Finding Non-direct RNA refolding paths (in silico)

Ronny Lorenz
ronny@tbi.univie.ac.at

Institute for Theoretical Chemistry
University of Vienna

Bled, Slovenia, February 16, 2009

1. Basics
   - RNA secondary structure energy landscape and neighborhood
   - obtaining direct refolding paths
   - taking detours

1. Basics
   - RNA secondary structure energy landscape and neighborhood
   - obtaining direct refolding paths
   - taking detours

2. finding meshpoint structures
   - Non-deterministic Method(s)
   - Deterministic Method

1. Basics
   - RNA secondary structure energy landscape and neighborhood
   - obtaining direct refolding paths
   - taking detours
2. finding meshpoint structures
   - Non-deterministic Method(s)
   - Deterministic Method
3. Results
   - Artificial RNA switches
   - add A-riboswitch

Secondary structure energy landscape

$$\text{Energy Landscape} = \{\mathcal{X}, \mathcal{H}, f\}$$

$\mathcal{X}$ . . . set of configurations
$\mathcal{H}$ . . . topological structure on $\mathcal{X}$
f . . . fitness/energy function with $f : \mathcal{X} \to \mathbb{R}$

Secondary structure energy landscape

$$\text{Energy Landscape} = \{\mathcal{X}, \mathcal{H}, f\}$$

$\mathcal{X}$ . . . set of configurations *(RNA secondary structures)*
$\mathcal{H}$ . . . topological structure on $\mathcal{X}$
f . . . fitness/energy function with $f : \mathcal{X} \to \mathbb{R}$

Secondary structure energy landscape

$$\text{Energy Landscape} = \{\mathcal{X}, \mathcal{H}, f\}$$

$\mathcal{X}$ . . . set of configurations *(RNA secondary structures)*
$\mathcal{H}$ . . . topological structure on $\mathcal{X}$ *(insert/delete bp)*
f . . . fitness/energy function with $f : \mathcal{X} \rightarrow \mathbb{R}$

Secondary structure energy landscape

$$\text{Energy Landscape} = \{\mathcal{X}, \mathcal{H}, f\}$$

$\mathcal{X}$ . . . set of configurations *(RNA secondary structures)*
$\mathcal{H}$ . . . topological structure on $\mathcal{X}$ *(insert/delete bp)*
f . . . fitness/energy function with $f : \mathcal{X} \to \mathbb{R}$ *(free energy)*

Folding path p between two secondary structures s_1 and s_2

sequence of secondary structures where:

- $p[0] = s_1$
- $p[n] = s_2$
- $\forall p[i] : p[i+1]$ is reachable from $p[i]$ by applying one single move from the moveset

direct paths:

- $n = 1 + d_{BP}(s_1, s_2)$

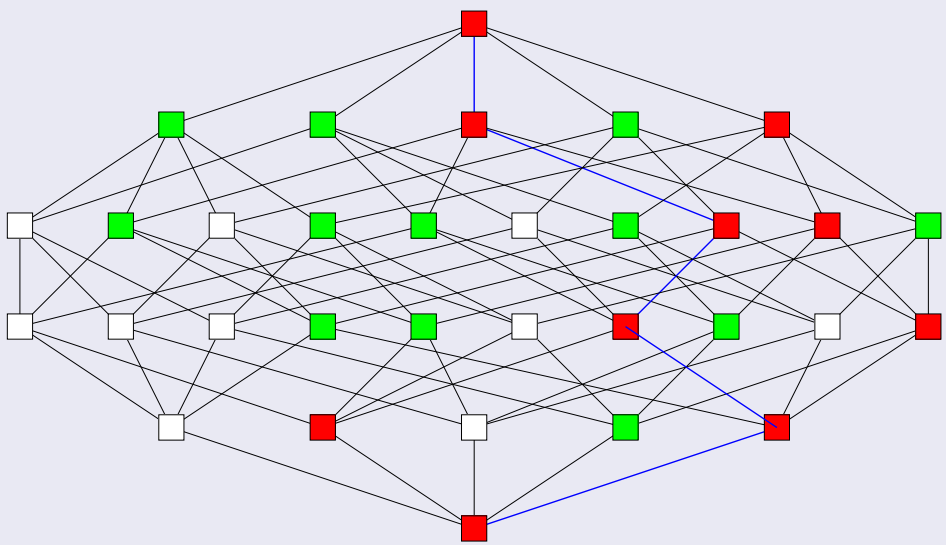
Energy barrier between two secondary structures along a folding path

- $B(p) = \max_{0 \leq i \leq n} E(p[i]) - E(p[0])$
- finding the lowest energy barrier for any path is too costly in terms of computational effort
- heuristics are applied to obtain a good estimate of the lowest barrier between s_1 and s_2

Flamm et al. approach

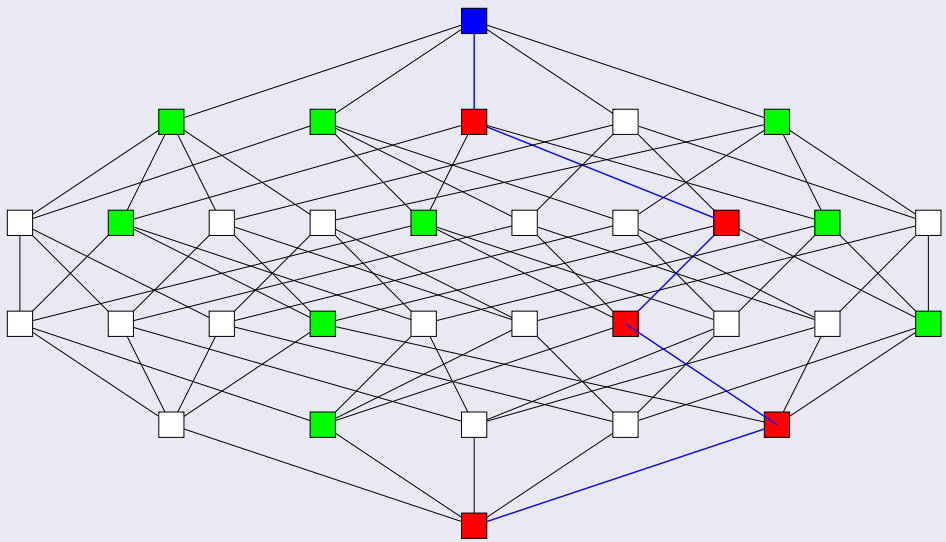
starting from s_1

1. generate all neighbors that are one step closer to the target structure
2. keep the best m neighbors
3. if target structure is not reached, proceed with (1) for each of the m neighbors



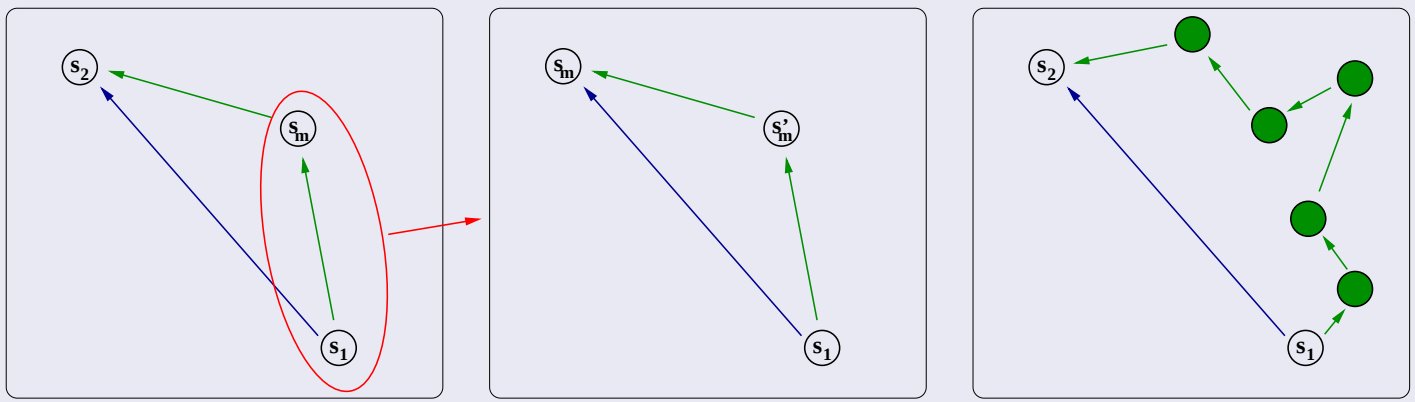
Morgan Higgs et al. approach

1. let the current structure A be s_1
2. find (i,j) in s_2 which has least incompatible pairs in A
3. remove incompatible pairs in A and insert (i,j)
4. if $A! = s_2$ go to (2)



Approach with detours

1. let $A = s_1$ and $B = s_2$
2. generate best direct folding path p_d between A and B
3. find x other structures s_{x_m} (mehspoint x)
4. generate best direct paths p_{x_1} from A to s_m and p_{x_2} from s_m to B
5. get complete paths $p_{x_c} = p_{x_1} + p_{x_2}$
6. accept if $B(p_{x_c}) < B(p_d)$
7. if deeper iteration required continue from (2) to level p_{x_1} and p_{x_2}
8. otherwise return p_{x_c} with lowest barrier



So, how to obtain meshpoint structures s_{x_m}?

Non-deterministic meshpoint generation

Monte Carlo with Metropolis rule

- take s_b that exhibits the barrier along the known path between s_1 and s_2
- start a random walk from s_b
- probability to move from structure s_i to one of its neighbors s_j:

$$p(s_i \rightarrow s_j) = \frac{1}{N} \begin{cases} exp^{-\beta \cdot (E(s_i) - E(s_j))} & \text{if } E(s_i) < E(s_j) \\ 1 & \text{else} \end{cases} \quad (1)$$

- probability to reject the next move at all p_r:

$$p_r = 1 - \sum_j p(s_i \rightarrow s_j) \quad (2)$$

- rejectionless selection of neighbor s_j with probability

$$p_{\mathrm{rl}}(s_i \rightarrow s_j) = \frac{1}{\sum_j p(s_i \rightarrow s_j)} \cdot p(s_i \rightarrow s_j) \quad (3)$$

Move penalties

- walks backward on trajectory can be penalized
- penalty contribution of $2 \cdot kT$ added to free energy of previously visited neighbor

Stop criterion

- if all neighbors show heigher free energy than current state, MC stops with probability

$$p_{\text{stop}} = 1 - e^{-\beta \cdot (E(s_i) - \min_j E(s_j))} \tag{4}$$

Simulated annealing

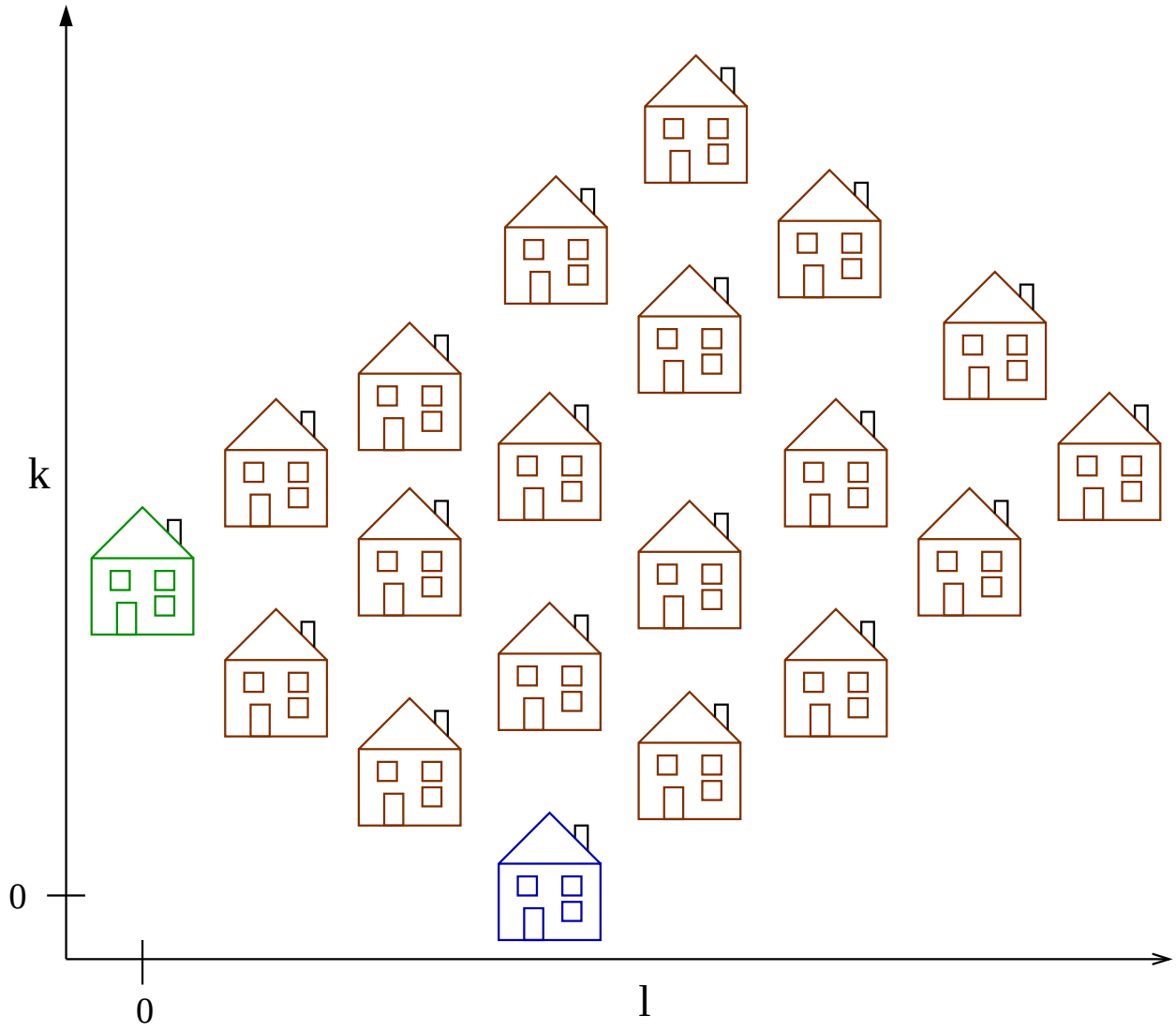
- given a start and stop temperatures t_{start}, t_{stop} and cooling rate r
- acceptance probabilites of next states is altered according to falling temperature of the system

Deterministic meshpoint generation

k,l - neighborhood of two RNA secondary structures

- similar to approach of finding optimal k-neighbors (Freyhult et al. 2007)
  a.k.a. `RNAbor`
- optimal k-neighbor is optimal secondary structure s_k with $d_{BP}(s_k, s_{\text{target}}) = k$
- extension to second target structure s_{target_2}
- optimal k, l-neighbor is optimal secondary structure s_{kl} with $d_{BP}(s_{kl}, s_{\text{target}}) = k$ and $d_{BP}(s_{kl}, s_{\text{target}_2}) = l$

k,l - neighborhood of two RNA secondary structures



Finding all optimal k,l neighbors

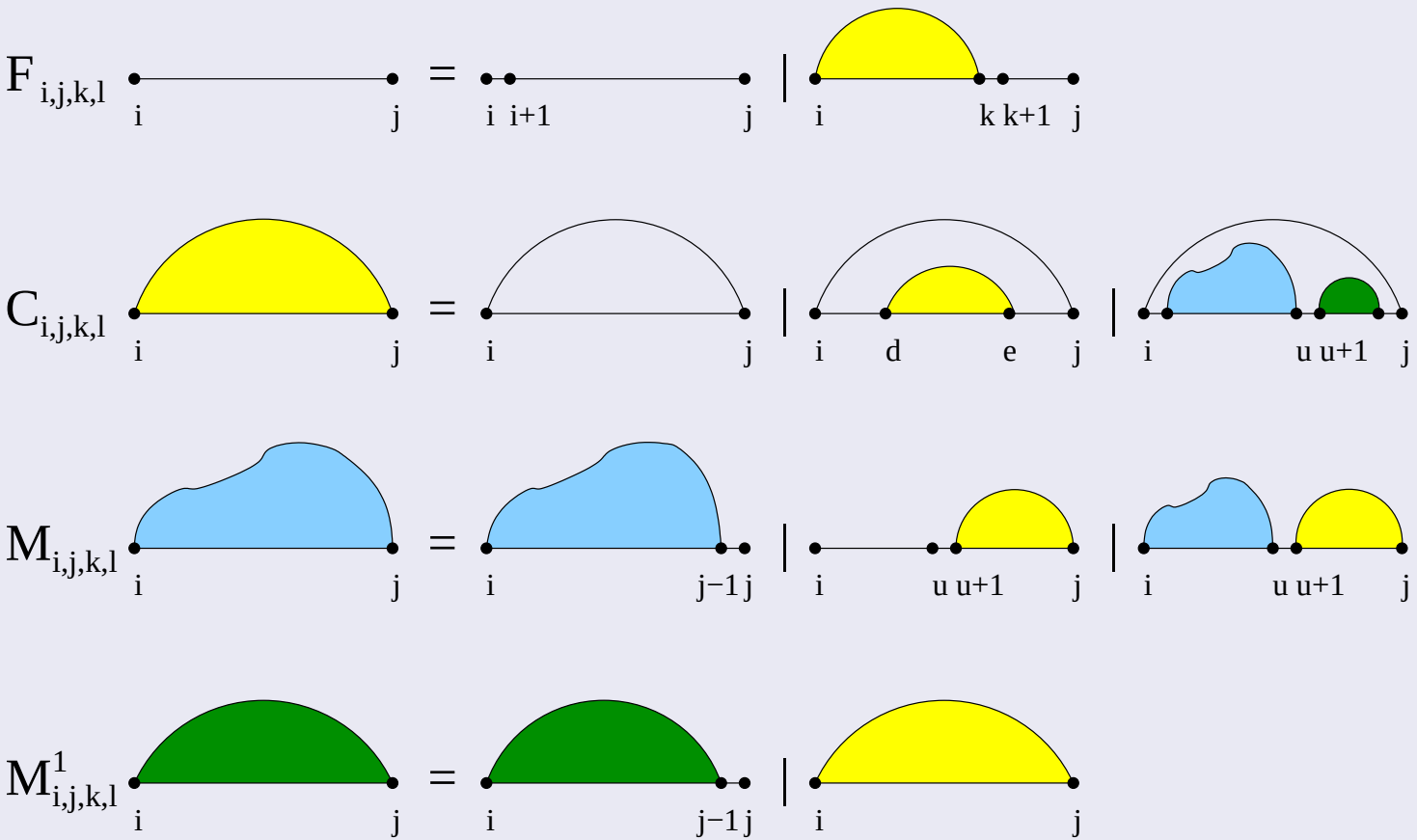
Definitions

$$
\begin{aligned}
d_{\mathrm{BP}}(s_1, s_2) &= |s_1 \cup s_2| - |s_1 \cap s_2| \\
s[i,j] &= \{(p,q) \in s : i \leq p < q \leq j\} \\
\delta(x,y) &= 1, \text{ iff } x = y \text{ (Kronecker function)} \\
\\
\lambda_1(i,j,s) &= d_{\mathrm{BP}}(s[i,j], s[i,j-1]) \\
\lambda_2(i,j,s) &= d_{\mathrm{BP}}(s[i,j], s[i,u] \cup s[u+1,j]) \\
\lambda_3(i,j,p,q,s) &= d_{\mathrm{BP}}(s[i,j], \{(i,j)\} \cup s[p,q]) \\
\lambda_4(i,j,u,s) &= d_{\mathrm{BP}}(s[i,j], \{(i,j)\} \cup s[i+1,u] \cup s[u+1,j-1]) \\
\lambda_5(i,j,u,s) &= d_{\mathrm{BP}}(s[i,j], s[u,j])
\end{aligned}
$$

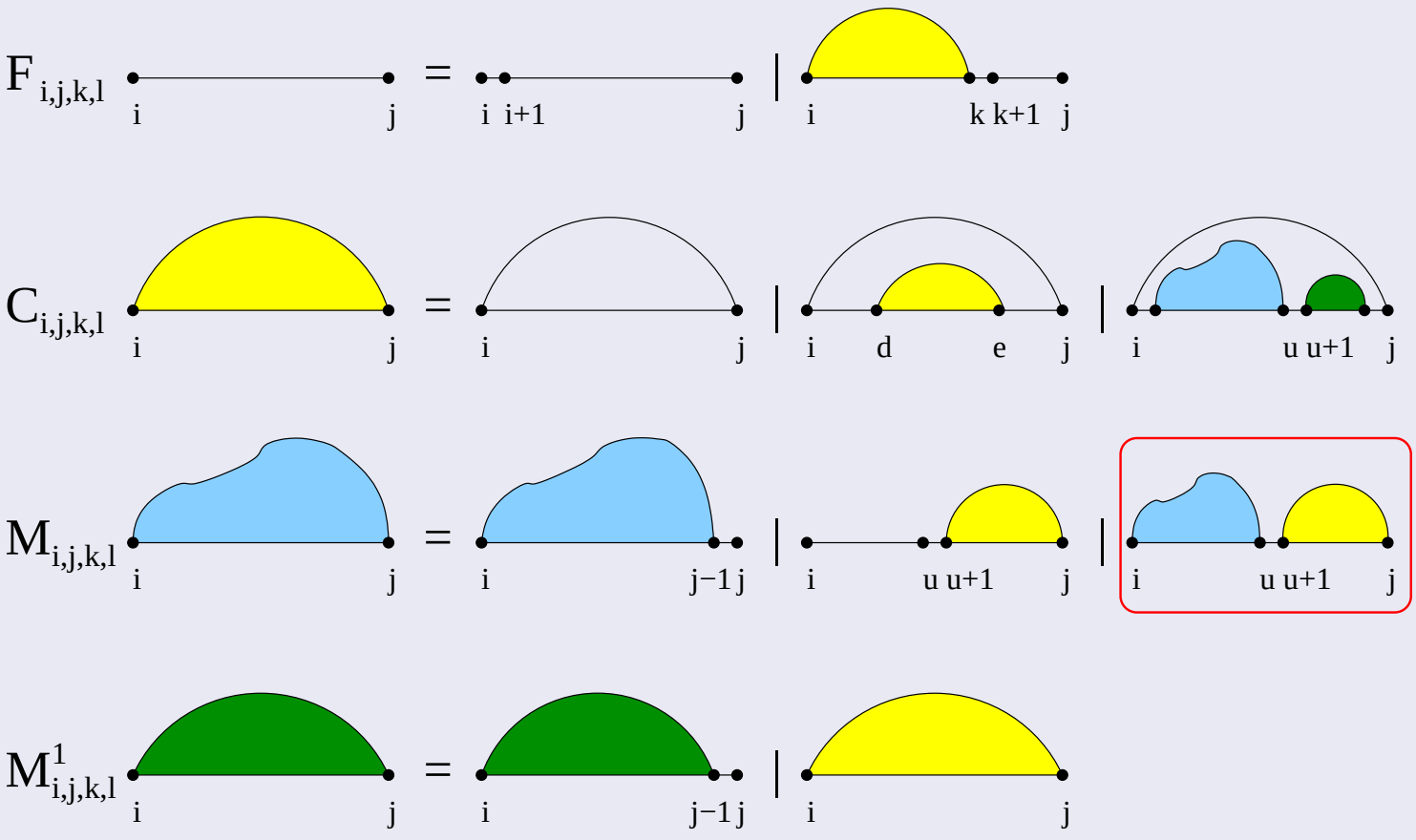
MFE Algorithm for finding all optimal k, l-neighbors

$$
\begin{aligned}
F_{i,j,k,l} &= \min \left\{ \begin{array}{l} F_{i+1,j,k-\lambda_1(i,j,s_1),l-\lambda_1(i,j,s_2)}, \\ \min\limits_{i<u\leq j} \ \min\limits_{\omega_1+\hat{\omega}_1=k-\lambda_2(i,j,u,s_1)} \ \min\limits_{\omega_2+\hat{\omega}_2=l-\lambda_2(i,j,u,s_2)} C_{i,u,\omega_1,\omega_2} + F_{u+1,j,\hat{\omega}_1,\hat{\omega}_2} \end{array} \right. \\
C_{i,j,k,l} &= \min \left\{ \begin{array}{l} \delta(d_{\mathrm{BP}}(s_1[i,j],\{(i,j)\})) \cdot \delta(d_{\mathrm{BP}}(s_2[i,j],\{(i,j)\})) \cdot \mathcal{H}(i,j), \\ \min\limits_{i<p<q<j} \left\{ C_{p,q,k-\lambda_3(i,j,p,q,s_1),l-\lambda_3(i,j,p,q,s_2)} + \mathcal{I}(i,j,p,q) \right\}, \\ \min\limits_{i<u<j} \ \min\limits_{\omega_1+\hat{\omega}_1=k-\lambda_4(i,j,u,s_1)} \ \min\limits_{\omega_2+\hat{\omega}_2=l-\lambda_4(i,j,u,s_2)} \left\{ M_{i+1,u,\omega_1,\omega_2} + M^1_{u+1,j-1,\hat{\omega}_1,\hat{\omega}_2} + a \right\} \end{array} \right. \\
M_{i,j,k,l} &= \min \left\{ \begin{array}{l} \min\limits_{i\leq u<j} \left\{ (u-i)\cdot c + C_{u,j,k-\lambda_5(i,j,u,s_1),l-\lambda_5(i,j,u,s_2)} + b \right\}, \\ \min\limits_{i<u<j} \ \min\limits_{\omega_1+\hat{\omega}_1=k-\lambda_2(i,j,u,s_1)} \ \min\limits_{\omega_2+\hat{\omega}_2=l-\lambda_2(i,j,u,s_2)} \left\{ M_{i,u,\omega_1,\omega_2} + C_{u+1,j,\hat{\omega}_1,\hat{\omega}_2} + b \right\}, \\ M_{i,j-1,k-\lambda_1(i,j,s_1),l-\lambda_1(i,j,s_2)} + c \end{array} \right. \\
M^1_{i,j,k,l} &= \min \left\{ \begin{array}{l} M^1_{i,j-1,k-\lambda_1(i,j,s_1),l-\lambda_1(i,j,s_2)} + c, \\ C_{i,j,k,l} + b \end{array} \right.
\end{aligned}
$$

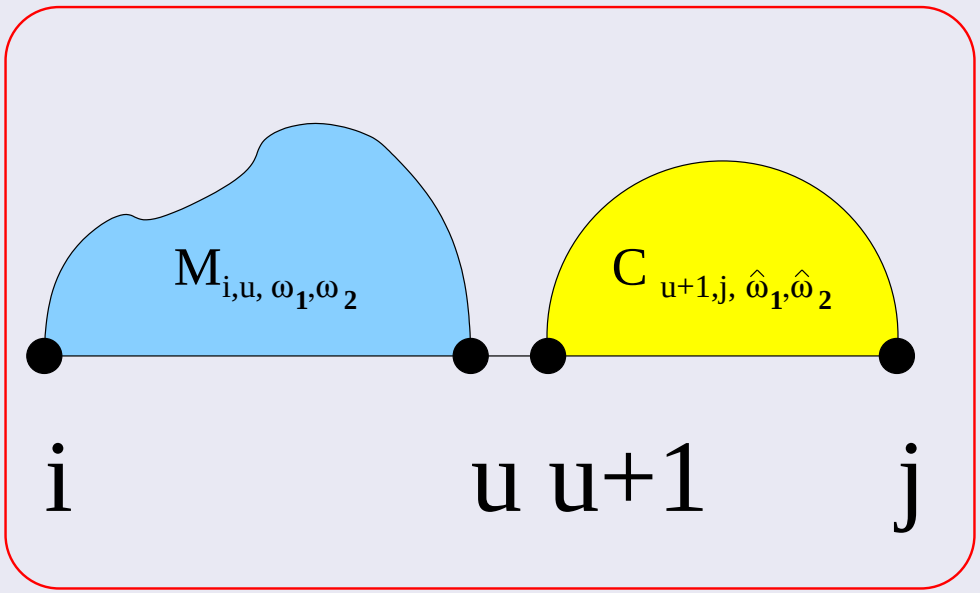
Recursion scheme for finding k, l-neighbors



Recursion scheme for finding k, l-neighbors



Recursion scheme for finding k, l-neighbors



$$\omega_1 + \hat{\omega}_1 = k - \lambda_2(i,j,u,s_1) = k - d_{\mathrm{BP}}(s_1[i,j], s_1[i,u] \cup s_1[u+1,j])$$
$$\omega_2 + \hat{\omega}_2 = l - \lambda_2(i,j,u,s_2) = l - d_{\mathrm{BP}}(s_2[i,j], s_2[i,u] \cup s_2[u+1,j])$$

Output of RNA2Dfold

```
AGCCAGCCAGCCUGUAGCCCUCAAUAAAAGGCAGCUGCCUCUGCUCCCCAU
((.(((((..((((.............)))).)))))..)).........  (-10.06)
free energy of ensemble = -11.49 kcal/mol
((.(((((..((((.............)))).)))))..)).........  <target 1>
..................................................  <target 2>
```

0	10	0.09861506	1.00000000	0.09861513	-10.06	-10.06	((.(((((..((((.............)))).)))))..))..........
1	9	0.06603776	0.91781333	0.06061030	-9.76	-9.81	.(.(((((..((((.............)))).)))))..)...........
1	11	0.01572989	0.49883057	0.00784655	-8.50	-8.93	((.(((((..((((..(........)...)))).)))))..))..........
2	8	0.04061951	0.91709499	0.03725198	-9.46	-9.51	...(((((..((((.............)))).)))))...............
2	10	0.01190747	0.40500697	0.00482261	-8.20	-8.76	.(.(((((..((((..(........)...)))).)))))..)...........
2	12	0.02935243	0.83232812	0.02443084	-9.20	-9.31	((.(((((..((((.((........)).)))).)))))..))..........
3	7	0.00286429	0.56773426	0.00162616	-7.53	-7.88	...(((((..(((.................)))..))))).............
3	9	0.04069049	0.77837766	0.03167258	-9.36	-9.51	.(.(((((..((((.............)))).)))))..)...........
3	11	0.02578489	0.58233897	0.01501555	-8.90	-9.23	.(.(((((..((((.((........)).)))).)))))..)...........
3	13	0.00086728	0.41464082	0.00035961	-6.60	-7.14	((.(((((..((((((.(....).))...)))).)))))..))..........
4	6	0.00022925	0.55531321	0.00012731	-5.96	-6.32	.....((..((((.............)))).)).................
4	8	0.00263955	0.52380121	0.00138260	-7.43	-7.83	.(.(((((..(((.................)))..))))))..........
4	10	0.07164758	0.61152396	0.04381424	-9.56	-9.86	.((((((..((((.............)))).))))))....)).......
4	12	0.00253061	0.31984127	0.00080939	-7.10	-7.80	((.(((((..((((((........)))...)))).)))))..))..........
4	14	0.00041320	0.87030617	0.00035961	-6.60	-6.69	((.(((((..((((.((........)).)))).)))))..)).((....)).
5	5	0.00000856	0.64958465	0.00000556	-4.03	-4.30	.....((..(((.................))).)).................
5	7	0.00012481	0.30207706	0.00003770	-5.21	-5.95	.(.(((((..(((.................))).))))))..........
5	9	0.03880912	0.81611104	0.03167258	-9.36	-9.49	.((((((..((((.............)))).))))))....)).......
5	11	0.09057879	0.78701893	0.07128722	-9.86	-10.01	((((((((..((((.............)))).))))))....)).......
5	13	0.00053389	0.41398274	0.00022102	-6.30	-6.84	.(.(((((..((((.((........)).)))).)))))..)..((....)).
5	15	0.00000622	0.85088938	0.00000529	-4.00	-4.10	((.(((((..((((((.(....).))...)))).)))))..)).((....)).
6	4	0.00003512	0.98976507	0.00003476	-5.16	-5.17	.........((((.............))))...................
6	6	0.00000461	0.29377326	0.00000135	-3.16	-3.91	.....(.(((((.............)))).)).................
6	8	0.00197876	0.69871921	0.00138260	-7.43	-7.65	.((.(((((..(((.................)))..))))))..........
6	10	0.01330700	0.23385391	0.00311189	-7.93	-8.83	((((((((..(((.................)))..))))))....)).......
6	12	0.02565282	0.42313124	0.01085451	-8.70	-9.23	.((((((..((((.((........)).)))).))))))....)).......
6	14	0.00003311	0.35984238	0.00001191	-4.50	-5.13	((.(((((..((((((........)))...)))).)))))..)).((....)).
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
22	14	0.00000004	0.65243047	0.00000003	-0.70	-0.96	(((.((.(((((((.(....).)))...)...)))))..))..)).....
23	13	0.11005782	0.30707807	0.03379634	-9.40	-10.13	....(((.((...(((((((.........)))...)))))..))))).....
23	15	0.00000000	0.52881538	0.00000000	9.40	9.01	(((.((.(((((((.(....).)))...)...)))))..))..))(...))
24	14	0.04966769	0.18581086	0.00922879	-8.60	-9.64	.(..(((.((...(((((((.........)))..)))))..)))))..)...
25	15	0.00140967	0.48817756	0.00068817	-7.00	-7.44	(((.((..((...(((((((.........)))..)))))))))))).....)
26	16	0.00000000	0.49744775	0.00000000	3.10	2.67	(((.((..((...(((((((.........)))..)))))))))))(....))

Improving runtime properties of the algorithm

Time complexity

- finding all optimal k, l-neighbors with $k \leq d_1$ and $l \leq d_2$: $\mathcal{O}(d_1 \cdot d_2 \cdot n^3)$
- $d_x \leq MM(seq) + |s_x|$
- $MM(seq) < n/2 - 1$ and $|s_x| < n/2 - 1$
- $\mathcal{O}(n^5)$

Memory complexity

- finding all optimal k, l-neighbors with $k \leq d_1$ and $l \leq d_2$: $\mathcal{O}(m \cdot n^2)$, $m = \frac{1}{2} \cdot d_1 \cdot d_2$
- checkerboard like memory fill pattern
- $\mathcal{O}(n^4)$

Improving runtime properties of the algorithm

Observations

straight implementation results:

- more than 3.5h runtime for sequence of length $100nt$
- more than 30GB RAM for sequences with length $> 200nt$

Improving runtime properties of the algorithm

Observations

straight implementation results:

- more than 3.5h runtime for sequence of length $100nt$
- more than 30GB RAM for sequences with length $> 200nt$

BUT: Energy arrays are filled by less than 2%

Improving runtime properties of the algorithm

Observations

straight implementation results:

- more than 3.5h runtime for sequence of length $100nt$
- more than 30GB RAM for sequences with length $> 200nt$

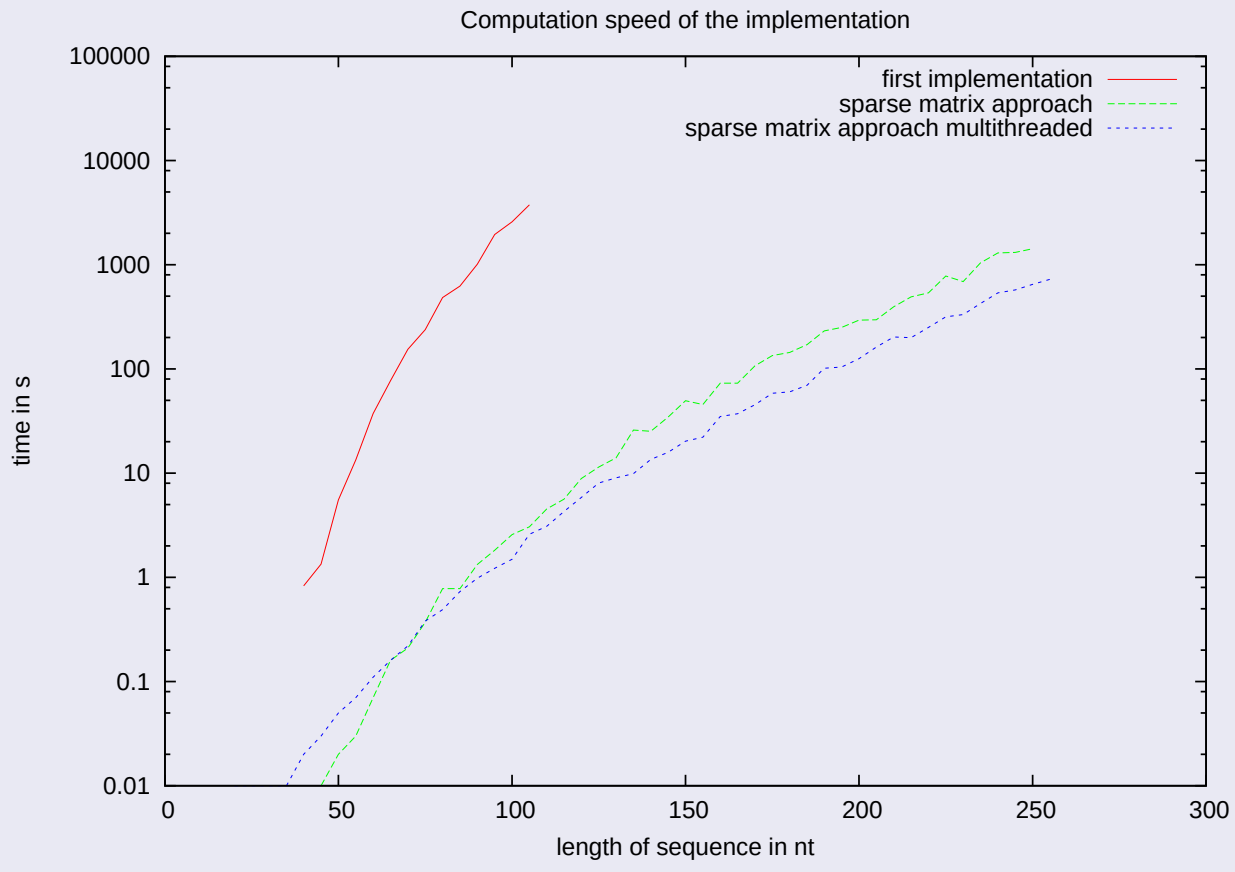
BUT: Energy arrays are filled by less than 2%

Improvements

- operating with sparse matrices to lower runtime and memory requirements
- parallelization of recursive algorithm (parallel computation of elements in diagonals of energy matrices)

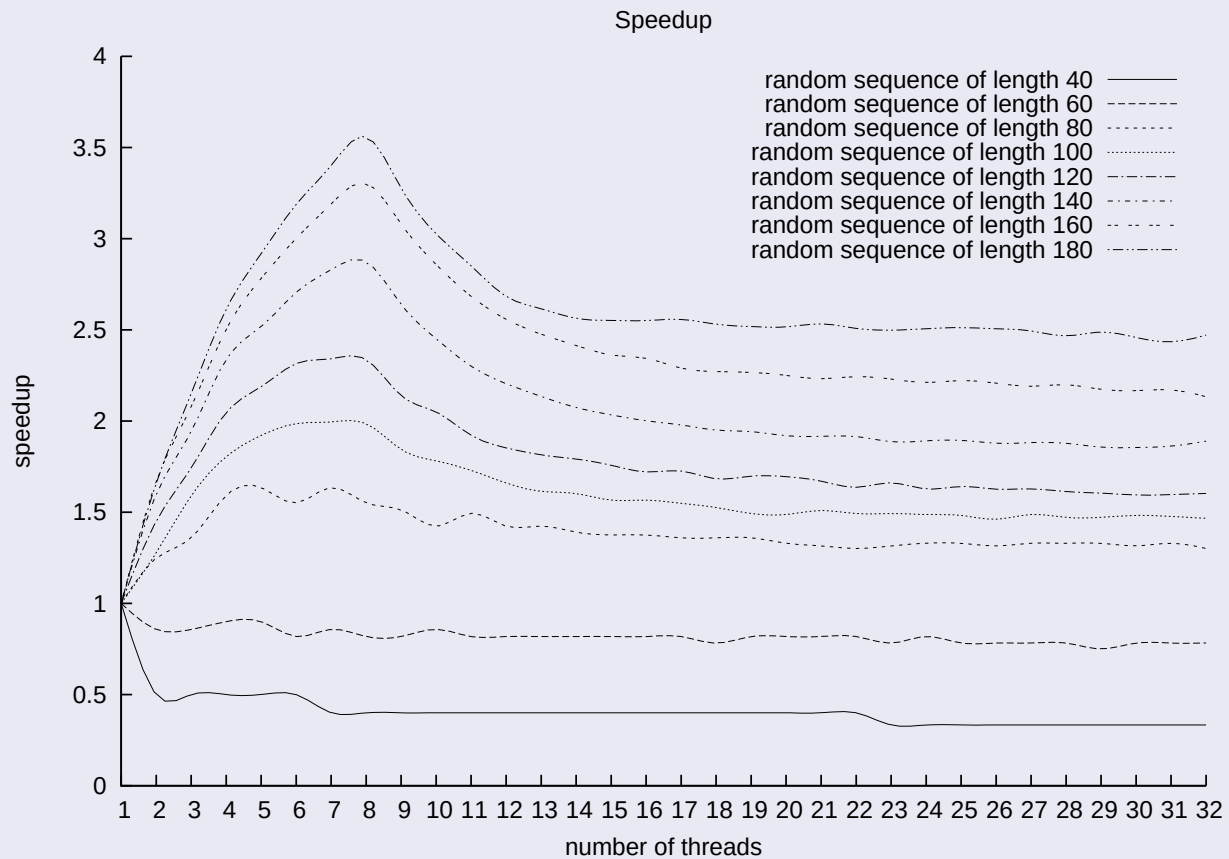
Improving runtime properties of the algorithm

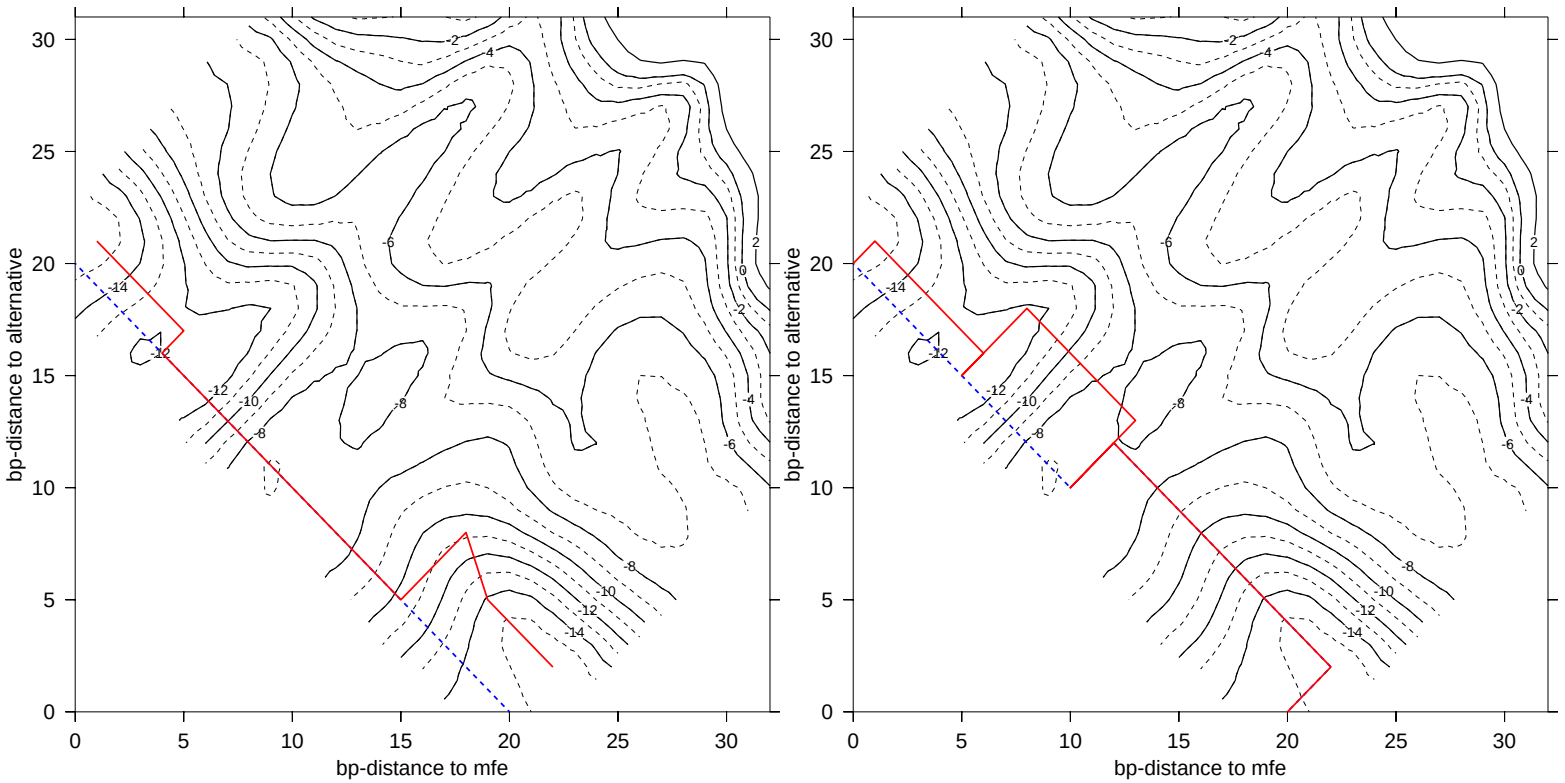
Stats



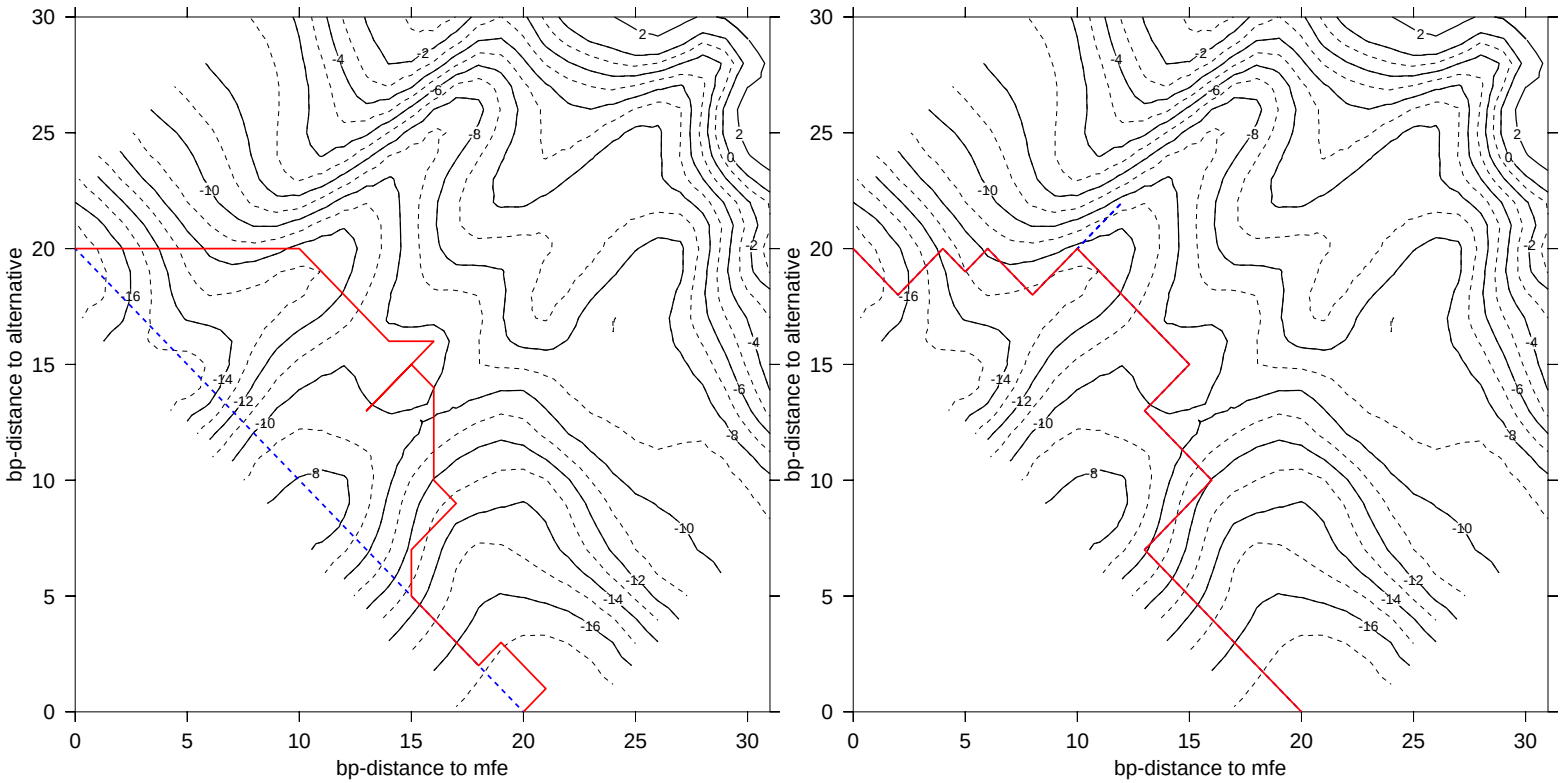
Improving runtime properties of the algorithm

Stats

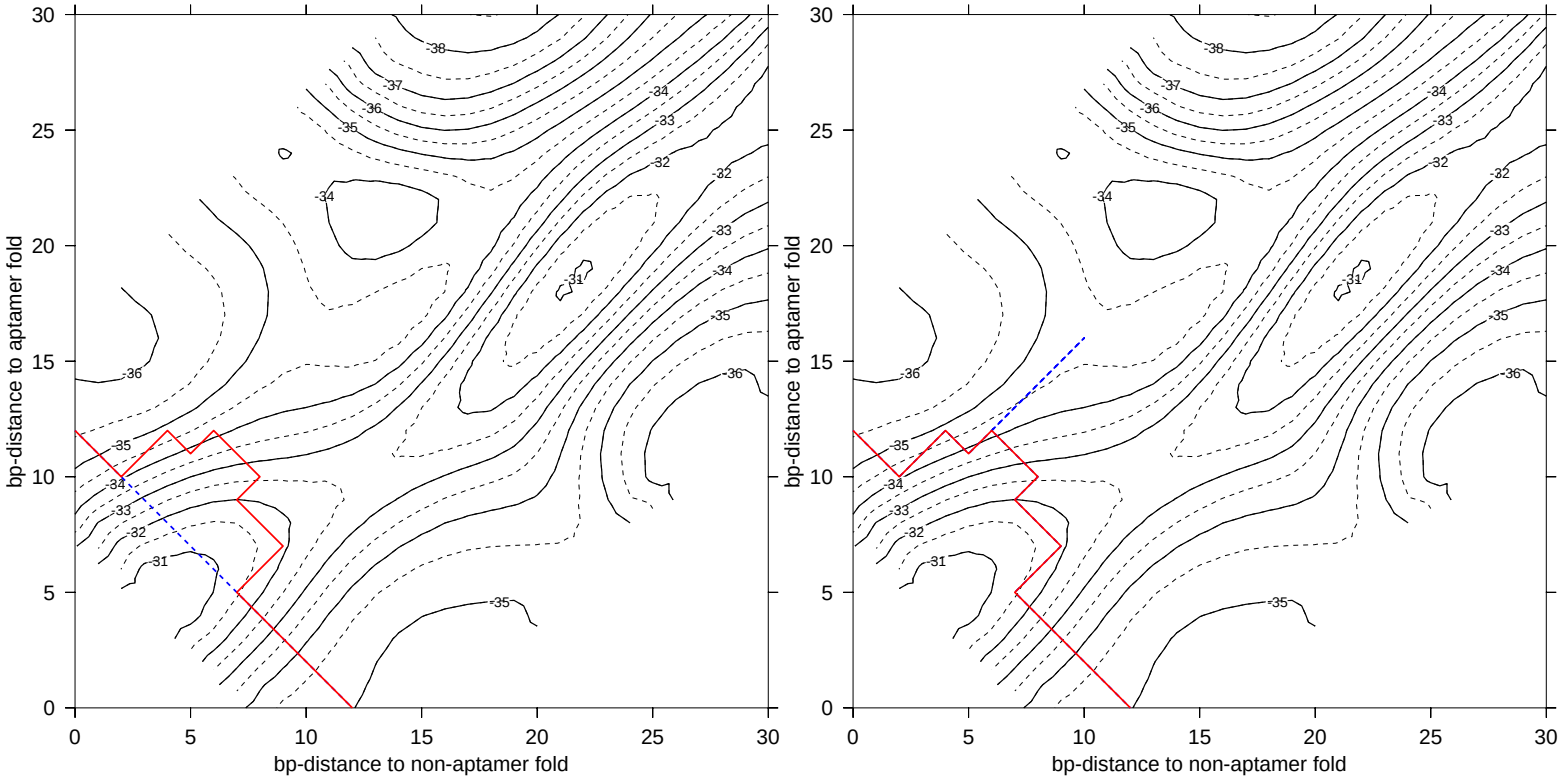




- GGCUGUAUGGCAGCUGCCUCGUUAAGAGGUGAUUACUAUGUAGUC
  (((((.....)))))(((((.....)))))(((((.....))))) -15.40 *(MFE structure)*
  ((((((((((.....(((((.....)))))).....)))))))))) -14.70 *(alternative structure)*
- Height field contours: Gibbs free energy of the corresponding k, l-neighborhood
- Left side: Trajectories of
  (barrier tree generated path) 9.6 kcal/mol (−5.8 kcal/mol)
  (direct folding path) 10.5 kcal/mol (−4.9 kcal/mol)
- Right side: Trajectories of
  (Monte carlo method (SA)) 9.8 kcal/mol (−5.8 kcal/mol)
  (Deterministic method) 9.6 kcal/mol (−5.8 kcal/mol)



- GGGCGCGGUUCGCCCUCCGCUAAAUGCGGAAGAUAAAUUGUGUCU
  (((((.....)))))(((((.....)))))(((((.....))))) -18.20 *(MFE structure)*
  ((((((((((.....(((((.....)))))).....)))))))))) -17.70 *(alternative structure)*
- Height field contours: Gibbs free energy of the corresponding k, l-neighborhood
- Left side: Trajectories of
  (barrier tree generated path) 10.7 kcal/mol (−7.5 kcal/mol)
  (direct folding path) 13.33 kcal/mol (− − 4.87 kcal/mol)
- Right side: Trajectories of
  (Monte carlo method (SA)) 10.7 kcal/mol (−7.5 kcal/mol)
  (Deterministic method) 10.7 kcal/mol (−7.5 kcal/mol)



- GCUUCAUAUAAUCCUAAUGAUAUGGURUGGGAGUUUCUACCAAGAGCCUUAAACUCUUGAUUAUGAAGUCUGUCGCUUUAUCCGAAAUUUUAUAAAGAGAAGACUCAUGA
  ..........((((((.....)).)))).....((((((.....))))))..((((.((((((.((.((((((.........)))))).)))))).))))))
  -36.03 *(non-aptamer fold)*
  ..((((((...((((((.....)).)))).....((((((.....))))))..))))))((((.((.((((((.........)))))).)))))).....
  -35.32 *(aptamer fold)*
- Height field contours: Gibbs free energy of the corresponding k, l-neighborhood
- Left side: Trajectories of
  (barrier tree generated path) 6.77 kcal/mol (−29.26 kcal/mol)
  (direct folding path) 7.28 kcal/mol (−28.75 kcal/mol)
- Right side: Trajectories of
  (Monte carlo method (SA)) 6.77 kcal/mol (−29.26 kcal/mol)
  (Deterministic method) 6.77 kcal/mol (−29.26 kcal/mol)

Thanks to:

Stephan Bernhart
Ilenia Boria
Christoph Flamm
Andreas Gruber
Christian Höner
Ivo Hofacker
Andrea Tanzer
Caroline Thurner
Alexander Ullrich

The audience for listening