# The Vienna RNA Package 2.0

Ronny Lorenz
ronny@tbi.univie.ac.at

Institute for Theoretical Chemistry
University of Vienna

, Czech Republic, October 8, 2010

## Overview

- extended nearest neighbor model (Turner ’04)
- standardization of user interfaces
- standardization of implementations

## Current state

- *testing*
- release in the next weeks

## Review: Turner ’04 nearest neighbor model

- tri-, hexa-loop parameters
- mismatch energies for exterior and multibranch loops
- 1xN-, 2x3-interior loops
- enthalpies for all energy contributions
- updated energy contributions

## Changes for users

### Command line options

Up to version 1.8.4

- `POSIX` standard (e.g. `-p`)
- `X toolkit` standard (e.g. `-noLP`)

Since version 2.0:

- `GNU` standard
  `'-'` prefix for short options (e.g. `-p`)
  `'--'` prefix for long options (e.g. `--noLP`)
- combination of command line options
  `RNAsubopt -c -4 -D -e 5` equals
  `RNAsubopt -c4De 5`

more detailed help output (`-h | --help`) since usage `gengetopt`

## Changes for users

### New programs included

- `RNA2Dfold` - free energy landscape partitioning
- `RNALfoldz` - locally stable secondary structures + z-score
- `RNALalifold` - locally stable secondary structures for alignments
- `RNAplex`, `RNAsnoop`
- several updates on interfaced library functions
- `RNAparconv` - energy parameter file conversion tool (version 1.x $\rightarrow$ 2.x)

### Parameter sets

- Andronescu 07
- Mathews DNA
- Turner ’99

## Changes for developers

### C/C++ API extensions

- unified design of functions and function wrappers
- several new or replacement functions in RNAlib
  (e.g. `findpath` direct path heuristic)
- several *deprecated* functions $\rightarrow$ obsolete/replacement
- tons of utility functions
- a huge amount of API code documentation

consult the reference manual and source code

## API documentation with `doxygen`

- comments with detailed function description directly in sources (and there only)
- no need for 3+ way documentation (`code`, `man`, `TeX`)
- generate `man` pages, `HTML` and `PDF` reference manual

*Be chatty while writing a function and lazy afterwards ;)*

current reference manual available at
`http://www.tbi.univie.ac.at/~ronny/RNA`

# some examples...

```
#ifndef __VIENNA_RNA_PACKAGE_LOOP_ENERGIES_H__
#define __VIENNA_RNA_PACKAGE_LOOP_ENERGIES_H__

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <ctype.h>
#include <string.h>
#include "params.h"
#include "fold_vars.h"
#include "energy_par.h"

#ifdef __GNUC__
# define INLINE inline
#else
# define INLINE
#endif

INLINE  PRIVATE int E_MLstem(int type, int si1, int sj1, paramT *P);

INLINE  PRIVATE double exp_E_MLstem(int type, int si1, int sj1, pf_paramT *P);

INLINE  PRIVATE int E_ExtLoop(int type, int si1, int sj1, paramT *P);

INLINE  PRIVATE double exp_E_ExtLoop(int type, int si1, int sj1, pf_paramT *P);

INLINE  PRIVATE int E_IntLoop(int n1,
                              int n2,
                              int type,
                              int type_2,
                              int si1,
                              int sj1,
                              int sp1,
                              int sq1,
                              paramT *P);

INLINE  PRIVATE int E_Hairpin(int size, int type, int si1, int sj1, const char *string, paramT *P);

INLINE  PRIVATE int E_Stem(int type, int si1, int sj1, int extLoop, paramT *P);

INLINE  PRIVATE double exp_E_Stem(int type, int si1, int sj1, int extLoop, pf_paramT *P);

INLINE  PRIVATE double exp_E_Hairpin(int u, int type, short si1, short sj1, const char *string, pf_paramT *P);

INLINE  PRIVATE double  exp_E_IntLoop(int u1,
                                      int u2,
                                      int type,
                                      int type2,
                                      short si1,
                                      short sj1,
                                      short sp1,
                                      short sq1,
                                      pf_paramT *P);

/*
###################################
# SOME OTHER STUFF FOLLOWING    #
###################################
*/

 ...

// comment
/* another comment */
/** a doxygen comment */

#endif
```

# some examples...

```
1   /**
2   *** <H2>Compute the Energy of a hairpin-loop</H2>
3   *** To evaluate the free energy of a hairpin-loop, several parameters have to be known.
4   *** A general hairpin-loop has this structure:<BR>
5   *** <PRE>
6   ***        a3 a4
7   ***      a2     a5
8   ***      a1     a6
9   ***        X - Y
10  ***        |   |
11  ***        5'  3'
12  *** </PRE>
13  *** where X-Y marks the closing pair [e.g. a <B>(G,C)</B> pair]. The length of this loop
14  *** is 6 as there are six unpaired nucleotides (a1-a6) enclosed by (X,Y). The 5' mismatching
15  *** nucleotide is a1 while the 3' mismatch is a6. The nucleotide sequence of this loop is
16  *** &quot;a1.a2.a3.a4.a5.a6&quot; <BR>
17  *** \note The parameter sequence should contain the sequence of the loop in capital letters
18  *** of the nucleic acid alphabet if the loop size is below 7. This is useful for unusually
19  *** stable tri-, tetra- and hexa-loops which are treated differently (based on experimental
20  *** data) if they are tabulated.
21  *** @see scale_parameters()
22  *** @see paramT
23  *** \warning Not (really) thread safe! A threadsafe implementation will replace this function
24  *** in a future release!\n
25  *** Energy evaluation may change due to updates in global variable "tetra_loop"
26  ***
27  *** \param   size   The size of the loop (number of unpaired nucleotides)
28  *** \param   type   The pair type of the base pair closing the hairpin
29  *** \param   si1    The 5'-mismatching nucleotide
30  *** \param   sj1    The 3'-mismatching nucleotide
31  *** \param   string   The sequence of the loop
32  *** \param   P      The datastructure containing scaled energy parameters
33  *** \return The Free energy of the Hairpin-loop in dcal/mol
34  **/
35  INLINE  PRIVATE int E_Hairpin(int size, int type, int si1, int sj1, const char *string, paramT *P);
36
37  /**
38  *** <H2>Compute the Energy of an interior-loop</H2>
39  *** This function computes the free energy \f$\Delta G\f$ of an interior-loop with the
40  *** following structure: <BR>
41  *** <PRE>
42  ***        3'  5'
43  ***        |   |
44  ***        U - V
45  ***    a_n         b_1
46  ***     .          .
47  ***     .          .
48  ***     .          .
49  ***    a_1         b_m
50  ***        X - Y
51  ***        |   |
52  ***        5'  3'
53  *** </PRE>
54  *** This general structure depicts an interior-loop that is closed by the base pair (X,Y).
55  *** The enclosed base pair is (V,U) which leaves the unpaired bases a_1-a_n and b_1-b_n
56  *** that constitute the loop. In this example, the length of the interior-loop is \f$(n+m)\f$
57  *** where n or m may be 0 resulting in a bulge-loop or base pair stack.
58  *** The mismatching nucleotides for the closing pair (X,Y) are:<BR>
59  *** 5'-mismatch: a_1<BR>
60  *** 3'-mismatch: b_m<BR>
61  *** and for the enclosed base pair (V,U):<BR>
62  *** 5'-mismatch: b_1<BR>
63  *** 3'-mismatch: a_n<BR>
64  *** \note Base pairs are always denoted in 5'->3' direction. Thus the enclosed base pair
65  *** must be 'turned arround' when evaluating the free energy of the interior-loop
66  *** \see scale_parameters()
67  *** \see paramT
68  *** \note This function is threadsafe
69  ***
70  *** \param   n1       The size of the 'left'-loop (number of unpaired nucleotides)
71  *** \param   n2       The size of the 'right'-loop (number of unpaired nucleotides)
72  *** \param   type     The pair type of the base pair closing the interior loop
73  *** \param   type_2   The pair type of the enclosed base pair
74  *** \param   si1      The 5'-mismatching nucleotide of the closing pair
75  *** \param   sj1      The 3'-mismatching nucleotide of the closing pair
76  *** \param   sp1      The 3'-mismatching nucleotide of the enclosed pair
77  *** \param   sq1      The 5'-mismatching nucleotide of the enclosed pair
78  *** \param   P        The datastructure containing scaled energy parameters
79  *** \return The Free energy of the Interior-loop in dcal/mol
80  **/
81  INLINE  PRIVATE int E_IntLoop(int n1,
82                                int n2,
83                                int type,
84                                int type_2,
85                                int si1,
86                                int sj1,
87                                int sp1,
88                                int sq1,
89                                paramT *P);
```

# Concurrent computations capability

All folding algorithms are `OpenMP` threadsave[1]

changes in energy model might result in unexpected behavior!

Simple to use

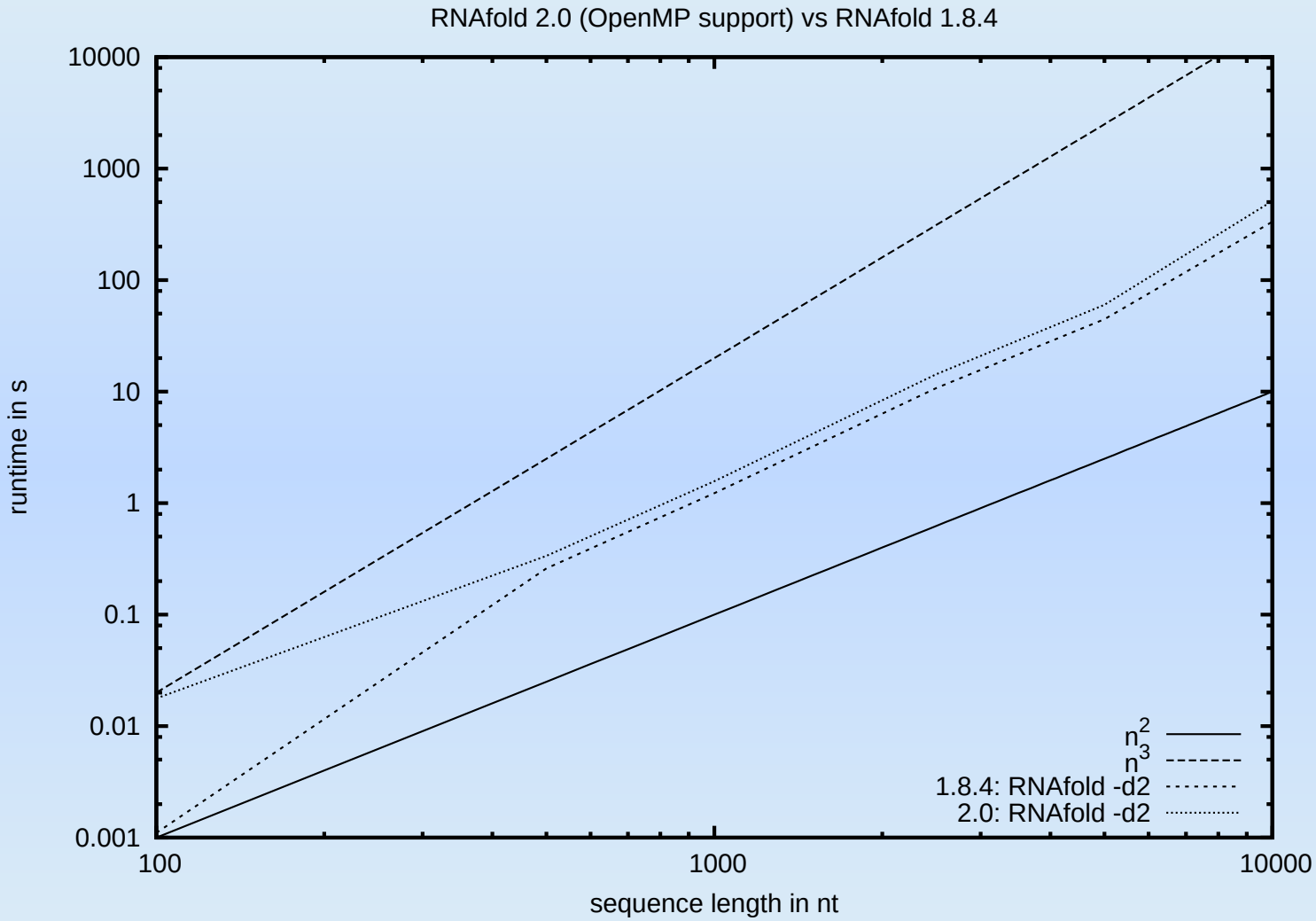
```
1    #pragma omp parallel for private(i, structure, en)
2    for(i=0;i<n;i++){
3      en[i] = fold(sequence[i], structure[i]);
4      free_arrays();
5    }
```

8 sequences, 8 threads, `Intel(R) Xeon(R) X5550`

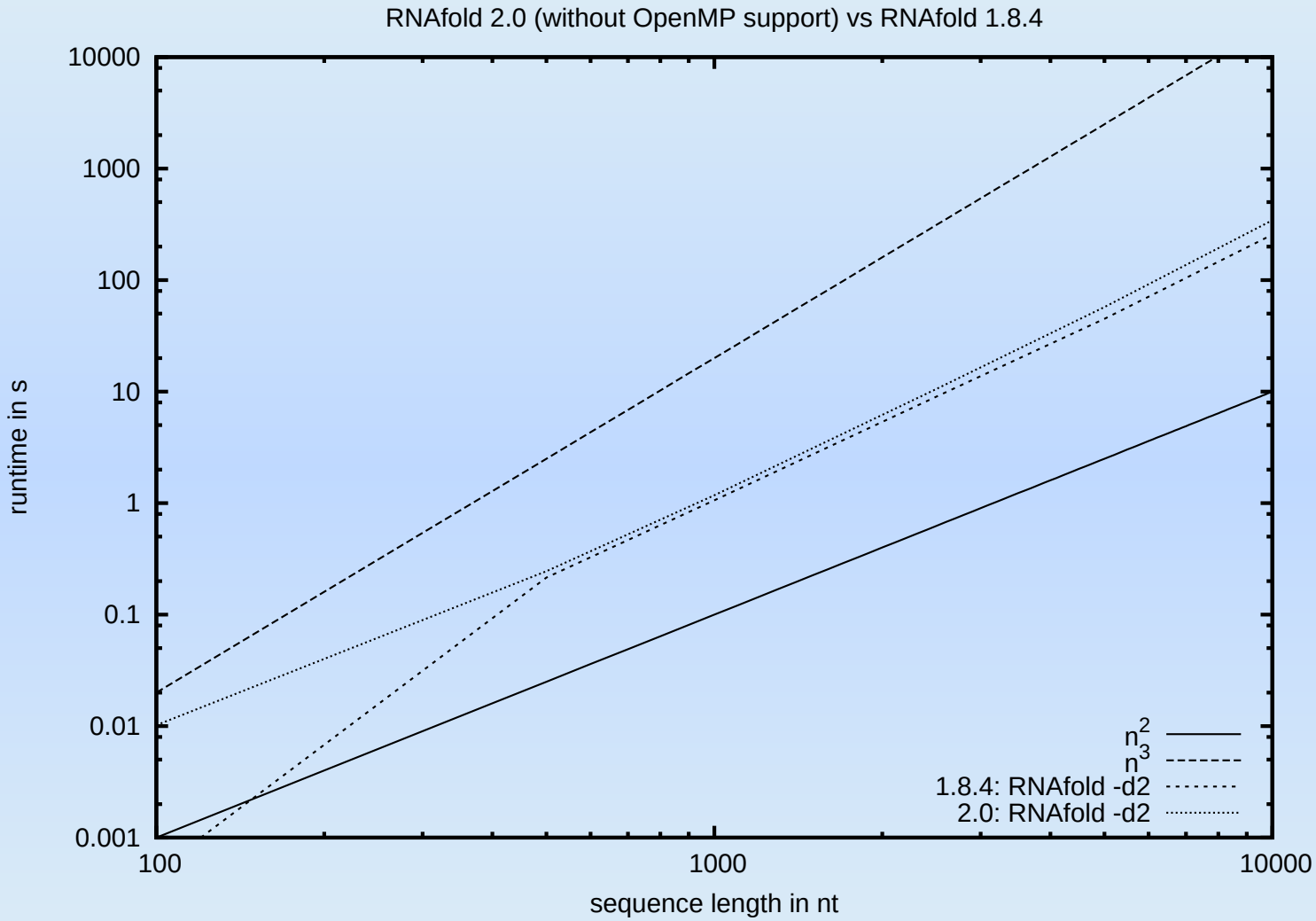
- **1000nt**: speedup 7.26, overhead 1.08
- **2500nt**: speedup 6.02, overhead 1.33
- **5000nt**: speedup 4.94, overhead 1.6
- **10000nt**: speedup 4.16, overhead 1.91

[1] at least w/o model change

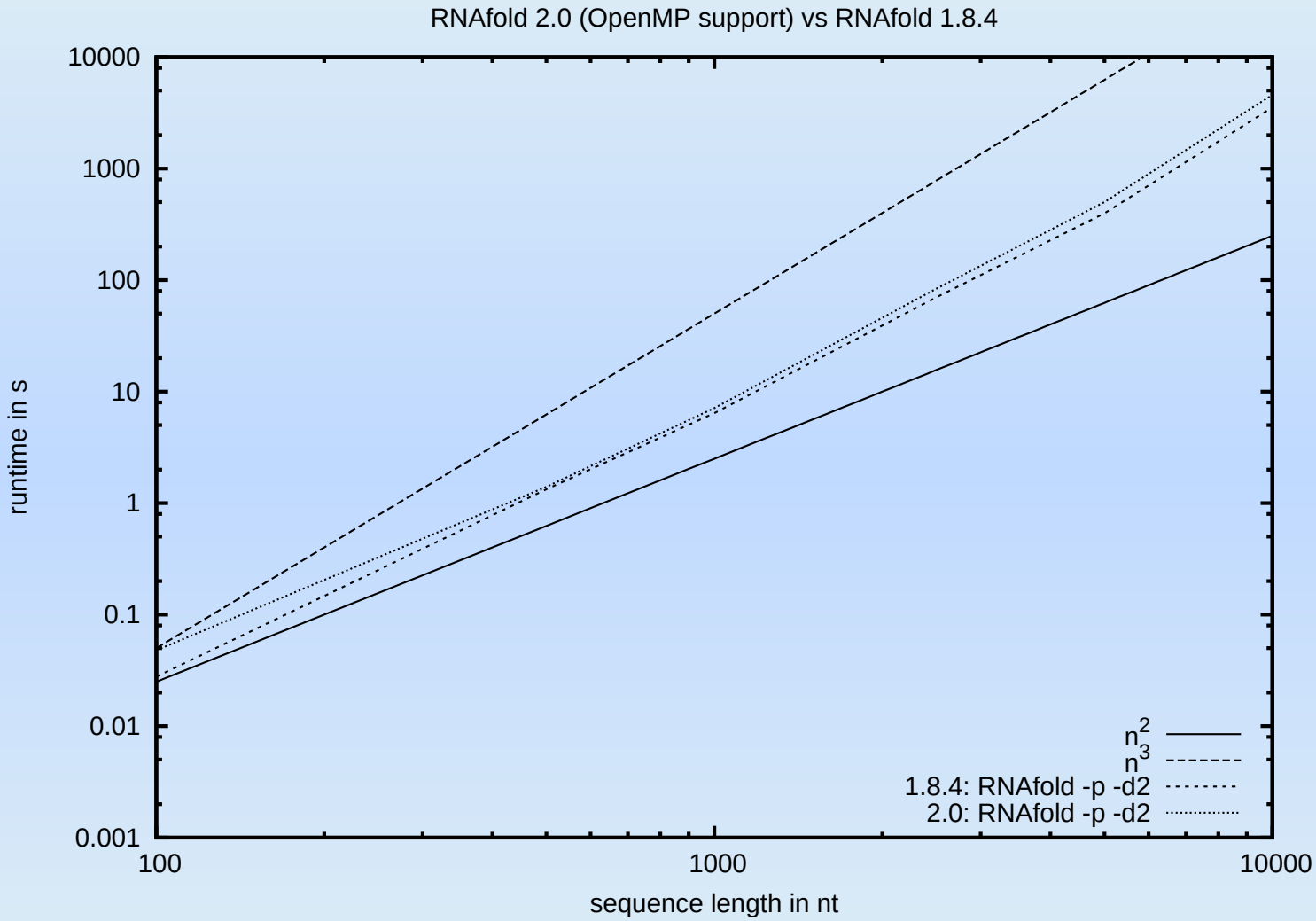
# Runtime comparison 2.0 vs 1.8.4



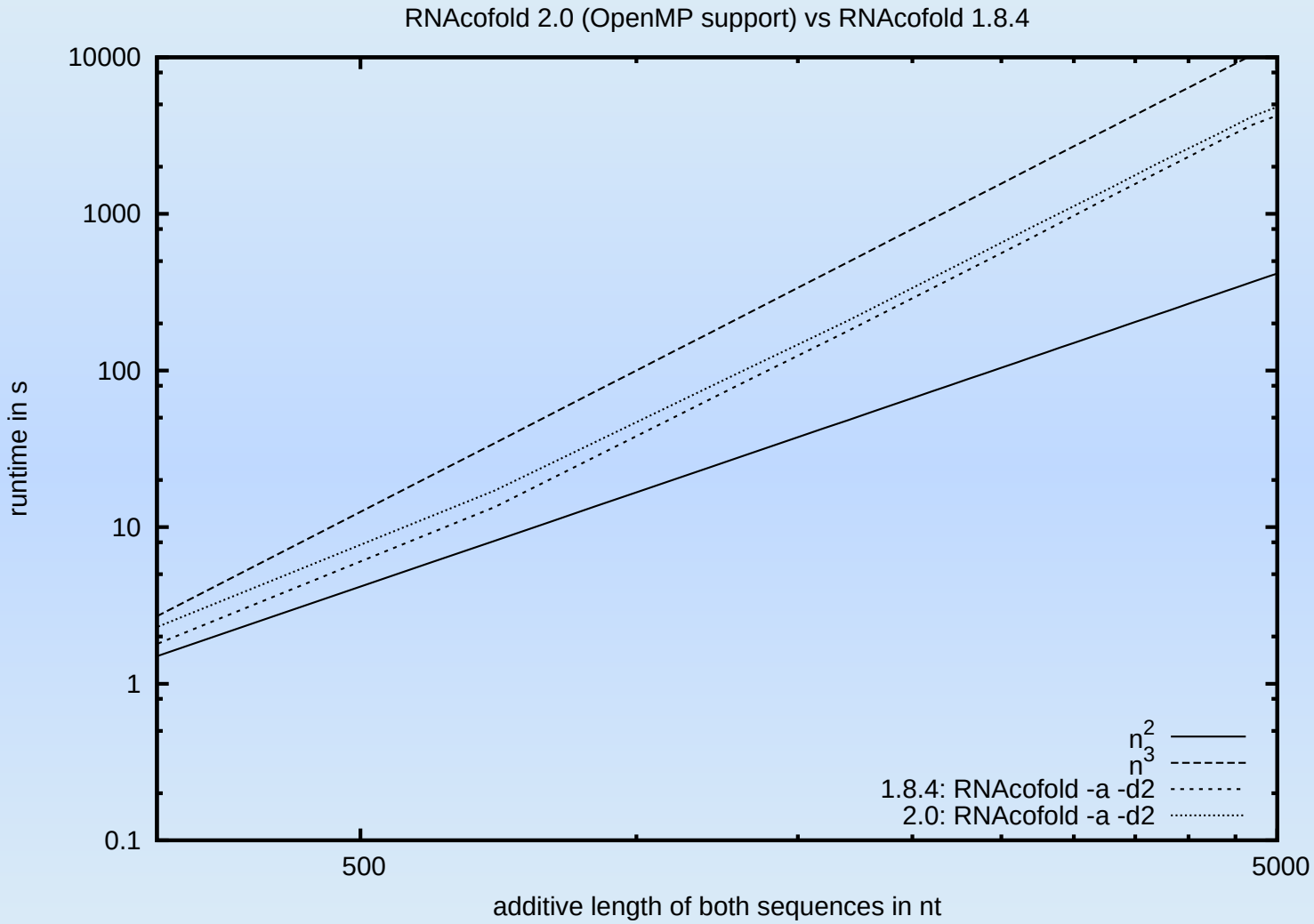
# Runtime comparison 2.0 vs 1.8.4



# Runtime comparison 2.0 vs 1.8.4



# Runtime comparison 2.0 vs 1.8.4



# Prediction accuracy

will be done next week

## Summary

- soon to be released!
- improved energy model
- more energy parameter sets:
  Turner ’99, Mathews DNA & Andronescu ’07
- standardized user interface
- improved API:
  one structure $\rightarrow$ one function
- much more documentation for developers
- re-use existing code from RNAlib!
  (instead of re-inventing the wheel)

latest snapshots available

`http://www.tbi.univie.ac.at/~ronny/RNA`

## Thanks to

Stephan Bernhard
Christoph Flamm
Andreas Gruber
Ivo Hofacker
Christian Höner zu Siederdissen
rest of TBI team

**You for listening**