GenSeed

Quick Guide

GenSeed – Quick Guide

1. System requirements

GenSeed is a Perl script designed for Linux machines. The software requires PERL::GD library to generate a graphical output. This library can be downloaded from http://www.cpan.org/.

GenSeed also requires the following software:

- **BLAST package.** BLAST can be downloaded from ftp://ftp.ncbi.nih.gov/blast/executables/release/. GenSeed uses the following executables from the package: `formatdb`, `blastall`, `fastacmd` and `bl2seq`.
- GenSeed uses **Cross_match**, only if vector masking is required. GenSeed generates `*.ace` assembly files that can be easily inspected using **Consed**. Cross_match and Consed are components of the Phred/Phrap/Consed package. Instructions for obtaining this package are provided at http://www.phrap.org/phredphrapconsed.html.
- **CAP3.** CAP3 assembly program can be downloaded from http://seq.cs.iastate.edu/.
- **HTML browser.** GenSeed produces text and graphical output files in HTML format. These files can be inspected in any HTML browser such as Mozilla Firefox or Internet Explorer.

2. Algorithm and implementation

GenSeed is a program for the recursive reconstruction of sequences flanking the ends of a starting sequence, referred from now on as the seed sequence. The principle of the program is based on the use of a seed sequence in a similarity search to select reads from a database that presents overlaps to the ends. These reads are then assembled together with the seed itself, resulting in a longer consensus sequence. The ends of this longer sequence are in turn used in a subsequent walking round, and the process is iterated a number of times, resulting in a progressive extension of the original seed sequence. GenSeed accepts either nucleic acid or protein sequences in FASTA format as seeds (Figure 1A). In principle, there is no restriction to the seed length, and we have succeeded to reconstruct sequences starting from 17-base SAGE tags and from 8-residue proteomic data. However, whenever possible, we suggest using a minimum of 50 bp and 15 amino acid residues for nucleic acid and protein sequences, respectively. The seed sequences must not contain low-complexity regions, as they could recruit physically unrelated sequences. Any set of multiple nucleotide sequences in FASTA format can be used to compose a database, with no trace files or quality values being required or used (Figure 1B). GenSeed invokes `formatdb`, a program from BLAST package that generates index files from the database. A nucleotide sequence seed is then submitted to a BLASTN similarity search against the database (Figure 1C). In case of a protein seed, GenSeed automatically recognizes its character and runs a TBLASTN search against the database. The positive reads are then retrieved using `fastacmd`, another BLAST package program that fetches sequences in a very rapid way using the database index files. The sequences can then be masked against a database of vectors using `Cross_match` (Phil Green, unpublished) and finally be assembled with `CAP3`. The assembly output files are checked to identify which contig contains the original seed sequence, and the consensus is stored. In order to save computational resources, GenSeed does not use the whole consensus sequence as a seed for the next round. Instead, the user can define the size of the sequence ends that will be used as seeds for the subsequent round. From the second round on, the assemblies are carried out using all positive hits of that round, plus the whole consensus sequence generated in the previous one.

An additional procedure is executed to avoid the incorporation of potential chimeric sequences in each round. All positive reads selected to extend a sequence end are submitted to a similarity search against the whole consensus sequence of the previous round. A read is considered a potential chimera if the alignment does not extend either to the end of the consensus sequence or to the end of the read itself. The assembly then proceeds with the reads that fulfilled the criteria depicted above. All positive reads are fetched and assembled, and the resulting consensus is then used in the further cycles, as described for nucleotide sequences. The iterative process (Figure 1C) is concluded when a user-defined (1) number of rounds or (2) sequence length have been attained, or until the exhaustion of the process, as detected either by the (3) absence of further extending reads in the database, or by (4) no increase of the sequence length in two successive rounds. In any case, GenSeed executes a final assembly step (Figure 1D), where all reads recruited during the several cycles are used, together with the consensus sequence generated in the last round. At the end of the process, a final reconstructed sequence is stored. Also, some files of the intermediate steps can be stored, including the BLAST output files, CAP3 assembly files and the consensi of all rounds. Finally, GenSeed produces three report files in HTML format, which list all reads incorporated in each round, the corresponding consensus sequences, and display a graphical output that shows a schematic view of the progressive sequence reconstruction.

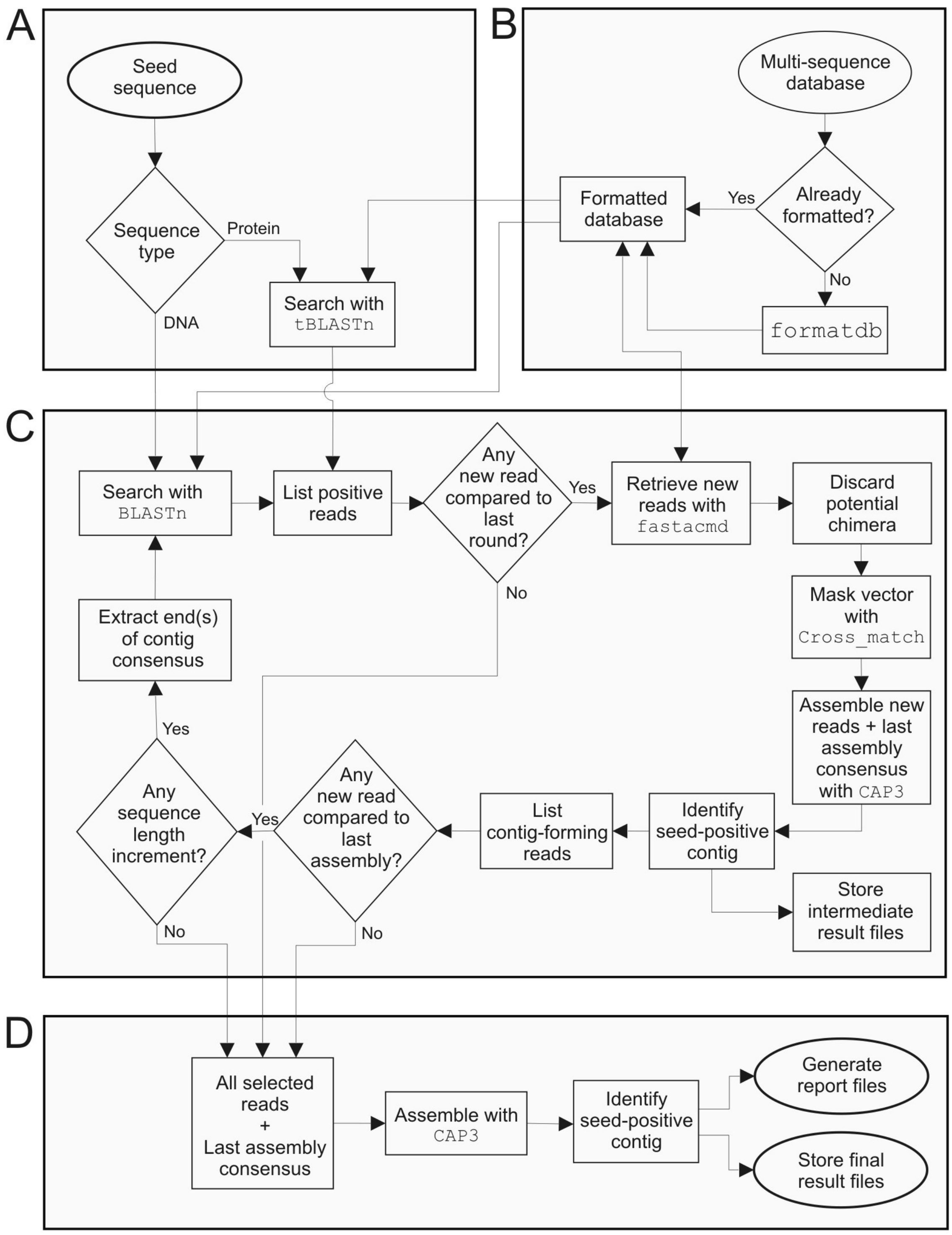
A
Seed sequence
Sequence type
Protein
DNA
Search with tBLASTn
B
Multi-sequence database
Already formatted?
Yes
No
Formatted database
formatdb
C
Search with BLASTn
List positive reads
Any new read compared to last round?
Yes
No
Retrieve new reads with fastacmd
Discard potential chimera
Mask vector with Cross_match
Assemble new reads + last assembly consensus with CAP3
Identify seed-positive contig
Store intermediate result files
List contig-forming reads
Any new read compared to last assembly?
Yes
No
Any sequence length increment?
Yes
No
Extract end(s) of contig consensus
D
All selected reads + Last assembly consensus
Assemble with CAP3
Identify seed-positive contig
Generate report files
Store final result files

Fig. 1. Workflow of GenSeed algorithm. GenSeed automatically detects the type of biological sequence (A) used as a seed and (B) formats, if necessary, a database FASTA file to generate BLAST index files. The seed sequence is introduced into a seed-driven assembly iterative process (C), which shortly consists in a similarity search, sequence retrieval and assembly. The walking process is concluded when a user-defined number of rounds or sequence length have been attained, or alternatively, when GenSeed detects no additional increase of the sequence length or incorporation of novel reads in comparison to the previous round. All reads selected during the walking cycles are then recruited and assembled (D).

3. The database

GenSeed requires a database of DNA sequences, which will be recursively used to select reads for the progressive seed-driven assembly process. The database must follow the multiple sequence FASTA format. Shortly, all sequences are composed by a header starting with a "greater than" sign (>), followed by a short name and/or description. The header is separated from the sequence by a "carriage return" (end-of-line) character. The so called multiFASTA format is composed by multiple sequences in a concatenated version of the FASTA format.

Example:

```
>My_sequence_002
ATGACGATGTAGCGAGCAGTCGATGCTCTCTTAAAAAGGTGTGCATCATG
>My_sequence_003
GGGGTTTTAGGCACGCACGACGGCAAAACGTGCGTGGTGGAATCGTACGT
```

In order to retrieve subsets of specific sequences in a rapid and efficient way, GenSeed uses NCBI's `formatdb` program to create BLAST index files, and `fastacmd` to fetch the sequences based on a list. In order to correctly identify and retrieve the sequences from the database, `fastacmd` has some constraints on the header content and identifier format. Thus, no duplicate names are accepted, and each sequence must have a unique identifier. Also, some public databases use a nomenclature composed by the organism name followed by an identifier. Some examples of sequence identifier formats that are accepted by `fastcmd` are provided below:

```
>TnOp07401E02.g
Ggtagtttttccatgccatttggtcagattttgacttatacttgagatcgcgccccgaaagatcctccagta
caaatttgccacaccattttcacaatgttc

>T62292 2006-10-02 EST dbEST
CAGCTTTGATTAAAGAAACTTTAAGTGTGTCTGTTTCTTTCACAACATTTTTTTGAGTGG
GACGAGACTGAAGGTCCGTACGCCGCTTGTGACCACACGCTTTTAATGAGTGAAAGCCTG
GGGCCGCGGAGCTCCTCCACTGTTGTCCTGA

>cam100a10.p1t
TTTTTTCAATGCTGTCCTCGAAAATGCTCGCTCTAGGGCTTATGGAATTTTAGCAAGTTG
ATTTTTAAAGCAA

>gnl|ti|1284152938 1041048645874
GTACGGTTGGAGGCTACAGTTGGCCCACATCGGGGGAAGCGACGTCGGCAGATGGTTCTCCAGAGCCTAC
TGCGGCGGTGAGAGTCCAGTG
```

```
>gi|115357965|gb|DV111293.1|DV111293   EST-MOA151H10   Toxoplasma   gondii
Tachyzoite  Library  Toxoplasma  gondii  cDNA  clone  MOA151H10  5',  mRNA
sequence
ATTCGGCACGAGGCTGCTCCGGTTCACGTATTTTGCCCTGAGCCAGCTTCCACCTCTTAGCACCGAGGCG
TTCCCACGGCCATGGCCGAGTACTCCTACGTGAAGTCCACCAAACTCGTGCTCAAGGGAACCAAG
```

However, sequence identifiers composed solely by numerical characters are not accepted, and must be converted to one of the formats displayed above.

Not accepted:

```
>115357965
ATTCGGCACGAGGCTGCTCCGGTTCACGTATTTTGCCCTGAGCCAGCTTCCACCTCTTAGCACCGAGGCG
TTCCCACGGCCATGGCCGAGTACTCCTACGTGAAGTCCACCAAACTCGTGCTCAAGGGAACCAAG
```

Accepted:

```
>115357965.
ATTCGGCACGAGGCTGCTCCGGTTCACGTATTTTGCCCTGAGCCAGCTTCCACCTCTTAGCACCGAGGCG
TTCCCACGGCCATGGCCGAGTACTCCTACGTGAAGTCCACCAAACTCGTGCTCAAGGGAACCAAG
```

Accepted:

```
>115357965A
ATTCGGCACGAGGCTGCTCCGGTTCACGTATTTTGCCCTGAGCCAGCTTCCACCTCTTAGCACCGAGGCG
TTCCCACGGCCATGGCCGAGTACTCCTACGTGAAGTCCACCAAACTCGTGCTCAAGGGAACCAAG
```

Also, `formatdb/fastacmd` only work with identifiers contained in a restricted number of text characters. Thus, if this restricted number of characters does not define a unique identifier, `fastacmd` will not be able to retrieve the sequence. For instance, the format below is not accepted:

```
>Toxoplasma_gondii|T62291|2006-10-02|EST|dbEST
GTGCCGTGCCAGCTTGGTGGTGGTATGGACCCGAGCGAAGCAGTGCATGCTTGTGACGAAAGGTACGGACCGATTCTGCT
TTGAGTGACTTGGAGCAAGCGTCAGACTAGGACCATCTGGTTACGTGGGGTGTTACCGTTGAAGAAATGGAGCGCGTTTC
TTCCAAGTGGGTTTTGCAAATGTTGTGTCGCTGCAGGGGCCAGGGGACGCCTGTCACGGTCCTCAAATATGCAGCATG
```

But if you remove the "`Toxoplasma gondii|`" string, then the database will work fine:

```
>T62291|2006-10-02|EST|dbEST
GTGCCGTGCCAGCTTGGTGGTGGTATGGACCCGAGCGAAGCAGTGCATGCTTGTGACGAAAGGTACGGACCGATTCTGCT
TTGAGTGACTTGGAGCAAGCGTCAGACTAGGACCATCTGGTTACGTGGGGTGTTACCGTTGAAGAAATGGAGCGCGTTTC
TTCCAAGTGGGTTTTGCAAATGTTGTGTCGCTGCAGGGGCCAGGGGACGCCTGTCACGGTCCTCAAATATGCAGCATG
```

Before using GenSeed, the user is advised to check if the database file is compliant with `fastacmd` sequence identifier constraints.

4. Getting started with GenSeed

4. 1. Mandatory arguments

To run GenSeed, you should type the following command:

```
genseed.pl -s <seed_input_file_name> -d <database_file_name>
```

Please note that `-s` and `-d` are the only mandatory arguments.

- `-s <seed_input_file>` - this is a text file in FASTA format that contains the seed sequence(s). Both nucleotide and protein sequences are accepted as input, and GenSeed automatically detects the type of sequence. GenSeed also accepts multiple seeds, which can be provided in a multiFASTA file. A complete path must be provided if the file is not located in the same directory where GenSeed is invoked from. Seeds containing both DNA and protein sequences in the same file are not accepted by GenSeed.

- `-d <database_file> -` this is a multiple nucleotide sequence file in a multiple sequence FASTA format. This file will be used as a database to select the reads presenting overlaps with the seed sequence(s). A complete path must be provided if the file is not located in the same directory where GenSeed is invoked from.

4.2. Help

- `-h`: prints a help screen

4.3. Changing parameters for output files

- `-o <output_directory_name> -` name of the output directory that will store GenSeed´s output files. If the output directory name is not specified, `genseed_dir#` will be used by default. If GenSeed determines that `genseed_dir1` already exists, it will create a directory named `genseed_dir2`, `genseed_dir3`, and so forth. However, if a user-defined directory name already exists, then the specified directory and respective files will be overwritten.

- `-x <option>` – defines the sets of output files that will be saved.
    - `simple` – saves the final assembled sequence(s), the corresponding CAP3 files, and the HTML files.
    - `complete` – saves all files above, plus the following files produced in each round: BLAST output files, seed sequences, CAP3 assembly files and assembled sequences.

GenSeed always generates the following output files:

- `genseed.log` – This is a log file that stores the parameters utilized to run GenSeed, the size of the contig (assembled sequence) obtained in each round, the number of reads used for the last assembly cycle, the size of the final contig(s), and a list of errors, if there were any, reported by the third-party programs.
- `report.html` – This file contains the information on the size of the contig(s), number of reads included in the final assembly step, and graphical outputs that display the assembly of each walking round. The graphic files are stored in the `image_dir` directory.
- `seed_contigs.html` – This file contains the consensus sequence(s) obtained in each round.
- `list_of_reads.html` – This file contains a list of names of the reads incorporated in each round.
- `final_contigs.fasta` – This file contains the final consensus assembled sequence(s) obtained by GenSeed.
- `CAP3_dir` – This directory contains CAP3 input and output files. If `-x` (`simple`) parameter is used, only the files of the last assembly round will be stored.

Additionally, if the option `-x` (`complete`) is used (default), the following directories and/or files will be stored:

- `BLAST_dir` – This directory contains BLAST output files produced in each assembly round. Files typically are named as `blast_1.out`, `blast_2.out`, etc.
- `fasta_dir` – This directory contains all seed sequences used as inputs for BLAST searches in all rounds. At the first round the user provides the seed sequence(s) to be used. After the first round, GenSeed automatically extracts the ends from the assembled

sequence and then use them as seeds in the subsequent round. Seed sequences are then named as `seed_1.fasta`, `seed_2.fasta`, etc. A second set of files is composed by the consensus sequences generated by CAP3 during the assembly phase of each round (`consensus_1.fasta`, `consensus_2.fasta`, etc.). A specific requirement is that each of these contigs must contain the consensus sequence of the previous round. This feature guarantees that the progressive assembly process will always keep the original seed sequence as part of the newly assembled sequence. Finally, a file named `final_reads.fasta` contains all read sequences selected during the whole set of cycling rounds, and that were used for the final assembly phase.

5. Fine tuning GenSeed with additional parameters

The whole workflow shortly described in Section 2 is highly configurable, and GenSeed provides many parameters to fine tune the progressive assembly process. We will now cover the optional parameters that allow adjusting GenSeed to the several possible applications.

5.1. Parameters to limit the walking process

- `-e <option>` – defines the expansion direction of the sequence in regard to the seed. NOTE: unidirectional expansion is only allowed for a single nucleic acid seed sequence. Multiple nucleic acids sequences and protein seeds cannot be extended unidirectionally.
  - `l` - expand only from the 5' end of the seed(s)
  - `r` - expand only from the 3' end of the seed(s)
  - `b` - expand from both ends of the seed(s) (default)
- `-f <integer>` – defines the maximum length (bp) of the final consensus sequence (default: no limit). If an assembled consensus sequence is longer than this value, then GenSeed will interrupt the walking process and proceed with the final assembly step. Please notice that the length of the final consensus will most of times be slightly longer than the `f` value, since this value is just a reference limit for GenSeed to interrupt the walking cycling.
- `-r <integer>` – defines the maximum number of assembly rounds (default: 100 rounds).

5.2. Defining the size of the intermediate seeds

To save computational resources, GenSeed does not use the whole consensus sequence of a given round as a seed for the next one. Instead, GenSeed extracts one or both end sequences of a user-defined length (parameter `-l`), and these sequences are then used as seeds in the subsequent assembly round. Once the overlapping reads are identified, CAP3 is used to assemble these reads together with the whole consensus sequence of the previous round. Only at the last assembly step, GenSeed uses the whole set of reads selected during all rounds. This approach allows GenSeed to perform much faster in the intermediate assembly steps. By using high `-l` values, BLAST tends to find more positive matches searches, as the query sequence is bigger, but at the cost of a slower performance. Also, because the number of recruited reads is usually higher, the assembly steps will perform slower. In general, if we reduce the `-l` value, the opposite effects will be noticed but, most of times, a higher number of rounds will be necessary to cover a given sequence length, since the sequence increment of each round is usually shorter.

- `-l <integer> –` Length of the sequence end(s) that will be used as seed sequences(s) in the next round (default: 500). NOTE: * (asterisk) characters, sometimes added by CAP3, are automatically removed from the sequence.

5.3. Defining the quality of the sequence ends to be used as seeds

GenSeed implements a routine to scan the ends of the assembled consensus sequence of each round to check the assembly confidence. GenSeed examines the `*.ace` (assembly) file created by CAP3 and utilizes a sliding window of a user-defined length (parameter `-q`) to scan the sequence, starting from both ends towards the central part. An end sequence is accepted when the region within the sliding window presents a percentage of bases (parameter `-m`) with quality values above a user-defined threshold value (parameter `-p`). Once the sliding window finds a region fulfilling these criteria, the scanning process ends. End sequences starting from high-quality regions, and presenting a user-defined length (parameter `-l`, see above), will then be used as seeds for the next walking cycle. Please notice that base qualities here are not related to the classical “Phred quality”, as we are not using PHD files. Every time CAP3 uses FASTA sequences with no quality values, it ascribes a flat value of 20 to all bases. However, during the assembly process, base redundancy may imply in a higher confidence, such that CAP3 may ascribe higher values to the consensus bases. Conversely, base discrepancies for the same position among different reads may decrease the confidence value. Therefore, "confidence values" must be distinguished from Phred values and only represent an indication of the level of redundancy and discrepancies across different reads of the assembly.

- `-q <integer>` – Length of the sliding window (default: 0)
- `-p <integer>` – Minimum quality value for the sliding window (default: 0)
- `-m <integer>` – Maximum percentage of bases within the sliding window presenting quality values below the minimum quality value (`-p`) (default: 10)

5.4. Changing the stringency of the alignment seed/reads

The seed-driven assembly process starts through a similarity search of the seed against a database in order to select the reads that will in turn be assembled to generate the seed of the next round. The reads are selected if the alignment e-value is lower (and therefore more significant) than the value defined by BLAST's `-e` parameter (see section 6.2). Hence, by default, the top scoring hits are always chosen, even if the absolute scores or e-value of the respective alignments are very poor. To avoid the reconstruction of unrelated sequences, the user can establish a priori criteria for the read(s) to be accepted. Thus, by using `-i 90` and `-j 90`, GenSeed will only select reads and proceed with the sequence reconstruction process if the seed has found matches in the database presenting alignment blocks with a length equal or higher than 90% of its own length (parameter `-i`). Also, as a second parameter, the read(s) will only be accepted if the alignment blocks present a percentage identity equal to or higher than 90% (parameter `-j`). If the top hit alignments do not fulfill these criteria, than GenSeed will return a message informing that no hit has been found in the database, and suggest using another seed. Please notice that these parameters are only functional for the first BLAST search. From the second round on, BLAST's `-e` parameter is the only criterion defining stringency.

- `-i <integer>` – Minimum length of the alignment block of the first BLAST search, in percentage relative of the length of the seed sequence (default: 0)
- `-j <integer>` – Minimum identity percentage of the alignment block of the first BLAST search (default: 0)

5.5. Avoiding the incorporation of chimeric reads

GenSeed tries to avoid the incorporation of chimeric reads in the newly growing sequence. This is particularly problematic when repetitive sequences are present. In each cycle, GenSeed takes the seed sequence and runs a BLAST search to select a new set of reads to be incorporated. Before proceeding with the assembly step, these reads are aligned with BLAST to the consensus sequence of the previous round. In case a read presents an alignment that does not extend either to the end of this consensus sequence or to the end of the read itself, it can be considered as a potential chimera. The parameter `-t` defines what is the minimum length of this misaligned region for a read to be considered as chimeric. If a read falls in this case, it will not be incorporated in the assembly phase and, as such, will not be used to extend chimeric sequences.

NOTE: In our experience, GenSeed does a reasonable job in discarding potentially chimeric sequences. However, if the original seed sequence itself presents a match to a chimeric read, then GenSeed will probably result in a wrong assembly. Choosing a seed from another region will solve the problem. Alternatively, manually identifying the potential chimeric read and removing it from the database can also solve the problem. This can be accomplished by BLAST searches followed by manual inspection, and also by looking at the assembly using Consed. The chimeric reads are usually at the end of the contig, and present matches to several reads in one end (the internal one in relation to the contig), but to no other read at the external end.

- `-t <integer>` – Minimum length of misaligned ends to discard potentially chimeric sequences (default: 75)

5.6. Driving final assembly with the last-round consensus sequence

During the intermediate walking cycles, GenSeed selects the reads and assemble them together with the whole consensus sequence of the previous round. Because GenSeed uses only the end sequences as seeds, the total number of reads used in each intermediate assembly is rather low, and the assembly steps perform in a relatively fast pace. Conversely, the last assembly step uses the whole set of reads selected during all rounds. The incorporation of a long consensus sequence, plus a high number of reads, causes a computational overload. Hence, if the consensus sequence is included as a template, CAP3 may require a much longer processing time (sometimes more than three times). On the other hand, deactivating this option may eventually lead to a final consensus sequence being divided into two or more distinct contigs. For this reason, we recommend, whenever possible, and despite the higher computational load, to maintain this option activated (`-g yes`).

- `-g <integer>` – Use the last assembly consensus sequence as a template in the final assembly step (default: yes)

5.7. Masking vector sequences

- `-v <vector_file_name>` – name of the vector database file to be used for masking purposes. A complete path to the file must be provided if the file is not located in the same directory where GenSeed is invoked from.

6. Parameters for third-party programs

IMPORTANT: the parameters below MUST be defined under quotes.

The parameters below will be incorporated as a string of characters in the command used to invoke third-party programs. For this reason, they must all be specified under quotes.

- `-a <option>` – CAP3 assembly parameters (default: none)
- `-b <option>` – BLAST parameters (default: `"-e 1e-06 -F F -b 500"`)
- `-c <option>` – Cross_match parameters for vector masking (default: `"-minmatch 12 -penalty -2 -minscore 20"`)

6.1. Some examples on how to use the third-party program parameters:

```
genseed.pl -s seed_file_name -d seed_data_base -a "-o 100 -p 95"
```

will result in CAP3 running as it were invoked with the command below:

```
cap3 input_file -o 100 -p 95
```

```
genseed.pl -s seed_file_name -d seed_data_base -c "-minmatch 12
-penalty -2 -minscore 20"
```

will result in Cross_match running as it were invoked with the command below:

```
cross_match    input.fasta    vector.fasta  -minmatch  12  -penalty  -2
-minscore 20
```

6.2. Some important considerations on BLAST parameters and their impact on GenSeed performance and efficiency

GenSeed uses as default the following BLAST parameters: `-e 1e-06,` `-F F` and `-b 500.` Let's understand each one of these parameters and how they can change the way how GenSeed runs.

Parameter `-b`

The parameter `-b` defines how many alignments will be displayed by BLAST in a similarity search. Since GenSeed obtains the list of positive matches from this output, using a low `-b` value implies that a low number of reads will be used in each assembly round. This can be advantageous if a high performance is more important than a high coverage. On the other hand, choosing a low `-b` value tends to increase the number of rounds necessary to complete the progressive assembly of a given sequence length. Conversely, choosing a high `-b` number may imply in a much better read coverage, provided that the database has a good overall coverage overall. Also, less walking rounds will probably be necessary to complete a job, but at the cost of loss of performance due to heavier loads in the assembly steps.

Parameter `-e`

The parameter –e defines what is the expectation value (e-value) cutoff for BLAST to display sequence alignments. The e-value of a given alignment block is the number of expected alignments that you should expect to occur by chance, given the database size you used, that would yield the same score as the one you have obtained. Because the e-value changes in accordance to the database size you use, there is no way to correctly choose one single "most appropriate" value. BLAST's default is 10, which is a very high and non-stringent value, meaning that 10 subject sequences are expected to produce a similar alignment by chance. GenSeed was implemented to use a default value of 1e-06 (one to one 1 million), so supposedly only highly significant alignments will be reported by BLAST. Since GenSeed uses BLAST's list of hits to compose the assembly, if this list comprises only highly significant matches, chances are that you will be using the most appropriate reads. This is just an assumption, since CAP3, the DNA assembler, will also use its own stringency parameters to select those reads that significantly presents overlaps with each other. This way, unrelated reads will theoretically not be assembled in the same contig, even if you use a low-stringency `-e` value. Probably, the most important advantage of being stringent (using a very low `-e` parameter) is to reduce to a minimum the total number of reads that will be used in each assembly round. Hence, the computational load will be also diminished (less reads will have to be compared to each other in the assembly process) and, therefore, an improvement of overall performance is expected to occur.

A caution note, however, must be presented. If you use very short seed sequences, especially protein sequences, the e-value of the alignment will be high (even higher than 1). Thus, if you use a too stringent `-e` value, the seed will not be able to progress, since all alignments' e-values will fall above defined number, hence no alignment will be reported by BLAST. In our experience, we have successfully reconstructed full-length cDNAs using protein seeds as short as 8-amino acid long. However, for these particular cases, we had to use `-e` values as high as 1000. To avoid the selection of spurious reads, we also use GenSeed parameters `-i 90` and `–j 90` (see section 5.4). As a rule-of-thumb, we recommend using relatively low values (1e-06) for sequences longer than 100 bp or 33 amino acid residues, and extremely high values (1000) for very short sequences, especially for proteomic data (peptides of 8 to 20 aa residues). For LongSAGE data composed by 17-bp tags, we have successfully reconstructed the corresponding cDNAs using BLAST parameter `-e 1`, and GenSeed parameters `-i 90` and `-j 90`.

Parameters –e versus –b

Since using a low `-e` value (high stringency) has a reducing effect on the number of reads to be reported by BLAST, we recommend a balance between `-e` and `-b` values. If you are using a very high –e value for proteomic data, then lowering `-b` value looks sensible. On the other hand, when using highly stringent conditions, `-b` value can be relaxed. GenSeed's default values are 1e-06 for `-e` and 500 for `-b`. It is noteworthy that even using high `-b` values, the total number of reads selected in each round can be very small, particularly if the database coverage is poor.

Parameter `-F`

The parameter `-F F` turns BLAST's low-complexity filter off. This filter is used by BLAST to mask off low-complexity regions, so they are not considered for sequence alignments. Depending on the sequence composition of your query, this default choice may be appropriate. However, if your target sequence (the one that will be used for walking) is very rich in low-complexity regions, it may be necessary to activate this filter. Since it is hard to predict if the filter will or not represent the best choice, we recommend using GenSeed with the filter turned off (default) and, if the assembly is not acceptable, trying to rerun GenSeed with the filter this time turned on.