CrocoBLAST:UserManual

From WebChem Wiki

Below you can find the **CrocoBLAST** user manual, which contains all the information you need in order to make efficient use of **CrocoBLAST**. Note that you need not install **CrocoBLAST**, as it is sufficient to download the files from the webpage and unzip them. You may run **CrocoBLAST** from the graphical interface or directly from the command line. Using the graphical interface currently requires Java - but don't worry, you probably have it already. If you get in trouble, see the Technical details.

Happy sequence munching! Nom nom!

What can I do with CrocoBLAST? [Collapse]

Everyone loves BLAST. We love BLAST too. What's not to love? BLAST is easy to understand and it still represents the gold standard in local alignment algorithms. But BLAST doesn't like parallel computing. Not even in the latest NCBI release. If your lab produces small data sets, you are probably familiar with the NCBI BLAST service. If you have medium or large data sets, perhaps you have access to a powerful computer and you can afford to wait a few weeks or even months to see whether your alignment is finished or you need to start over. Or perhaps you are among the lucky few who have access to a GPU machine equipped with a GPU BLAST code. Whatever your situation, if you are producing data from Next Generation Sequencing (NGS) experiments, you have likely encountered the limitations of currently available BLAST implementations. **CrocoBLAST** is for you.

CrocoBLAST offers a platform for planning, running, monitoring, and managing your BLAST calculations. With **CrocoBLAST**, you will always know *how much time* it will take to run a BLAST job, and you will be able to *pause* or interrupt a BLAST job at any time and *retrieve partial results*.

Another key aspect of **CrocoBLAST** is that it is extremely hungry. As such, it takes the sequences in the input file, and munches them into small pieces that will be fed into the classical BLAST algorithm. This enables you to run *very large* BLAST jobs efficiently even with minimal computational resources (say, your *desktop machine*), while ensuring that the output is *identical* to what you would obtain if you were to run BLAST (minus the headache and the constant frustration while you wait for the job to complete... or not).

Terminology [Collapse]

There are a few basic terms you need to keep in mind when running BLAST within CrocoBLAST.

Input file and Database

It its essence, BLAST takes an unknown nucleotide or protein sequence, tries to align it against a set of reference sequences, and then reports the score of each alignment, in an effort to help you identify the unknown sequence. In practice, this translates into taking an *input file* with many query sequences, and aligning each of the query sequences against a *database* of known sequences. Such databases are typically stored in suitable repositories such as NCBI, or may be obtained in-house.

Therefore, in order to run BLAST, you will need to specify an *input file* containing the query sequences, and a *database file* containing the reference sequences. CrocoBLAST accepts input files in FASTA and FASTQ format. BLAST uses a specific *database format* for database file. You may indicate the database file either in database format or in FASTA or FASTQ format, which will be converted to database format before BLAST is run. Within CrocoBLAST you may directly download databases from the NCBI server.

BLAST program

Depending on the nature of the query and reference sequences, there are several BLAST programs you may use within CrocoBLAST:

- blastp - compares an amino acid query sequence against a protein sequence database
- blastn - compares a nucleotide query sequence against a nucleotide sequence database
- blastx - compares a nucleotide query sequence translated in all reading frames against a protein sequence database
- tblastn - compares a protein query sequence against a nucleotide sequence database dynamically translated in all reading frames
- tblastx - compares the six-frame translations of a nucleotide query sequence against the six-frame translations of a nucleotide sequence database

Therefore, in order to run BLAST, you will need to indicate which BLAST program you intend to use.

BLAST options

The BLAST algorithm for sequence alignment is relatively complex, and the default settings are not always optimal for identifying suitable hits in a database or retrieving only the relevant results. You may wish to fiddle with the default BLAST settings by changing the general BLAST *options*, as well as the *options* specific to each BLAST program. Please see the NCBI web pages for a full description of accessible BLAST options (http://www.ncbi.nlm.nih.gov/books/NBK279675/).

Job

Within CrocoBLAST, a *job* is defined by the BLAST program (with or without non-default options), the database, the input file, and the output location (folder). When created, each job receives a unique job ID that can be referenced whenever you wish to perform an operation on that job.

Queue

All BLAST jobs created within the CrocoBLAST environment are included in a list, which we further refer to as *queue*. The concept of *queue* is useful because it allows you to plan your work in advance and manage your jobs as you need. While CrocoBLAST only runs one *job* at a time, all your interaction with the created jobs will be via the *queue*. For example, you may pause one job to obtain the partial alignment results, and start another job while you analyze the partial results of the original job. This enables you to retain the settings and progress of the original job, which you may later choose to resume.

Job management

[Collapse]

CrocoBLAST is built to help you plan your BLAST jobs and run them efficiently. CrocoBLAST operates with the concept of queue, which is basically a list of BLAST jobs scheduled to run. Thus, you can plan several BLAST job and let CrocoBLAST manage their execution for you.

All CrocoBLAST functionality is available via the command line utility and the graphical user interface. In fact, the graphical user interface does precisely what its name suggests: it provides an interface for the command line utility. In a nutshell, while you can interact with CrocoBLAST via simple commands, you may also use the interface to generate the commands or read the output of such commands.

Create BLAST jobs

As already mentioned, BLAST takes an input file with unknown sequences and aligns each such sequence against a database of known sequences. To create a job, you must first specify the BLAST program (http://www.ncbi.nlm.nih.gov/BLAST/blast_program.shtml) you plan to use, which depends on the nature of the unknown sequences in your input file, and the nature of the sequences in the reference database. Then, you need to specify the name of the *database* listed in the CrocoBLAST index (more details on that below) that contains the reference sequences you wish to use. Finally, provide the input file and the location where you want CrocoBLAST to place the output files. Keep in mind that the output files may be quite large. Finally, if you want to change the default BLAST settings, you can do so by specifying the names and values of the BLAST options (http://www.ncbi.nlm.nih.gov/books/NBK279675/) of interest.

```
CrocoBLAST -add_to_queue blast_program database input_file output_folder
CrocoBLAST -add_to_queue blast_program database input_file output_folder --options option1 value1
option2 value2...
```

Note that, when you create a BLAST job, CrocoBLAST automatically assigns each BLAST job a unique job ID, and updates the CrocoBLAST queue (more on this later).

Manage databases

To submit a BLAST job, you must specify which database you wish to align against. The first time you indicate a database for a BLAST job, CrocoBLAST will remember it and add it to its index, so that in the future it is easier for you to access this database. You can see which databases are already indexed in CrocoBLAST:

```
CrocoBLAST -list_databases
```

If you want to remove a database from the CrocoBLAST index (for example, because it has become obsolete), you need to first specify the type of **sequences** it holds, and, of course, the name of the database.

```
CrocoBLAST -remove_database nucleotide database_name
CrocoBLAST -remove_database protein database_name
```

There are two ways to add a new database to the CrocoBLAST index. In both cases, you should provide a simple name for new each database, so that you may later refer this database easily whenever you need to run a BLAST job.

Retrieve database from the NCBI servers

In the most typical scenario, you will use the established reference sequence databases maintained by NCBI (ftp://ftp.ncbi.nlm.nih.gov/refseq/release/). CrocoBLAST allows you to specify the name of such a database, and will download or update the database for you:

```
CrocoBLAST -add_database --ncbi_download ncbi_database_name output_folder
CrocoBLAST -update_ncbi_database ncbi_database_name output_folder
```

When adding or updating a database in this manner, you need not worry about the format of the database, as NCBI provides pre-formatted database files.

Add database from your computer

If you have already downloaded the databases from NCBI, or if you do not have internet connection, you may add to the CrocoBLAST index database files stored on your computer. Remember to provide a unique and representative name for each database you add, so that it is easy to call the databases later. If the database files are appropriately formatted (e.g., psq or nsq):

```
CrocoBLAST -add_database --formated_db nsq_database_file
CrocoBLAST -add_database --formated_db psq_database_file
```

If your database is in FASTA or FASTQ format, you will need to tell CrocoBLAST the type of **sequence** it will find in the database:

```
CrocoBLAST -add_database --sequence_file nucleotide fasta_file database_name output_folder
CrocoBLAST -add_database --sequence_file protein fasta_file database_name output_folder
CrocoBLAST -add_database --sequence_file nucleotide fastq_file database_name output_folder
CrocoBLAST -add_database --sequence_file protein fastq_file database_name output_folder
```

Manage CrocoBLAST queue

The efficiency of CrocoBLAST lies in its ability to parallelize the execution of your BLAST jobs. This is related to breaking each big calculation into smaller pieces, and then organizing the execution of the pieces. Having smaller pieces means that you need less memory to run each job, and if you can analyze several pieces at once you can speed up the total calculation time. CrocoBLAST takes care of these things for you.

Execution

Say you have *created one or more BLAST jobs* and are ready to start munching some sequences. It's easy:

```
CrocoBLAST -run
```

This tells CrocoBLAST to take the input file, break it into little fragments, and submit each fragment for sequence alignment as soon as a core becomes free. This means that, if your computer has only one core, the alignment will start only after fragmentation of the input file is complete. However, if your computer has two cores (or one core that supports multi-threading), the alignment will start as soon as at least one fragment has been generated, which means immediately. The alignment of each fragment runs as an independent thread. The more threads you can run simultaneously, the faster your job will finish. This depends on the number and type of cores your computer has.

When you run CrocoBLAST without any additional options, you will make the most efficient use of your computational resources, as CrocoBLAST will figure out how to best parallelize the calculation on your machine. Nonetheless, if you want to limit the number of threads running simultaneously, you may do so:

```
CrocoBLAST -run --num_threads number_of_threads
```

Similarly, you can easily stop or pause the execution at any time. The difference between *pause* and *stop* rests with how long you are willing to wait before your computational resources become available, and how much partial output you need. To immediately kill a CrocoBLAST job and free up the memory and cores:

```
CrocoBLAST -stop
```

On the other hand, if you are more interested in the output:

```
CrocoBLAST -pause
```

This lets CrocoBLAST know that no new threads should be initiated, and the output produced by each running thread will be incorporated in the partial results as soon as the thread finishes. Therefore, you will have to wait until all running threads have completed. Depending on the type of BLAST program you are running, the size of the database, and the similarity between your input sequences and the sequences in the database, you may have to wait a considerable amount of time. However, this will ensure that you can resume the calculation at a later time. To resume, simply tell CrocoBLAST to start munching.

```
CrocoBLAST -run
```

It will automatically detect the current state of each job in the queue, and continue from where it left off, unless you have made changes to the queue in the meantime. While CrocoBLAST operates with the concept of queue, it is important to note that only one job is active at any given time. You can check the current state of the CrocoBLAST queue:

```
CrocoBLAST -status
```

This will provide you with information regarding which jobs are queued, with full details regarding the job ID and BLAST setup, as well as a description about the progress of the alignment. The progress of each job is described in three main directions: fragmentation of the input file, alignment, and assembly of results.

Administration

If you want to change anything about the queue (say, pause one job and start another, or change the order of the jobs in a queue), you need to first pause or stop the current run. Subsequently, you may perform operations like adding, removing, or reordering jobs in the queue:

```
CrocoBLAST -add_to_queue blast_program database input_file output_folder
CrocoBLAST -remove_from_queue job_id
CrocoBLAST -remove_from_queue job_id_1 job_id_2 ...
CrocoBLAST -move_top_queue job_id
CrocoBLAST -move_top_queue job_id_pos_1 job_id_pos_2 ...
```

Note that, once a job is added to the queue, you may perform operations with it (remove, reorder) if you refer to the job by its job ID. You can obtain the job IDs by checking the current state of the queue:

```
CrocoBLAST -status
```

Technical details

[Collapse]

CrocoBLAST is free to use within the conditions of the licence, and has been available for download since July 2016 at http://webchem.ncbr.muni.cz/CrocoBLAST. There is no login requirement for downloading or running **CrocoBLAST**.

Software requirements

CrocoBLAST runs on Windows and Linux, and all results are provided in the typical BLAST output. Once you download the .zip archive with all necessary **CrocoBLAST** files, you will need a program to unpack the archive. Such a program (e.g., unzip, 7zip, etc.) will likely already be installed on your computer, as unzipping archives is a common procedure. Obviously, you will need to have BLAST available on your computer before you can run CrocoBLAST. If you don't already have BLAST, please get it from the NCBI website (https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastDocs&DOC_TYPE=Download). No further requirements exist for running the command line utility of **CrocoBLAST**, or inspecting the results. The graphical user interface requires Java, which is likely already installed on your computer. If not, please visit https://java.com/en/download/.

Hardware requirements

Because of the nature of the data being processed, it is better if your computer has at least 200 MB of RAM per core. Nonetheless, it is possible to run CrocoBLAST on big data files even if less memory is available, but you will need to specify this fact during job submission. Furthermore, if you need to analyze NGS data,the input and output files involved in such calculations can be quite large, and therefore you will need to have sufficient space on your hard disk. NCBI databases range from 10 MB to 500 GB (whole genomes). Depending on the type of sequencing experiment you ran, your input files may range from a few kB to 100 GB. If you don't specify any BLAST options, the size of the output file may be up to 1300 times the size of the input file. Nevertheless, in the typical use case, requesting relevant BLAST options (e.g., provide only the first 20 hits) will greatly reduce the size of the output file.

Limitations

As mentioned above, all results are provided in the typical BLAST output, which is a text file. CrocoBLAST does not currently offer facilities for graphical visualization and analysis of the BLAST results, partly due to the fact that it targets big data, and partly because many great tools are already available for such purposes. We recommend that you obtain the alignments using CrocoBLAST, and then select the best alignments to be viewed and analyzed in some specialized software (e.g., MEGAN (http://ab.inf.uni-tuebingen.de/software/megan6/)).

CrocoBLAST currently does not implement a parallelization of the BLAST calculation via the network. This aspect may be addressed in a future version of CrocoBLAST, once we have gathered sufficient information regarding the most common use case for network-distributed calculations. Your feedback is greatly valued.

Finally, while CrocoBLAST will run on most versions of Windows (XP or newer) and Linux, CrocoBLAST will not run on OS X. It is unlikely that this should change in the immediate future, but do check back with us just in case.

Troubleshooting

CrocoBLAST typically checks that the necessary files and permissions exist before starting the demanding BLAST calculation. Furthermore, a specific CrocoBLAST function is available for fixing data consistency errors automatically:

```
CrocoBLAST -repair_inconsistencies
```

If you get into trouble while trying to run CrocoBLAST, please check the error messages, which are quite informative and should help you overcome the most common issues you are likely to encounter. If you experience further issues, please contact us and describe the problem in detail.

Retrieved from "http://webchem.ncbr.muni.cz/w/index.php?title=CrocoBLAST:UserManual&oldid=2204"

- This page was last modified on 25 July 2016, at 14:43.