MCFlashfold (mcff version 34)
user manual

Paul Dallaire
Labo Major,
IRIC, DIRO, UdeM

2013-Sept-17
f34 2015-May-14

Introduction

This document focusses on installing and using mcff (MC-Flashfold) to compute RNA secondary structures. See the yet unpublished article for background information and the original MCfold paper for the concept of NCM.

Motivations
I wrote this program because I needed MCfold to run faster and on larger input RNA sequences. It is not however a drop-in replacement. For one, it's feature set and interface are somewhat different but also their predictions differ in a number of ways. Also mcff generates more detailed structure variants therefore increasing the output size and scores slightly differently the predictions.

Installation

Platform
mcff was used on MacOSX (mountain lion) and Linux.

Installation on a Mac OS X
Precompiled binaries are provided for the OS X platform.

1- Move the MC-Flashfold directory to /usr/local/bin/MC-Flashfold or to your home directory under the name ~/MC-Flashfold
2- Copy the executables under MC-Flashfold/
(mcff and flashScan) to /usr/local/bin/ or elsewhere on your path (or set your path to include this directory).

Compilation

this basic compilation should always work:

```
cc -o mcff flashfold.c -lm -std=c99 -O3
```

You may experience speedup in some situations when specifying these:

```
-mfpmath=sse -msse2
-march=native -msse4.1 -D___SIMD_COMPILATION__=1
```

On some Linux boxes you may need to set this if the compiler complains about __TIMESTAMP__ not being available"

```
-DNOTIMESTAMP=1
```

If you prefer to visualize shape signatures using square brackets (as is standard) you may set this compilation option:

```
-DSQUARE_BRACKET_SHAPES=1
```

Energy files details

Five external files are needed for mcff to compute structure: energies, hinges, junctions, strands and transitions. A textual representation of these are provided in a directory named 'tables' and distributed with the source code. Each of these text files have the extension .f2.csv. This directory is expected to be in the current running directory otherwise it's location can be specified at the command line. mcff uses binary versions of these files which are not distributed with the source code. On first launch, mcff computes the required binary versions from the text ones and stores them with the suffix .f8 in the same directory. That way a platform specific binary version is computed for each installation. If you obtained the software by copying the directory in your private hard drive, you may discover that the 'tables' directory contains files that do not end in .f2.csv, it is safe to remove them. If you use a shared disk environment such as nfs and you run heterogeneous machines, you may copy the tables directory to locations that reflect the name of your computer node, remove from them all files that do not end in .f2.csv and create aliases for mcff on each node using a fully specified path for the -tables command line parameter. However under intense use (such as genome scanning) when mcff is to be repeatedly called, it may be best to use local copies of this directory.

mcff uses the first valid tables directory that it finds. You can verify which is used by running the program in verbose mode (-v or -v2). The search order is as follows:
1) path specified using the parameter `-tables PATH` at the command line,
2) Value of the `MCFTABLES` environment variable,
3) ./tables
4) ~/MC-Flashfold/tables
5) /usr/local/MC-Flashfold/tables

Running mcff

The behavior of mcff is entirely controlled at the command line using parameters and options. The user specifies a parameter by prepending one or two dash '-' character(s) to the relevant keyword or alias and as required by specifying relevant options values.

```
-parameter OPTION1 OPTION2 ...
```

Also each parameter name can be specified using alternative forms. For example, to specify a threshold value of 9.2 kcal/mol, the following forms are equivalent:

```
-t 9.2
--t 9.2
-threshold 9.2
--threshold 9.2
```

You can obtain the list of parameters, by typing -h at the command line (or --h, -help...). The only mandatory parameter is -s which specifies the sequence data.

Obtaining the MFE secondary structure(s)

To obtain the secondary structure of minimum free energy of sequence AAACCUUU, type

```
mcff -s AAACCUUU
```

this will output the dot bracket followed by the folding energy and the *shape* level 5 abstraction of the structure.

```
(((..))) -2.537 ()
```

You can use the following characters and no other in the sequence: ACGUTacgut.

Sometimes the software will output more than one MFE structure. This is because two secondary structures or more can exhibit the same energy level at the resolution used to compute them. On long RNA sequences such as 23S ribosomal sequences this becomes unavoidable.

Formatting the output

The verbose flag can be set to obtain additional information on the run. This is useful in particular for logging.

type:

```
mcff -s AAACCUUU -v
```

to obtain:

```
Explored
>(null)
AAACCUUU
(((..))) -2.537 ()
```

The value (null) can be set to a name you provide if you use the parameter -n as follows:

```
mcff -s AAACCUUU -n 'A sequence' -v
```

The -alt parameter expands the dot brackets format to include <> for non-canonical base pairs. Here the set of canonical base pairs is taken as {AU,GC}. This is useful when used in conjunction with output masks (discussed later) in particular.

Typing:

```
mcff -s AAAAACCUUGUU -alt
```

outputs:

```
((<((..))>)) -5.871 ()
```

emphasizing that the pair A-G is non canonical.

Getting a little more information

Sometimes it is appropriate to get an insight into what the program is doing. You can use the parameter -v2 (or -vv) to increase the verbosity level. An example of this will be seen when we cover the *floating threshold* parameter.

Computing suboptimal structures

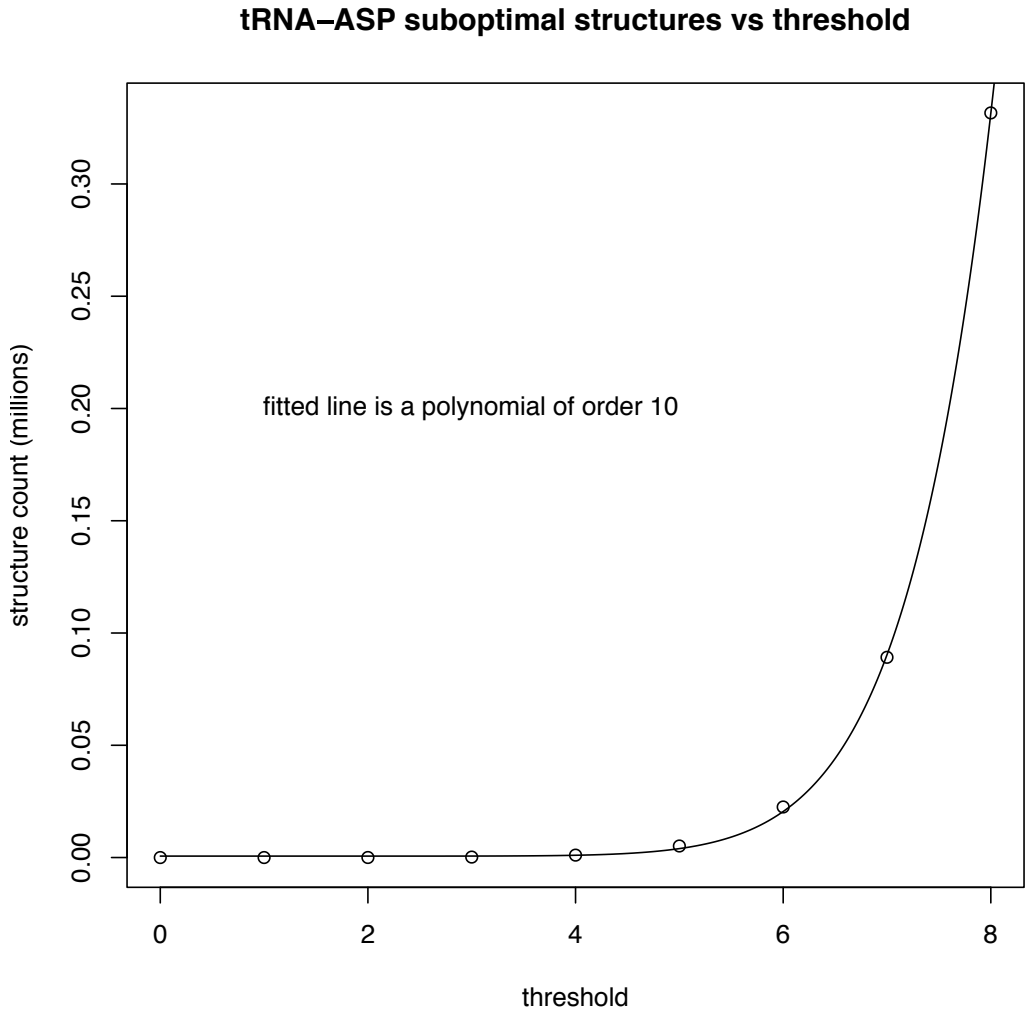
Any secondary structure whose free energy is not the MFE is a suboptimal structure. The closer its energy to that of the MFE, the more likely the structure. mcff can be used to list less and less likely structures as their energies are further and further from the MFE value. To guide this search a threshold must be provided in kcal/mol that defines the boundary of the search. Use the parameter -threshold (or -t) to specify this boundary and mcff will list all complying possibilities. (Since version 33 of mcff, you can use -explore (-e) to specify the threshold as percentage of MFE instead of as an asbolute kcal/mol value.) The toy sequence AAACCUUU outputs 4 structures when we specify -t 1 at the command line:

```
(((..))) -2.537 ()
((...)). -2.452 ()
((....)) -2.082 ()
.((...)) -1.983 ()
```

It is important not to abuse of this parameter because the number of suboptimal structures grows rapidly as the threshold is raised. If the threshold is too large, the number of structures will take a proportionally long time to compute and the output will occupy a large space on your disk. One available option is to ask mcff to count the structures that would be generated for a given value of -t by using the command line parameter -count (or -c) and to approach your goal by increasing its value appropriately.

The following script was used to output the structure counts for the yeast tRNA ASP (GCCGUGAUAGUUUAAUGGUCAGAAUGGGCGCUUGUCGCGUGCCAGAUCGGGGUUCAAUUCCCCGUCGCGGCGC) as the threshold in increased. This data was used to generate the plot.

```
for T in 0 1 2 3 4 5 6 7 8; do
     echo -n "$T ";
     ./mcff -c -s $(cat tRNA-ASP.seq) -t $T ;
done > counts
```



Using -count, it is thus possible for you to estimate the duration of the calculation and the space required to store its output. However the software still needs to perform the computations and count the solutions when using -c because the growth rate is very different for different sequences and computation times for an exaggerated -t value is still potentially long.

If you know the number of suboptimal structures that you seek but do not know the curve growth rate for your sequence, you may use the parameter -floating_threshold (or -ft). When you use -ft, mcff approaches the ideal threshold from below and then uses an adaptive threshold algorithm to determine the exact threshold required for the target structure number. Then mcff generates output using the measured threshold. -ft can be combined with the graph calculations too but not with the simple duplex modes. To follow the decisions made during a run using -ft you can specify -v2.
For example, to list the 1000000 best suboptimal structures of yeast tRNA-ASP, the worst energy explored should be -57.57329.

```
mcff -s $(cat tRNA-ASP.seq) -ft 1000000 -v2
```

And here are the first few lines of output:

```
Explored
>(null)
GCCGUGAUAGUUUAAUGGUCAGAAUGGGCGCUUGUCGCGUGCCAGAUCGGGGUUCAAUUCCCCGUCGCGGCGC
input threshold +0.00000

INFO: Found 1 solutions.
Trying with new threshold of -65.083 (delta=+1.370)

INFO: Found 6 solutions.
Trying with new threshold of -63.713 (delta=+2.740)

INFO: Found 122 solutions.
Trying with new threshold of -60.974 (delta=+5.479)

INFO: Found 10700 solutions.
Trying with new threshold of -57.294 (delta=+9.159)
Explored
>(null)
GCCGUGAUAGUUUAAUGGUCAGAAUGGGCGCUUGUCGCGUGCCAGAUCGGGGUUCAAUUCCCCGUCGCGGCGC
mfe(-66.45300), th(-57.57329)
(((((((((((((((((((((((((((((((...)))))))))))))))))..)))))))))))))).. -66.453 ()
(((((((((((((((((((((((((((..(...)))))))))))))))..)))))))))))))).. -66.118 ()
(((((((((((((((((((((((((((((.....)))))))))))))))..)))))))))))))).. -65.457 ()
(((((((((((((((((((((((((((((...)))))))))))))))..)))))))))))))).. -65.378 ()
((((((((((((...))))))(((((((((...)))))))))))))))(((...)..)))))))))))).. -65.326 (()()())
(((((((((((((((((((((((((((((....).)))))))))))))..)))))))))))))).. -65.150 ()
(((((((((((((((((((((((((((..(...)))))))))))))))..)))))))))))))).. -65.043 ()
((((((((((..))))))))(...(((((((...)))))))))))))))(((...)..)))))))))))).. -65.032 (()()())
((((((((((((...))))))(((((((((..(...)))))))))))))))(((...)..)))))))))))).. -64.991 (()()())
```

On-line mode

The default behavior of mcff is to generate all structures internally and to sort them before output. In cases where the number of solutions is potentially very big, keeping copies of the solutions internally may cause memory problems. In those cases you can specify -no_sort (or -ns). In this mode the solutions are output as soon as they are discovered, downstream processing tools can directly use the results from a Unix pipe and save both RAM and storage requirements.
Note that -ns generated outputs are not sorted and do not output the shape string.
This mode is incompatible with graph, duplex and floating threshold computations.

Masking

Masks serve two main purposes: (a) limit the search by making use of prior knowledge and (b) identify suboptimal structures from the output. These two goals are independent and can be used together or separately. mcff offers two mask categories corresponding to these usage scenarios: (a) full masks make use of prior knowledge during calculation and (b) output masks do not modify computing but recognize matching structures on output. Also, masks come in two flavors: (a) balanced and (b) unbalanced. The balanced masks are used to specify base pairs whereas unbalanced masks specify the base pairing status of individual nucleotides. You can thus simultaneously specify four types of masks to mcff.

Here are the corresponding command line parameters:

	full mask	**output mask**
balanced	-mask (-m)	-output_mask (-om)
unbalanced	-unbalanced_mask (-um)	-output_unbalanced_mask (-oum)

Each mask must have exactly the same length as the input sequence (except when used in one of the duplex modes: see the section on duplex modes). Otherwise, an error will be reported and execution will not start. Balanced masks are composed of characters from the following table:

BALANCED MASK character	**BALANCED MASK meaning**
.	unpaired
x	don’t care
(	pairs with matching)
)	pairs with matching (

The balanced mask ‘.(x.xx).’ would match ‘.((..)).’ and ‘.(....).’ but not ‘.(...)..’ for example.

Unbalanced masks are composed of characters from the following table and allow for flexible specifications:

UNBALANCED MASK character	UNBALANCED MASK meaning
.	unpaired
x	don't care
(	forward paired
)	reverse paired
l (vertical bar)	paired
- (minus sign)	not canonically paired
[	forward canonically paired
]	reverse canonically paired
+	canonically paired
_ (underscore)	not paired non canonically
< (less than)	forward paired non canonically
> (greater than)	reverse paired non canonically
!	paired non canonically
p	not reverse paired
q	not forward paired

For example an unstructured terminal loop (.....) that would not allow bases to pair when only canonical base pairs are considered may require a more flexible mask if non canonical base pairs are allowed to form (-----) using the NCM model.

It is an error to specify more than one balanced full mask (-m) and one unbalanced full mask (-um).

It is possible to use unbalanced masks symbols in a full balanced mask if this causes no confusion. When a -m mask is specified, a process called *conciliation* is performed that removes unbalanced masks symbols from the balanced mask and places them in the unbalanced mask. At this time, if a -um mask was used and specifies a symbol that is not compatible with the symbol from the balanced

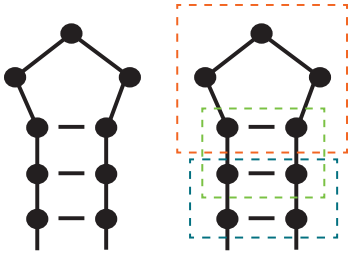
mask an error is reported and mcff does not start. If the combined symbols are compatible a symbol describing their combined effect is placed in the unbalanced mask and no error is reported.

Output masks are named. Many output masks can be given. All names should be unique and there is no conciliation performed on output masks. You may well specify a mask for a ‘native’ structure another for a ‘bound’ structure and yet others for defined hairpins or family like folds. The names of the matching output masks will appear on the output lines.

When placing masks on the command line it is a good idea and may be necessary to use single or double quotes to prevent the system from trying to *interpret* your string. Among other characters, parenthesis and vertical bars have special meanings in shells.

Fine tuning predictions

The parameter -NO_LONG_TERMINAL_LOOP (-nltl) blocks the exploration of stretches of looping unpaired nucleotides at the apex of a stem. The figure shows a stem capped with a terminal loop. On the right, rectangles illustrate the NCMs that are used in the computation of this structure. The red rectangle encompasses the terminal loop NCM. Statistical data can be collected for a number of structures from known structures but some structures are so rare that statistics would have little predictive value and that is the case for terminal loops composed of more than a few nucleotides. These structures normally collapse on themselves, forming intricate structures involving non canonical base pairs. Nonetheless we find that the addition of a default rule allowing for very unlikely structures to lay unstructured is sometimes beneficial and does not interfere with predictions. You can turn this default rule off by selecting -nltl at the command line.



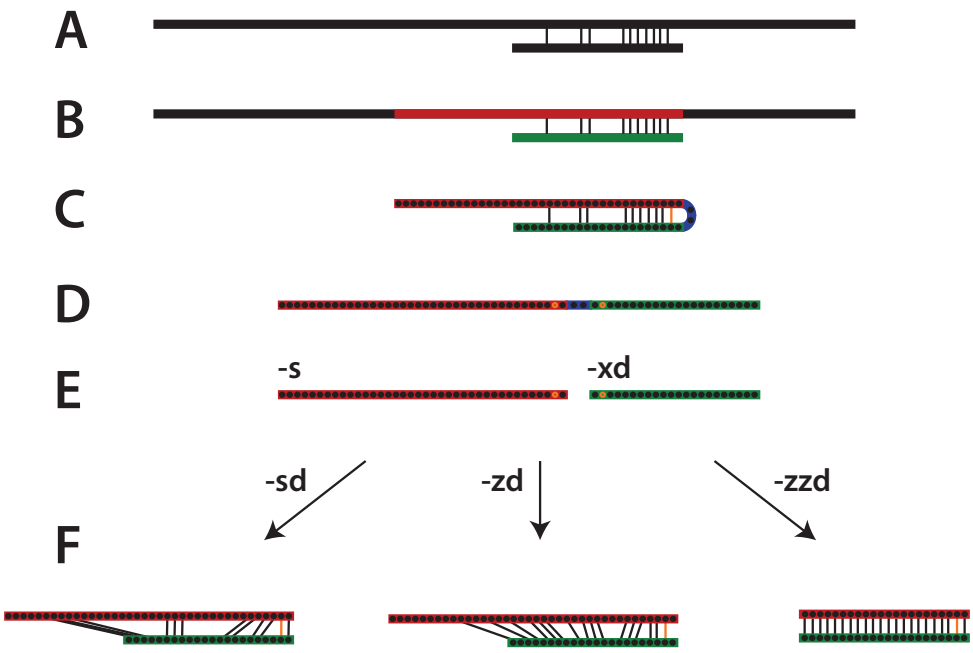
Also you can allow or prevent the algorithm from exploring tiny stems composed of single base pairs and caped with a terminal loop. In version 30, the default behavior is to prevent them but if you want to allow them, use the parameter -STEM_OF_ONE (-soo). Depending on algorithmic implementation variations some tools are unable to predict these.

Overall, if your type of work involves exploring the details of non canonical base pairing patterns you probably wish to set -soo and examine numerous suboptimal structures. If on the other hand your work aims at identifying main structural elements you may find that the default behavior or using -nltl

produce less variants. Your mileage will vary according to your sequences, the type of masks that you use and so on.

Simple duplex modes

Simultaneous folding of two sequences in a single structure space (co-folding) calls for a higher order computation that is not currently implemented in mcff. But the usage scenario that involves pairing short segments together as in a microRNA with a target site on an mRNA is a straightforward adaptation of mcff. Three variations on the theme are available and illustrated in the figure below.



(A) The mRNA (top strand) is expected to hybridize to a microRNA (bottom strand) through a variable number of base pairs whose exact matches are to be computed.
(B) The user selects a region of the mRNA shown in red.
(C) Shows the hairpin structure that mcff will internally compute. To do so, mcff adds a short terminal loop of known composition to guide its calculation. But the energy contribution of that loop will later be removed from all predictions. In the current version of mcff, the single base pair shown in orange (microRNA seed nucleotide #2) is forced to form a base pair with its mRNA facing partner.
(D) Illustrates the sequence that is used for computation.

(E) The user specifies the sequences shown via the command line parameters -s and -*x*d. Here -xd is one of -sd, -zd or -zzd and these correspond to the modes: simple-duplex, zip-duplex (or NCM-duplex) and zip-zip-duplex (or NCM4-duplex)

(F) The computed structures are imposed different constraints according to the exact mode used according to the following table.

parameter	mode name	constraints on folding	constraints on input
--all cases--		Orange base must pair. No terminal loop in either strand.	
-sd, -simple_duplex	Simple Duplex	Long internal loops and bulges are possible everywhere.	
-zd -NCM_DUPLEX, -zip_duplex	NCM Duplex	All NCMs allowed. Only short internal loops and bulges.	
-zzd, -NCM4_DUPLEX, -zipzip_duplex	NCM4 Duplex	All base paired using 2_2_2 NCMs.	Both sequences must be exactly of the same length

NOTE: By default mcff outputs only the energy of the solution. To obtain the corresponding dot bracket structure you must add parameter -v or -v2.

Masks can be applied to simple duplex modes but the effective sequence that will be masked was shown in (D) in the figure. This implies that your masks must conform to the general form ABC where A is your mRNA sequence, B is the inserted terminal loop and C is your microRNA sequence. The previous to last character in A must be '('. B must be '..' and the second character in C must be ')'. So a mask without effect could be 'xxxxx(x..x)xxx' where the A and C segments are underlined.

For example the following run would compute the NCM Duplex but enforcing nucleotides 2 to 8 of the miR in a consecutive stem and enforcing some form of pairing from nucleotides 13 up the end. Nucleotides at positions 9-12 in the microRNA are not forced to base pair

```
./mcff -s ACGUACGUCGUACGUACGUACGUACGUCGUACGUACGUACGU \
-zd ACGUACGUCGUACGUACGUACGU \
-m 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx(((((((x..x)))))))xxxxx||||||||||' \
-vv -t 2
```

output:

```
INFO: using bmask: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx(((((((x..x)))))))xxxxxxxxxxxxxxx
INFO: using umask: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx(((((((x..x)))))))xxxxx||||||||||
Explored
>(null)
ACGUACGUCGUACGUACGUACGUACGUACGUCGUACGUACGUACGUAAACGUACGUCGUACGUACGUACGU
input threshold +2.00000
mfe(-59.89038), th(-57.89037) (INFO: Duplex mode solutions are not sorted.)
..................(..((((((((((...(((((.(((((((.  ))))))))))))))))))))) -58.764
..................(..((((((((...((((((.(((((((.  ))))))))))))))))))))) -57.999
...................(((((((((((...(((.(((((((.  ))))))))))))))))))))) -58.352
...................((((((((((...((((.(((((((.  ))))))))))))))))))))) -58.897
...................((((((((((...(((((.(((((((.  ))))))))))))))))))))) -59.890
...................(((((((((...((((((.(((((((.  ))))))))))))))))))))) -59.125
```

Scanning for high energy microRNA sites on mRNA

MC-Flashfold comes bundled with a script that automates calling MC-Flashfold in duplex modes on a miRNA-mRNA pair. It returns either a user selectable number of the best binding sites or those whose energy is better than some threshold.

```
flashScan -help
```

Provides all the details of the command line uses.

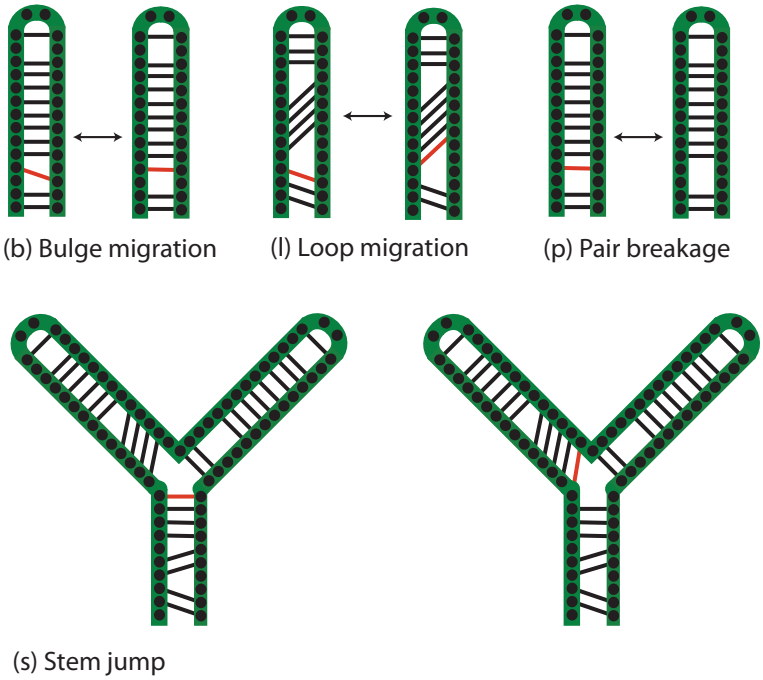
Graph of Disconnected Components

We can link every potential structures for an RNA molecule in a network where one structure can be transformed in another one by the application of simple rules such as base pair breakage for example. In computer sciences, the terms *graph* and *network* are essentially equivalent and we will now speak of the *graph* of structures. This graph of RNA secondary structures for a given molecule can be very (very) big and out of reach to computation but it is nonetheless always implicitly defined. When mcff computes the structures below an energy level it can be thought of as taking samples from this graph. Some of the computed structures will share features and others will be unrelated. If you specify -g or -gs at the command line, mcff will compute the sets of connected secondary structures from its output where each structure in each set can be transformed into another one in the same set but not in structures from any other sets. We call these sets the *disconnected components.* Many people find it more natural to think of them as *clusters* of similar structures. This is almost ok but not quite. The difference between connected components and clusters is that an element in a cluster is required to *be more similar to* other elements in its cluster than to elements in another cluster whereas this is not assumed of an element in a connected components system. However, as a first approximation, elements in a connected component can all be transformed into one another by

adding energy to a lesser level than the value specified for the threshold (-t) parameter, for example by binding to a protein or small molecule. Elements in different components cannot be transformed into one another by repeated application of the specified rules and within the given energy threshold.

We know that RNA molecules undergoe fluctuations and that they are in contact with a myriad of molecules affecting their internal states and we use the graph disconnected components calculation to determine which secondary structures are reachable from one another at a given energy threshold.

As shown in the figure, a number of transformation rules can be applied to this process and in some situations a faster algorithm can be used in the calculation of the disconnected components.



mcff calculates the components if you specify -graph (-g) or -graph_summary (-gs) at the command line. You may also give a value to -edge_type (-et) to control the set of rules used to compute connectivity. By default two structures are connected if a single application of one rule suffices. The parameter -ex allows you to override this behavior in certain situations. Type -ex 10 to compute the components separated by up to 10 applications of the rule set generating very robust separations.

NOTE: Values, parameters and defaults may change with further versions and you should check your version with -h before assuming that the following holds.

parameter	rules (see figure)	notes	version of mcff
(none) or -et 0	b,l,p,s	fastest	f10
-et 1	b, l, p	allows use of -ex	f30
-et 2	b, p	planned	(f32?)

Measuring time

The parameter -times causes the output of statistics collected during the execution. This outputs a vector comprising various data, including elapsed seconds (actual CPU time). A description of the data contained in this vector can be obtained with -times_header. Like the -h and -v parameters, using -times_header will not calculate anything and only output a header for the -times vector.