Multisubjective Nucleic acid design tool

User documentation

Overview

Multisubjective is a nucleic acid design tool that identifies and removes strong spurious base pairing in arbitrary user-defined systems, with special features targeted towards dynamic strand-displacement systems used in developmental self-assembly. Multisubjective identifies the small subset of bases responsible for the spurious base pairing, and uses a client designer to select new identities for those bases.

Multisubjective's algorithm is based on multiple iterations of an analysis–redesign cycle with the goal of finding a design that has very few strong undesired base pairs. Multisubjective uses a thermodynamic analysis to identify a minimal set of bases that need to be changed because they are responsible for undesired secondary structure, or because they are part of a user-defined prevented sequence. This analysis is passed to the client designer to redesign only those bases identified as problematic by Multisubjective, holding the others constant. Multisubjective interacts automatically with an implementation of the combinatorial heuristics-based Domain Design (DD) software, and is also capable of automatically submitting a design to the NUPACK multi-objective designer on their web server and automatically retrieving the results.

Installation

In all cases:
- Install NUPACK 3.0.4 (http://www.nupack.org/).
- Obtain the file multisubjective.cpp (v1.1.3) and compile it using Xcode, gpp, or another compliant compiler.

If automatic interaction with the NUPACK web server is to be used:
- Install cURL 7.21.6 (http://curl.haxx.se/).

If automatic interaction with DD is to be used:
- Install Node.js 0.6.12 (http://nodejs.org/).
- Create the directory $HOME/Documents/Multisubjective , and run the command "npm install commander underscore format" in that directory.
- Obtain the files dd.js (v0.3) and cldd.js (v0.4.7.3) and place them in that directory. Set cldd.js to be executable.

Input

Multisubjective requires a specification file that is an augmented version of the NUPACK ".np" specification, and optionally a file containing the base sequences for the system. These should be placed in the Multisubjective working directory, which can be specified within the program interface. The shown filenames are the defaults, but most of them can be changed through the user interface or the command line call.

specification.np: Augmented NUPACK multi-objective specification

The file *specification.np* contains an augmented from of the NUPACK multi-objective design specification for the structure. The file must also contain extra lines needed by Multisubjective that relate to the function of the strands within the developmental hairpin system. The syntax is identical to the NUPACK syntax with different keywords. The format of this file is explained in a later section in this documentation.

candidate.npo, candidate.dd, or candidate-N.dd: Strand or block sequences

Multisubjective loads the actual sequences in one of two formats. The file *candidate.npo* contains strand sequences, and can be automatically obtained from the NUPACK web server. (Note that Multisubjective will overwrite this file if a new job is requested.) The file *candidate.dd* contains block sequences, and is used if Multisubjective is told to load data in DD format. If multiple DD files are to be run, the series of ten files *candidate-1.dd* to *candidate-10.dd* are used. Multisubjective can also create random sequences according to constrains in the specification file, in which case no sequence input file is needed.

Output

Multisubjective outputs data into three files, with an extra two generated if autosubmission to the NUPACK web server is utilized. These are all placed into the Multisubjective working directory specified in the program interface. If Multisubjective is set to run in a loop, the files *final.dd* and *final.msq* will be generated, containing a copy of the best candidate sequence from the final iteration of the analysis–redesign loop.

analysis.dd and analysis.msq: New block sequences

The file *analysis.msq* contains the sequences of all the blocks, including mixed bases as determined by the Multisubjective algorithm, in the NUPACK "domain=" format. These can be copy-pasted into a NUPACK multi-objective design specification for manual input to the web server. The file *analysis.dd* contains the same sequences in DD format, but with each mixed base set to a random base consistent with that mixed base, with all the non-mixed bases set to be immutable by DD, and with each domain's sequence constraint set to the union of all the mixed bases in that domain.

analysis.log and favorites.txt: Internal data log

The file *analysis.log* contains a mirror of internal data stored after Multisubjective has finished loading and processing the three input files, as well as a record of Multisubjective's secondary structure analysis. It is useful for debugging purposes. The file *favorites.txt* contains information about the best candidate sequence for each round, and is useful in monitoring the progress of the designs.

analysis.post and response.html: NUPACK web server autosubmission data

If automatic submission to the NUPACK web server is utilized, Multisubjective will generate two additional files. The file *analysis.post* contains a machine-readable version of the full NUPACK multi-objective design file including the new block sequences with mixed bases, used in the HTTP POST request to the NUPACK webserver. The file *response.html* contains the web server's response to the POST request. If successful, the file contains the URL of the newly created job, and a subsequent run of Multisubjective can use this file to autofill the job number and token, to simplify the design process.

Program operation

Upon starting, the program gives the user a choice of input modes:

- Load a DD file (d). The program loads sequence data from the file *candidate.dd* . Multisubjective automatically converts the block sequences contained in the DD file into the strand sequences used by Multisubjective.

- Load a NUPACK-MO file (n). The program loads sequence data from the file *candidate.npo* .

- Load multiple DD files (m). The analysis will run ten times using the files *candidate-1.dd* to *candidate-10.dd* . This option is useful for quickly comparing several related DD outputs and determining which is optimal.

- Autofill from last MO web submission (a). The program uses the job number and token from the job most recently submitted by Multisubjective to the NUPACK web server. This is useful when iterating sequences between Multisubjective and NUPACK multi-objective design.

- Fill with random bases (f). The program creates ten sets of random block sequences consistent with the base constraints in the specification file, and uses these to construct ten sets of strand sequences. This option should be used if sequences are to be directly loaded from the specification file, even if none of them are degenerate. (Note that the constraints will not be honored in subsequent rounds unless they are set as immutable bases.)

- Seed with independent DD trials (i). DD is run ten times from ten independent seeds consistent with the base constraints in the specification file, and the resulting ten sets of sequences are used as the initial candidate sequences. (Note that the constraints may not be honored by DD and will not be honored by Multisubjective in subsequent rounds unless they are set as immutable bases.)

- Input by job number (j). If the user inputs a number, the program checks whether the job has already been downloaded within the current working directory. If not, the program asks for a token, uses cURL to download the job from the NUPACK web server to the file *mo-output.zip*, and unzips the job into the folder */mo_output* . It then asks for the trial id and copies the requested sequence file to *candidate.npo*, overwriting a previous file if necessary. (The keyword "all" can be used in place of a trial id to cause Multisubjective to test trials 0–9 of in that job.) If the job has already been downloaded, it asks only for a trial id, and copies the desired file from */mo_output* to *candidate.npo* .

- Set other options (s). A submenu allows the user to change the Multisubjective working directory (where the input and output files are stored), the local NUPACK home directory, the filenames of the two specification files, and the filename prefixes of the sequence input and output files. These can be entered manually, or a filename can be provided that should contain these data each on a single line. These settings are stored in the default configuration file, and automatically reloaded on subsequent runs of Multisubjective.

The user also has a choice of iteration modes to specify what designer is to be used to select new identities for those bases that were identified by Multisubjective as problematic. In all cases, the sequence given to the designer has only those bases unlocked that were identified by Multisubjective as problematic. If multiple sequences are tested by Multisubjective, only the one with the fewest number of undesired bases is passed to the designer. The iteration modes are:

- Run DD once (o). The program runs one round of DD, yielding ten new designs.

- Run DD in loop (l). Ten rounds of DD are run, with ten new designs generated in each round with only the one with the fewest number of undesired bases advancing to the next round.

- Submit to NUPACK-MO web server (w). Multisubjective generates a formatted version of the NUPACK multi-objective design file, with the new block sequence assignments inserted in place of the grave character (`) delineated block in the *specification.np* input file, and replacing exclamation point characters (!) with hash characters (#) to comment out the Multisubjective-defined lines. Note that the submitted request copies the contents of *specification.np* verbatim, and it is recommended to verify that the file contains no errors beforehand. Multisubjective then uses cURL to submit an HTTP POST request to the NUPACK web server.

- Use random bases in loop (r). Ten rounds are run, with ten new random designs generated in each round (within the prevented sequence constraints) with only the one with the fewest number of undesired bases advancing to the next round.

- No designer (x). No designer is used after Multisubjective analyzes the input design for undesired secondary structure.

After data have been loaded from the input files, Multisubjective uses the local installation of NUPACK to analyze the secondary structure of the strands. This is done in (up to) three passes. The first pass contains the full sequence of each hairpin, testing the "closed" hairpin structures. The second pass includes the hairpins with their input domains removed, giving the "open" hairpin structures. If there are bridge structures defined, and intermolecular analysis is turned on, the open hairpin structures involved in the bridges are split into their own pass without intermolecular interactions. All NUPACK analysis files are stored in the directory */nupack* .

Multisubjective then tabulates the undesired secondary structure from the NUPACK analysis, and decides which bases need to be changed to disrupt this secondary structure, avoiding changing the desired bases in the hairpin and bridge complexes, as well as the immutable bases calculated from user input. All such bases are changed to N. It also checks for the existence of prevented sequences, and inserts the proper mixed base to disrupt this sequence (e.g., if there are too many A's in a row, one of them will be changed to a B; if there are too many W's in a row, one will be changed to an S), again avoiding changing immutable bases. The algorithm seeks to minimize the number of changed bases by changing an N to the mixed base if one is in the proper range of positions. If the presence of immutable bases prevents any of these changes from being made, a warning is output to the screen.

Once these changes have been made to the strand sequences, the block sequences must be extracted for output. Since each block may appear many times throughout the system, and each instance may have been changed in different ways, a base collision scheme is used to resolve these differences. Mixed bases other than N take precedence over N's, N's take precedence over single bases, and for two non-N bases, the intersection of the two bases is taken unless the intersection is empty, in which case the earlier base is used and a warning is output to the screen.

Once the strand sequences have been compiled, they are written to the output files. If appropriate, the output files are then sent to the designer chosen by the user. If Multisubjective is to be run in a loop, the input mode is reset to 'm' and the program then loops ten times, and then prompts the user to run more loops in blocks of five. The pair probability threshold may be changed after each block of rounds. After all desired loops are complete, Multisubjective cheerfully exits.

4

Command line operation

If Multisubjective is called with any command line arguments, the user interface will be skipped and information will be taken from the arguments. The syntax is:

multisubjective -m mode [-j job_number] [-r trial_id] [-k token] [-c config_filename] [-d working_dir]
[-i specification_infile_prefix] [-s sequence_infile_prefix] [-o outfile_prefix] [-h NUPACK_homedir] [-w]

where:
- The argument *mode* is a two-letter code specifying the input and iteration modes. The first letter is one of the letters 'd', 'm', 'f', 'i', 'n', 'a', or 'j', representing the input mode as explained above. The second letter is one of the letters 'o', 'l', 'w', 'r', or 'x', representing the iteration mode as explained above. This argument is required. If option 'j' is used, the arguments *job_number* and *trial_id* are always required, and the argument *token* is also required if the job is not available locally.

- The argument *config_filename* is the filename including full path of a configuration file. If this argument is used, the options *-d, -i, -s, -o,* and *-h* should be omitted as these data are being loaded from the configuration file instead.

- The other command line arguments correspond to items listed in "Format of the configuration file" below; the filenames should not include a path as *working_dir* is used. If any or all of these are omitted and *-c config_filename* is not used, the omitted data will be loaded from the Multisubjective default configuration file if it exists, otherwise the default values will be used.

- The option *–w* enables Workbench mode. This generates the extra file *analysis.mso,* which contains Multisubjective's analysis data in JSON format.

Format of the *specification.np* input file

The file *specification.np* contains the NUPACK multi-objective design specification for the structure. Multisubjective accepts both the old and new NUPACK formats. Multisubjective only uses the following lines: "structure", "strand", ".seq", and the reaction condition lines; any other lines used by NUPACK may be present and will be ignored by Multisubjective. Structure inputs in DU notation are preferred; dot-paren notation is also supported, with a slightly modified format in some specific cases. (For dot-paren structures that have two stems immediately adjacent to each other, i.e., there are no unpaired bases between them, a colon character (:) must be inserted to separate the sets of parentheses from each stem. If the colon character is omitted, an error will usually result, but there are some structures that will be translated into an incorrect structure without causing an error.) Multisubjective ignores strand breaks ('+' characters) and concatenates each structure into a single strand.

If submission to the NUPACK web server is to be used, the "domain" group of lines must be bracketed with two lines each containing a single grave character as a comment (#`); alternatively, the entire group of lines may be omitted and replaced with a line containing two grave characters in a comment (#``). The grave characters tell Multisubjective where to insert the new sequences in the HTTP POST request after they have been generated. The old block of assignments, if present, is discarded by Multisubjective.

The file *specification.np* must also contain data needed by Multisubjective that is absent in the NUPACK specification. Most of these data relate to the function of the strands within the developmental

hairpin system. The format of this file is explained in a later section in this documentation. The syntax is identical to the NUPACK syntax with different keywords, all of which begin with an exclamation point character (!). Note that **two** hash characters are required to comment out a line with an exclamation point (##!); this is so that using one (#!) will cause NUPACK to ignore the line while Multisubjective recognizes it, allowing the same specification file to be used for both platforms.

In most cases there are two varieties of specification for each type of strand function. The *automatic* notation causes Multisubjective to calculate some of the needed information based upon the user specification and information in *specification.np*, while in the *explicit* notation the user directly supplies the result of these calculations. The automatic notation contains a colon (:) as its operator, while the explicit notation contains an equals sign (=). The automatic notation is easier to use, but the explicit notation is useful for non-standard structures and in troubleshooting.

Static structures

```
!static [name]
```

Structures declared in this way are included in the "closed" analysis pass, but are omitted from the "open" analysis pass. This is useful for analyzing strand that are not dynamic, or for directly analyzing strands whose configurational changes are too complex for Multisubjective's preprogrammed strand types; the closed and open structures can be individually passed to Multisubjective using this notation. Note that "static" is the default setting for structures defined in the regular ".np" format but not accounted for with a strand function statement.

Hairpins

```
!hairpin [name] : + / -
!hairpin [name] = [offset]
```

A hairpin is specified using the syntax "hairpin A1 : +" where A1 is the name of the strand (which must correspond to the name used in the NUPACK specification in *specification.np*), and one of the characters '+' or '-' to specify the polarity of the hairpin. A hairpin has positive polarity if its toehold is at the 5' end of the strand, and has a negative polarity if its toehold is at the 3' end.

The explicit syntax instead specifies a number called the *offset*, which tells Multisubjective what portion of the hairpin sequence to remove to obtain the open hairpin sequence. This has the same sign as the polarity, and its magnitude is either the length of the input port (if the polarity is positive), or the length of the open portion (if the polarity is negative).

Cooperative hybridization complexes

```
!coop [name] = [size]
```

The *size* of a cooperative complex is the length of the longer strand in the complex. In the structural specification in *specification.np*, the longer strand should be listed first, i.e., it should have a form like "structure C1 = U8 D37 (U8 +)".

Bridge interactions

```
!bridge [name1] [name2]
```

Bridge interactions involve two separate complementary regions that are independently revealed on different hairpins and then hybridize in a separate step once both are present within the same complex.

Because bridge regions necessarily involve large regions of complementarity between open hairpins, Multisubjective needs to know that these interactions are desired so that it can avoid handling them as spurious base pairing. This statement has the effect of removing both named structures from the intermolecular interaction pool and analyzing their intramolecular interactions only.

Specifying immutable bases

```
!immutable = off / auto / blocks [blocknames]
```

Multisubjective requires the location of bases desired to be immutable. The "off" option is the default and will result in no bases being immutable. The "auto" option will cause immutable bases to be set using the "domain=" lines, where each position with an unambiguous base (A, C, G, or T) will be set as immutable. (Note that if a sequence is loaded from a file, the locations of the immutable bases will be preserved, but the base identities will be overwritten.) The "blocks" option is followed by a list of block names for which all bases within those blocks will be set to be immutable. Specifying immutable bases is useful in the case of endonuclease restriction sites, or for pre-specifying sequences of clamping regions.

Specifying the pair probability thresholds

```
!threshold = [value] / worst [N]
!toethreshold = [value]
!singlethreshold = [value]
!intermolecular = on / off
```

The threshold can be specified here, as well as an optional lower threshold for base pairs involving an exposed toehold. If either of these lines is absent, the default value of 0.67 is used for that threshold. The keyword "worst" followed by an integer N causes the threshold to be automatically set to the pair probability of the Nth worst undesired base pair for each candidate sequence. There is also an optional threshold for bases desired to be unpaired, which defaults to 1 (which has no effect). Lower values for these parameters are more stringent. The "intermolecular" line specifies whether intermolecular interactions between open hairpins are to be considered; this increases the quality of the sequences but is much slower.

Specifying the prevented sequence limits

```
!prevent [base] = [value]
```

The specified base, which may be a mixed base, is prevented from appearing in consecutive repeats of the given value. The default values are: 4 for A, C, G, and T; 6 for S, W, R, Y, K, and M; and 100 for B, D, H, and V. Note that X and N are not valid arguments for this statement.

Format of the configuration file

Configuration files contain the following information, each item on a separate line:

- The Multisubjective working directory, where input and output files are stored
- The filename prefix, with an extension, of the specification input files (the default is *spec*)
- The filename prefix, without an extension, of the sequence input file (the default is *candidate*)
- The filename prefix, without an extension, of the output files (the default is *analysis*)
- Optionally, the NUPACK home directory preset, which is used only if $NUPACKHOME is not set

The default configuration file is *$HOME/Documents/Multisubjective/multisubjective.cfg* . If it exists, it is loaded at the beginning of the program. A new copy is saved after the user inputs new data through the user interface, but not if these options are changed through the command line. Deleting *multisubjective.cfg* will cause the Multisubjective's original default values to be restored.

Exception handling, signal handling, and exit status

If an error condition occurs during the program's execution, an exception will be thrown with a unique identifying number in the range 1–150. A short descriptive message containing the exception number will be output to the screen, and the exception number will be noted at the end of the *analysis.log* file. Signals indicating a program error when encountered result in exceptions.

Upon successful termination, Multisubjective will have exit status 0. If an exception was thrown, the exit status will be equal to the number of the exception if it is in the range 1–250, otherwise the exit status will be 255. If the program terminated because it received the signals SIGINT, SIGHUP, SIGTERM, or SIGQUIT, the exit status will be one of the numbers 251–254, respectively.