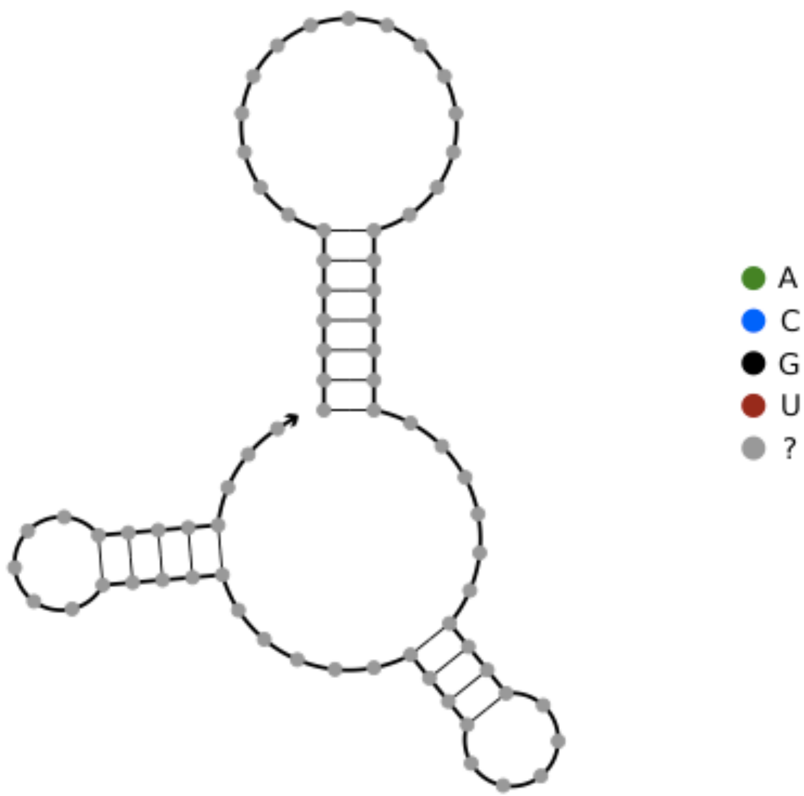
EteRNA-RL: Designing RNA secondary structures with reinforcement learning

Isaac Kauvar, Ethan Richman, Will Allen

The Problem

Specify an RNA sequence that will fold into a desired secondary structure.

Applications include:
- genetic tool design
- drug discovery



Why use reinforcement learning?
- There is evidence that humans can perform well at sequentially optimizing a structure.
- Can we train an agent that learns ‘intuition’ about good sequences?

Environment

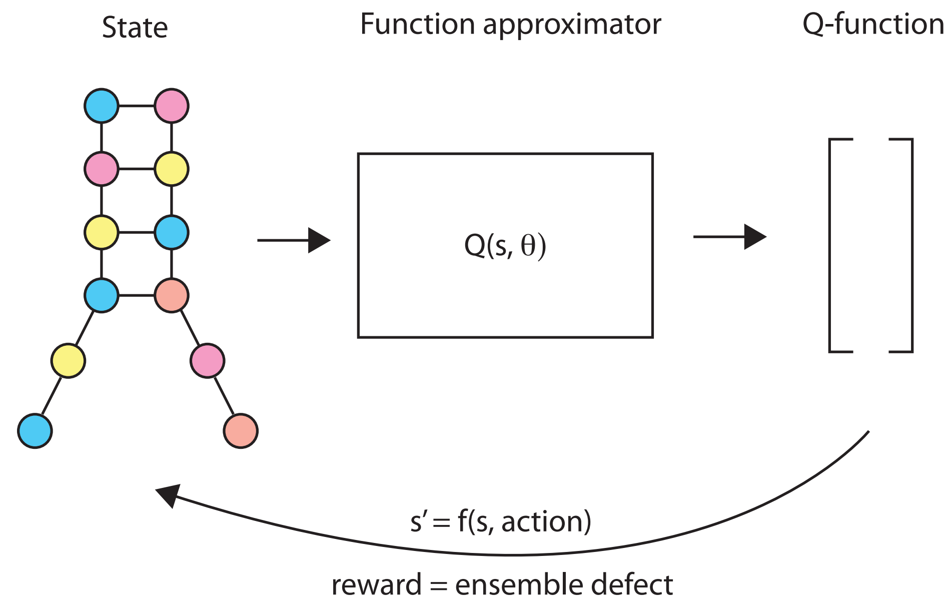
Screenshot of EteRNA computer game:

Ensemble Defect - the reward function:

'Average number of nucleotides that are incorrectly paired at equilibrium relative to the specified secondary structure.'

- Calculated using Nucleic Acid Package (NUPACK)

Algorithm



$Q(s, \theta)$: fully connected, ReLu activation, 1-5 layers

Loss function:

$$L = r + \gamma \max_{a'} Q(s', a', w^-) - Q(s, a, w)$$
$$\Delta w = \alpha(L) \nabla_w Q(s, w)$$

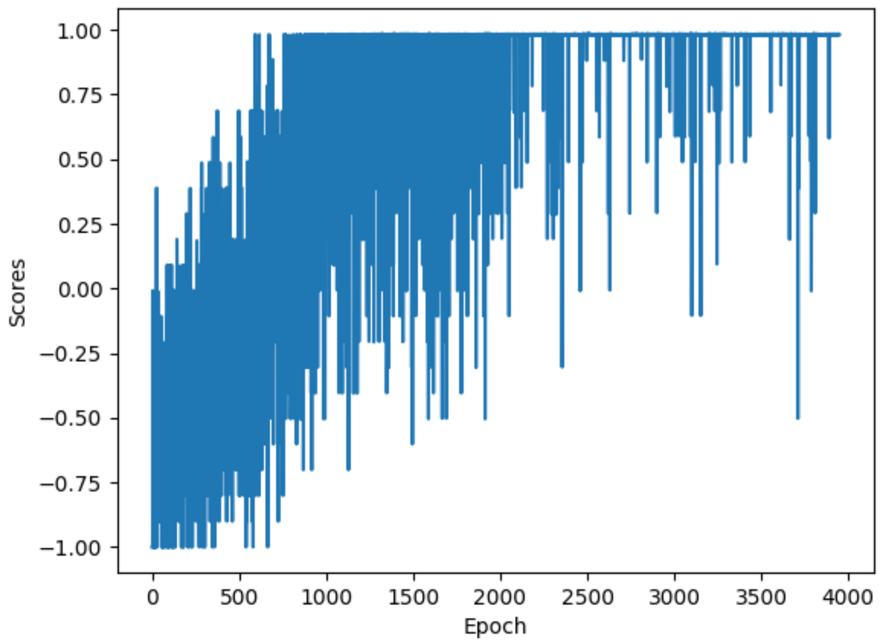
Transition model:

$$f(s, a) : \textbf{set } s[i] = x \in [1, 4]$$

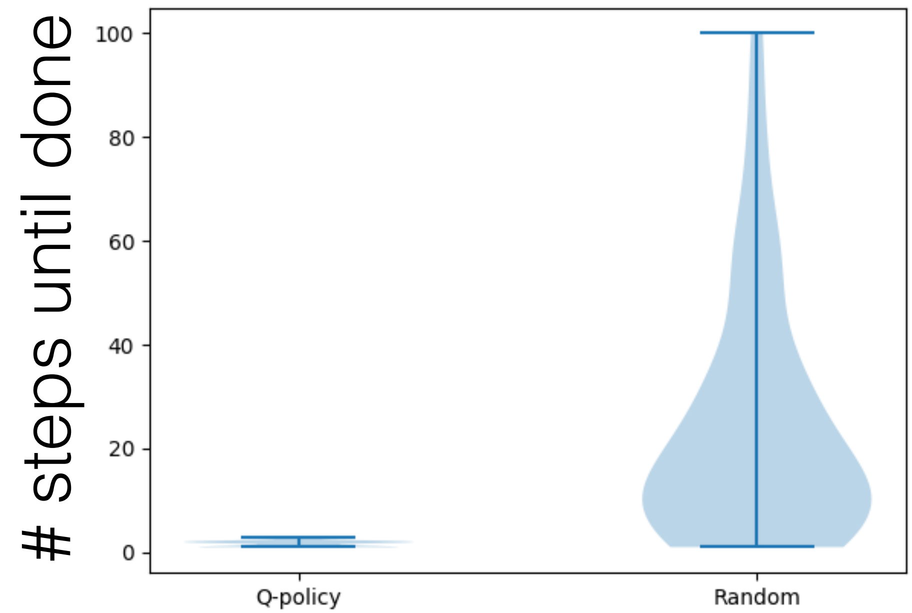
Result #1 - Simple state space, simple reward function

Setup:
- sequence length n=3
- reward = 1 if coloring matches predefined target coloring for a given adjacency matrix, else reward = 0
- multiple target colorings for each adjacency matrix

Convergence



Versus random policy

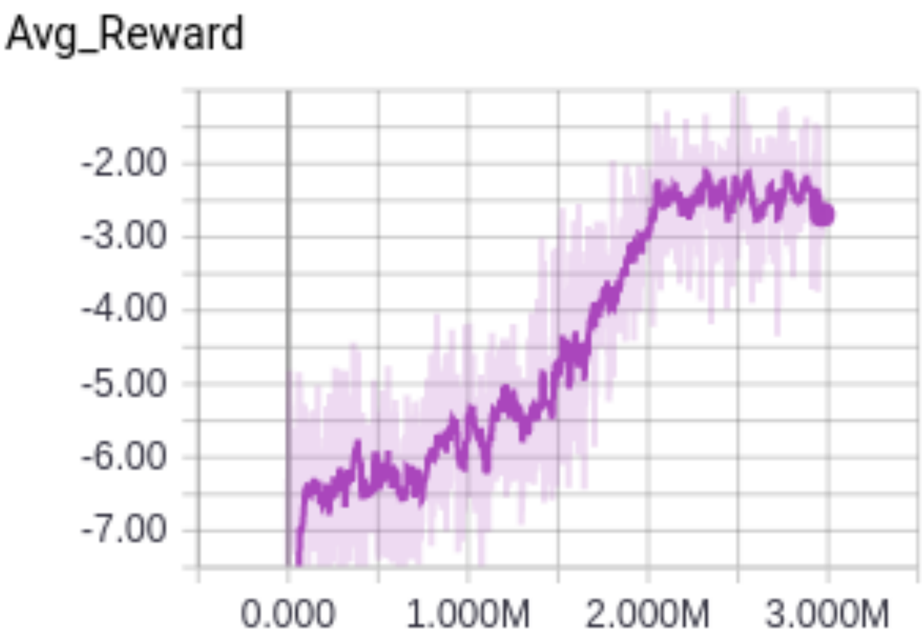


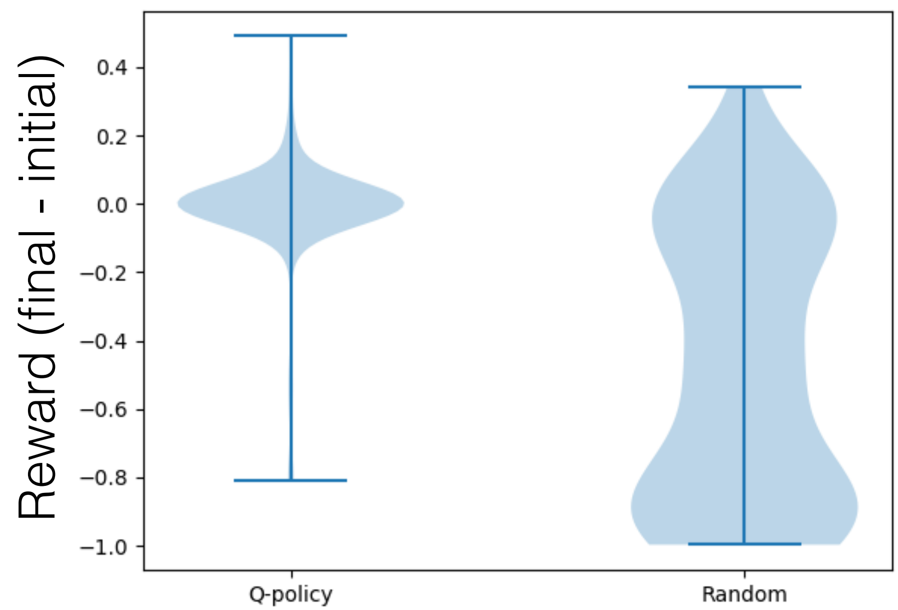
Conclusion: RL successfully learned a policy for reaching target in fewest possible number of steps.

Result #2 - Larger state space, realistic reward

Setup:

- sequence length n=20
- reward = ensemble defect of a coloring given an adjacency matrix

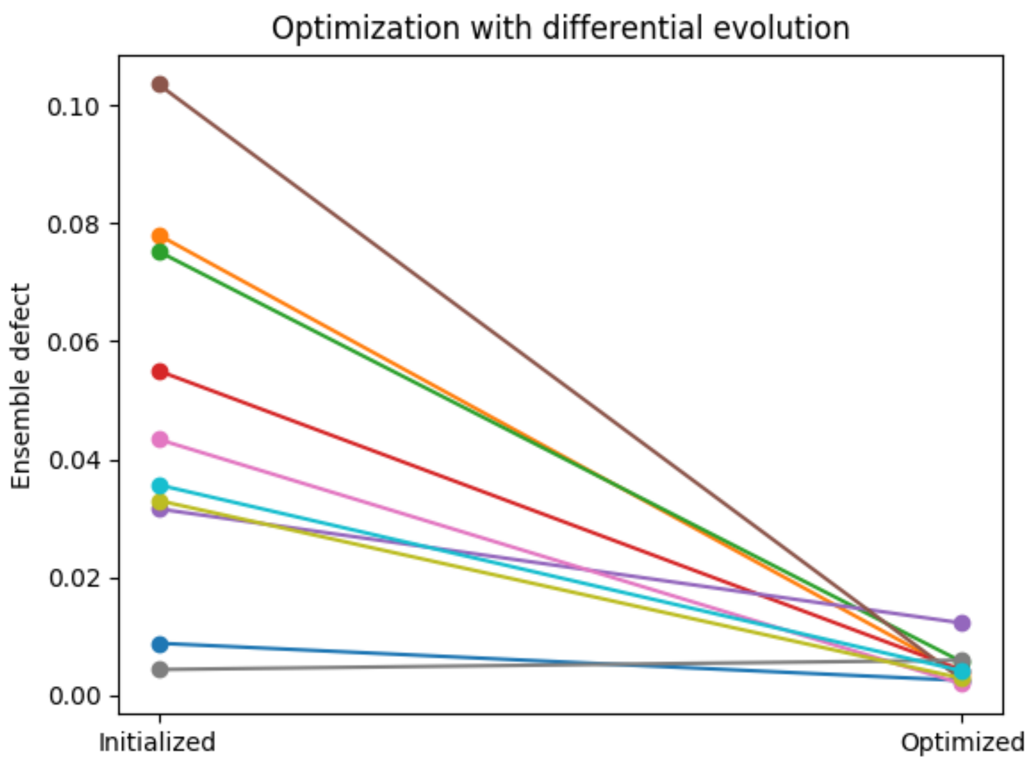




Conclusion: Converged to choosing actions that do not 'mess up' the initialization.

Result #3 - Direct evolutionary optimization

Setup:
- sequence length n=20
- reward = ensemble defect of a coloring



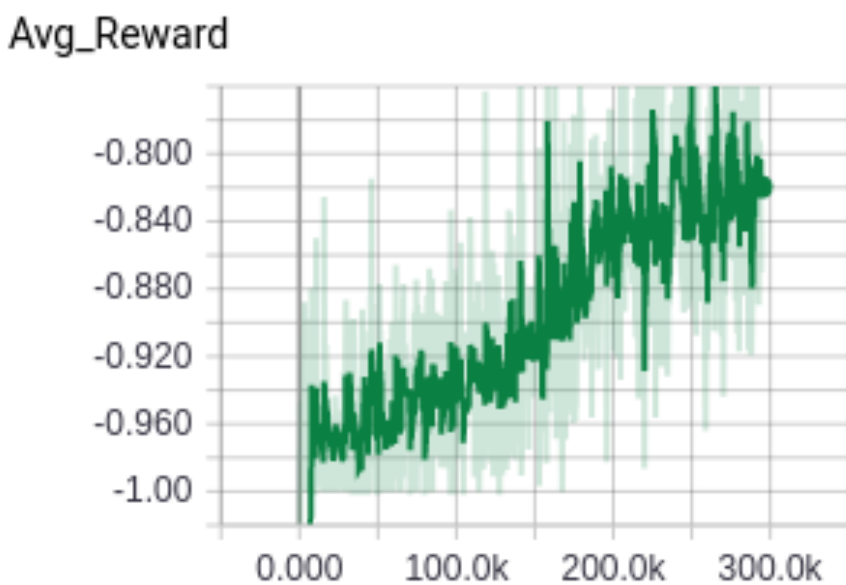
Mean defect after smart initialization:
0.05 +/- 0.03

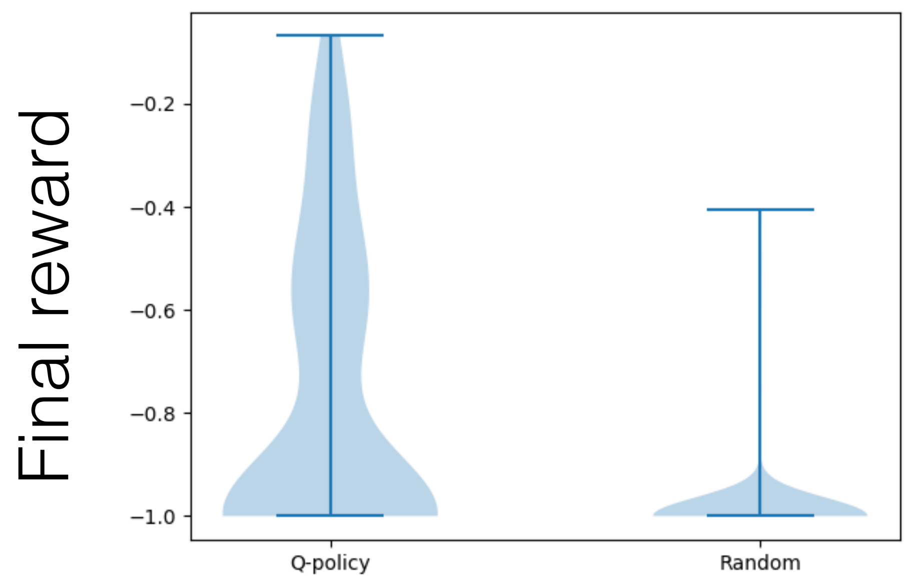
Mean defect after 100 iterations:
0.004 +/- 0.002

Conclusion: Direct optimization for a given target sequence works well within 100 iterations.

Result #4 - Training network to take a single good action

Setup:
- sequence length n=20
- reward = ensemble defect of a coloring
- initial coloring is ‘invalid’; has minimum possible reward
- goal is to select *one* action that will lead to valid coloring (note: most possible actions will not accomplish this)





Conclusion: With proper reward function, RL can indeed converge to a policy that selects good actions.

Conclusions

- Reward function selection is extremely important:
  The reward we defined made the punishment for entering bad states much larger than the potential gain in eventually reaching good states.

- Decreasing the sparsity of the reward is also important for obtaining a good policy.

- Direct optimization of the ensemble defect with an evolutionary algorithm is easy and significantly outperforms our trained agent.

Potential improvements:
- adjust the reward function enable ‘exploratory excursions’
- pre-train the network on simpler test cases

Postscript - is this actually a good application of RL?

Yes, it can be formulated as a game - but in retrospect, that does not mean it is a good target of RL.

- Characteristics of this problem: Deterministic state-transition function, reward is available at each step, every state is reachable from every other state.

- We were trying to 'learn an optimizer' - an agent that gained intuition for performing an optimization.

- Direct optimization can work quite well

- Curricular learning/pre-training is likely important

- Is this even a game humans should be playing?

References:

Lee, J, et al. "RNA design rules from a massive open laboratory." PNAS 111.6 (2014): 2122-2127.

Zadeh, JN et al. "NUPACK: analysis and design of nucleic acid systems." Journal of computational chemistry 32.1 (2011): 170-173.

Mnih, V, et al. "Human-level control through deep reinforcement learning." Nature 518.7540 (2015): 529-533.