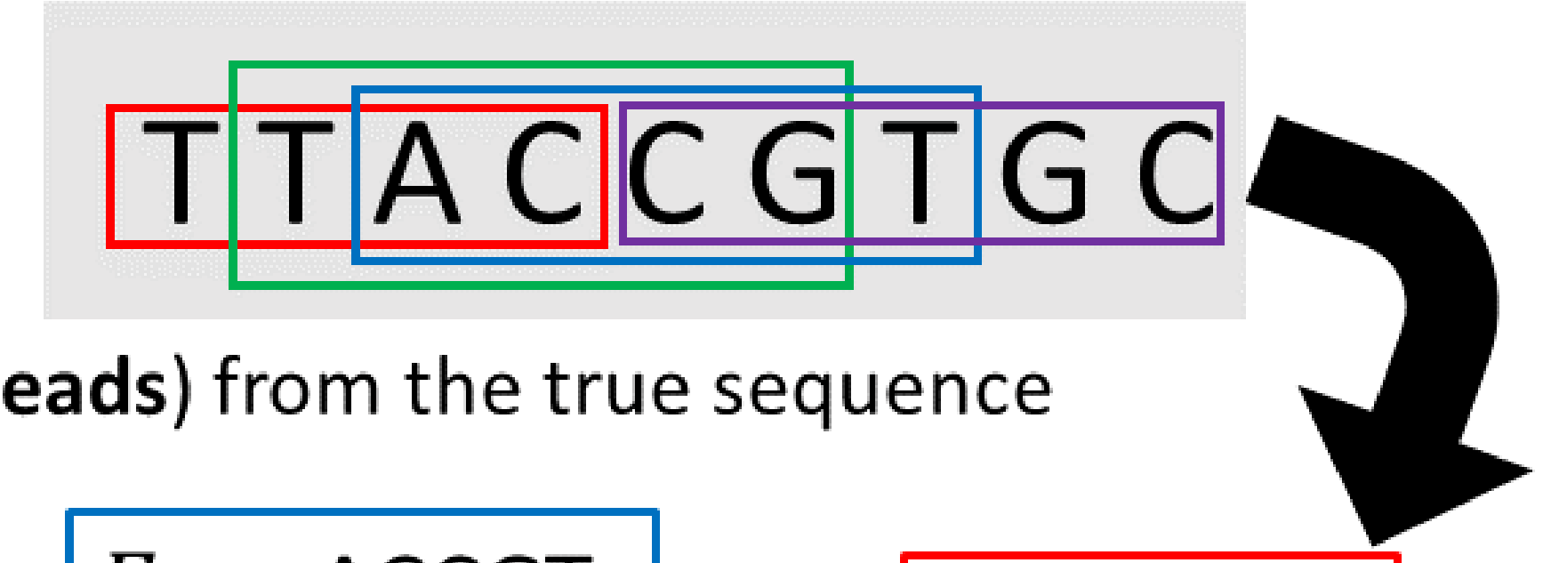
Investigating MDP Optimization Approaches to the De Novo DNA Fragment Assembly Problem

Srihari Ganesh
sganesh@college.harvard.edu
HARVARD UNIVERSITY

DNA Fragment Assembly Problem

Need to sequence DNA for biological research, but can only read **fragments**

True DNA sequence (**unknown**):



Data: Unordered set of fragments (**reads**) from the true sequence

NP-Hard!

F_1 = ACCGT

F_3 = TTAC

F_4 = TACCG

F_2 = CGTGC

Goal: find the correct permutation of reads…

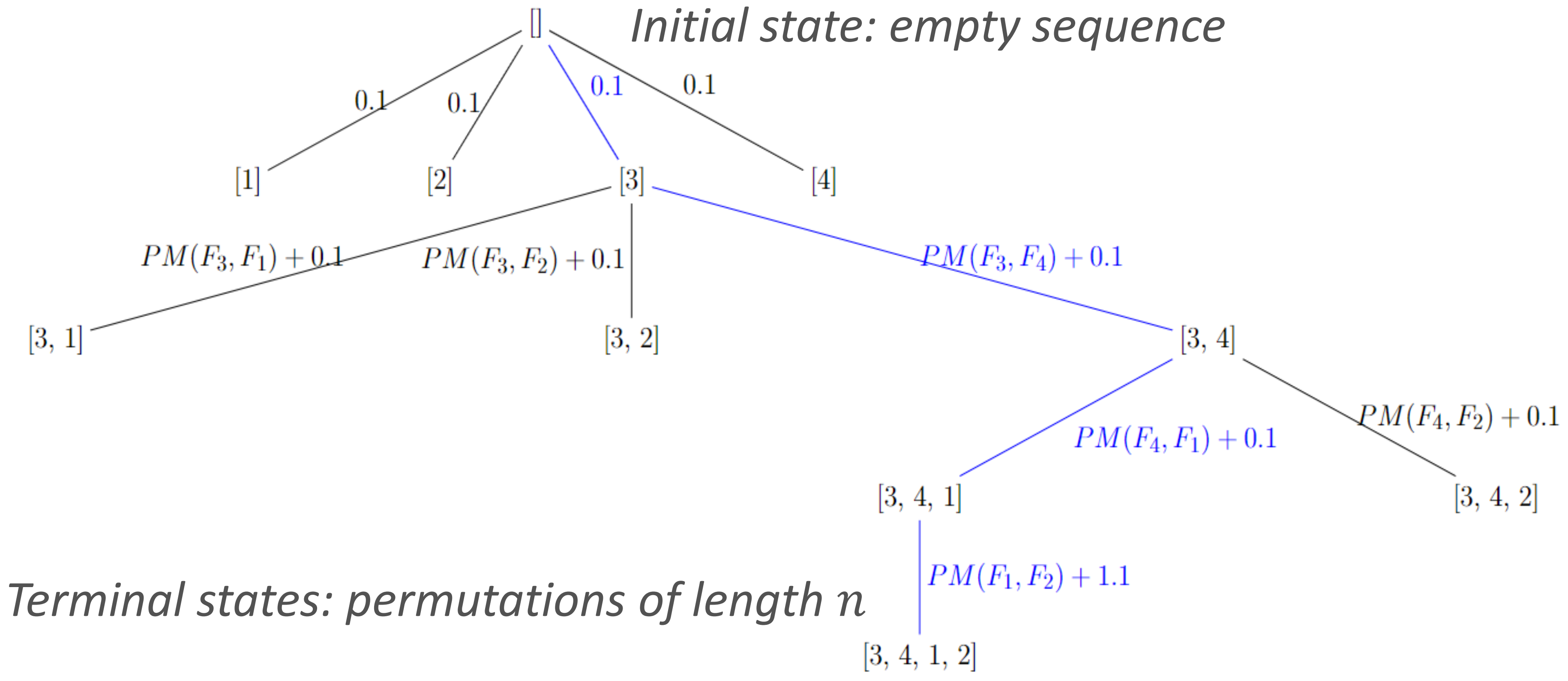
$[F_3, F_4, F_1, F_2]$

… which can then give the solution sequence when aligned

F_3	T	T	A	C	-	-	-	-	-
F_4	-	T	A	C	C	G	-	-	-
F_1	-	-	A	C	C	G	T	-	-
F_2	-	-	-	-	C	G	T	G	C
Solution	**T**	**T**	**A**	**C**	**C**	**G**	**T**	**G**	**C**

Prior RL work[1][2] models problem as optimization on episodic MDP

- **Data**: n error-free reads with the same orientation
- **State space**: permutations of up to n reads
- **Action space**: any read not in current state
  - **Deterministic transitions**: state is a list of actions taken
- **Rewards**: overlap (given by PM) between the action taken and the previous action – i.e. the two most recent reads



Initial state: empty sequence

Terminal states: permutations of length n

Proposed Improvements

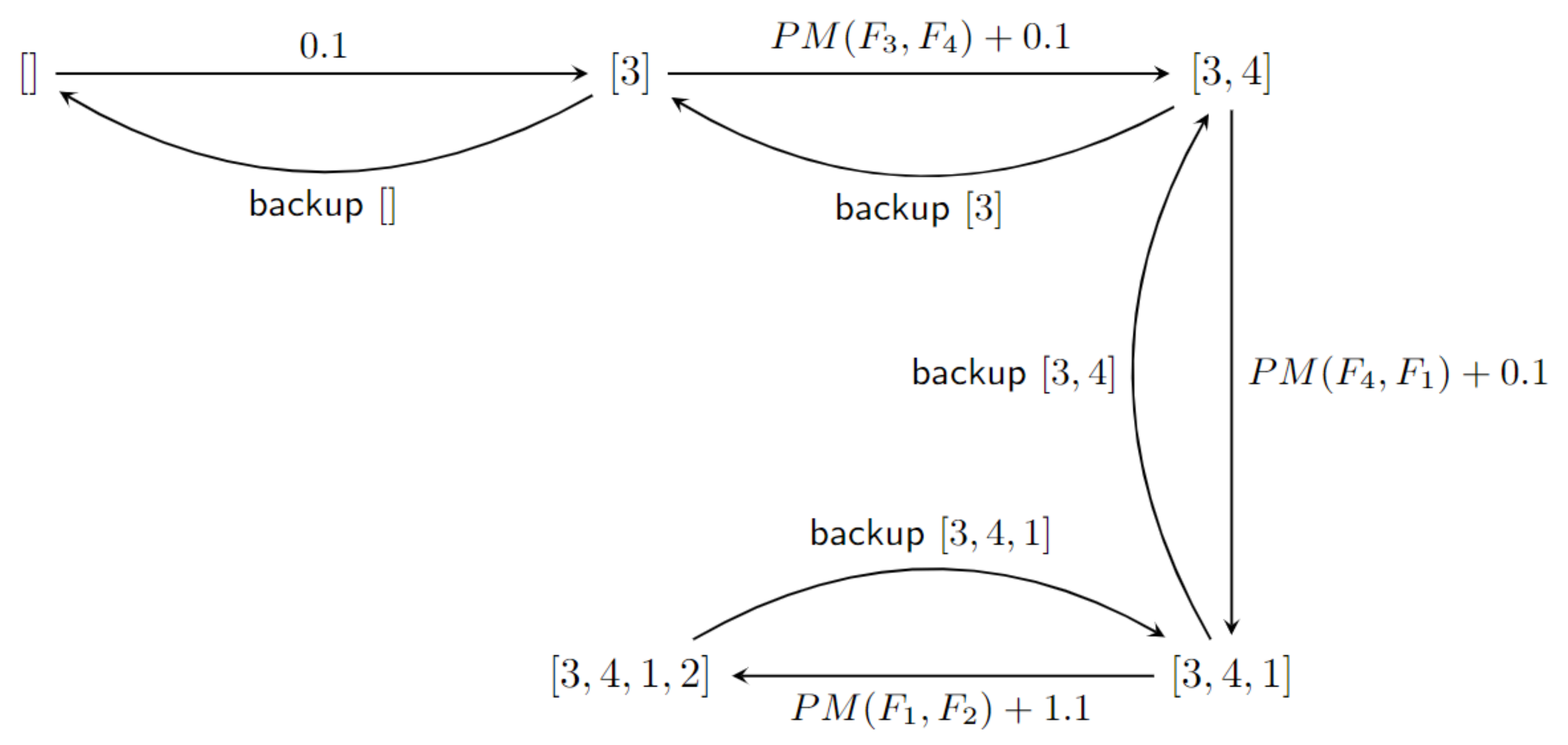
Shortcoming: Learning Algorithm

Only ε-greedy Q-learning has been tested:

- Struggles to propagate values in large state space
- Unnecessary exploitation in optimization setting

Improvement: Real-Time Dynamic Programming[3]

Heuristic search: maintains upper bounds of state values
Select actions greedily
After each episode, backup values in reverse with Bellman operator

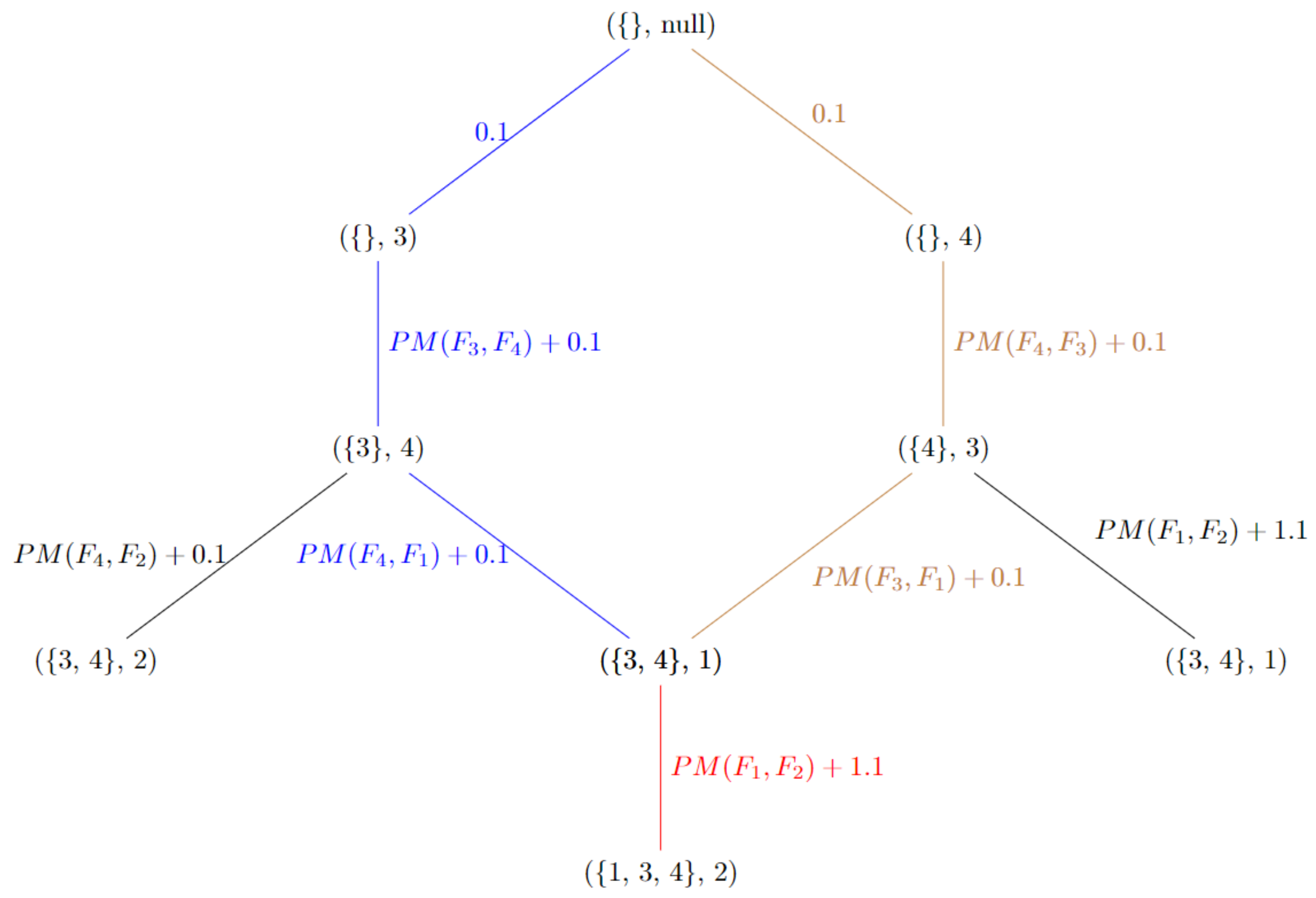


Shortcoming: MDP is a tree

Wasted time in re-learning values of suffixes

Improvement: Represent states with tuple of (set of previous actions, latest action)

ex. $[3, 4, 1, 2] \Rightarrow (\{1, 3, 4\}, 2)$



Simulation Experiments

Experiments run on simulated microgenomes for 500,000 episodes

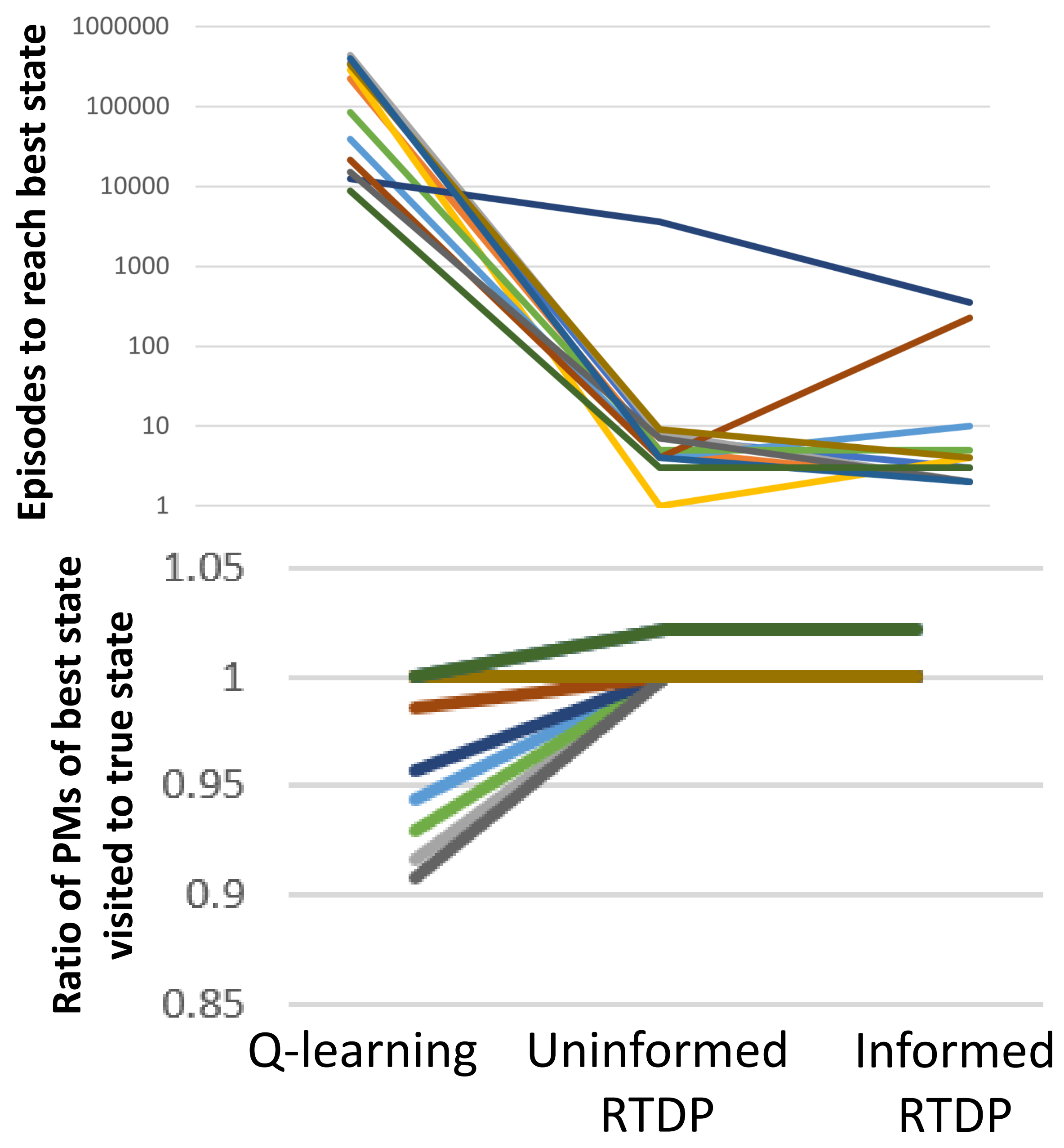
Algorithms: ε-greedy Q-learning vs. Real-Time Dynamic Programming (RTDP)

- Uniformed RTDP (degenerate heuristic) vs. Informed RTDP

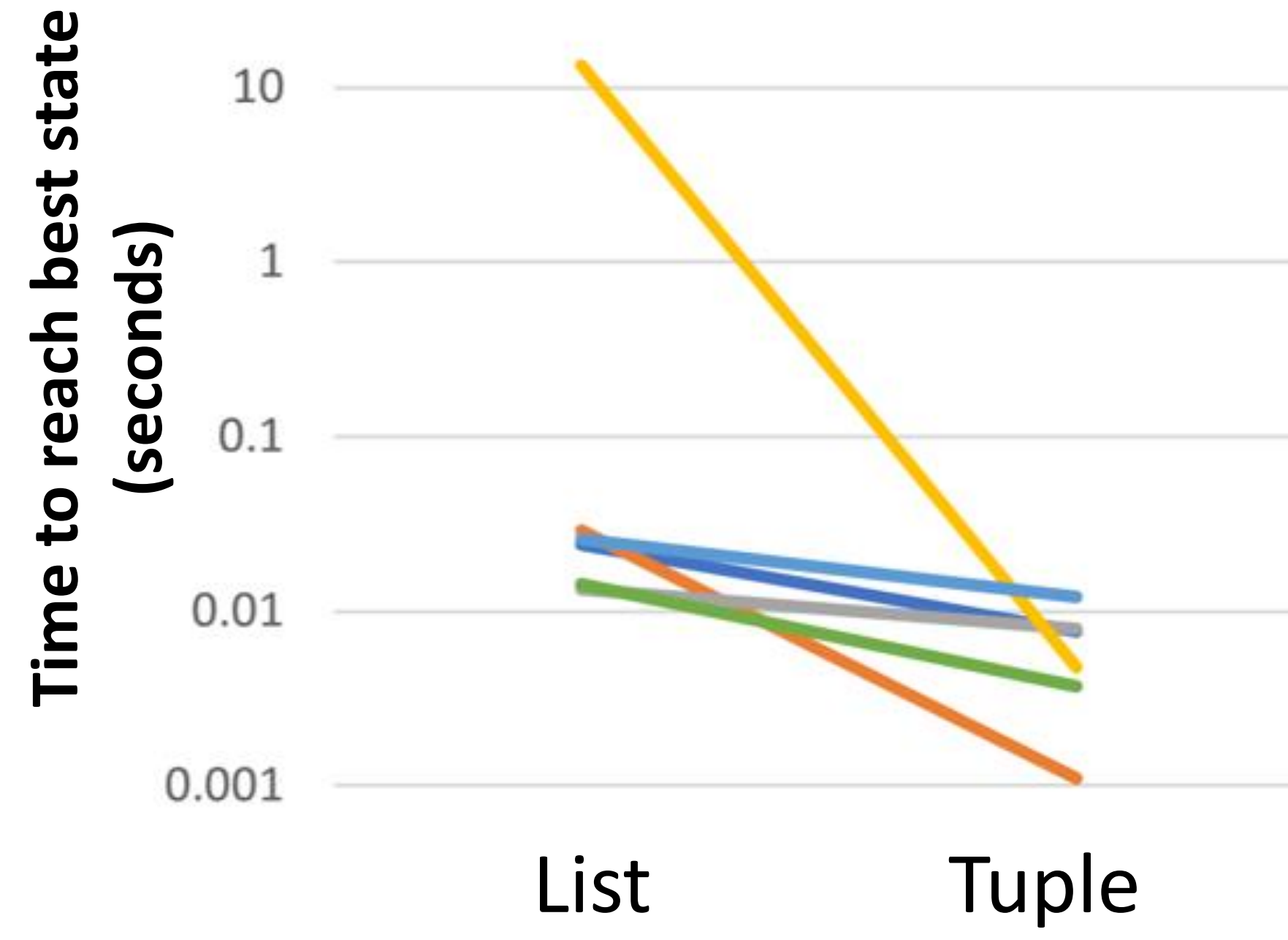
State representations: list vs. tuple; ex. $[3, 4, 1, 2]$ vs. $(\{1, 3, 4\}, 2)$

Preliminary Results

RTDP performs suspiciously well



Tuple representation speeds up uninformed RTDP



Next Steps

Test on larger and messier data

- Testbeds from prior papers are too ideal

Compare to genetic algorithms[4]

- May be more flexible generalization of MDP setup

References:

[1] Bocicor, M., Czibula, G., Czibula, I. (2011). A Reinforcement Learning Approach for Solving the Fragment Assembly Problem. *2011 13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*. https://doi.org/10.1109/SYNASC.2011.9
[2] Padovani, K., Xavier, R., Carvalho, A., Reali, A., Chateau, A. & Alves, R. (2021). A Step Towards a Reinforcement Learning *De Novo* Genome Assembler. *arXiv*. https://doi.org/10.48550/arXiv.2102.02649
[3] Barto, A. G., Bradtke, S. J., Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence.* https://doi.org/10.1016/0004-3702(94)00011-O
[4] Oliviera, R. R. M., Damasceno, F., Souza, R., Santos, R., Lima, M., Kawasaki, R., Sales, C. (2017). GAVGA: A Genetic Algorithm for Viral Genome Assembly. *Progress in Artificial Intelligence*. https://doi.org/10.1007/978-3-319-65340-2_33