Example:

Sequence:
CUCGACCCCCUUGACGUCGCAAGUGUUGGGG

Ground truth:

bb_x	bb_y	siz_x	siz_y	bb_type
0	29	6	6	stem
9	22	4	4	stem
5	24	5	3	internal_loop
12	19	8	8	hairpin_loop

Proposals (at 0.1 threshold):
6 stems, 18 iloops, 3 hloops

Step 1: Filter out invalid bounding boxes

For iloop, we need at least one compatible stem on each side, i.e. iloop.top_right == some_stem1.bottom_left and iloop.bottom_left == some_stem2.top_right. Iloop not satisfying this is discarded.

For hloop, we need at least one compatible stem on the outer side, i.e. hloop.top_right == some_stem.bottom left. Also hloop needs to be on the diagonal of the matrix. Hloop not satisfying the above is discarded.

Result: after filtering, we're left with much fewer bounding boxes:

Stem

bb_x	bb_y	siz_x	siz_y	prob	bl_x	bl_y
0	29	6	6	[0.41025621353158326, 0.44290844275857927, 0.2...	5	24
5	30	4	4	[0.20544070627534108]	8	27
6	30	4	4	[0.4038486250672587, 0.27864132850155837, 0.27...	9	27
9	22	4	4	[0.6770828362027589, 0.573858923373826, 0.8764...	12	19
12	26	4	4	[0.09296749466427809]	15	23
13	23	2	2	[0.4102197996540831, 0.32933917103708865, 0.41...	14	22

iloop

bb_x	bb_y	siz_x	siz_y	prob	bl_x	bl_y	num_compatible_stem_outer	num_compatible_stem_inner
5.0	24.0	5.0	3.0	[0.22653995386454878, 0.4762805845032449, 0.40...	9.0	22.0	1.0	1.0
8.0	27.0	2.0	6.0	[0.21078391904750435]	9.0	22.0	1.0	1.0
8.0	27.0	5.0	2.0	[0.05582155712225877]	12.0	26.0	1.0	1.0
9.0	27.0	5.0	5.0	[0.10652713805518142, 0.0927118477820101]	13.0	23.0	1.0	1.0

hloop

bb_x	bb_y	siz_x	siz_y	prob	bl_x	bl_y	num_compatible_stem_outer
12	19	8.0	8.0	[0.5592991751604471, 0.6358038212924311, 0.709...	19.0	12.0	1.0
14	22	9.0	9.0	[0.5634663251435189, 0.2603944252215587, 0.666...	22.0	14.0	1.0

Step 2: Extend each bounding box by one step, towards the outer direction

For each bounding box, we find all compatible "next" bounding boxes where bounding_box.top_right == next.bottom_left. Following local constraints need to be satisfied:

hloop.next = stem
iloop.next = stem
stem.next = iloop or None

Result: each bounding box is associated with a list (can be empty) of other bounding boxes as candidate "next":

```
{'iloop_0': iloop iloop_0 top right (5.0, 24.0), bottom left (9.0, 22.0). Next: ['stem_0'],
 'iloop_1': iloop iloop_1 top right (8.0, 27.0), bottom left (9.0, 22.0). Next: ['stem_1'],
 'iloop_2': iloop iloop_2 top right (8.0, 27.0), bottom left (12.0, 26.0). Next: ['stem_1'],
 'iloop_3': iloop iloop_3 top right (9.0, 27.0), bottom left (13.0, 23.0). Next: ['stem_2'],
 'stem_0': stem stem_0 top right (0, 29), bottom left (5, 24). Next: N/A,
 'stem_1': stem stem_1 top right (5, 30), bottom left (8, 27). Next: N/A,
 'stem_2': stem stem_2 top right (6, 30), bottom left (9, 27). Next: N/A,
 'stem_3': stem stem_3 top right (9, 22), bottom left (12, 19). Next: ['iloop_0', 'iloop_1'],
 'stem_4': stem stem_4 top right (12, 26), bottom left (15, 23). Next: ['iloop_2'],
 'stem_5': stem stem_5 top right (13, 23), bottom left (14, 22). Next: ['iloop_3'],
 'hloop_0': hloop hloop_0 top right (12, 19), bottom left (19.0, 12.0). Next: ['stem_3'],
 'hloop_1': hloop hloop_1 top right (14, 22), bottom left (22.0, 14.0). Next: ['stem_5']}
```

Step 3: Assemble a full "chain"

Since all the one-step parent-child relationship is given from step 2, in this step, we enumerate all possible short and long chains subject to constraint:

- start/root can be hloop or stem
- end can only be stem

Note that a particular chain can be a subset of others, since we're enumerating.

Also note that this can be implemented efficiently, since if all possible chains have been constructed for bounding box A, and we're trying to grow a new chain whose last element is A, then we can stitch the already known chains (starting from A) to the current one without exploration. (to be implemented)

Result:

```
[FullChain ['stem_0'] Completed,
 FullChain ['stem_1'] Completed,
 FullChain ['stem_2'] Completed,
 FullChain ['stem_3'] Completed,
 FullChain ['stem_3', 'iloop_0', 'stem_0'] Completed,
 FullChain ['stem_3', 'iloop_1', 'stem_1'] Completed,
 FullChain ['stem_4'] Completed,
 FullChain ['stem_4', 'iloop_2', 'stem_1'] Completed,
 FullChain ['stem_5'] Completed,
 FullChain ['stem_5', 'iloop_3', 'stem_2'] Completed,
 FullChain ['hloop_0', 'stem_3'] Completed,
 FullChain ['hloop_0', 'stem_3', 'iloop_0', 'stem_0'] Completed,
 FullChain ['hloop_0', 'stem_3', 'iloop_1', 'stem_1'] Completed,
 FullChain ['hloop_1', 'stem_5'] Completed,
 FullChain ['hloop_1', 'stem_5', 'iloop_3', 'stem_2'] Completed]
```

Note that in the above, we already recovered the ground truth solution:
hloop_0 top right (12, 19), bottom left (19.0, 12.0),
stem_3 top right (9, 22), bottom left (12, 19),
iloop_0 top right (5.0, 24.0), bottom left (9.0, 22.0),
stem_0 top right (0, 29), bottom left (5, 24))
(due to the fact that this is a short sequence with only one "chain")

Step 4: Global assembly and filter w.r.t. global constraints

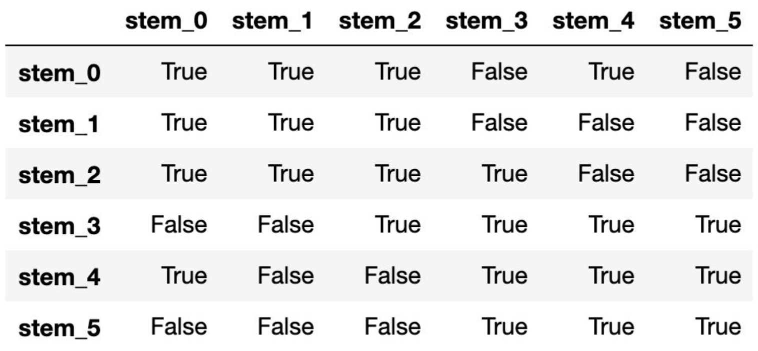
Given the short and long chains from step 3, in this step we create all valid global assemblies. Although in theory the number of combinations is 2^n where n is the number of chains, in practise it's far less, due to these constraints and considerations:

- Chain A and chain B are not compatible if there exists one stem in A that has base pair conflict with one stem in B. Since we can pre-compute the pairwise bp pair compatibility between all stems, this is easy to check.

- In a global assembly, for a particular chain, if the innermost bounding box is a stem, then it’s implied it connects to a multi-loop but NOT a hairpin loop, since hloop is not in the chain. This means there exists some base pairing between i and j, where (i,j) is the closing base of the stem. We can check this is the case and discard those not satisfying this.

This part is WIP.

Stem bp conflict:



	stem_0	**stem_1**	**stem_2**	**stem_3**	**stem_4**	**stem_5**
stem_0	True	True	True	False	True	False
stem_1	True	True	True	False	False	False
stem_2	True	True	True	True	False	False
stem_3	False	False	True	True	True	True
stem_4	True	False	False	True	True	True
stem_5	False	False	False	True	True	True

True: conflict (thus all diagonals are True since we can’t have the same stem bounding box twice)

Step 5: Ranking of global structures (model? Heuristic using stage 1 probability?)

WIP