Bridges User’s Manual

Description of the Program

Bridges is a heuristic search tool that uses short exact word matches to identify local similarities between sequences. It is written in the C programming language and can be run on all platforms. Any questions about the program should be directed to kondrash@umich.edu.

Input File Format

Bridges requires two files as input: a database and a query file. The database file can contain either a naked sequence or one in FASTA format. The same rules apply to the query file when there is only one query. Otherwise, the query file should be in FASTA format.

Output File Format

The main output file is named “Bridges.out”. The first four lines list the parameters chosen by the user. Next, the length of the database and fraction of nucleotides masked in the database are given. After this, the length, orientation, and fraction masked of a query are listed. All similarities pertaining to this query and orientation follow. If both orientations are searched, the reverse-complemented similarities will follow those on the direct strand. If there are multiple queries, they will be listed consecutively.

Similarities for each query are numbered consecutively. Coordinates and scores of similarities are given on the same line as the similarity number. First listed are the database start and stop positions, then the query start and stop positions, and then the score. If the similarity occurs on the reverse-complemented strand, the query start coordinate will be greater than the query stop coordinate. If the user chooses to print sequences, the next two lines will consist of the database and query sequences, respectively.

Filtered database and query sequences are also output as two separate files and are named "Seq1M.txt" and "Seq2M.txt", respectively. All masked nucleotides are printed in lowercase.

Description of Parameters

inp1: Database file path

inp2: Query file path

SeqL1Max: Total length of database sequence

SeqL2Max: Total length of query sequence(s)

MaxSimA: Maximum number of active similarities stored

- Increase if parameters are lax and you get a MaxSimA error message.

MaxSimF: Maximum number of finished similarities

- Increase if parameters are lax and you get a MaxSimF error message.

MaxWordDB: Maximum number of occurrences of a particular KS word in database sequence

- Increase if KS is small, SeqL1Max is large, and you get a MaxWordDB error message.

LowCase: Unmask (0) or keep masked (1) lowercase letters in input sequences

InvComp: Look for similarities on the direct (0), reverse-complemented (1), or both (2) strands of query sequence(s)

FullMatrix: Search only lower triangle (0) or full (1) alignment matrix

- Mode 0 should be used if the database and query sequences are identical.

KM: Length of words used for masking

- You may want to choose this so that an average word appears no more than ten times in a sequence (i.e., 4^{KM} > sequence_length/10).

FilterDBase: Maximum allowed occurrences of a KM word in the database sequence

- It usually makes sense to filter words that occur more than 100 times. However, the desired fraction of masked letters depends on the nature of a sequence, as well as on the specific goal of the user.

FilterQuery: Maximum allowed occurrences of a KM word in the query sequence(s)

- See FilterDBase above

KS: Length of words for searching

- When KS is large, Bridges runs faster, but sensitivity declines. Long segments with similarities >85% can be reliably found with KS = 12.

CoeffMis: Mismatch penalty

- This is not a very important parameter, but probably should be kept low.

CoeffGap: Gap penalty

- See CoeffMis above.

FlatGap: Gap length after which the penalty does not increase

- We use an affine gap penalty function, and FlatGap = 10 works well for most purposes.

MaxDist: Maximum distance, in ether sequence, between words that form a chain

- The longer MaxDist is, the laxer the search.

MinWeight: Minimum weight/score of a local similarity

- The lower MinWeight is, the laxer the search.

PostProcess: Skip (0) or enable (1) post-processing of local similarities

MaxOver: Maximum coverage, of either sequence, allowed under a similarity

- MaxOver = *n* means that all similarities reported will have a maximum copy number of *n*.

MinOver: Minimum coverage, of either sequence, allowed under a similarity

- MinOver = *n* means that all similarities reported will have a minimum copy number of *n*.

FractMaxOver: Maximum fraction of a similarity residing over maximum coverage regions

- This is the fraction of overlap used to decide if the copy number of a similarity is above MaxOver.

FractMinOver: Minimum fraction of a similarity residing over minimum coverage regions

- This is the fraction of overlap used to decide if the copy number of a similarity is below MinOver.

MaxDistPost: Maximal distance for merging local similarities

- This can be used to find long, perhaps distant, similarities.

CoeffMisPost: Gap penalty used for merging local similarities

- This should probably be kept low.

PrintSeq: Skip (0) or print (1) sequences of local similarities to output file

Memory Requirement

Filtering stage:

$2(4^{KM}) + \max(SeqL1Max, SeqL2Max)$

Similarity searching stage:

$4^{(KS+1)} + 5(SeqL1Max + SeqL2Max)$