory program **sortgrcd** sorts the gene loci found by **spaln** in the order of chromosomal position and orientation.

References

[1] Gotoh, O. "A space-efficient and accurate method for mapping and aligning cDNA sequences onto genomic sequence," *Nucleic Acids Research* **36** (8), 2630-2638 (2008).
[2] Gotoh, O. "Direct mapping and alignment of protein sequences onto genomic sequence," *Bioinformatics* **24** (21), 2438-2444 (2008).
[3] Iwata, H. and Gotoh, O. "Benchmarking spliced alignment programs including Spaln2, an extended version of Spaln that incorporates additional species-specific features," *Nucleic Acids Res.*, **40** (20) e161 (2012).

New features in Version 2.2

From this version, **spaln** formally supports rapid similarity search and (semi-)global alignment of protein sequence queries against a protein sequence database. In this connection, one-time formatting-mapping modes are made available.

Major changes Version 2.1.4

Three-byte addressing has been abolished for better performance and compatibility at an increased cost of memory usage. In this connection, the formats of data files are changed; data files formatted with an older version of spaln can be used by the new version, but the opposite will cause a failure.

Major changes Version 2.1.3

From this version, **Spaln** can deals with non-standard genetic codes. See option –C*N*.

Major changes in Version 2.1

From this version, the formats of the database files and output of –O12 option of **Spaln** have been changed. These modifications have been done to eliminate the limitations on the lengths of identifiers of both genomic and query sequences (formally up to 20 letters). First, **Makdbs** program now produces four (.ent, *.grp, *.idx, *.seq) or optionally the fifth (.odr) files, instead of four (*.grp, *.hed, *.idx, *.seq) files. The internal data structures of the .seq and .grp files are unchanged, while that of .idx file has been changed, and the .ent and .odr files are new comers. Second, **Sortgrcd** now requires the third file (.qrd) in addition to .grd and .erd files. Despite these modifications in the file formats, the user interface of each program is virtually unchanged. **Spaln** can read database sequences formatted by an older version, provided that the length limitations are satisfied. However, **Sortgrcd** can no longer process outputs from an older version of **Spaln**.

Major changes in Version2

The major revision number of spaln has been updated from 1 to 2. The new version incorporates additional features for intron recognition together with other modifications. The major points of revisions are as follows.

1. The codes have been extensively rewritten in C++, so that they are no longer compiled by a C compiler.
2. A heuristic routine is added after the HSP search to reduce the number of calls of the restricted DP routine.
3. **Spaln** supports parallel processing; -t[*N*] option specifies the number of CPUs involved in a multi-thread operation. If *N* is omitted, all available CPUs are used.
4. A branch point signal and an intron propensity based on oligomer compositions are newly incorporated in the scoring system.
5. Splice junction signals derived from two sources can be mixed at an arbitrary rate, where the two sources are (1) the dinucleotide pair at the ends of an intron and (2) the somewhat longer sequence around each splice junction. The relative contributions of these two terms can be freely adjusted by –yS*N* option.
6. The intron penalty is automatically adjusted depending on the values of other parameters.
7. Species specific parameters are available for 61 divergent eukaryotes.

Caveats

Files formatted on different platforms or in different configurations (e.g. –m32 and –m64) may not be inter-compatible.

Install

In the rest of this document, we refer to *work* as the directory of your workplace, *seqdb* as the directory in which genomic and database sequences are stored, and *download,* which may or may not be identical to *work*, as the directory used for installation. These directories are assumed to be in full-path name.

Compile from source

To compile the source codes in the default settings, follow the instructions below. To modify the location of executables and/or other settings, run "configure --help" at step 6 below. (**Warning**: Full path name rather than relative path name must be given for executables or other directories as the arguments of the **configure** command.) These locations are hard coded in **spaln**.

1. `% mkdir` *download*
2. `% cd work`
3. Download spalnXX.tar.gz (XX: version code).
4. `% gzip –cd spalnXX.tar.gz | tar xf –`
5. `% cd spalnXX/src`
6. `% configure [--help]`
   // Please manually edit Makefile if $(CXX) does not indicate a C++ compiler
7. `% make`
8. `% make install`
   // Executables are copied to ../bin
   // **makmdm** program makes mutation data matrices of various PAM levels in the directory ../table.
9. `% make clearall`
10. Add *download*/spalnXX/bin to your PATH
    `% setenv PATH $PATH:`*`download`*`/spalnXX/bin` (csh/tsh)
    `$ export PATH=$PATH:`*`download`*`/spalnXX/bin` (sh/bsh)
    Preferably, you may add the above line in your start up rc file (e.g. ~/.bashrc)
    Alternatively, move or copy *download*/spalnXX/bin/* to a directory on your PATH, if you have not specified the location of executables at step 6 above.
11. Only if you have changed the location of *table* and/or *seqdb* directory after installation, set the env variables ALN_TAB and/or ALN_DBS as explained in the following subsection.
12. Now proceed to "**Sequence data**".

Use binaries

Binaries for a 64 bit Linux machine are available. To use the binaries, follow the instructions below.

Case I: Assume every material is stored in *work*. In this case, *seqdb*=*work*.

1. `% mkdir` *work*
2. `% cd` *work*
3. Download spalnXX.PC.tar.gz (XX: version code, PC: platform code).
4. `% gzip -cd spalnXX.PC.tar.gz | tar xf –`
5. Add *work* /bin to your PATH or move *work* /bin/* to a directory on your PATH
6. `% mv ./table/* .; rmdir ./table`
7. `% mv ./seqdb/* .; rmdir ./seqdb`
8. Now proceed to "**Sequence data**".

Case II: Assume *work* is distinct from *seqdb*.

1. `% mkdir` *download*
2. `% cd` *download*
3. Download spalnXX.PC.tar.gz (XX: version code, PC: platform code).
4. `% gzip -cd spalnXX.PC.tar.gz | tar xf -`
5. Add *download*/bin to your PATH or move *seqdb*/bin/* to a directory on your PATH
6. `% setenv ALN_TAB` *download*`/table` (csh/tsh) or
   `$ export ALN_TAB=`*download*`/table` (sh/bsh)
7. `% setenv ALN_DBS` *download*`/seqdb` (csh/tsh) or
   `$ export ALN_DBS=`*download*`/seqdb` (sh/bsh)
8. Add the above lines to your rc file, so that you don't have to repeat the commands at every login time.
9. Now proceed to "**Sequence data**".

Sequence data

All sequence files should be in (multi-)fasta format. The genomic sequence and the protein sequence database (if relevant) must be formatted before use. From ver. 2.2, the formatting, mapping, and alignment phases can be done in a single job. In this mode, you can skip steps 4-7 below, although you cannot reuse the formatted data because they are not stored in disk files.

1. `% cd` *seqdb*
2. Download or copy genomic sequences or protein sequence database in multi-fasta format.
3. Chromosomal sequences may be concatenated into a single file. To render the "make" command effective, the extension of the genomic sequence file should be ".mfa" or ".gf', and protein database sequence should be ".faa". Hereafter, the file name is assumed (but not obligatory) to be genuspec_g.gf or prosdb.faa, where "genuspec" indicates the combination of 4-letter prefixes of the genus and species of the relevant organism (*e.g.* "homosapi" for human). The total number of bases or amino acids in a file must not be greater than or equal to 2**32.
4. `% ./makeidx.pl –i[n|p|np] genuspec_g.gf` or
   `% ./makeidx.pl –i[a] prosdb.faa`
   These commands are shortcuts that replace the following series of operations 5-7, if (i) the input is a single sequence file and (ii) the default parameter setting is used. The `-ix` option specifies the "block file(s)" .bkx to be constructed, where x is 'a', 'n' or 'p'. The `–inp` option will construct both .bkn and .bkp files together with the .idx and associated files. If `-ix` is omitted or `x` is empty, no block file is constructed.
5. `% make genuspec_g.idx` or `prosdb.idx`
   This command converts the sequence into a binary format. Either four or five files, genuspec_g.seq, genuspec_g.idx, genuspec_g.ent, genuspec_g.grp, and occasionally genuspec_g.odr) are constructed (prosdb instead of genuspec_g in case of make prosdb.idx). It may take several tens of minutes to construct the files for mammalian genome.
6. Alternatively, chromosomal sequences may be fed to **makdbs** and **spaln** as multiple arguments. In that case, make commands described above will not function properly. See **Change from previous version** for version 1.4.3.
7. `% make genuspec_g.bkn` (for cDNA queries) or

`% make genuspec_g.bkp` (for protein queries) or
`% make prosdb.bka` (for protein database)

- This command makes the block index table. This process may take another several tens of minutes.
- Internally, the make command invokes
  `spaln –Wgenuspec_g.bkn -KD [Options] genuspec_g.gf` or
  `spaln –Wgenuspec_g.bkp -KP [Options] genuspec_g.gf` or
  `spaln –Wprosdb.bka -KA [Options] prosdb.faa`.
- If genuspec_g.grp or prosdb.grp were successfully constructed at step 4 above, the option values below would be automatically calculated by script makblk.pl. **Warning:** The estimated maximal gene size can be inadequately small if only a part of the genome (ex. a single chromosome) is formatted. At that time, explicitly specify the maximum gene size by the `-XG`*N* option of makblk.pl or at the runtime of **spaln**.
- Options: (default value)
  1. `–Xk`*N* word size (11 for DNA, 5 for protein)
  2. `–XG`*N* maximum gene length (estimated from genome size)
     N can have suffix 'k' and 'M' to indicate that the number is measured in kbp and Mbp, respectively.
  3. `–Xb`*N* block size (4096)
     N < 65536 and "genome size" < 65536 * *N* must be satisfied. An estimate of *N* is sqrt(genome size). For mammal, *N*≒54000.
  4. `–Xa`*N* a parameter used to filter excessively abundant words (10)
  5. `–Xg`*N* maximal distance in block number between 5' terminal and 3' terminal blocks
  6. `–Xs`*N* Distance between adjacent seeds within a block (= word size).
     This option allows for overlapping k-mer seeds at the block search phase, where *N* (1≥N≥k) indicates the distance between adjacent seeds. The sensitivity of block search will be improved with a small *N*, which is particularly beneficial for mapping of short queries. Compared with the default setting (*N*=*k*, i.e. non-overlapping tiling seeds), however, the memory consumption for the k-mer table will be increased by the factor of *k*/*N*.

Execution

1. `% cd` *work*
2. Prepare protein, cDNA, or genomic segment sequence(s) in (multi-)fasta format (denoted by *query* below).
3. Run **spaln** in one of the following four modes. Don't mix different query types in a single run. **Spaln** does not support comparison between two genomic segments.
   `(A)% spaln –Q[0|1|2|3] [–O`*N*`] [other options]` *genome_segment query*
   `(B)% spaln –Q[4|5|6|7] [–O`*N*`] [other options] –d`*genuspec_g query*
   `(C)% spaln –Q[4|5|6|7] [–O`*N*`] [other options] –a`*prosdb query*
   `(D)% spaln –Q[4|5|6|7] [–O`*N*`] [other options]`
   `[`*genuspec_g.mfa*`|`*prosdb.fa*`]` *query*
4. Only a subset of queries may be examined if *query* is replaced with '*query* (from to)', where "from" and "to" are the first and last entry numbers in *query* to be examined. To run **spaln** on multiple CPUs, for example, the following commands may be used and the results may be summarized with **sortgrcd,** as explained later.
   `(A)% spaln -Q7 -O12 -oxxxO1 -dgenuspec_g '`*query* `(1 1000)'`

```
(B)% spaln -Q7 -O12 -oxxxO2 -dgenuspec_g 'query (1001 2000)'
(C)% spaln -Q7 ...
```

However, the multi-thread operation simplifies the procedure as follows:

```
(D)% spaln -Q7 -O12 -oxxx -dgenuspec_g -t[N] query
```

5. Options: (default value)
   - `-CN`: Use the genetic code specified by the "transl_table number" defined in http://www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi (1).
   - `-HN`: Output is suppressed if the alignment score is less than *N*. See also `–pw`. (35)
   - `-K[D|P|A]`: Format either genomic DNA for sequence search with DNA (D) or Protein (P) queries or protein database (A). Use in combination with `–W` option below.
   - `-LS`: Smith-Waterman-type local alignment. This option may prune out weakly matched regions.
   - `-M[N]`: Single or multiple output for each query
     - No option (default): single locus
     - No argument: Multiple loci up to the maximum number specified by the program (4 in the present implementation)
     - *N*=1: Re-search the *query* region not aligned in the first trial. May be useful to detect chimera or fragmented genomic region
     - *N*>1: Output multiple loci maximally up to *N*
   - `-ON`: Select output format. (4)
     - *N*=0: GFF3 gene format
     - *N*=0: (semi-)global alignment statistics in protein-protein mode
     - *N*=1: Alignment
     - *N*=2: GFF3 match format
     - *N*=2: Sugar format in protein-protein mode
     - *N*=3: Bed/psl format
     - *N*=4: Exon-oriented format similar to the output of megablast –D 3
     - *N*=4: coordinate + match length in protein-protein mode
     - *N*=5: Intron-oriented output
     - *N*=6: Concatenated exon sequence
     - *N*=7: Translated amino-acid sequence. Presently not very useful for cDNA queries because the entire exon rather than an ORF is translated
     - *N*=8: Cigar format
     - *N*=9: Vulgar format
     - *N*=10: SAM format
     - *N*=11 BAM format (in preparatoin)
     - *N*=12: Output the same information as `-O4` in the binary format. If `-oOutput` is set, three files named *Output.grd*, *Output.erd* and *Output.qrd* will be created. Otherwise, *query.grd*, *query.erd*, and *query.qrd* will be created.
     - *N*=15: Inference of the exon structure of the query sequence, based on spliced alignment with the closest genomic region in the database
   - `-QN`: Select algorithm. (3)
     - $0 \leqq N \leqq 3$: Genomic segment or amino acid sequence in the FASTA format given by the first argument vs. *query* given by the second argument. One may skip the formatting step described above if only this mode of operation is used.
     - $4 \leqq N \leqq 7$: Genome mapping and spliced alignment or protein database search

and ordinary alignment. The genomic sequence or protein database must be formatted beforehand.
    - *N*=0,4: DP procedure without HSP search. Considerably slow
    - *N*=1,2,3,5,6,7: Recursive HSP search up to the level of (*N*% 4)
- `-RS`*S*: Read block index table from file *S*. If omitted, the X.bkn, X.bkp, or X.bka file will be read depending on the type of query. The appropriate file is searched for in the current directory, the directory specified by the `env` variable 'ALNDBS', and the 'seqdb' directory specified at the compile time in this order.
- -S*N*: Specify the orientation of query. (0)
    - *N*=0: depend on the query annotation
    - *N*=1: forward direction only
    - *N*=2: reverse direction only
    - *N*=3: both directions
- `-T`*xxx*: Specify the species-specific parameter tables. For genome vs. DNA comparison, –yS flag should also be set in combination with this option. *xxx* corresponds to the subdirectory in the ~/table/ directory. From version 2.2.2, genus name (e.g. Homo) or eight-character genus-species identifier (e.g. homosapi) may be used as xxx.
- `-V`*N*: Minimum space to induce Hirschberg's algorithm (16M)
- `-W`*S*: Write block index table to file *S*.
- `-i[a|p]`:Input mode with –Q[0≦*N*≦3].
    - `-ia`: Alternative mode; a genomic segment of an odd numbered entry in the input file is aligned with the query of the following entry.
    - `-ip`: Parallel mode; the *i*-th entry in the file specified by the first argument is aligned with the *i*-th entry in the file specified by the second argument.
    - default: The genomic segment specified by the first argument is aligned with each entry in the file specified by the second argument.
- `-i[a|p]`:Input mode with -Q[0>4]:.
    - `-ia`: Alternative paired-end mode; 5' and 3' matching pairs must appear alternatively in a file.
    - `-ip`: Parallel paired-end mode; 5' and 3' matching pairs must appear in the same order in the two files.
    - default:. Normal input mode.
- `-o`*S*: Destination of output file name (stdout)
- `-pa`: Terminal polyA or polyT sequence is not trimmed.
- `-pq`: Quiet mode; messages to stderr are suppressed.
- `-pw`: Report result even if alignment score is below threshold value.
- `-xB`*S*: Bit pattern of seeds used for HSP search at level 1
- `-xb`*S*: Bit pattern of seeds used for HSP search at level 3
- `-u`*N*: Gap-extension penalty (3, 2, 2)
- `-v`*N*: Gap-opening penalty (8, 6, 9)
- `-ya`*N*: Accept intron ends other than canonical pairs
    - *N*=0: canonical only (GT..AG, GC..AG, AT..AC)
    - *N*=1: relaxed to (GT..AG, GC..AG, AT..AN)
    - *N*=2: *N*=1 + allow 1 mismatch from GT..AG
    - *N*=3: any dinucleotide pairs
- `-yi`*N*: Intron penalty (11, 8, 11).
- `-yj`*N*: Incline of long gap penalty (0.6)
- `-yk`*N*: Flex point where the incline of gap penalty changes (7)

- `-ylN`: Double affine gap penalty if *N*=3; otherwise affine gap penalty
- `-ymN`: Score for a nucleotide match (2, 2)
- `-ynN`: Penalty for nucleotide mismatch (6, 2)
- `-yoN`: Penalty for a in-frame termination codon (100)
- `-ypN`: PAM level used in the alignment (third) phase (150)
- `-yqN`: PAM level used in the second phase (50)
- `-yxN`: Penalty for a frame shift (100)
- `-yyN`: Relative contribution of splicing signal (8)
- `-yzN`: Relative contribution of coding potential (2)
- `-yAN`: Relative contribution of the translational initiation or termination signal (8)
- `-yBN`: Relative contribution of branch point signal (0)
- `-yEN`: Minimum exon length (2)
- `-yIS`: Intron distribution parameters
- `-yJN`: Relative contribution of the bonus given to a conserved intron position in the transcripts
- `-yLN`: Minimum intron length (20)
- `-yS`: For genome vs. cDNA queries, use species-specific exon-intron boundary signals. For protein queries, invoke the "salvage" procedure to examine all blocks with positive scores.
- `-ySN`: *N* specifies the percentile contribution of the species-specific splice signal. The other part is derived from the ubiquitous signal given to the dinucleotides at the ends of an intron. An omission of *N* implies *N*= 100. By default, *N*= 0 for DNA and *N*= 100 for protein queries.
- `-yX`: For a DNA query, this option sets parameter values for cross-species comparison. -yS100 is also automatically invoked. Conversely, this option specifies an intra-species mode for a protein query, whereby -yS30 is invoked.
- `-yYN`: Relative contribution of length-dependent part of intron penalty (8)
- `-yZN`: Relative contribution of oligomer composition within an intron (0)

6. % sortgrcd [options] *.grd
   - **Sortgrcd** is used to recover the output of **spaln** with –O12 option, to apply some filtering, and also to rearrange the output of multiple **spaln** runs.
   - Options:
     - `-CN`: Minimum cover rate = % nucleotides in predicted exons / length of *query* (3 times if *query* is protein) (0-100)
     - `-FN`: Filter level *N*=0 (default): no; *N*=1: mild; *N*=2: medium; *N*=3: stringent
     - `-PN`: Minimum sequence identity (0-100)
     - `-HN`: Minimum alignment score
     - `-ON`: Output mode (*N*=0: Gff3 gene form; *N*=4: same as –O4 of **spaln;** *N*=5: same as –O5 of **spaln**; N=15: unique introns)
     - `-VN`: Internal memory size used for core sort. If the data size is greater than *N*, the sorting procedure will be done in pieces. Suffix k or K indicates *N* being in kilo byte. Likely, suffix m or M indicates *N* being in mega byte.
     - `-mN`: Maximum number of mismatches within 20 bp from the nearest exon-intron boundary
     - `-n1`: Allows for non-canonical（other than GT..AG, GC..AG, AT..AC）intron ends

- `-uN`: Maximum number of unpaired (gap) sites within 20 bp from the nearest exon-intron boundary
- `-Sa`: Sort chromosomes/contigs in the alphabetical order of their identifiers
- `-Sb`: Sort chromosomes/contigs in the order of abundance of the hits
- `-Sc`: Sort chromosomes/contigs in the order of appearance in the genomic database (default)
- `-Sr`: Sort records in the reverse order of chromosomal position when they are mapped on the minus strand

- By default, no filter listed above is applied.
- When the output of **spaln** is separated into several files, the combined results are subjected to the sorting. Although *.grd files are assigned as the argument, there must be corresponding *.erd and *.qrd files in the same directory.
- In the default output format, the gene structure corresponding to each transcript is delimited by a line starting with '@', whereas each gene locus is delimited by a line starting with '!'. Two transcripts belong to the same locus if their corresponding genomic regions overlap by at least one nucleotide on the same strand.
- With `-O0` option, the outputs follow the Gff3 gene format (http://www.sequenceontology.org/gff3.shtml) where a gene locus is defined as described above.

Example

- To experience the flow of procedures, type in the following series of commands after moving to ~/seqdb or spalnXX/seqdb.

```
%   make dictdisc_g
%   makeidx.pl -inp dictdisc_g
%   make dictdisc_c
%   make dictdisc_c.srd
%   make dictdisc.spn
%   make ddiaa.idx
%   make ddiaa.bka
```

- See also some other examples shown in the man page

```
%   man ./spaln.1
```