Multisubjective Nucleic acid design assistant v1.0b6/2012-06-19

by John P. Sadowski

Overview

Multisubjective improves the quality of sequence designs for nucleic acid hybridization chain reaction (HCR) systems and other systems used in DNA nanotechnology, and automates several tasks in the process of sequence design. Multisubjective can work in concert with another designer, such as the NUPACK multiobjective designer or Dave Zhang's Domain Design (DD), to minimize undesired secondary structure in these systems.

It has been observed that the NUPACK multiobjective designer does a poor job of avoiding undesired secondary structure in regions intended to be unpaired, when used to design larger systems (~9 hairpins). Multisubjective identifies a minimal set of specific bases which are responsible for undesired secondary structure, and instructs the designer to redesign only those bases, holding the others constant. Multisubjective also enforces exclusion of sequences consisting of a certain number of a single base type in a row, including mixed bases such as S or W.

Multisubjective also streamlines the design process by interacting directly with the NUPACK web server's multiobjective design software. Multisubjective is capable of automatically downloading and processing NUPACK job results, and of automatically submitting new design jobs to the web server. Multisubjective is also capable of automatic interaction with DD, as it can read and write files in the DD format.

Installation

In all cases:

- Install NUPACK 3.0 (http://www.nupack.org).
- Obtain the file *multisubjective.cpp* (v1.0 beta 6) 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 *$HOME/Documents/Multisubjective* . (Another location may be used, in which case the *$CLDDPATH* environment variable should be set to specify this.)
- Run the command "npm install commander underscore format" in that directory.
- Obtain the files *dd.js* (v0.3) and *cldd.js* (v0.4.6) and place them in that directory. Set *cldd.js* to be executable.

<u>Input</u>

Multisubjective requires three input files, which 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: NUPACK multiobjective specification

The file *specification.np* contains the NUPACK multiobjective 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 but valid 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 “sequence=” or “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.

specification.ms: Multisubjective specification

The file *specification.ms* contains data needed by Multisubjective that is absent in the NUPACK specification. Most of these data relate to the function of the strands within the HCR 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.

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

Multisubjective loads the actual sequences in one of two formats. The file *sequences.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 *sequences.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 *sequences-0.dd* to *sequences-9.dd* are used. Multisubjective can also create random sequences, in which case no sequence input file is needed.

<u>Output</u>

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.

output.dd and output.msq: New block sequences

The file *output.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 multiobjective design specification for manual input to the web server. The file *output.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.

output.log: Internal data log

The file *output.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.

output.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 *output.post* contains a machine-readable version of the full NUPACK multiobjective 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.

<u>Program operation</u>

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 *sequences.dd* . Multisubjective automatically converts the block sequences contained in the DD file into the strand sequences used by Multisubjective.
- Load multiple DD files (m). The program will run ten times using the files *sequences-1.dd* to *sequences-10.dd* . This option is useful for quickly comparing several related DD outputs and determining which is optimal.
- Fill with random bases (f). The program creates ten sets of random block sequences and uses these to construct ten sets of strand sequences.

- Load a NUPACK-MO file (n). The preexisting *sequences.npo* file is used. This is useful for running the same sequence many times.
- Autofill from last MO web submission (a). The program gets the job number and token from *response.html*, which contains the information for the job most recently submitted to the NUPACK web server, in a previous run of Multisubjective. This is useful when iterating sequences between Multisubjective and NUPACK multiobjective design.
- 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 copies the requested sequence file to *sequences.npo*, overwriting a previous file if necessary. If the job has already been downloaded, it asks only for a trial id, and copies the desired file from */mo_output* to *sequences.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. In all cases, the sequence given to the designer has only those bases unlocked that were identified by Multisubjective as problematic. If multiple sequences were 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 multiobjective design file, with the new block sequence assignments inserted in place of the grave character (`) delineated block in the *specification.np* input file. 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 two passes. The first pass contains the full sequence of each hairpin, testing the "closed" hairpin structures. The second pass includes both the hairpins with their input domains removed, giving the "open" hairpin structures, as well as the bridge complexes. 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.

<u>**Command line operation**</u>

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] [-t pair_threshold] [-p prevented_seq] [-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', '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 *pair_threshold* is the threshold used by Multisubjective to identify strong undesired base pairs. The argument *prevent_seq* is a string specifying the prevented sequence limits in the following format: sets of a number followed by the identities of the (possibly mixed) bases for which that number is to be used, all with no spaces. For example, the default could be represented by “4ACGT6SWRYKM100BDHV”. These arguments will be overridden by any contrary settings in the *specification.ms* file.
- 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 three extra files: *output.mso* contains Multisubjective’s analysis data in JSON format, while *nupack/ms0.ocx-mfe* and *nupack/ms2.ocx-mfe* are the NUPACK-generated minimum free energy structures for the closed and open strands, respectively.

Format of the *specification.ms* input file

In general, the syntax used in the *specification.ms* file is similar to that used in the NUPACK design format. Most lines in the *specification.ms* file specify the function of the various strands, so that Multisubjective knows how to process them. Any strands that appear in *specification.np* but not in *specification.ms* are ignored by Multisubjective. The *specification.ms* file also contains specifications for the block lengths, and the location of any bases the user desires to be immutable.

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.

Hairpins

Automatic: hairpin [name] : [+/-]
Explicit: 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

Explicit: 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 complexes [deprecated]

Automatic: bridge [name1] [name2] : [complex_name]
Explicit: bridge [name1] [name2] = [pos1] [pos2] [size]

Including a bridge complex as part of the NUPACK design specification is often useful when bridge domains are present in the system. It instructs Multisubjective to consider undesired interactions between the free ends of strands involved in a bridge interaction. For each bridge interaction, in *specification.np* a structure is specified containing the open portions (i.e., the sequence with the input domain removed) of the two strands involved in the bridge.

In the automatic syntax, bridge complexes are specified using the syntax "bridge B1 C2 : br12" where B1 and C2 are the names of the two individual strands involved in the bridge, and br12 is the name of the complex containing both open portions. Again, all three of these names must appear as structures in the NUPACK specification. For the automatic syntax, bridge complexes are expected to be open bridges (i.e., bridges between opposite-polarity hairpins) with the positive-polarity strand positioned first.

In the explicit syntax, the three numbers specify the position and length of the bridge interaction. The numbers *pos1* and *pos2* are the positions of the first base involved in the bridge interaction on each of the two strands respectively, using their positions on the full, closed hairpins. The *size* is the number of bases that are paired in the bridge interaction. The explicit syntax allows closed bridges (i.e., bridges between same-polarity hairpins) to be used.

Static structures

Automatic: 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 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.

Specifying block lengths and immutable bases

Explicit: length [blockname] = [specification]

Multisubjective requires information about the length of each block and the location of bases desired to be immutable. The syntax of this group of lines is almost identical to the “domain=” lines omitted from the NUPACK specification, but with the keyword “length”. Instead of specifying bases, the letter ‘N’ represents a normal base and the letter ‘Y’ represents an immutable base. For example, “length j = N5Y2” indicates that the length of block j is 7 nucleotides, and the last two bases are 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 threshold

Explicit: threshold = [value]

The threshold can be specified here. If this line is absent, the default value of 0.67 is used.

Specifying the prevented sequence limits

Explicit: 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; 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, without 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 *sequences*)
- The filename prefix, without an extension, of the output files (the default is *output*)
- 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.

<u>Exception handling, signal handling, and exit status</u>

If an error condition occurs during the program's execution, an exception will be thrown with a unique identifying number in the range 1–120. 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 *output.log* file. Signals indicating a program error are caught and handled in the same way as 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.