Topics in Reinforcement Learning: Lessons from AlphaZero for (Sub)Optimal Control and Discrete Optimization

Arizona State University
Course CSE 691, Spring 2023

Links to Class Notes, Videolectures, and Slides at
http://web.mit.edu/dimitrib/www/RLbook.html

Dimitri P. Bertsekas
dbertsek@asu.edu

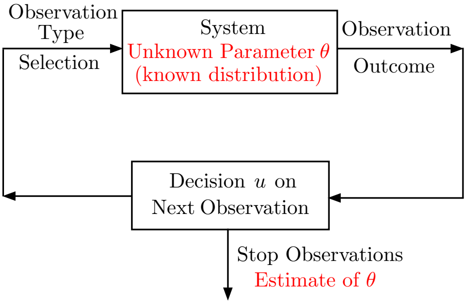
Lecture 9

Combined Estimation/Control: Sequential Estimation, Bayesian Optimization, and Adaptive Control with a POMDP Approach

Application to the Wordle Puzzle

Outline

Sequential Estimation of a Parameter Vector θ



Use of costly observations to estimate a parameter vector θ

- The number and type of observations are subject to choice
- Instead, the outcomes of the observations obtained are evaluated on-line with a view towards stopping or modifying the observation process
- This involves sequential decision making, thus bringing DP to bear

Example: Select one of two hypotheses using costly sequential observations

Given a new observation, we can accept one of the hypotheses or obtain a new observation at cost C (cf. quality control, the sequential probability ratio test, 1940s).

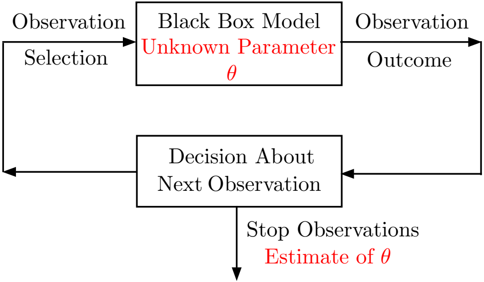
Applications of Sequential Estimation

- Classical sequential experiment design problems or sequential sampling strategies in statistics.
  - Select one of multiple hypotheses.
  - Design of clinical trials or tests for medical diagnosis.
- Classical sequential search problems (e.g., search and rescue).
- Route planning through a sensor network for sequential information collection.
- Sequential decoding problems (e.g., the Mastermind and Wordle puzzles, to be discussed later).
- Surrogate and Bayesian optimization for minimizing “black box" functions (to be discussed first).

An important distinction: Does the current choice of observation affect the availability, the quality, or the cost of future observations?

- If no, we call this a simple sequential estimation problem (we will discuss it first in the context of Bayesian optimization).
- If yes, this can be viewed as a combined estimation and control problem, and can be viewed within the context of adaptive control.

Surrogate Optimization of “Black Box" Functions



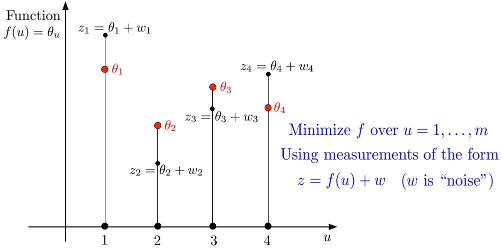
Minimize approximately a function whose values at given points are obtained only through time-consuming calculation, simulation, or experimentation

- Introduce a parametric model of the cost function with parameter θ.
- Observe sequentially the true cost function at a few observation points.
- Construct a model of the cost function (the surrogate) by estimating θ.
- Minimize the surrogate to obtain a suboptimal solution.
- How to select observation points based on results of previous observations?
- Exploration-exploitation tradeoff: Observing at points likely to have near-optimal value vs observing at points in relatively unexplored areas of the search space.

Surrogate Optimization Examples

- Geostatistical interpolation ("kriging" pioneered by the South African engineers Matheron and Krige in a goldmining context): Identify locations of high gold distribution based on samples from a few boreholes.
- Design optimization, e.g., aerodynamic design using hardware prototypes, materials design, drug development, etc.
- Hyperparameter selection of machine-learning models, including the architectural parameters of the deep neural network of AlphaZero.

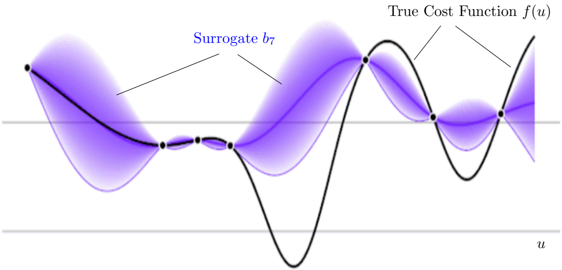
Bayesian Optimization of a Black Box Function f



- Introduce a parameter vector $\theta = (\theta_1, \ldots, \theta_m) \in \Re^m$ where $\theta_u = f(u)$, i.e., θ is f
- Observations are of the form $z = f(u) + w$ (important special case is $w = 0$)
- Estimate θ with $N << m$ noisy measurements at chosen points $u_1, \ldots, u_N$
- We assume that θ has a given a priori distribution $b_0 = (b_{0,1}, \ldots, b_{0,m})$ over $\Re^m$ (values of f at "neighboring" points should be correlated)
- After observations at points $u_1, \ldots, u_k$ of the form $z_{u_i} = \theta_{u_i} + w_{u_i}$, we choose the next point u_{k+1} at which to observe the value of f.
- Update the posterior distribution b_k with an estimator $b_{k+1} = B_k(b_k, u_{k+1}, z_{u_{k+1}})$ (b_k is essentially the surrogate cost function after the kth observation)
- Gaussian case: If b_0 and the noises w_u are Gaussian, b_k can be updated using closed form Gaussian process regression formulas.

Black is the true cost function

Purple is the surrogate cost function



After 7 noise-free observations

The surrogate is specified by the posterior distribution b_k (mean and standard deviation at the different points are shown in the figure)

Myopic Bayesian Optimization

Key Question: How to select sequentially the observation point u_{k+1} given the observation results $z_{u_1}, \ldots, z_{u_k}$ from previously selected points $u_1, \ldots, u_k$

A DP view

- Introduce a POMDP model: The posterior b_k (given the observations up to time k) is the belief state, u_k is the control, the belief estimator $b_{k+1} = B_k(b_k, u_{k+1}, z_{u_{k+1}})$ is the system. The cost function is based on the cost of the observations, and the "quality" of the surrogate obtained at the end.
- The dominant method in practice: Use a greedy/myopic policy, based on an acquisition function.
- The acquisition function $A_k(b_k, u_{k+1})$ is a heuristic measure of "benefit" for selecting point u_{k+1} for observation when the belief state is b_k.
- Myopic policy: Selects the next point at which to observe, $\hat{u}_{k+1}$, as

$$\hat{u}_{k+1} \in \arg \max_{u_{k+1} \in \{1,\ldots,m\}} A_k(b_k, u_{k+1})$$

- An alternative method: Use rollout with a myopic base policy; it has been advocated in several research works since 2016, with promising results.

Examples of Acquisition Functions for Myopic Bayesian Optimization

The myopic policy maximizes over u_{k+1} the acquisition function $A_k(b_k, u_{k+1})$:

$$\hat{u}_{k+1} \in \arg \max_{u_{k+1} \in \{1,\dots,m\}} A_k(b_k, u_{k+1})$$

A common example of acquisition function: Upper confidence bound

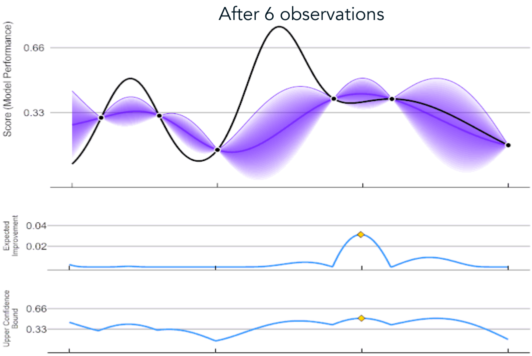
$$A_k(b_k, u) = T_k(b_k, u) + \beta R_k(b_k, u), \qquad \beta > 0 \text{ is a tunable parameter}$$

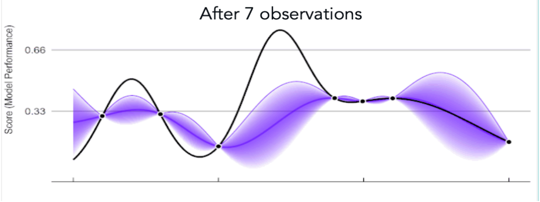
- Here $T_k(b_k, u) = -$Mean of $f(u)$, and $R_k(b_k, u) =$ Standard deviation of $f(u)$ (under the posterior distribution b_k).
- $T_k(b_k, u)$ can be viewed as an exploitation index (encoding our desire to search within parts of the space where f takes low value), while $R_k(b_k, u)$ can be viewed as an exploration index (encoding our desire to search within parts of the space that are relatively unexplored).

Another example of acquisition function: Expected improvement

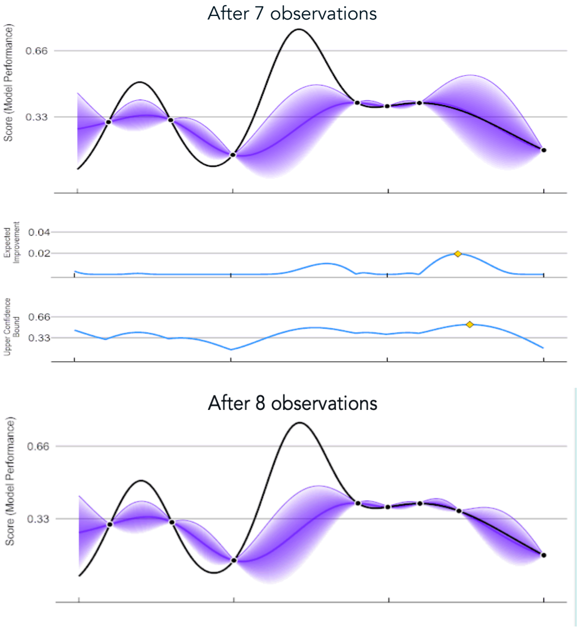
$A_k(b_k, u)$ is the expected value of the reduction of $f(u)$ relative to the minimal value of f obtained up to time k (under the posterior distribution b_k).

Maximization Example (From Wikipedia Article on BO): True Function is Black, Surrogate Function is Purple; Observations are Noise-Free

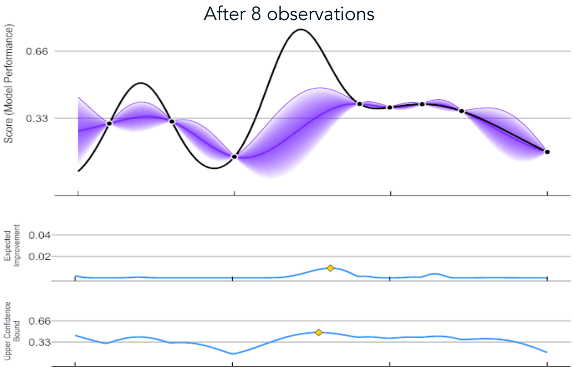


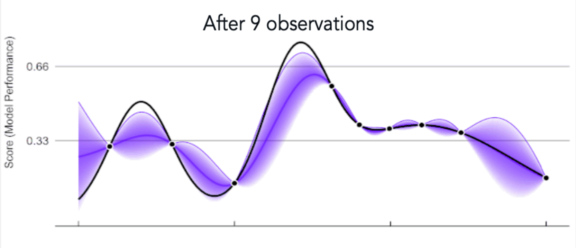


Maximization Example II

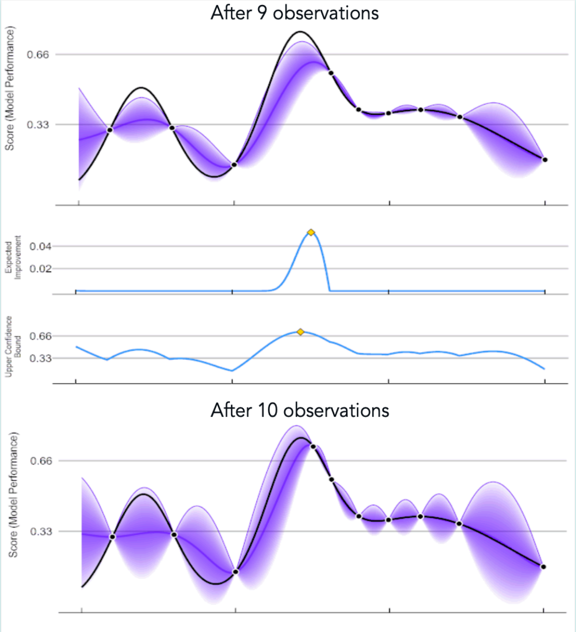


Maximization Example III





Maximization Example IV



DP Algorithm for POMDP Formulation of Bayesian Optimization

$$J_k^*(b_k) = \min_{u_{k+1} \in \{1,\ldots,m\}} \left[c(u_{k+1}) + E_{z_{u_{k+1}}} \left\{ J_{k+1}^*\big(B_k(b_k, u_{k+1}, z_{u_{k+1}})\big) \mid b_k, u_{k+1} \right\} \right]$$

where $c(u)$ is the cost of observation at u. Proceeds backwards from a terminal cost

$$J_N^*(b_N) = G(b_N) \qquad \text{(measures the quality of the surrogate obtained at the end)}$$

Approximation in value space (replace J_{k+1}^* with $\tilde{J}_{k+1}$)

$$\tilde{u}_{k+1} \in \arg \min_{u_{k+1} \in \{1,\ldots,m\}} Q_k(b_k, u_{k+1})$$

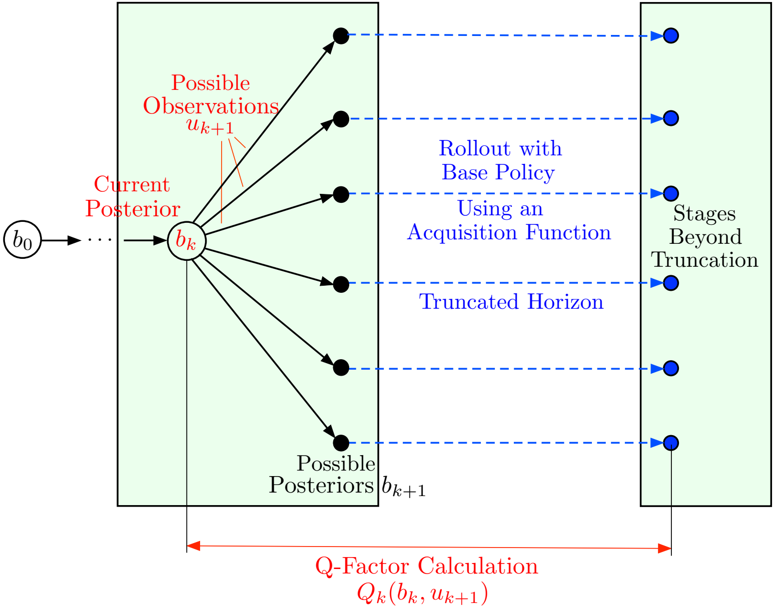
where $Q_k(b_k, u_{k+1})$ is the (approximate) Q-factor corresponding to the pair (b_k, u_{k+1}):

$$Q_k(b_k, u_{k+1}) = c(u_{k+1}) + E_{z_{u_{k+1}}} \left\{ \tilde{J}_{k+1}\big(B_k(b_k, u_{k+1}, z_{u_{k+1}})\big) \mid b_k, u_{k+1} \right\}$$

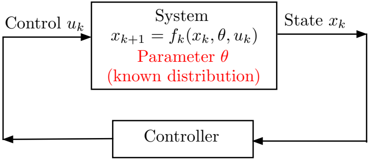
Rollout

Use as $\tilde{J}_{k+1}$ the cost function of a myopic base heuristic based on an acquisition function (or approximation thereof); first proposed by Lam, Wilcox, and Wolpert (2016), and followed up by others (promising, but relatively untested at present).

Truncated Rollout with a Myopic Base Heuristic



Adaptive Control with a POMDP Formulation and Rollout



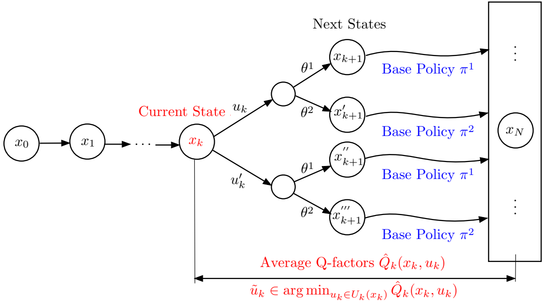
Deterministic system $x_{k+1} = f(x_k, \theta, u_k)$, $\theta \in \{\theta^1, \ldots, \theta^m\}$: unknown parameter

- θ has known initial distribution b_0 and stays constant. It is observed indirectly through perfect observation of x_k
- View θ as part of an augmented state (x_k, θ) that is partially observed
- Bellman equation for optimal cost function J_k^*:

$$J_k^*(I_k) = \min_{u_k} \sum_{i=1}^{m} b_{k,i}\Big(g(x_k, \theta^i, u_k) + J_{k+1}^*\big(I_k, u_k, f(x_k, \theta^i, u_k)\big)\Big)$$

  where $I_k = (x_0, \ldots, x_k, u_0, \ldots, u_{k-1})$ is the information state at time k, and $b_{k,i} = P\{\theta = \theta^i \mid I_k\}$, $i = 1, \ldots, m$, is the belief state (estimated on-line)
- Approximation in value space: Use approximation $\tilde{J}^i\big(f(x_k, \theta^i, u_k)\big)$ in place of $J_{k+1}^*\big(I_k, u_k, f(x_k, \theta^i, u_k)\big)$. Minimize over u_k to obtain a one-step lookahead policy
- Example 1: $\tilde{J}^i$ is the cost function of the optimal policy corresponding to θ^i
- Example 2: $\tilde{J}^i$ is the cost function of a known policy assuming $\theta = \theta^i$ (this is rollout)

Rollout for Adaptive Control with a POMDP Formulation



At x_k, we minimize $\hat{Q}_k(x_k, u_k)$, the average Q-factor of u_k, defined by

$$\hat{Q}_k(x_k, u_k) = \sum_{i=1}^{m} b_{k,i} Q_k(x_k, u_k, \theta^i),$$

where $Q_k(x_k, u_k, \theta^i)$ is the Q-factor computed assuming that $\theta = \theta^i$

$$Q_k(x_k, u_k, \theta^i) = g_k(x_k, \theta^i, u_k) + J_{k+1,\pi^i}\big(f_k(x_k, \theta^i, u_k)\big)$$

If $\pi^i \equiv \pi$, cost improvement over π can be proved

A Rollout Approach for Solving On-Line the Wordle Puzzle (Joint Work with Siddhant Bhambri and Amrita Bhattacharjee)

Overview

- There is a hidden mystery word/code word θ drawn from an initial mystery list according to a known distribution. In the standard version of the puzzle this distribution is uniform.
- The mystery list shrinks as a result of guesses/observations.
- The guesses are chosen based on feedback about the mystery word provided by the preceding guesses.
- The puzzle is solved when the mystery list shrinks to a single element.
- We want to minimize the expected number of guesses to solve the puzzle.
- Important fact: The belief distribution over the current mystery list remains uniform through the solution process.
- This makes possible the solution by exact DP, with days of computation (Selby 2022).
- Without the uniform initial belief distribution assumption (and/or small variations in the problem structure), the exact solution would be impossible.
- Rollout can solve near optimally the puzzle (and its variations) on-line much faster.

Online Rollout Solution for Deterministic POMDP with Unknown Parameters

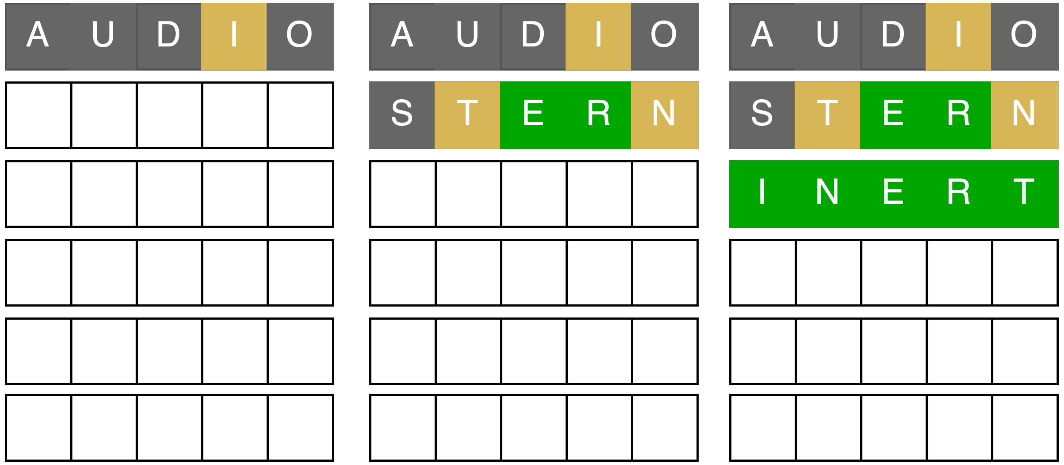
Siddhant Bhambri, Amrita Bhattacharjee, and Dimitri Bertsekas

sbhambr1@asu.edu

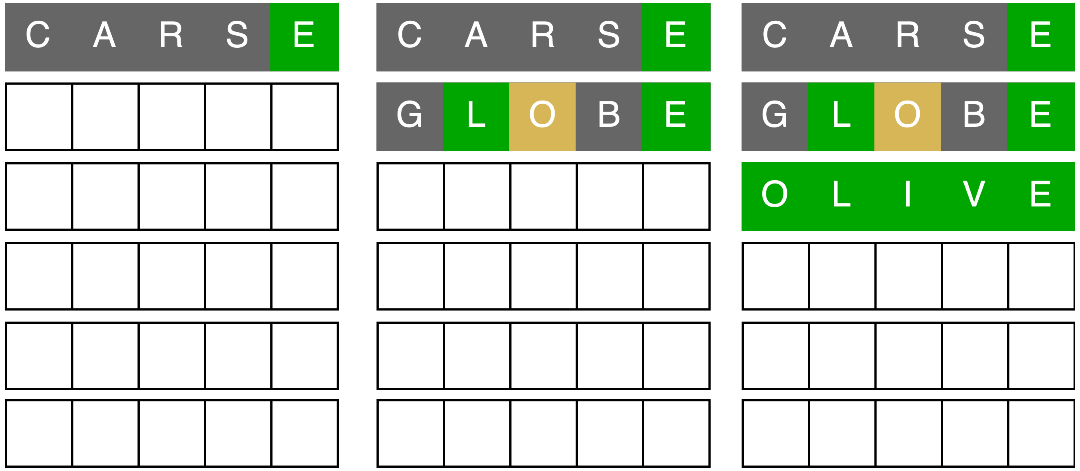
Revisiting the POMDP Model

- S, a finite set of states, which includes a cost-free and absorbing termination state;
- A, a finite set of actions, and for each state s_k, a constraint subset $A(s_k) \subset A$ from within which a_k must be chosen when at state s_k;
- T, a deterministic transition function: $T(s_k,\theta,a_k)$ that gives the θ-dependent next state s_{k+1} when action a_k is applied in state s_k at time k;
- C, a cost function: $C(s_k, \theta, a_k)$ that gives the θ- dependent cost (or negative rewards) incurred by the agent when action a_k is applied in state s_k at time k.

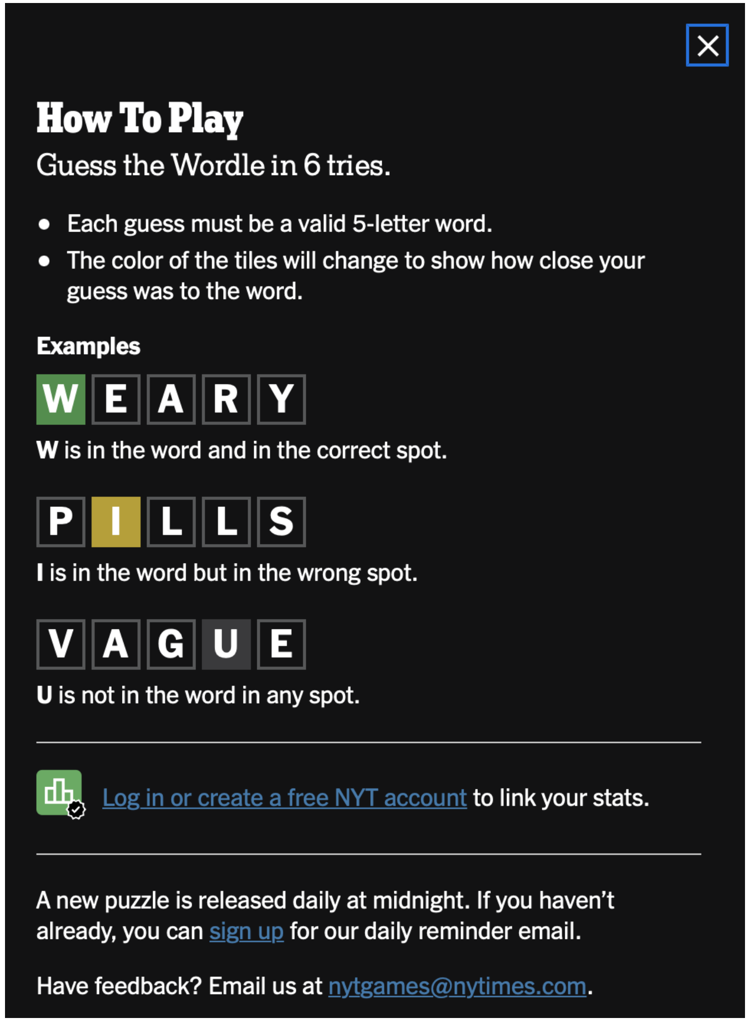
The Wordle puzzle



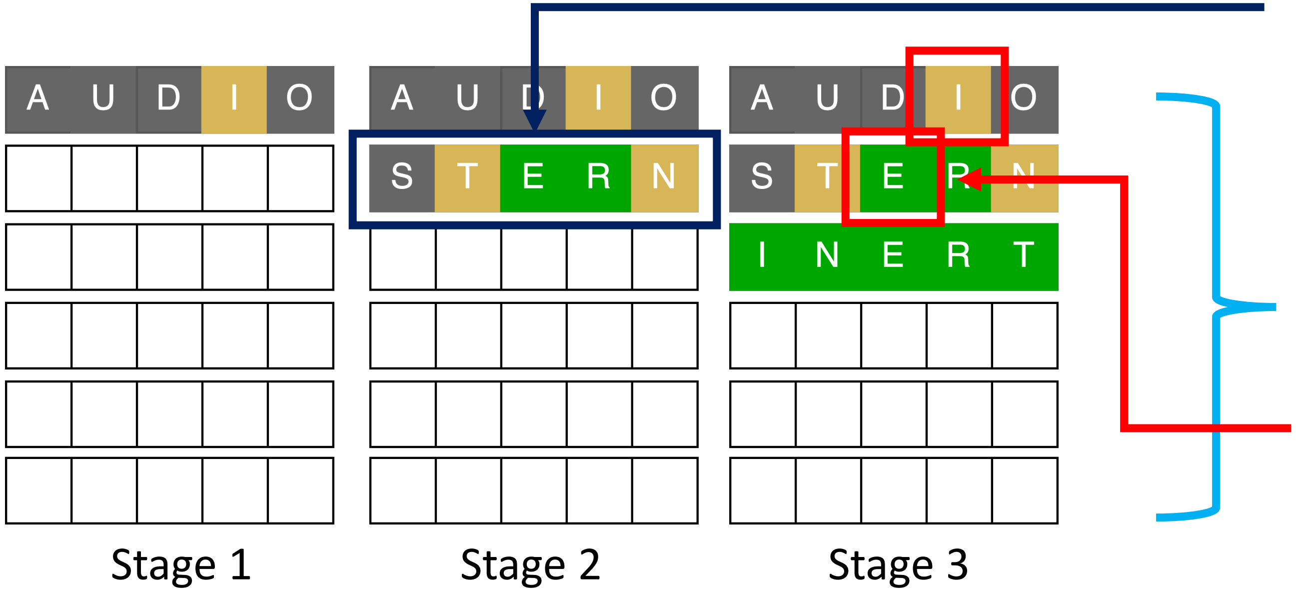
Easy mode



Hard mode



Wordle as a POMDP



States (S): Subset of the initial mystery list of 2,315 words

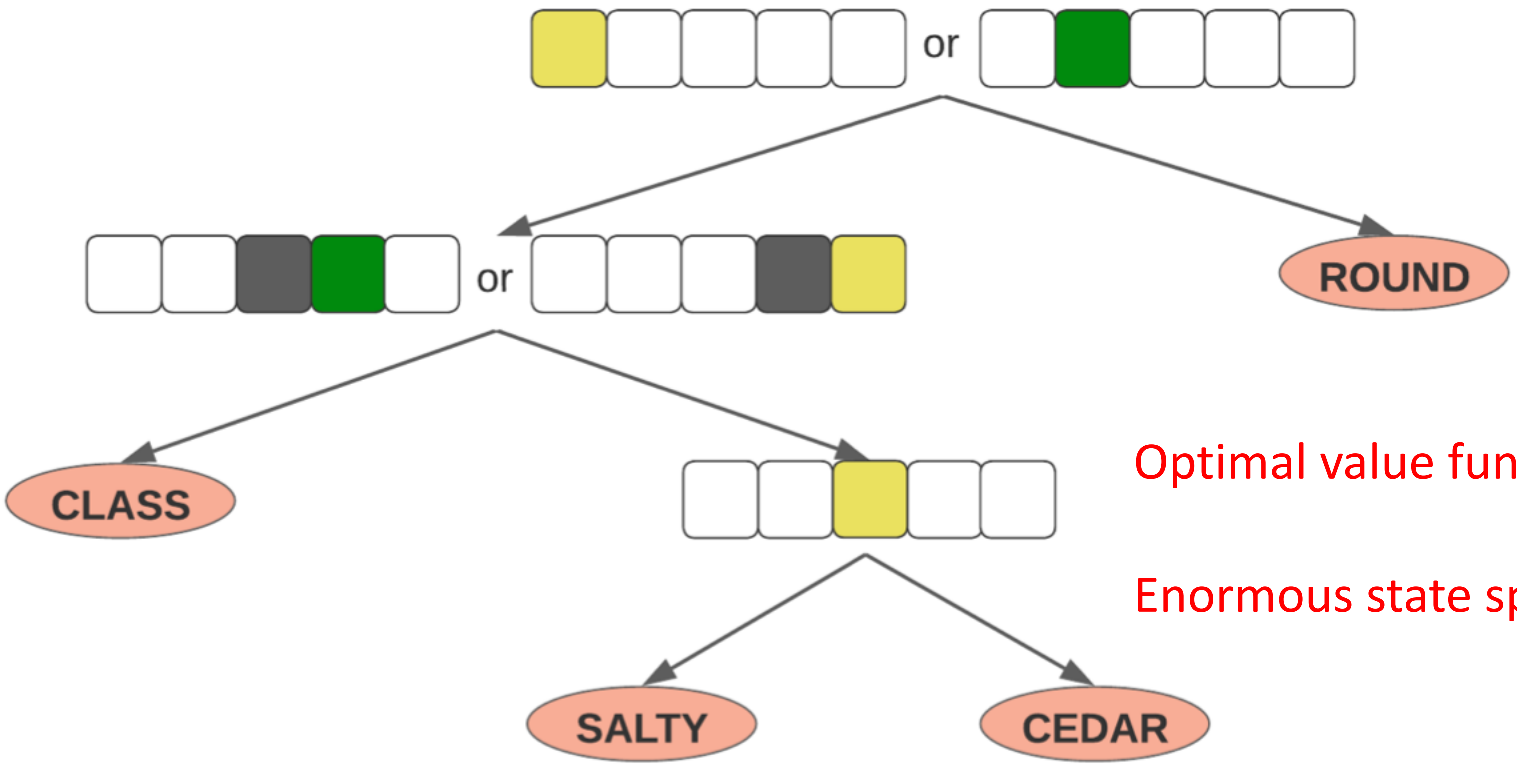
Actions (A): Set of 12,972 guess words

Transitions (T): probability of going from one mystery word list to the next.

Cost (C): cost of utilizing a guess word (=1)

Observations from the game: colored observations for each letter

Optimal solution using Dynamic Programming



Optimal value function required to compute!

Enormous state space: $2^{2315} \cong 10^{697}$

Figure adapted from Bertsimas, D. and Paskov, A., 2022. An Exact and Interpretable Solution to Wordle. *Available at URL.(Accessed: 14 November 2022)*.
Selby A., 2022.\ ``The best strategies for Wordle (last edited on 17 March 2022)." (Accessed: 14 November 2022).

Approximate the value function using Rollout

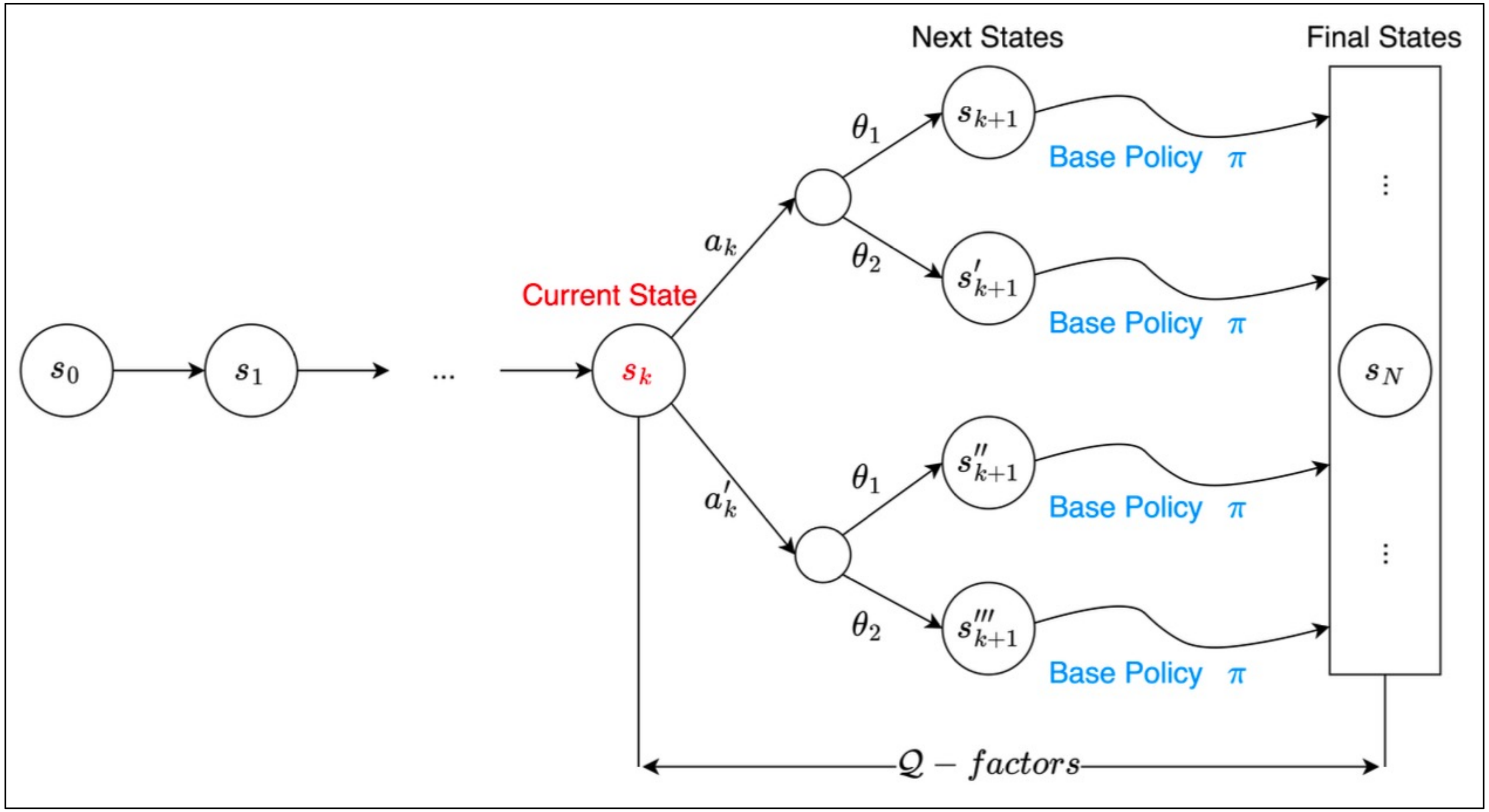


Figure: Schematic illustration of the rollout approach.

Base Heuristic for Wordle – Information Gain!

Information gain - calculating entropy (roughly based on how much using a word reduces the uncertainty about the mystery word)

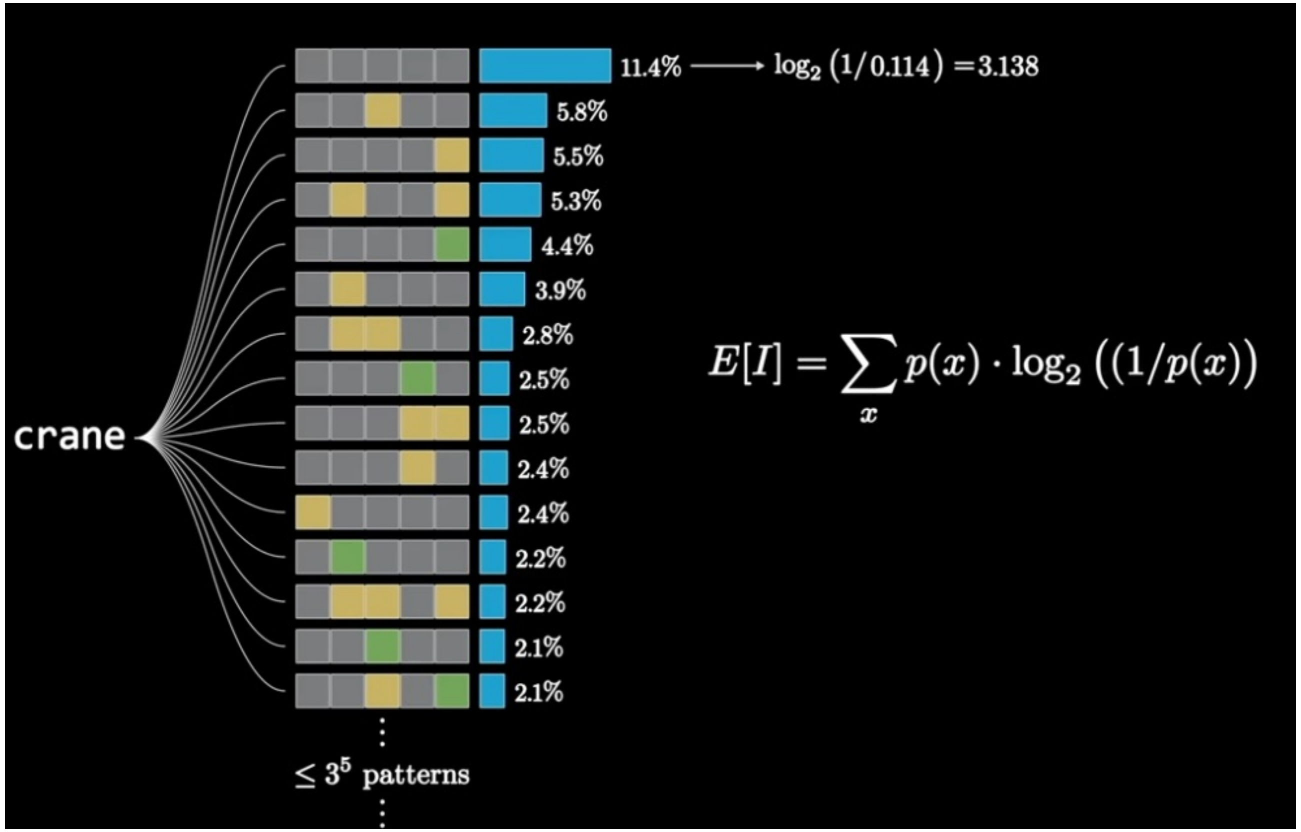


Figure adapted from Grant's video (3Blue1Brown on YouTube)

Solving Wordle using Rollout

Line 1: empty set to store the average Q-factors for each possible action at stage k.

Algorithm 1: Action and next state selection w/ rollout.

Data: Current state $s_k \in \mathcal{S}$, Set of unknown parameters Θ, Current belief distribution b_k, Action space $\mathcal{A}$, Transition function $\mathcal{T}$, Cost function $\mathcal{C}$, Base policy cost function J_{k+1}.

Result: Action $\tilde{a}_k$ at state s_k, Next state s_{k+1}.

1 $\boxed{avg\_Q\_list \leftarrow [\,];}$

2 **for** a_k *in* $\mathcal{A}$ **do**

3     $Q\_list \leftarrow [\,];$

4   **for** $\theta^i \in \Theta$ **do**

5         $\mathcal{Q}_k \leftarrow \mathcal{C}(s_k, \theta^i, a_k) + J^i_{k+1}(\mathcal{T}(s_k, \theta^i, a_k));$

6         $Q\_list \leftarrow Q\_list \cup [\mathcal{Q}_k];$

7     $avg\_Q\_factor \leftarrow \sum_{i=1}^{m} b_{k,i}(Q\_list);$

8     $avg\_Q\_list \leftarrow avg\_Q\_list \cup [avg\_Q\_factor];$

9 $\tilde{a}_k \leftarrow \arg\min_{\forall a_k \in \mathcal{A}} avg\_Q\_list\ ;$

10 $s_{k+1} \leftarrow \mathcal{T}(s_k, \theta, \tilde{a}_k);$

11 **return** $\tilde{a}_k, s_{k+1}$

Solving Wordle using Rollout

Line 4-6: for all possible $\theta_i \in \Theta$, we perform the rollout by applying the next action as selected by our base heuristic cost function Jk+1 and compute the Q-factor until we reach the terminating state.

Algorithm 1: Action and next state selection w/ rollout.

Data: Current state $s_k \in \mathcal{S}$, Set of unknown parameters Θ, Current belief distribution b_k, Action space $\mathcal{A}$, Transition function $\mathcal{T}$, Cost function $\mathcal{C}$, Base policy cost function J_{k+1}.

Result: Action $\tilde{a}_k$ at state s_k, Next state s_{k+1}.

1 $avg\_Q\_list \leftarrow [\,]$;

2 **for** a_k *in* $\mathcal{A}$ **do**

3     $Q\_list \leftarrow [\,]$;

4   **for** $\theta^i \in \Theta$ **do**

5         $Q_k \leftarrow \mathcal{C}(s_k, \theta^i, a_k) + J^i_{k+1}(\mathcal{T}(s_k, \theta^i, a_k))$;

6         $Q\_list \leftarrow Q\_list \cup [Q_k]$;

7     $avg\_Q\_factor \leftarrow \sum_{i=1}^{m} b_{k,i}(Q\_list)$;

8     $avg\_Q\_list \leftarrow avg\_Q\_list \cup [avg\_Q\_factor]$;

9 $\tilde{a}_k \leftarrow \arg\min_{\forall a_k \in \mathcal{A}} avg\_Q\_list$;

10 $s_{k+1} \leftarrow \mathcal{T}(s_k, \theta, \tilde{a}_k)$;

11 **return** $\tilde{a}_k, s_{k+1}$

Solving Wordle using Rollout

Line 7: find the average Q-factor for taking an action a_k weighed by the current belief distribution.

Algorithm 1: Action and next state selection w/ rollout.

Data: Current state $s_k \in \mathcal{S}$, Set of unknown parameters Θ, Current belief distribution b_k, Action space $\mathcal{A}$, Transition function $\mathcal{T}$, Cost function $\mathcal{C}$, Base policy cost function J_{k+1}.

Result: Action $\tilde{a}_k$ at state s_k, Next state s_{k+1}.

1 $avg\_Q\_list \leftarrow [\,]$;

2 **for** a_k *in* $\mathcal{A}$ **do**

3 $\quad Q\_list \leftarrow [\,]$;

4 $\quad$ **for** $\theta^i \in \Theta$ **do**

5 $\quad\quad \mathcal{Q}_k \leftarrow \mathcal{C}(s_k, \theta^i, a_k) + J^i_{k+1}(\mathcal{T}(s_k, \theta^i, a_k))$;

6 $\quad\quad Q\_list \leftarrow Q\_list \cup [\mathcal{Q}_k]$;

7 $\quad avg\_Q\_factor \leftarrow \sum_{i=1}^{m} b_{k,i}(Q\_list)$;

8 $\quad avg\_Q\_list \leftarrow avg\_Q\_list \cup [avg\_Q\_factor]$;

9 $\tilde{a}_k \leftarrow \arg\min_{\forall a_k \in \mathcal{A}} avg\_Q\_list$;

10 $s_{k+1} \leftarrow \mathcal{T}(s_k, \theta, \tilde{a}_k)$;

11 **return** $\tilde{a}_k, s_{k+1}$

Solving Wordle using Rollout

Line 9-10: we select the action $\tilde{a}_k$ that corresponds to the minimum average Q-factor and apply it to state s_k.

Algorithm 1: Action and next state selection w/ rollout.

Data: Current state $s_k \in \mathcal{S}$, Set of unknown parameters Θ, Current belief distribution b_k, Action space $\mathcal{A}$, Transition function $\mathcal{T}$, Cost function $\mathcal{C}$, Base policy cost function J_{k+1}.

Result: Action $\tilde{a}_k$ at state s_k, Next state s_{k+1}.

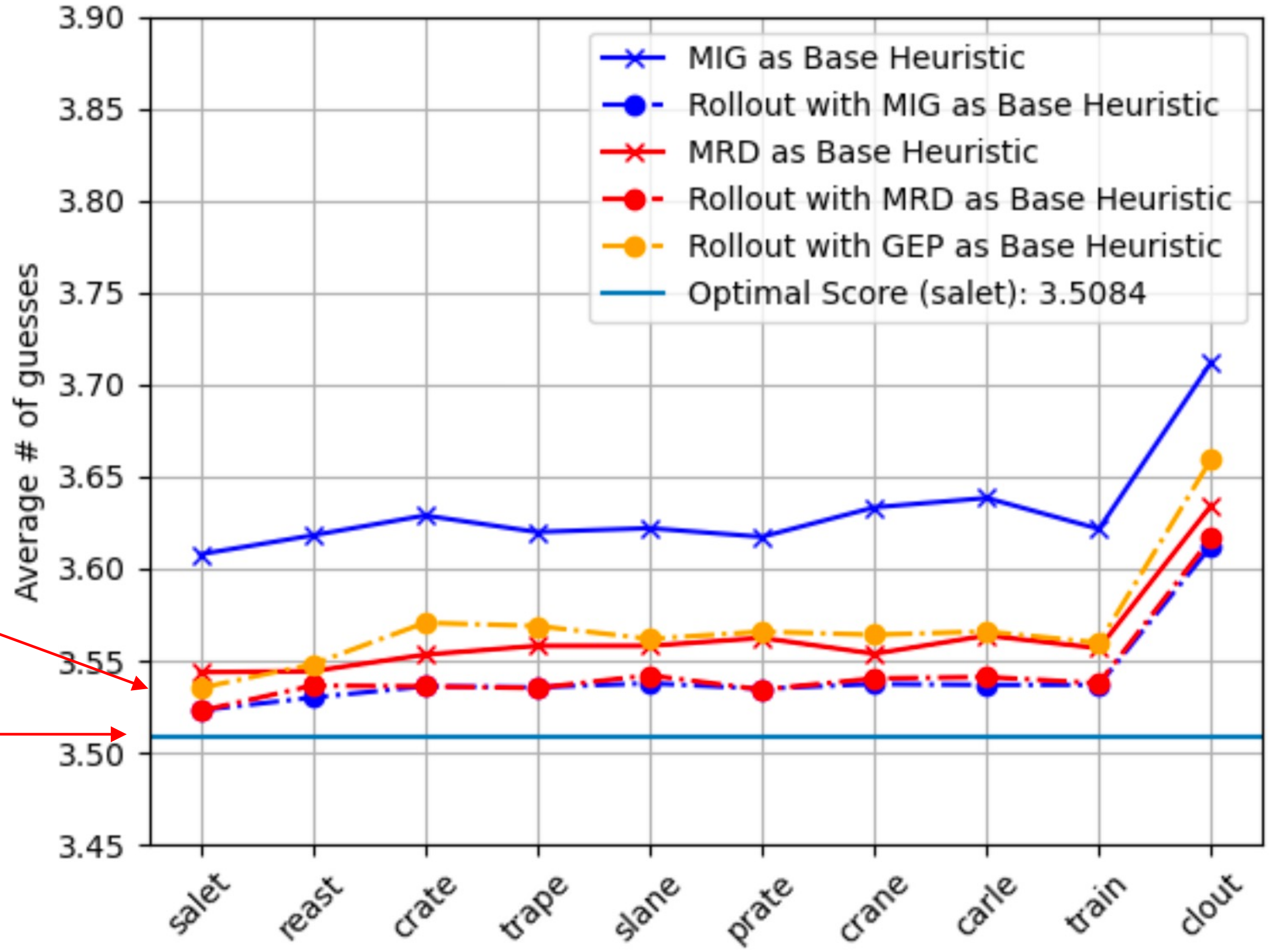
1 $avg\_Q\_list \leftarrow [\,]$;
2 **for** a_k *in* $\mathcal{A}$ **do**
3 $\quad Q\_list \leftarrow [\,]$;
4 $\quad$ **for** $\theta^i \in \Theta$ **do**
5 $\quad\quad Q_k \leftarrow \mathcal{C}(s_k, \theta^i, a_k) + J^i_{k+1}(\mathcal{T}(s_k, \theta^i, a_k))$;
6 $\quad\quad Q\_list \leftarrow Q\_list \cup [Q_k]$;
7 $\quad avg\_Q\_factor \leftarrow \sum_{i=1}^{m} b_{k,i}(Q\_list)$;
8 $\quad avg\_Q\_list \leftarrow avg\_Q\_list \cup [avg\_Q\_factor]$;
9 $\tilde{a}_k \leftarrow \arg\min_{\forall a_k \in \mathcal{A}} avg\_Q\_list$;
10 $s_{k+1} \leftarrow \mathcal{T}(s_k, \theta, \tilde{a}_k)$;
11 **return** $\tilde{a}_k, s_{k+1}$

Results for Rollout vs Optimal Scores

Opening Word	Easy Mode			Hard Mode		
	MIG as Base Heuristic	Rollout with MIG as Base Heuristic	Optimal Score	MIG as Base Heuristic	Rollout with MIG as Base Heuristic	Optimal Score
salet	3.6108 (5.54%)	3.4345 (**0.39%**)	3.4212	3.6078 (2.83%)	3.5231 (**0.42%**)	3.5084
reast	3.6 (5.19%)	3.4462 (**0.69%**)	3.4225	3.6181 (2.97%)	3.53 (**0.47%**)	3.5136
crate	3.6177 (5.64%)	3.4414 (**0.51%**)	3.4238	3.6289 (3.17%)	3.5361 (**0.53%**)	3.5175
trape	3.6319 (5.41%)	3.4604 (**0.43%**)	3.4454	3.6199 (2.9%)	3.5356 (**0.50%**)	3.5179
slane	3.6255 (5.67%)	3.4444 (**0.39%**)	3.4311	3.622 (2.89%)	3.5378 (**0.5%**)	3.5201
prate	3.6333 (5.69%)	3.4535 (**0.46%**)	3.4376	3.6173 (2.73%)	3.5348 (**0.39%**)	3.5210
crane	3.6091 (5.36%)	3.4380 (**0.36%**)	3.4255	3.6333 (3.14%)	3.5374 (**0.42%**)	3.5227
carle	3.6108 (5.32%)	3.4419 (**0.39%**)	3.4285	3.6384 (3.18%)	3.5369 (**0.31%**)	3.5261
train	3.6181 (5.07%)	3.4622 (**0.54%**)	3.4436	3.6216 (2.74%)	3.5369 (**0.34%**)	3.5248
clout	3.6955 (5.29%)	3.5248 (**0.43%**)	3.5097	3.7123 (3.32%)	3.6125 (**0.54%**)	3.5931

Table: Results using ‘Maximum Information Gain’ (MIG) as base heuristic and with rollout.

Advantage of rollout (vs only base heuristic)

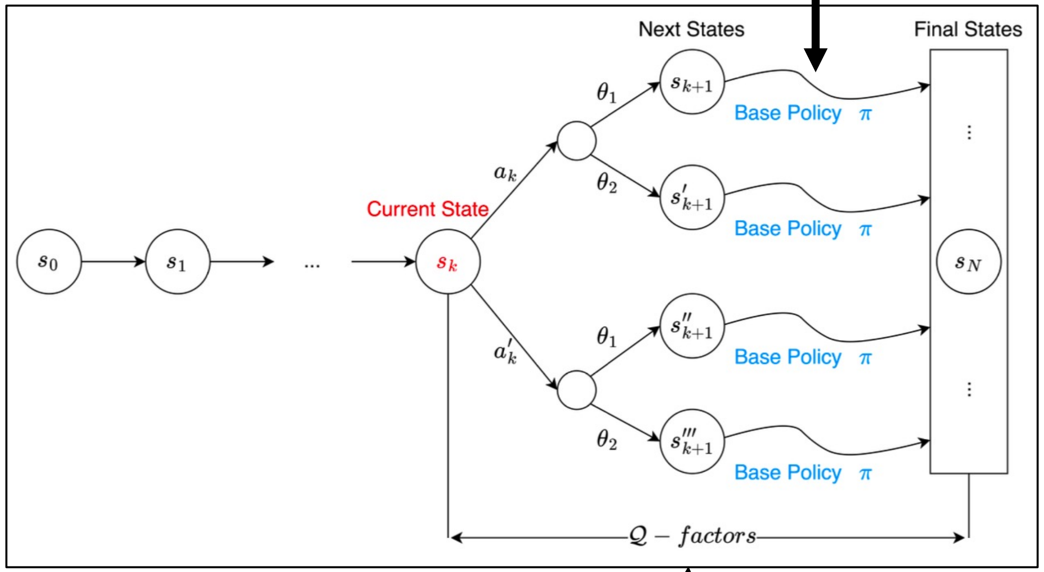


Our score: 3.5231

Optimal Score: 3.5084

Limitations of Rollout

- The need for a reasonable base policy – our experience with Wordle has been that the rollout algorithm is relatively insensitive to the base policy (e.g., the GEP heuristic).
- The need for a posterior distribution estimator - this is a limitation of most POMDP algorithms.
- The number of Q-factors that need to be computed by the algorithm online, particularly for a large action space - this difficulty may possibly be mitigated by intelligently pruning the action space or by offline training using a neural network.



Summary

- We introduced a DP-based online rollout strategy as a computationally efficient solution to deterministic POMDPs with unknown parameters, whose exact solution is intractable.
- We demonstrated our approach using the challenging online puzzle Wordle, and empirically show that our approach provides near-optimal performance and impressive improvement over the heuristic approaches that have been used so far.
- Through the Wordle computational demonstration, we identified the key obstacles in the way of solving other challenging POMDP problems that involve sequential estimation, possibly in conjunction with simultaneous adaptive control.

Access the pre-print:
(arXiv:2211.10298)