The DIAMOND protein aligner

Introduction

DIAMOND is a sequence aligner for protein and translated DNA searches, designed for high performance analysis of big sequence data. The key features are:

- Pairwise alignment of proteins and translated DNA at 500x-20,000x speed of BLAST.
- Frameshift alignments for long read analysis.
- Low resource requirements and suitable for running on standard desktops or laptops.
- Various output formats, including BLAST pairwise, tabular and XML, as well as taxonomic classification.

DIAMOND was published in Nature Methods in 2015 [1]. Please cite this paper when using the program in published research.

Support & updates are available at `http://github.com/bbuchfink/diamond/`.

1 Quick start guide

We assume to have a protein database file in FASTA format named `nr.faa` and a file of DNA reads that we want to align named `reads.fna`. In order to set up a reference database for DIAMOND, the makedb command needs to be executed with the following command line:

```
diamond makedb --in nr.faa -d nr
```

This will create a binary DIAMOND database file with the specified name (`nr.dmnd`). The alignment task may then be initiated using the `blastx` command like this:

```
diamond blastx -d nr -q reads.fna -o matches.m8
```

The output file here is specified with the `-o` option and named `matches.m8`. By default, it is generated in BLAST tabular format.

Note:

- The program may use quite a lot of memory and also temporary disk space. Should the program fail due to running out of either one, you need to set a lower value for the block size parameter `-b` (see 3.7).
- Use the options `--sensitive` or `--more-sensitive` for searches with higher sensitivity.
- The runtime of the program is not linear in the size of the query file and it is more efficient for large query files (> 1 million sequences) than for smaller ones.
- Low complexity masking is applied to the query and reference sequences by default. Masked residues appear in the output as X.
- The default e-value cutoff of DIAMOND is 0.001 while that of BLAST is 10, so by default the program will search a lot more stringently than BLAST and not report weak hits.

2 Installation & requirements

DIAMOND compiles as generic C++ code and has no particular requirements on the hardware architecture, but it makes use of the SSE instruction set of the Intel/AMD x86-64 platform if available and will run considerably faster on that platform. It runs on POSIX-compatible operating systems (Linux, FreeBSD, OS X) as well as on Microsoft Windows.

A high-memory server is recommended for better performance, but the program can be run on standard desktop computers or laptops.

Compiled binaries are available for download for Linux, macOS (via Bioconda), FreeBSD (via `pkg`) and Windows. For best performance, it is recommended to compile the software from source on the target system.

2.1 Binary download

2.1.1 Linux

A precompiled binary is available for recent Linux systems and may be downloaded for immediate use:

```
wget http://github.com/bbuchfink/diamond/releases/download/v0.9.25/diamond-linux64.tar.gz
tar xzf diamond-linux64.tar.gz
```

The binary requires a fairly current CPU that supports POPCNT and SSSE3. If the binary does not work on your system, i.e. you are getting error messages like `Illegal instruction` or `Kernel too old`, please try another installation method or compile the software from source.

2.1.2 Linux/macOS (via Bioconda)

Install Bioconda on your system if not already present, then install DIAMOND using this shell command:

```
conda install -c bioconda diamond
```

Regularly updating to the latest version is also recommended:

```
conda update diamond
```

2.1.3 Linux (via Docker)

To pull the latest version of the official Docker container:

```
docker pull buchfink/diamond
```

To pull a specific version:

```
docker pull buchfink/diamond:version0.9.25
```

2.1.4 FreeBSD

On FreeBSD, you can use `pkg install diamond` to install the software.

2.1.5 Windows

A binary executable for Windows can be downloaded at the DIAMOND github page. You also need to install the Visual C++ Redistributable.

2.2 Compiling from source

Compilation requires GCC 4.1 or later, CMake 2.6 or later as well as libpthread and zlib including development headers. To compile DIAMOND from source, invoke the following commands on the shell:

```
wget http://github.com/bbuchfink/diamond/archive/v0.9.25.tar.gz
tar xzf v0.9.25.tar.gz
cd diamond-0.9.25
mkdir bin
cd bin
cmake ..
make install
```

Note:

- Use `cmake -DCMAKE_INSTALL_PREFIX=...` to install to a different prefix.
- Use `cmake -DBUILD_STATIC=ON` to create a statically linked executable.

3 Command line options

3.1 Commands

Commands are issued as the first parameter on the command line and set the task to be run by the program.

`makedb`
Create a DIAMOND formatted reference database from a FASTA input file.

`blastp`
Align protein query sequences against a protein reference database.

`blastx`
Align translated DNA query sequences against a protein reference database.

`view`
Generate formatted output from DAA files.

`version`
Print version information.

`dbinfo`
Print information about a database file.

`help`
Print help message.

3.2 Makedb options

`--in <file>`
Path to the input protein reference database file in FASTA format (may be gzip compressed). If this parameter is omitted, the input will be read from `stdin`.

`--db/-d <file>`
Path to the output DIAMOND database file.

`--taxonmap <file>`
Path to mapping file that maps NCBI protein accession numbers to taxon ids (gzip compressed). This parameter is optional and needs to be supplied in order to provide taxonomy features. The file can be downloaded from NCBI: `ftp://ftp.ncbi.nlm.nih.gov/pub/taxonomy/accession2taxid/prot.accession2taxid.gz`.

`--taxonnodes <file>`
Path to the `nodes.dmp` file from the NCBI taxonomy. This parameter is optional and needs to be supplied in order to provide taxonomy features. The file is contained within this archive downloadable at NCBI: `ftp://ftp.ncbi.nlm.nih.gov/pub/taxonomy/taxdmp.zip`.

3.3 General options

`--threads/-p #`
Number of CPU threads. By default, the program will auto-detect and use all available virtual cores on the machine.

3.4 Input options

`--db/-d <file>`
Path to the DIAMOND database file.

`--query/-q <file>`
Path to the query input file in FASTA or FASTQ format (may be gzip compressed). If this parameter is omitted, the input will be read from `stdin`.

`--taxonlist <list>`
Comma-separated list of NCBI taxonomic IDs to filter the database by. Any taxonomic rank can be used, and only reference sequences matching one of the specified taxon ids will be searched against. Using this option requires setting the `--taxonmap` and `--taxonnodes` parameters for `makedb`.

`--query-gencode #`
Genetic code used for translation of query in BLASTX mode. A list of possible values can be found at the NCBI website. By default, the Standard Code is used. Note: changing the genetic code is currently not fully supported for the DAA format.

`--strand {both, plus, minus}`
Set strand of query to align for translated searches. By default both strands are searched.

`--min-orf/-l #`
Ignore translated sequences that do not contain an open reading frame of at least this length. By default this feature is disabled for sequences of length below 30, set to 20 for sequences of length below 100, and set to 40 otherwise. Setting this option to `1` will disable this feature.

3.5 Alignment options

`--sensitive`
This mode is a lot more sensitive that the default and generally recommended for aligning longer sequences. The default mode is mainly designed for short read alignment, i.e. finding significant matches of $>$50 bits on 30-40aa fragments.

`--more-sensitive`
This mode provides some additional sensitivity compared to the sensitive mode.

`--frameshift/-F #`
Penalty for frameshifts in DNA-vs-protein alignments. Values around 15 are reasonable for this parameter. Enabling this feature will have the aligner tolerate missing bases in DNA sequences and is most recommended for long, error-prone sequences like MinION reads.
In the pairwise output format, frameshifts will be indicated by \ and / for a shift by +1 and -1 nucleotide in the direction of translation respectively.
Note that this feature is disabled by default.

`--gapopen #`
Gap open penalty.

`--gapextend #`
Gap extension penalty.

`--matrix <matrix name>`
Scoring matrix. The following matrices are supported, with the default being BLOSUM62.

Matrix	Supported values for (gap open)/(gap extend)	Default gap penalties
BLOSUM45	(10-13)/3; (12-16)/2; (16-19)/1	14/2
BLOSUM50	(9-13)/3; (12-16)/2; (15-19)/1	13/2
BLOSUM62	(6-11)/2; (9-13)/1	11/1
BLOSUM80	(6-9)/2; 13/2; 25/2; (9-11)/1	10/1
BLOSUM90	(6-9)/2; (9-11)/1	10/1
PAM250	(11-15)/3; (13-17)/2; (17-21)/1	14/2
PAM70	(6-8)/2; (9-11)/1	10/1
PAM30	(5-7)/2; (8-10)/1	9/1

`--comp-based-stats (0,1)`

Compositional bias correction of alignment scores. 0 means no score correction, 1 means compositional bias correction as described in [2] (default). Compositionally biased sequences often cause false positive matches, which are effectively filtered by this algorithm in a way similar to the composition based statistics used by BLAST [3].

`--algo (0,1)`

Algorithm for seed search. 0 means double-indexed and 1 means query-indexed. The double-indexed algorithm is the program's main algorithm, but it is inefficient for very small query files, where the query-indexed algorithm should be used instead.
By default, the program will automatically choose one of the algorithms based on the size of the query and database files. The algorithm used will be displayed at program startup.
Note that while the two algorithms are configured to provide roughly the same sensitivity for the respective modes, results will not be exactly identical to each other.

3.6 Output options

`--out/-o <file>`

Path to the output file. If this parameter is omitted, the results will be written to the standard output and all other program output will be suppressed.

`--outfmt/-f #`

Format of the output file. The following values are accepted:

`0` BLAST pairwise format.

`5` BLAST XML format.

`6` BLAST tabular format (default). This format can be customized, the `6` may be followed by a space-separated list of the following keywords, each specifying a field of the output.

qseqid Query Seq - id

qlen Query sequence length

sseqid Subject Seq - id

sallseqid All subject Seq - id(s), separated by a ';'

slen Subject sequence length

qstart Start of alignment in query

qend End of alignment in query

sstart Start of alignment in subject

send End of alignment in subject

qseq Aligned part of query sequence

sseq Aligned part of subject sequence

full_sseq Full subject sequence

evalue Expect value

bitscore Bit score

score Raw score

length Alignment length

pident Percentage of identical matches

nident Number of identical matches

mismatch Number of mismatches

positive Number of positive - scoring matches

gapopen Number of gap openings

gaps Total number of gaps

ppos Percentage of positive - scoring matches

qframe Query frame

btop Blast traceback operations(BTOP)

staxids Unique Subject Taxonomy ID(s), separated by a ';' (in numerical order). This field requires setting the `--taxonmap` parameter for `makedb`.

stitle Subject Title

salltitles All Subject Title(s), separated by a '<>'

qcovhsp Query Coverage Per HSP

qtitle Query title

By default, there are 12 preconfigured fields: `qseqid sseqid pident length mismatch gapopen qstart qend sstart send evalue bitscore`.

`100` DIAMOND alignment archive (DAA). The DAA format is a proprietary binary format that can subsequently be used to generate other output formats using the `view` command. It is also supported by MEGAN and allows a quick import of results.

`101` SAM format.

`102` Taxonomic classification. This format will not print alignments but only a taxonomic classification for each query using the LCA algorithm. The output lines consist of 3 tab-delimited fields:

1. Query ID
2. NCBI taxonomy ID (0 if unclassified)
3. E-value of the best alignment with a known taxonomic ID found for the query (0 if unclassified)

The score range for the LCA algorithm is set by the `--top` parameter. The default value is 10 which means that all alignments whose score is at most 10% lower than the best score are considered for the LCA computation.

Using this format requires setting the `--taxonmap` and `--taxonnodes` parameters for `makedb`.

`103` PAF format. The custom fields in the format are AS (bit score), ZR (raw score) and ZE (e-value).

`--salltitles`

Include full length subject titles into the DAA format. By default, DAA files contain only the shortened sequence id (up to the first blank character).

`--sallseqid`

Include all subject ids into the DAA file. By default only the first id of each subject is included. As the subject ids are much shorter than the full titles this option will save space compared to the `--salltitles` option.

`--compress (0,1)`

Enable compression of the output file. 0 (default) means no compression, 1 means gzip compression.

`--max-target-seqs/-k #`

The maximum number of target sequences per query to report alignments for (default=25). Setting this to 0 will report all alignments that were found.

`--top #`

Report alignments within the given percentage range of the top alignment score for a query (overrides `--max-target-seqs` option). For example, setting this to 10 will report all alignments whose score is at most 10% lower than the best alignment score for a query.

`--range-culling`

Restrict hit culling to overlapping query ranges. This feature is designed for long query DNA sequences that may span several genes. In these cases, the default of reporting the 25 best overall hits could cause hits to a lower scoring gene to be overshadowed. But just increasing the number of alignments reported will bloat the output size and reduce performance.

Using this feature along with `-k 25` (default), a hit will only be deleted if at least 50% of its query range is spanned by at least 25 higher or equal scoring hits.

Using this feature along with `--top 10`, a hit will only be deleted if its score is more than 10% lower than that of a higher scoring hit over at least 50% of its query range.

The percentage is configurable using `--range-cover`. Note that this feature is currently only available in frameshift alignment mode.

`--evalue/-e #`

Maximum expected value to report an alignment (default=0.001).

`--min-score #`

Minimum bit score to report an alignment. Setting this option will override the `--evalue` parameter.

`--id #`

Report only alignments above the given percentage of sequence identity.

`--query-cover #`

Report only alignments above the given percentage of query cover.

`--subject-cover #`

Report only alignments above the given percentage of subject cover.

`--max-hsps #`

The maximum number of HSPs per subject sequence to report for each query. Note that in general any number of HSPs may exist for a single query/subject pair. The program's default policy is to report any HSP if its query and subject ranges are not enveloped by a higher scoring HSP and if it meets the e-value treshold.

This option may be used to limit the number of HSPs reported. Setting this option to 1 will cause only the single best HSP for each query/subject pair to be reported.

`--unal (0,1)`

Report unaligned queries (0=no, 1=yes). By default, unaligned queries are reported for the BLAST pairwise, BLAST XML and SAM format.

`--no-self-hits`
: Suppress reporting of identical self-hits between sequences.

3.7 Memory & performance options

`--block-size/-b #`
: Block size in billions of sequence letters to be processed at a time. This is the main parameter for controlling the program's memory usage. Bigger numbers will increase the use of memory and temporary disk space, but also improve performance. The program can be expected to use roughly six times this number of memory (in GB). So for the default value of `-b2.0`, the memory usage will be about 12 GB.

`--tmpdir/-t <directory>`
: Directory to be used for temporary storage. This is set to the output directory by default. The amount of disk space that will be used depends on the program's settings and your data. As a general rule you should ensure that 100 GB of disk space are available here. If you run the program in a cluster environment, and disk space is only available over a slow network based file system, you may want to set the `--tmpdir` option to `/dev/shm`. This will keep temporary information in memory and thus increase the program's memory usage substantially.

`--index-chunks/-c #`
: The number of chunks for processing the seed index (default=4). This option can be additionally used to tune the performance. It is recommended to set this to 1 on a high memory server, which will increase performance and memory usage, but not the usage of temporary disk space.

3.8 View options

`--daa/-a <file>`
: Path to input file in DAA format.

`--out/-o <file>`
: Path to output file. If this parameter is omitted, the results will be written to the standard output and all other program output will be suppressed.

These aligner parameters apply to the view command as well and work in the same way: `--outfmt`, `--compress`, `--max-target-seqs`, `--top`. Note that taxonomy features are currently not available for the DAA format.

3.9 Advanced options

`--freq-sd #`
: During the seed search, seeds with very high frequency in the queries or the database are ignored. This option sets the number of standard deviations above the average frequency for ignoring a seed. The default values are 50 in default mode, 10 in sensitive mode, and 200 in more-sensitive mode.

`--band #`
Dynamic programming band for seed extension. Initial seed hits are scanned across a band of this length in both directions.

`--xml-blord-format`
Use gnl|BL_ORD_ID style format for hit IDs in XML output.

`--range-cover #`
The percentage of a hit's query range that needs to be spanned by higher scoring hits for a hit to be deleted in range culling mode (default=50.0).

4 Support & FAQ

Support is provided by email or the GitHub issues page. For support requests it is usually helpful if you provide details about your data (size and type of sequences) and your execution environment (CPU cores, RAM, operating system).

Community feedback and feature requests are also appreciated.

1. **How to run the program on multiple input files?**

   It is not recommended to run more than one instance of DIAMOND simultaneously on the same machine, because it is more efficient if you allocate more ressources to a single task by increasing the block size parameter `-b` and using lower numbers for `-c`. For this reason, input files should be processed consecutively. Pipes can also be used process multiple input files. For example, this command will build a database from all `fasta.gz` files in the current directory:

   ```
   zcat *.fasta.gz | diamond makedb -d diamond_db
   ```

2. **The program runs for a very long time and the output file remains empty.**

   This is normal. Larger databases are split into chunks and the output is kept in temporary files at first and only merged in the end.

3. **There are no temporary files.**

   The temporary files are not visible using standard shell commands because `unlink` is invoked on them after creation, which ensures that the files will be removed in case of ungraceful termination of the program.

4. **More than one alignment is reported for a single query and subject sequence.**

   Due to the definition of HSPs, which are pairs of segments that form a linear alignment, any number of these may be contained in a single query/subject pair. The program reports additional aligned domains by default as they may contain valuable information. This behaviour can be changed by using the `--max-hsps` option.

5. **The program does not appear to be sensitive enough / it does not find an alignment I expect it to find or that another aligner reported.**

   There are a number of reasons that can cause hits to not be found or reported. In particular, even identical self-alignments of protein sequences may not be reported for some of the reasons listed below.

   (a) The hit may not be deemed significant, i.e. its e-value is above the configured cutoff. This is possible even for short 100% identity matches. Also note that e-value computations are not compatible for different aligners most of the time.

   (b) Low complexity masking and compositional bias correction may cause hits to be filtered. This is usually desirable as these methods are highly effective in eliminating false positives. Note that differences in false positive filtering can cause substantial deviations in the output for different aligners used on the same data.

   (c) Due to its seed configuration, the program is not able to align sequences shorter than 12 or 10 amino acids, for the fast mode and sensitive modes respectively.

   (d) Seeds that occur too frequently in the input are ignored, which may cause problems if your data is extremely repetitive. This behaviour can be adjusted using the `--freq-sd` option.

   (e) The program may not be sensitive enough for your application. You should try the more sensitive modes (options `--sensitive` and `--more-sensitive`). Still, perfect results cannot be guaranteed by heuristical algorithms, but only by a rigorous Smith-Waterman computation.

6. **The program terminates with an unclear or no error message.**

   The task may have been killed by the system for running out of memory. Use `dmesg` to check for a respective kernel message. See 3.7 for instructions on memory usage. If you run the program on a cluster, you may need to allocate more memory to the task.

7. **Are results consistent for multiple calls of the program?**

   The program has a basic consistency guarantee that the same version run on the same input with the same parameters will produce identical results, regardless of the hardware, operating system, compiler, or number of CPU threads. If this was not the case, I would consider it a bug.

   Beyond that, no guarantees of consistency can be made. DIAMOND like all fast aligners uses numerous heuristics that can alter results in unexpected ways. For example, it has been noted that the top hit of a BLAST search can change depending on the `-max_target_seqs` option. Because gapped extension is expensive, BLAST ranks all the subject sequences that were hit for a query based on ungapped extensions at seed hits and considers only a subset of the subjects for gapped extension. The pool of subjects considered is bigger for higher values of `-max_target_seqs`, so in rare cases the true best hit may or may not make it to the gapped extension stage depending on this parameter.

   Effects like this seem counterintuitive to many users, but they are a natural byproduct of heuristics used by all fast aligners. The only tools that can guarantee optimal results in any sense of the word are rigorous search tools like SSEARCH.

5 Optimizing performance

You should take the time to configure the program to your needs instead of just running it with default settings, as this will substantially optimize the use of system resources. Some points to consider:

- Set the `-e` parameter (maximum expected value) as low as possible.
- Set the `-k` parameter (number of target sequences to report alignments for) as low as possible. This will improve performance and reduce the use of temporary disk space and the size of the output files.
- Consider using the `--top` parameter instead of `-k`. For example, setting `--top 5` will report all alignments whose score is at most 5% lower than the top alignment score for a query.
- Configure the tabular output format to only report the output fields that are actually required (see 3.6.).
- Use the BLAST tabular format for machine-based processing of the output. Use the XML and SAM format only if required by your downstream tools or if you prefer it as a human-readable format.
- Use the option `--compress 1` to automatically generate gzip-compressed output files.

About

DIAMOND is developed by Benjamin Buchfink and distributed as Open Source under the GNU Affero General Public License.
Bug reports, support and other inquiries may be sent to Benjamin Buchfink.

References

[1] Benjamin Buchfink, Chao Xie, and Daniel H. Huson. Fast and sensitive protein alignment using diamond. *Nature methods*, 12(1):59–60, Jan 2015.

[2] Maria Hauser, Martin Steinegger, and Johannes Söding. Mmseqs software suite for fast and deep clustering and searching of large protein sequence sets. *Bioinformatics (Oxford, England)*, 32(9):1323–1330, May 2016.

[3] Yi-Kuo Yu and Stephen F. Altschul. The construction of amino acid substitution matrices for the comparison of proteins with non-standard compositions. *Bioinformatics (Oxford, England)*, 21(7):902–911, Apr 2005.