Playing Wordle Using an Online Rollout Algorithm for Deterministic POMDPs

Siddhant Bhambri
School of Computing & AI
Arizona State University
Tempe, USA
siddhantbhambri@asu.edu

Amrita Bhattacharjee
School of Computing & AI
Arizona State University
Tempe, USA
abhatt43@asu.edu

Dimitri Bertsekas
School of Computing & AI
Arizona State University
Tempe, USA
dbertsek@asu.edu

2023 IEEE Conference on Games (CoG) | 979-8-3503-2277-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/CoG57401.2023.10333228

***Abstract*— In this paper, we consider an important class of Partially Observable Markov Decision Processes (POMDP) with unknown parameters, which contains the Wordle puzzle as a special case. For this class of POMDP, we develop a new on-line solution method, which is based on the rollout approach. Our method relies on the use of a base heuristic policy and guarantees cost improvement over that policy. When applied to Wordle, our algorithm solves the puzzle on-line. The performance is within 0.4% of the known optimal results, and is substantially better than that of the base heuristic policies we have tested.**

***Index Terms*—POMDP, Wordle, Rollout, Dynamic Programming**

I. Introduction

In this paper, we introduce a rollout solution methodology for an important class of POMDP, and apply it to the popular Wordle puzzle that appears daily in the New York Times[1]. Wordle involves a list of 5-letter mystery words, which is a subset of a larger list of guess words. A word is selected at random from the mystery list, and the objective is to find that word with sequential selection of as few words as possible from the guess list. Each guess word selection provides information about the letters contained in the hidden mystery word according to a given set of rules, which involves color coding of letters shared by the guess word and the mystery word.

Our solution method may be viewed as a form of approximation in value space in the context of Reinforcement Learning (RL) or approximate Dynamic Programming (DP). It is based on the ideas of the rollout algorithm, an online policy iteration method, which improves upon the performance of any given heuristic/sub-optimal policy. We present a suitably modified rollout approach which maintains this performance improvement property.

While in this paper we focus on Wordle, our approach applies to a wider class of problems that includes sequential decoding, the Mastermind class of puzzles, and sequential estimation. Broadly speaking, our approach relates to POMDP involving a controlled deterministic dynamic system with an unknown model parameter, but with a fully observable state.

In our wider POMDP formulation, the system can have a finite number of states, and evolves according to a discrete-time equation of the form $s_{k+1} = \mathcal{T}(s_k, \theta, a_k)$, where s_k is the state at time k, a_k is the control or action to be chosen at time k out of a give finite set, θ is the unknown parameter, and $\mathcal{T}$ is an available transition function. It is assumed that θ can take a finite number of values with an a priori known probability distribution. The perfect observations of s_k serve to estimate θ asymptotically, as the system is being controlled. In the Wordle context, θ is the unknown mystery word, a_k is the guess word chosen at step k, and s_k is the current list of mystery words that are consistent with the results of the guess word selections up to step k. The belief state of the corresponding POMDP is the pair (b_k, s_k), where b_k is the posterior probability distribution of θ, given the observations $s_0, \ldots, s_k$ and the past controls $a_0, \ldots, a_{k-1}$. Here, the choice of controls has a dual objective: to produce states and controls of small current and future cost, and to identify θ.

Our rollout solution methodology relies on the use of one out of several heuristic guess word selection policies for Wordle that have been proposed in the literature (see Section II). In the terminology of rollout algorithms, the heuristic policy is referred to as the *base policy*. The rollout algorithm can be viewed as a single step of the classical policy iteration method, which starts from the base policy. The major guarantee that our rollout algorithm offers is that its performance can be shown to be better than the performance of the corresponding base policy. Indeed we will show computationally (see Section V) that the rollout policy performs better than all of the heuristic policies that we have tried by a substantial margin, and is very close to optimal (it performs within 0.4% of the optimal assuming the best opening word selection `salet`).

Our primary contributions in this paper are: (1) We present a DP-based online rollout strategy as a computationally efficient solution to deterministic POMDP. (2) We motivate our approach using the popular online puzzle Wordle as our case study for deterministic POMDP and empirically show that our approach provides near-optimal performance. (3) Our empirical evaluations further show that our online rollout approach is generalizable across various heuristics as it significantly

[1]https://www.nytimes.com/games/wordle/index.html

979-8-3503-2277-4/23/$31.00 ©2023 IEEE

improves the performance even over heuristics that may not be computationally efficient for the given problem.

We begin with a discussion of existing approaches for Wordle in Section II, we provide a formal definition of deterministic POMDP and we provide a brief description of the maximum information gain heuristic in Section III. We present our proposed rollout approach and its application to Wordle in Section IV. We then describe our experiments and results in Section V, and finally conclude this work in Section VI. An appendix is also attached[2].

II. Related Work

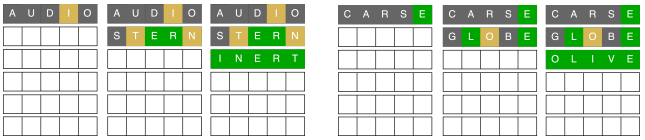
Wordle has attracted the attention of quite a few scientists, with a growing body of analysis and algorithmic development from both mathematical and computer science perspectives. Most of the discussions have focused on attempts to find the optimal strategy for guess word selection, or to propose good sub-optimal strategies. In particular, a widely known work by Selby [7] was the first to implement optimal strategies, and gave the corresponding optimal scores. Bertsimas and Paskov [3] confirmed these optimal scores by using a similar DP-based solution method.

Since we aim to device an observation strategy that finds the mystery word as quickly as possible, it is logical to try guess words that are as "informative" as possible, given the observations already available. An information theoretic approach to identify such guess words, was first suggested by Sanderson through his 3Blue1Brown[3] channel, and will be described in the next section. Additional heuristic strategies have been proposed in [4] and [5]. RL solution methods such as Deep-Q Learning and Advantage Actor Critic were tested [6], with much worse results than the MIG heuristic.

III. Background

Generally, a deterministic POMDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{C}, \Omega, \mathcal{O})$ that consists of $\mathcal{S}$, a finite set of states with the initial distribution over these states, denoted by b_0; $\mathcal{A}$, a finite set of actions; $\mathcal{T}$, a deterministic transition function: $\mathcal{T}(s_k, \theta, a_k)$ gives the next state s_{k+1} when action a_k is applied in state s_k at stage k for $s_k, s_{k+1} \in \mathcal{S}$ and $a_k \in \mathcal{A}$; $\mathcal{C}$, a cost function: $\mathcal{C}(s_k, a_k)$ that gives the cost (or negative rewards) incurred by the agent when action a_k is applied in state s_k at stage k for $s_k \in \mathcal{S}, a_k \in \mathcal{A}$; Ω, a finite set of observations; and $\mathcal{O}$, a deterministic observation function: $\mathcal{O}(s_k, a_k)$ that gives the observation $o_k \in \Omega$ when action a_k is applied in state s_k at stage k for $s_k \in \mathcal{S}, a_k \in \mathcal{A}$. We assume that the state space contains a cost-free and absorbing termination state. The total cost of a policy is the expected value (over the initial distribution b_0) of the cost incurred up to entering the termination state.

In Wordle, players have to guess a five-letter word in six attempts, which becomes the absorbing goal state for the agent. The hidden word, which we will refer to as the "mystery word" in this paper, is chosen every day from a set of 2,315 words according to some distribution. In published studies as well as the computations reported in the present paper, the distribution is assumed to be uniform. The set of mystery words, referred to as the "mystery list," is known to the public. Each guess attempt consists of a word chosen from a "guess list" of 12,972 words, also known to the public, and provides information about the letter in each of the five positions of the mystery word. This "guess list" comprises the POMDP action space $\mathcal{A}$. We assume that every guess costs +1, and that once the guess word is equal to the mystery word, we enter a cost-free termination state. Thus the total cost of an instance of the game is the number of guesses required to find the mystery word. We want to minimize the expected number of guesses to solve the puzzle, with the expectation taken over the mystery word randomly chosen according to some known distribution (the uniform distribution, in our computational experiments).



(a) Wordle in Easy mode. (b) Wordle in Hard mode.

Fig. 1: In the Easy mode (1a), the user can choose any word as the next guess word. However, in the Hard mode (1b), the user is constrained to use the letters marked as "yellow" and the letters marked as "green" that have to be at the same position as in the previous guess word. For example, here, if we get "E" as green when we play CARSE, we need to use only the words that end with an "E", and so on.

Maximum Information Gain: Consider a random variable Θ that can take a finite number of values $\theta^1, \ldots, \theta^m$ with given probabilities. An information theoretic approach aimed at estimating θ using one out of a finite number of observations Z_u, $u \in U$, is to select u that results in maximum entropy reduction (or maximizes the information gain). In particular, the (a priori) entropy of Θ is given by $H(\Theta) = -\sum_{i=1}^{m} p(\theta^i) \log\big(p(\theta^i)\big)$, where $p(\theta^i)$ is the a priori probability that Θ has the value θ^i. The a posteriori entropy of Θ given Z_u is given by

$$H(\Theta \mid Z_u) = -\sum_{j=1}^{n} p(z^j) \sum_{i=1}^{m} p(\theta^i \mid Z_u = z^j) \times \log\big(p(\theta^i \mid Z_u = z^j)\big), \tag{1}$$

where $p(z^j)$ is the probability that Z_u takes the value z^j and $p(\theta^i \mid Z_u = z^j)$ is the conditional probability that $\Theta = \theta^i$ given that Z_u has taken the value z^j. The entropy reduction (or information gain) provided by a choice $u \in U$ is the function of u given by $H(\Theta) - H(\Theta \mid Z_u)$, and the information theoretic approach suggests selecting $u \in U$ that maximizes this expression, or equivalently minimizes the a posteriori entropy $H(\Theta \mid Z_u)$. This is the basis for the maximum information gain (MIG) heuristic, which has received a lot of attention in the context of Wordle.

[2]https://tinyurl.com/appendix-wordle

[3]https://www.youtube.com/c/3blue1brown

IV. The Rollout Methodology

Assume that we have a base heuristic algorithm $\mathcal{H}$ that can choose the next guess at any stage of the game. The rollout algorithm at a given stage k considers each available action and completes the solution of the problem by using repeatedly the base heuristic to select the subsequent actions. The corresponding cost is recorded in the action's $\mathcal{Q}$-factor. The action selected for stage k is the one with minimal $\mathcal{Q}$-factor, as described in the following algorithm.

Algorithm 1: Rollout with One Step Look-ahead

Data: Current state $s_k \in \mathcal{S}$, Currently possible goal states $\mathcal{G}_k \subseteq \mathcal{S}$, Action space $\mathcal{A}$, Transition function $\mathcal{T}$, Cost function $\mathcal{C}$, Base Heuristic $\mathcal{H}$.
Result: Next state s_{k+1}.

```
1  Q̂_factors ← [ ];
2  for a in A do
3      Cost_total ← [ ];
4      for g ∈ G_k do
5          s_{k+1} ← T(s_k, a_k), cost ← 0;
6          while s_{k+1} ≠ g do
7              s_{k+1} ← arg min_{a_k ∈ A} H(T(s_k, a_k));
8              cost ← cost + C(s_k, a_k)
9          Cost_total.append(cost);
10     mean_cost ← (1/|G_k|) Σ(Cost_total);
11     Q̂_factors.append(mean_cost);
12 â_k ← arg min_{∀a_k ∈ A} Q̂_factors, s_{k+1} ← T(s_k, â_k) ;
13 return s_{k+1}
```

We will next explain in some detail Algorithm 1. We intend to calculate (approximate) $\mathcal{Q}$ factors for taking an action a_k in state s_k, i.e., $\hat{\mathcal{Q}}(s_k, a_k) = C(s_k, a_k) + \mathcal{H}(\mathcal{T}(s_k, a_k))$ (see Appendix B). In `line 1`, we begin with an empty set to store these $\mathcal{Q}$ factors for each possible action at stage k. We also assume a goal state set $\mathcal{G}_k$ that may be a subset or equal to the set of all states $\mathcal{S}$. Hence, for any possible goal state $g \in \mathcal{G}_k$, we perform rollout (`lines 4-9`) by applying the next action as selected by our base heuristic $\mathcal{H}$, and compute the total cost until termination. In `line 10`, we find the mean cost for taking action a leading to all possible terminating states in $\mathcal{G}_k$. Finally, in `line 12` we select the action $\hat{a}$ that corresponds to the minimum $\hat{\mathcal{Q}}$ factor and apply it at state s_k.

At a typical stage of the rollout algorithm, we know the current *mystery list* $\mathcal{G}_k$, i.e., the subset of possible mystery words that are consistent with the information received in response to the preceding guess selections. We also know (or can simulate) the posterior distribution over $\mathcal{G}_k$ given this information. We also know the *guess list* $\mathcal{A}_k$, which is defined as the subset of words in the original guess list $\mathcal{A}$ that are allowable for our next selection based on the information received in response to the preceding $k-1$ guess word selections. Note that in the easy mode we have $\mathcal{A}_k = \mathcal{A}$, but in the hard mode $\mathcal{A}_k$ may be a strict subset of $\mathcal{A}$. The rollout algorithm chooses the k^{th} guess word $\hat{a}_k \in \mathcal{A}_k$, the puzzle responds with some information regarding the identity and position of some letters within the mystery word according to the rules, the mystery list distribution and the guess list are updated, and the process is repeated until the mystery word is identified. The method for choosing $\hat{a}_k$ for the current stage k is described next.

For each pair (g, a_k), consisting of a possible mystery word $g \in \mathcal{G}_k$ and guess word $a_k \in \mathcal{A}_k$, and starting from the current state s_k, we calculate $\mathcal{Q}(s_k, a_k)$, the *$\mathcal{Q}$-factor of a_k, conditioned on g being the true mystery word*. This is the number of guesses required to find g, assuming that we select a_k as our first guess, and then we select the subsequent guess words using the base heuristic $\mathcal{H}$. This $\mathcal{Q}$-factor can be simply computed by simulating the base heuristic forward from stage $k+1$, knowing g and the k^{th} stage selection a_k. We also compute for each $a_k \in \mathcal{A}_k$, starting from stage s_k, the *average $\mathcal{Q}$-factor of a_k*, denoted by $\hat{\mathcal{Q}}_k(s_k, a_k)$, as $\hat{\mathcal{Q}}(s_k, a_k) = \frac{1}{|\mathcal{G}_k|} \sum_{g \in \mathcal{G}_k} \mathcal{Q}(s_k, a_k)$, where, $|\mathcal{G}_k|$ denotes the cardinality of the mystery word list $\mathcal{G}_k$. This corresponds to the $mean\_cost$ computation shown in `line 10` of Algorithm 1. The algorithm then selects in `line 12` a guess word $\hat{a}_k$ whose average $\mathcal{Q}$-factor is minimal:

$$\hat{a}_k \in \arg\min_{a_k \in A_k} \hat{\mathcal{Q}}(s_k, u_k). \tag{2}$$

In summary, the $\mathcal{Q}$-factors are averaged using the posterior distribution over the possible values of g (i.e., the current mystery list $\mathcal{G}_k$), and the guess word with minimal averaged $\mathcal{Q}$-factor is chosen, as in Eq. (2). An important fact is that the posterior distribution over $\mathcal{G}_k$ is uniform when the initial mystery word is chosen according to the uniform distribution, as we have assumed. Otherwise, Monte Carlo simulation would be needed to compute the averaged $\mathcal{Q}$-factors.

V. Experiments And Results

We focus here primarily on the maximum information gain heuristic (MIG), discussed in Section III, and we also test two other heuristics: most rapid decrease (MRD) and greatest expected probability (GEP) (see Appendix C).

Briefly, the MRD heuristic aims to select the next guess word that yields the smallest set of possible mystery words once it is selected and an observation is received. In essence, it keeps track of the number of mystery words with a non-zero probability in the belief distribution over $\mathcal{G}_k$ at any stage k of the game. On the other hand, the GEP heuristic keeps track of the expected probability that a guess word, if selected, may be the correct mystery word in the next stage of the game. This expected probability is calculated as 1 over the size of the smallest mystery word set that an observation from the game could lead to. We have re-implemented the MRD and GEP heuristics, based on our best understanding of [8], so our implementations may differ from the ones of [8].[4] Still

[4]Actually, our implementation of MRD performs very well relative to the optimal. This is not true for GEP, which performs rather poorly. Remarkably, however, our rollout algorithm that uses GEP as its base heuristic, yields near-optimal performance; see Table 2.

TABLE I: Results using 'Maximum Information Gain' as base heuristic, and with rollout.

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	3.4345	3.4212	3.6078	3.5231	3.5084
reast	3.6	3.4462	3.4225	3.6181	3.53	3.5136
crate	3.6177	3.4414	3.4238	3.6289	3.5361	3.5175
trape	3.6319	3.4604	3.4454	3.6199	3.5356	3.5179
slane	3.6255	3.4444	3.4311	3.622	3.5378	3.5201
prate	3.6333	3.4535	3.4376	3.6173	3.5348	3.5210
crane	3.6091	3.4380	3.4255	3.6333	3.5374	3.5227
carle	3.6108	3.4419	3.4285	3.6384	3.5369	3.5261
train	3.6181	3.4622	3.4436	3.6216	3.5369	3.5248
clout	3.6955	3.5248	3.5097	3.7123	3.6125	3.5931

our implementations represent legitimate heuristics, so they are suitable for comparison with the corresponding rollout algorithm, which is our principal aim.

TABLE II: Results for hard mode using 'Most Rapid Decrease' and 'Greatest Expected Probability' as base heuristic, and with rollout.

Opening Word	MRD as Base Heuristic	Rollout with MRD as Base Heuristic	GEP as Base Heuristic	Rollout with GEP as Base Heuristic
salet	3.5438	3.5227	5.8674	3.5352
reast	3.5443	3.5365	5.9244	3.5481
crate	3.5533	3.5361	5.8998	3.5706
trape	3.5581	3.5352	5.8479	3.5689
slane	3.5581	3.5421	5.9158	3.5619
prate	3.5624	3.5343	5.8462	3.5658
crane	3.5538	3.5404	5.9935	3.5641
carle	3.5637	3.5412	5.9788	3.5659
train	3.5568	3.5378	5.8907	3.5598
clout	3.6345	3.6168	5.9974	3.6596

In Table I, we give our results for the MIG heuristic for both the easy and the hard mode of the game along with optimal scores as shown in [7] and [3], and in Table II, we give our results for the MRD and GEP heuristics in just the hard mode. In both tables, we show the score for each opening word, averaged over all the 2,315 mystery words. We also compare the base heuristic and rollout performances in Fig. 2.

More specifically, we have evaluated the three heuristics and their use as base policies within the rollout approach for a selected set of opening words, which have been identified in earlier works as best or nearly best choices for initial guess selection [3], [5], [7].

In summary, our tests show that our rollout approach improves substantially upon the performance of the three heuristics. In particular, the rollout performance is very close to the optimal, as calculated in the papers [7] and [3], even in the case of the GEP base heuristic, whose performance is far from optimal. This is consistent with a general interpretation of rollout and approximation in value space methods as a single step of Newton's method for solving the Bellman equation associated with the underlying DP problem [1], [2]. The role of the base heuristic is to provide the starting point for the Newton step, and apparently all three base heuristics provide starting points that are within the region of fast convergence of Newton's method. The running time for solving any given instance of the puzzle was in the order of a few seconds, thus showing that our approach is suitable for online implementation.

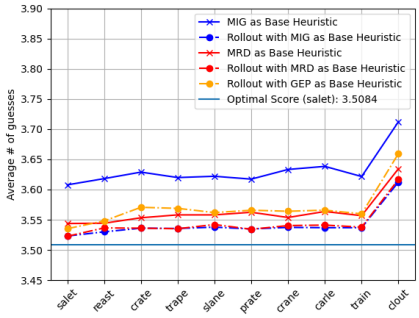


Fig. 2: Base heuristics against rollout in Hard mode.

VI. Concluding Remarks

In this paper, we described a rollout algorithm for solving online the Wordle puzzle. Our computational results show that the performance of our algorithm is very close to optimal, and much better than the ones of the three different base heuristics we used. Moreover, our rollout approach is capable of addressing extensions of Wordle (see Appendix D) and other types of online planning and adaptive control problems for which exact solution by DP is infeasible.

References

[1] D. Bertsekas, *Rollout, policy iteration, and distributed reinforcement learning*. Athena Scientific, 2020.

[2] ——, *Lessons from AlphaZero for optimal, model predictive, and adaptive control*. Athena Scientific, 2022.

[3] D. Bertsimas and A. Paskov, "An exact and interpretable solution to wordle," https://scholar.google.com/scholar?hl=en&as_sdt=0%2C3&q=An+Exact+and+Interpretable+Solution+to+Wordle&btnG=&oq=an+, 2022, accessed: 2022-11-14.

[4] M. Bonthron, "Rank one approximation as a strategy for wordle," *arXiv preprint arXiv:2204.06324*, 2022.

[5] N. de Silva, "Selecting seed words for wordle using character statistics," *arXiv preprint arXiv:2202.03457*, 2022.

[6] A. Ho, "Solving wordle with reinforcement learning," https://rb.gy/dvu2ft, 2022, accessed: 2022-11-14.

[7] A. Selby, "The best strategies for wordle," https://sonorouschocolate.com/notes/index.php?title=The_best_strategies_for_Wordle, 2022, accessed: 2022-11-14.

[8] M. B. Short, "Winning wordle wisely," *arXiv preprint arXiv:2202.02148*, 2022.

Appendix

A: Rollout, Adaptive Control and Planning

Solution methods based on a POMDP formulation such as the one that we present in our work have been discussed in the adaptive control literature since the 70s; see e.g., [12, 17, 19], and the survey [16]. Moreover, some of the pitfalls of performing parameter identification while simultaneously applying adaptive control have been described in [8], and in [15]; see the book [1], Section 6.8, for a related discussion. It is worth noting that the core objectives of adaptive control are shared with RL, which places special emphasis on the online identification of unknown environments.

Closely related problems and POMDP formulations have also been discussed extensively in the automated planning literature. For example Chapter 7 of the book [14] focuses on POMDP planning with a methodology that applies to the POMDP problems just described. In particular, online methods using heuristic cost function evaluations are discussed. Moreover, as noted in [7], planning problems with partial state observability can be formulated as search problems in belief space, with full observability of the belief state. In principle, this allows the use of online search methods for fully observable models. A survey of online planning algorithms for POMDP is given in the paper [25], with a brief mention of the rollout approach in Section 3.3.2. Moreover, special types of POMDP involving a fully observable state component, and a constant partially observable component, have been investigated in a number of works on planning. In particular, the papers [22–24], and [11] discuss Multiple-Environment Markov Decision Processes (MEMDP). These processes are described as Markov Decision Processes (MDP) equipped with multiple probabilistic transition functions, which represent the various possible unknown environments. The paper [11] describes how this special structure can be exploited to facilitate the computational solution, and notes applications in recommender systems and multi-environment robot navigation. Moreover, in addition to online POMDP planning works, there has been extensive research that deals with conditional and contingent (off-line) planning methods for POMDP; see e.g., the book [14], the papers [10, 18, 20], and the tutorial survey [9].

Generally speaking, the rollout approach has not received much attention in the automated planning literature, although it has been the subject of extensive research in control and discrete optimization; for detailed discussions and references, see the books [3, 4]. Indeed, one of our aims in this paper is to draw attention to the beneficial synergism of ideas from adaptive control theory on one hand, and POMDP planning and other RL contexts, on the other hand.

In this connection, it is also worth noting that rollout may be viewed as an online search method, which uses the cost function of the base policy as a heuristic function, in the terminology of online heuristic search (e.g., the books [13, 14, 21]). However, in the rollout algorithm the heuristic function is not admissible, i.e., it is not an underestimate of the optimal cost function. Instead, being the cost function of a policy, it is an *overestimate* of the optimal cost function. As a result, it does not offer a guarantee of asymptotically optimal performance, only a guarantee of cost improvement over the base policy. However, analytical insights (based on the Newton step interpretation noted earlier), as well as extensive computational practice (consistent with the computations reported in this paper) suggest that this cost improvement is typically substantial and often dramatic; see the discussion and the references to case studies given in the books [2, 3].

B: Rollout Methodology

Our proposed online rollout approach for the case of a deterministic POMDP is described in Section III of the paper. For a detailed mathematical discussion, including the connection of our rollout algorithm for deterministic POMDP, its relation to the one for MDP, and a discussion of the corresponding performance improvement property, see our ArXiv paper [6], and the books [4], Section 6.7, and [5], Section 2.11.

The starting point is a heuristic search algorithm, denoted by $\mathcal{H}$, which selects an action $a_k \in \mathcal{A}$ after receiving an observation $o_k \in \Omega$ when the action a_k was applied in state s_k by the agent at stage k. Our rollout algorithm uses a base heuristic $\mathcal{H}$, and aims to improve the performance of $\mathcal{H}$ and to achieve near-optimal performance while being computationally efficient.

At a typical stage of the rollout algorithm, say stage k, we know the updated belief distribution b_k based on the observations received in response to the preceding $k-1$ action selections. At this step, instead of the base heuristic, the rollout algorithm chooses the next action $a_k \in \mathcal{A}$ followed by the agent receiving the next observation $o_k \in \Omega$. The key difference between the action selection using the base

heuristic and that using the online rollout approach is that in latter case we evaluate each action a_k based on its average Q-factor, which is the total cost for using a_k in state s_k at stage k, followed by using the base heuristic in the subsequent stages, averaged over the belief distribution b_k, as indicated in Algorithm 1 of the paper.

C: Additional Experiments

The rollout approach also applies to several variations of the Wordle puzzle. Such variations include for example a larger length of mystery words, and/or a nonuniform distribution over the initial list of mystery words. For comparison purposes, we have implemented a 6-letter version of Wordle, where the mystery and guess lists consist of 6-letter words.

Table 1: Results for 6-letter Wordle using maximum information gain (MIG) as base heuristic and the corresponding rollout algorithm.

Opening Word	**Easy Mode**		**Hard Mode**	
	MIG as Base Heuristic	**Rollout with MIG as Base Heuristic**	**MIG as Base Heuristic**	**Rollout with MIG as Base Heuristic**
ambros	3.3235	3.1637	3.3024	3.1918
rabies	3.2346	3.0898	3.2294	3.1279
tances	3.2056	3.0769	3.1978	3.1041

We have used the 6-letter word list obtained from the Natural Language Corpus[1], and selected 12,972 words randomly for our guess word list, using a uniform distribution over the entire list of words. From this guess word list, we have randomly sampled 2,315 mystery words to create our mystery word list for the 6-letter Wordle experiments. In Table 1, we provide experimental results using the MIG base heuristic, for a few sample opening words.

D: Future Work

The most challenging variant of Wordle appears to be one where the probability distribution of selection of the mystery word is nonuniform; e.g., when mystery words are selected consistently with their frequency of usage within some domain. It appears computational infeasible to solve the puzzle optimally in this case, in the absence of special assumptions. For the same variant of Wordle, the main additional computation required by the rollout approach is the updating of the belief distribution over the mystery list, rather than updating the current mystery list [this is required to compute the average Q-factor of a guess word]. This can be done analytically or more likely by using online simulation-based methods such as particle filtering. Testing this approach computationally is an interesting subject for further research, either in the context of Wordle or more generally in the context of adaptive control or online planning problems.

Another important direction for further research is the use of overestimates of the optimal cost as heuristic functions (rather than underestimates as in A*, POMDP search, and related methods). In particular, rollout can be viewed as a form of search with overestimating heuristic functions that are provided by the cost functions of base policies, as we have noted in the Introduction. The use of this type of heuristic function provides a different type of performance guarantee: cost improvement over the corresponding base policy, rather than asymptotic optimality, as for example, in the case of A* search.

References

[1] D. Bertsekas, *Dynamic programming and optimal control: Volume I.* Athena scientific, 2012, vol. 4.

[2] ——, *Reinforcement learning and optimal control.* Athena Scientific, 2019.

[3] ——, *Rollout, policy iteration, and distributed reinforcement learning.* Athena Scientific, 2020.

[1] http://norvig.com/ngrams/

[4] ——, *Lessons from AlphaZero for optimal, model predictive, and adaptive control.* Athena Scientific, 2022.

[5] ——, *A Course in Reinforcement Learning.* Athena Scientific, 2023.

[6] S. Bhambri, A. Bhattacharjee, and D. Bertsekas, "Reinforcement learning methods for wordle: A pomdp/adaptive control approach," *arXiv preprint arXiv:2211.10298*, 2022.

[7] B. Bonet and H. Geffner, "Planning with incomplete information as heuristic search in belief space," in *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems*, 2000, pp. 52–61.

[8] V. Borkar and P. Varaiya, "Adaptive control of markov chains, i: Finite parameter set," *IEEE Transactions on Automatic Control*, vol. 24, no. 6, pp. 953–957, 1979.

[9] D. Bryce and S. Kambhampati, "A tutorial on planning graph based reachability heuristics," *AI Magazine*, vol. 28, no. 1, pp. 47–47, 2007.

[10] D. Bryce, S. Kambhampati, and D. E. Smith, "Planning graph heuristics for belief space search," *Journal of Artificial Intelligence Research*, vol. 26, pp. 35–99, 2006.

[11] K. Chatterjee, M. Chmelík, D. Karkhanis, P. Novotnỳ, and A. Royer, "Multiple-environment markov decision processes: Efficient analysis and applications," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 48–56.

[12] B. Doshi and S. E. Shreve, "Strong consistency of a modified maximum likelihood estimator for controlled markov chains," *Journal of Applied Probability*, vol. 17, no. 3, pp. 726–734, 1980.

[13] S. Edelkamp and S. Schrodl, *Heuristic search: theory and applications.* Elsevier, 2011.

[14] H. Geffner and B. Bonet, "A concise introduction to models and methods for automated planning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 8, no. 1, pp. 1–141, 2013.

[15] P. R. Kumar, "Optimal adaptive control of linear-quadratic-gaussian systems," *SIAM Journal on Control and Optimization*, vol. 21, no. 2, pp. 163–178, 1983.

[16] ——, "A survey of some results in stochastic adaptive control," *SIAM Journal on Control and Optimization*, vol. 23, no. 3, pp. 329–380, 1985.

[17] P. R. Kumar and W. Lin, "Optimal adaptive controllers for unknown markov chains," *IEEE Transactions on Automatic Control*, vol. 27, no. 4, pp. 765–774, 1982.

[18] S. Maliah, R. Brafman, E. Karpas, and G. Shani, "Partially observable online contingent planning using landmark heuristics," in *Twenty-Fourth International Conference on Automated Planning and Scheduling*, 2014.

[19] P. Mandl, "Estimation and control in markov chains," *Advances in Applied Probability*, vol. 6, no. 1, pp. 40–60, 1974.

[20] C. Muise, V. Belle, and S. McIlraith, "Computing contingent plans via fully observable non-deterministic planning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, 2014.

[21] J. Pearl, *Heuristics: intelligent search strategies for computer problem solving.* Addison-Wesley Longman Publishing Co., Inc., 1984.

[22] M. Randour, J.-F. Raskin, and O. Sankur, "Percentile queries in multi-dimensional markov decision processes," in *International Conference on Computer Aided Verification.* Springer, 2015, pp. 123–139.

[23] ——, "Variations on the stochastic shortest path problem," in *International Workshop on Verification, Model Checking, and Abstract Interpretation.* Springer, 2015, pp. 1–18.

[24] J.-F. Raskin and O. Sankur, "Multiple-environment markov decision processes," *arXiv preprint arXiv:1405.4733*, 2014.

[25] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, "Online planning algorithms for pomdps," *Journal of Artificial Intelligence Research*, vol. 32, pp. 663–704, 2008.