Competitive Analysis of the Explore/Exploit Tradeoff

John Langford JCL+@CS.CMU.EDU
Computer Science Department, Carnegie-Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213

Martin Zinkevich MAZ+@CS.CMU.EDU
Computer Science Department, Carnegie-Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213

Sham Kakade SHAM@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit, UCL, London WC1N 3AR, UK

Abstract

We investigate the explore/exploit trade-off in reinforcement learning using competitive analysis applied to an abstract model. We state and prove lower and upper bounds on the competitive ratio. The essential conclusion of our analysis is that optimizing the explore/exploit trade-off is much easier with a few pieces of extra knowledge such as the stopping time or upper and lower bounds on the value of the optimal exploitation policy.

1 Introduction

It is well known that efficient reinforcement learning algorithms must use some means for active exploration (see for example [9]). We consider an abstract class of algorithms where an *explicit* decision between exploration and exploitation is made. In this framework, we use competitive analysis to analyze the trade-off so as to maximize the cumulative reward the agent receives.

There are two relevant pieces of prior work that have well-defined notions of optimality in the on-line setting. First, the E^3 algorithm of Kearns and Singh [6] shows that, given certain assumptions about the world, the agent can achieve cumulative reward close to optimal in polynomial time (in the number of states). The notion of optimality used is the largest possible payoff among those policies that "mix" within some pre-chosen time. The time required to achieve near optimal payoff is then polynomial in this "mixing time". The other notion of optimality is in the competitive analysis of the N-armed bandit problem [1][4]. In this analysis we start with a box containing several levers each of which gives some reward when pulled. The goal is to choose the right lever in order to maximize reward and the expected reward is compared with the "optimal" algorithm which chooses the one lever which returns the largest reward.

Both of these notions of "optimality" are reasonable, and their respective algorithms specify means to get close to their respective notion of optimality. However, both notions assume the optimal algorithm always chooses actions under a policy that achieves the maximal reward. This is unrealistic since any practical algorithm must engage in exploration first before it finds this optimal policy. Hence, any feasible algorithm will behave in a manner quite different from optimal behavior.

We focus on analyzing *only* the exploration/exploitation trade-off. Informally, in the model we consider, the agent only has to decide between exploring and exploiting. We assume that if n exploration steps are carried out during the course of the game then this results in the same policy improvement, regardless of which state the agent is in or when this exploration was carried out. This is clearly an abstraction since exploration is, in general, state dependent. We also make a weaker assumption; that exploitation does not result in exploration.

The explore/exploit trade-off problem is a search problem. Competitive analysis has been previously applied to search problems (see [3] for examples) although this particular search problem appears new. The most strongly related previous problems are competitive financial decision making problems where the goal is to choose the optimal trading time [5][2]. The techniques applied to these problems are used here although our model is different enough such that no direct reduction exists.

In competitive analysis, the performance of our algorithm is compared with the performance that we could have achieved given all available information before decisions are made. This comparison is done on the worst sequence so the notion of an good algorithm is the algorithm that is not much worse than the optimal algorithm. This paper analyzes the worst case gap between how well an algorithm *could* have done and how well an algorithm *did*.

This paper has the following results:

(1) An abstract model for analyzing algorithms which make a trade-off between exploration and exploitation.
(2) Upper and lower bounds on competitive optimality for general algorithms in this model.
(3) Upper and lower bounds on the competitive optimality for algorithms which are given the stopping time, T, or upper and lower bounds ($R_{\min}$ and $R_{\max}$) on the average reward the optimal algorithm can achieve.

1.1 The abstract model

The common notion of optimal policy is one which does not need to engage in exploration: it simply always chooses the best actions from the start of the game. Any practical algorithm must engage in exploration and can thus never reasonably behave in a manner similar to an "optimal" algorithm for this notion of optimal. Our aim is to construct an abstract model in which an optimal algorithm must also explore. We construct an abstract model where the "optimal" algorithm engages in exploration and compare our algorithm to this (more realistic) notion of "optimal".

Our goal is to analyze the trade-off between exploration and exploitation. Suppose we have an agent, which can do two "meta-actions" which are labeled "explore " and "exploit". In addition, the agent has the knowledge of the current value of its "exploit" action. Let this *exploitation value* be R. When the "exploit" action is pressed, the agent receives reward R. When the "explore" button is pressed the agent receives 0 reward but the number R monotonically increases due to the information gained from exploration. The goal of the agent is simply to maximize the total reward that it receives.

The motivation for this problem is that it is an abstraction of a very common problem. It is often the case that an agent may choose between either exploiting with the best method derivable from the known knowledge of the world, or sacrificing the exploitation by instead gathering information which (hopefully) allows better exploitation in the future.

We insist that the *exploitation value* R increase monotonically with exploration simply because extra information acquired during exploration phases hopefully improves the exploitation efficiency. It is important to note that we do not require that the reward R increase *strictly.* An *algorithm* is some method to switch between exploration and exploitation, which can be based on the previous exploitation values. We define the *value* of an explore/exploit algorithm to be cumulative reward obtained in T steps, and the *optimal algorithm* is one which maximizes this cumulative reward.

An example sequence of changing exploitation values, R_i, might be:

$$0\ 0\ 0\ 0\ 1\ 1\ 2\ 3\ 9\ 9\ 9\ 9\ 9\ 10$$

Here, every time we explore we advance one increment in this sequence, and the starting exploitation value is R_0. Thus, after exploring 3 times, our exploitation policy has value 0. After 4 explorations, 1, and after 8, 9. We also assume that any exploitation does not advance any increments.

Let us consider the situation where the game ends in $T = 13$ rounds. In particular, we might imagine a strategy which explores 4 rounds, then exploits for 9 rounds, and a strategy which explores 12 rounds, then exploits for one round. The first strategy has cumulative reward:

$$1 * 9 = 9$$

and the second:

$$9 * 1 = 9$$

The optimal strategy explores 8 times and then exploits 5 times to achieve value:

$$9 * 5 = 45$$

Note here that we have achieved the goal of a realistic notion of optimal: our optimal algorithm engages in exploration. Also note that monotonicity implies that the optimal strategy always does exploration before exploitation. Thus, the optimal strategy always consists of exploration, and then exploitation. The problem of optimizing the trade-off is then simply finding the best switching time.

1.2 The analysis

We wish to compare the best possible strategy (given prior knowledge of the monotonically increasing reward sequence) with an algorithm that makes explore/exploit decisions based upon only observed values R. We do this in the common "competitive analysis" setting [3]. This comparison is done with a worst case over all problems.

Competitive analysis typically compares an algorithm with the "best possible" algorithm (given all information). Let C_* be the cumulative reward of the optimal algorithm. We want to use a definition which allows randomized algorithms for trading off between exploration and exploitation. A randomized algorithm has a source of uniform random bits independent of the problem. In general, any randomized algorithm, A can be regarded as a deterministic algorithm which takes it's random bits as an additional argument. If a randomized algorithm A which uses random bits r

achieves total cumulative reward $C_{A,r}$, then the competitive ratio is typically defined as:

$$\max_{\text{problems}} \frac{C_*}{E_r C_{A,r}}$$

Here r is all the random bits used by the algorithm A and E_r is the expectation over the random bits r.

2 Competitive Analysis

Before starting the analysis, we prove a meta-theorem which implies that we need only consider optimal algorithms which switch from exploration to exploitation, but not vice versa. The implication of this theorem is that the space of possible optimal algorithms is much smaller than expected.

Lemma 2.1. *(explore then exploit) For all sequences R_i and all algorithms which complete n explorations and m exploitations, the algorithm with the largest possible reward first does n explorations then m exploitations.*

Proof. Consider any algorithm which exploits during round t and explores during some later round t'. Consider the altered algorithm which explores during round t and exploits during round t'. This alteration leaves unaffected the values of exploitations at rounds earlier than t or later than t', and it monotonically improves the reward received from any exploitations in the interval $(t, t']$. The lemma follows by induction on all pairs (t, t') of exploitation followed by exploration. □

This lemma implies that the optimal algorithm (which already knows the value of exploration) *always* does exploration followed by exploitation. Our algorithm may or may not follow this behavior. Later, we prove that the best algorithm which knows the stopping time, T does all exploration before exploitation.

The optimal algorithm explores first then exploits for the remaining time. Let R_{opt} be the exploitation value per time step, and let t_{opt} be the number of times steps the optimal exploits. The cumulative reward received by optimal is then $R_{\text{opt}} t_{\text{opt}}$ and the competitive ratio is:

$$\max_{\text{problems}} \frac{R_{\text{opt}}(\text{problem}) t_{\text{opt}}(\text{problem})}{E_r C_{A,r}(\text{problem})}.$$

2.1 Results

We now present upper and lower bounds on the competitive ratio for deterministic and randomized algorithms with varying pieces of available information. In a randomized algorithm the decision to explore or exploit is stochastic. In competitive analysis, an upper bound is a bound on the *maximal* lower bound. In particular, an upper bound of X implies that there exists an algorithm with competitive ratio of X. Our proofs of the upper bounds presented involve explicitly defining such an algorithm.

Let T be the number of rounds before termination and let $R_{\max}$ and $R_{\min}$ be upper and lower bounds on R_{opt}. Note that we assume 0 is the minimum possible average reward.

We consider algorithms with knowledge of:

(1) nothing
(2) T
(3) $R_{\max}$ and $R_{\min} > 0$.
(4) T, $R_{\max}$ and $R_{\min} > 0$.

Here is a table of our results for deterministic algorithms:

	Upper bound	Lower Bound
Nothing	∞	∞
$R_{\max}, R_{\min}$	$\frac{R_{\max}}{R_{\min}}$	$\frac{R_{\max}}{2R_{\min}}$
T	T	T
$T, R_{\max}, R_{\min}$	$\min\left(T, \frac{R_{\max}}{R_{\min}}\right)$	$\min\left(T, \frac{R_{\max}}{2R_{\min}}\right)$

Note that with no information, the deterministic competitive ratio is very bad.

For randomized algorithms, it is convenient to define $f(x) = \Omega(\frac{\log x}{\log \log x})$ and $g(x) = 1 + \ln x$. Note that $g(x)$ differs by only a $\log \log x$ factor from $\frac{\log x}{\log \log x}$ implying that $g(x)$ and $f(x)$ have similar behavior. The table of results for randomized algorithms is considerably more encouraging:

	Upper bound	Lower Bound
Nothing	$\forall \epsilon > 0 \quad \frac{1+\epsilon}{\epsilon} T^{1+\epsilon}$	$T - 1$
$R_{\max}, R_{\min}$	$g\left(\frac{R_{\max}}{R_{\min}}\right)$	$f\left(\frac{R_{\max}}{R_{\min}}\right)$
T	$g(T)$	$f(T)$
$T, R_{\max}, R_{\min}$	$g(\min(T, \frac{R_{\max}}{R_{\min}}))$	$f\left(\min\left(T, \frac{R_{\max}}{R_{\min}}\right)\right)$

The competitive analysis of reinforcement learning shows that optimizing the explore/exploit trade-off is, in general, quite difficult. The analysis shows several important things:

(1) Randomized algorithms achieve significantly better performance than deterministic algorithms by amortizing worst-case situations.
(2) The lower bounds are near to the upper bounds (up to $\log \log$ factors).
(3) Knowledge of T (the termination time) implies there exists an algorithm with a competitive ratio of $\log T$.

(4) Knowledge of $R_{\max}$ and $R_{\min}$, which are upper and lower bounds on the exploitation value R_{opt} of the optimal policy, implies there exists an algorithm with a competitive ratio of $\ln \frac{R_{\max}}{R_{\min}}$.

2.2 No knowledge

The worst case is that our algorithm knows nothing. Deterministic algorithms perform remarkably poorly here.

Corollary 2.2. *(No Knowledge, deterministic lower bound) All deterministic algorithms have a competitive ratio of ∞.*

Proof. The proof is done by taking the limit as $R_{\max} \to \infty$ of the competitive ratio resulting from a theorem 2.10, which we prove later. □

The statements about what we can and cannot prove given no knowledge of T alter radically when randomization is allowed and we look at expected competitive ratios. Randomization allows us to greatly reduce both the upper and lower bounds.

Theorem 2.3. *(No Knowledge, randomized upper bound) For all $\epsilon > 0$, there exists a randomized algorithm with a competitive ratio of $\frac{1+\epsilon}{\epsilon}T^{1+\epsilon}$.*

Proof. Consider the randomized algorithm which switches from exploration to exploitation at time step t with probability:

$$\Pr(t) \propto \frac{1}{t^{1+\epsilon}}$$

The normalization for this distribution is bounded. In particular:

$$\sum_{t=1}^{\infty} \frac{1}{t^{1+\epsilon}} = 1 + \sum_{t=2}^{\infty} \frac{1}{t^{1+\epsilon}} \leq 1 + \int_{t=1}^{\infty} \frac{1}{t^{1+\epsilon}} dt = \frac{1+\epsilon}{\epsilon}$$

The optimal algorithm switches from exploration to exploitation at some value t_{opt}. Our randomized algorithm switches at time t_{opt} with probability greater than $\frac{\epsilon}{1+\epsilon}\frac{1}{t_{\text{opt}}^{1+\epsilon}}$. This implies that the expected value is at least $\frac{\epsilon}{1+\epsilon}\frac{C_{\text{opt}}}{t_{\text{opt}}^{1+\epsilon}}$ and so the competitive ratio is bounded by:

$$\frac{1+\epsilon}{\epsilon}T^{1+\epsilon}$$

□

This upper bound is near to the following randomized lower bound.

Theorem 2.4. *(No Knowledge, randomized lower bound) For all randomized algorithms, the competitive ratio is at least $T-1$.*

Proof. Since the competitive ratio is defined in a worst case scenario, we choose a particular sequence and show how all randomized algorithms have a competitive ratio of at least $T-1$ on this sequence.

Consider the sequence which has an ith entry of $R_i = (4i+4)^i$. This sequence has the property that the optimal policy always explores except for the final round where it exploits.

Let p_i be the chance the algorithm first switches from exploration to exploitation on the ith round in this sequence. Since $\sum_{t=1}^{\infty} \frac{1}{t} = \infty$ and probability distributions are normalized, we know that there must exist some time t where $p_t \leq \frac{1}{2t}$. Let us choose the stopping time to be $T = t+1$ and let C_i be the total cumulative reward obtained if the algorithm switches at time i. Since a property of the sequence is that $C_i \geq C_{i-1}$ and since $C_i = 0$ for $i \geq t+1$, the payoff is

$$\begin{aligned}
EA &= \sum_{i=1}^{t} p_i C_i \\
&\leq C_t p_t + C_{t-1} \sum_{i=1}^{t-1} p_i \\
&\leq \frac{C_t}{2t} + C_{t-1} \\
&= \frac{R_t}{2t} + 2R_{t-1} \\
&\leq \frac{R_t}{2t} + \frac{R_t}{2t}
\end{aligned}$$

where the last step follows since by definition of the sequence, $R_{t-1} \leq \frac{R_t}{4t}$. The optimal algorithm achieves total reward $C_{\text{opt}} = R_t$ so the competitive ratio is at least $t = T-1$. □

2.3 Knowledge of T only

The upper and lower bounds shift dramatically when T is known. The deterministic competitive ratio drops from ∞ to T.

Theorem 2.5. *(Know T, deterministic upper bound) Given knowledge of T, there exists a deterministic algorithm with competitive ratio of T.*

Proof. Consider the algorithm which explores until the last step, and then exploits once. This algorithm has a one round reward at least as large as R_{opt} which implies a competitive ratio of T, since optimal could obtain at most $R_{\text{opt}}T$. □

A lower bound of T also applies.

Corollary 2.6. *(Know T, deterministic lower bound) All deterministic algorithms have a competitive ratio of at least T.*

Proof. Optimization of the full knowledge deterministic lower bound (Theorem 2.16). □

Randomized algorithms also improve their competitive ratio significantly.

Theorem 2.7. *(Know T, randomized upper bound) There exists a randomized algorithm which achieves a competitive ratio of $1 + \ln T$ given knowledge of T.*

Proof. For any sequence, let t_{opt} be the number of times that the optimal algorithm exploits. Also let R_{opt} be the reward received each turn the optimal algorithm exploits.

We analyze a randomized algorithm which starts out doing exploration and randomly decides the amount of time t which it exploits. If we guess $t \le t_{\text{opt}}$, then we receive a return on each round of exploitation which is at least as large as R_{opt} because we explore at least as often as the optimal algorithm. If we guess a $t > t_{\text{opt}}$, then we could obtain 0 reward, since not enough exploration was conducted. Hence, the expected cumulative reward of our algorithm is bounded by:

$$\begin{aligned}
E_r[C_r] &\ge E_r[C_r | t \le t_{\text{opt}}] \Pr_r[t \le t_{\text{opt}}] \\
&\ge E_r[R_{\text{opt}} t | t \le t_{\text{opt}}] \Pr_r[t \le t_{\text{opt}}] \\
&\ge R_{\text{opt}} E_r[t | t \le t_{\text{opt}}] \Pr_r[t \le t_{\text{opt}}]
\end{aligned}$$

Now we specify how to choose this time. First, choose v from the real numbers in $[1, T]$ according to the following cumulative distribution:

$$\forall x \in [1, T], \Pr_r[v \le x] = \frac{1 + \ln x}{1 + \ln T}$$

Notice that this cumulative distribution places a point mass on $v = 1$. Let the switching time be $t = \lceil v \rceil$. Since t_{opt} is an integer, $t \le t_{\text{opt}}$ if and only if $v \le t_{\text{opt}}$. Therefore, $E_r[t | t \le t_{\text{opt}}] \ge E_r[v | v \le t_{\text{opt}}]$. Also, $\Pr_r[t \le t_{\text{opt}}] = \Pr_r[v \le t_{\text{opt}}]$. It follows that:

$$\begin{aligned}
& E_r[C_r] \\
\ge\ & R_{\text{opt}} E_r[v | v \le t_{\text{opt}}] \Pr_r[v \le t_{\text{opt}}] \\
\ge\ & R_{\text{opt}} E_r[v | v = 1] \Pr_r[v = 1] \\
& + R_{\text{opt}} E_r[v | 1 < v \le t_{\text{opt}}] \Pr_r[1 < v \le t_{\text{opt}}] \\
\ge\ & R_{opt} \left(\frac{1 + \ln 1}{1 + \ln T} + \int_1^{t_{opt}} \frac{x dx}{x(1 + \ln T)} \right) \\
\ge\ & R_{opt} \left(\frac{1}{1 + \ln T} + \frac{1}{1 + \ln T} \int_1^{t_{opt}} dx \right) \\
\ge\ & \frac{R_{\text{opt}} t_{\text{opt}}}{1 + \ln T}
\end{aligned}$$

This proves a competitive ratio of $1 + \ln T$. □

A similar lower bound also holds for randomized algorithms.

Corollary 2.8. *(Know T, randomized lower). For all randomized algorithms, the competitive ratio is $\Omega\left(\frac{\log T}{\log \log T}\right)$.*

Proof. The corollary is implied by optimization of the full knowledge lower bound 2.17. □

2.4 Knowledge of $R_{\max}$ and $R_{\min}$

It turns out that knowledge of just $R_{\min}$ (a lower bound on R_{opt}) or just $R_{\max}$ (an upper bound on R_{opt}) is not very interesting, since we can show the competitive ratio is infinite in both of these cases. Intuitively, consider the case where $R_{\min}$ is known, an adversarially chosen sequence can just use an infinite $R_{\max}$ to force an infinite competitive rate.

However, when *both* R_{max} and $R_{\min}$ are known, the lower bounds no longer apply. Essentially, knowledge of lower and upper bounds which are not separated by too many orders of magnitude can be regarded as a limit on the precision of exploitation values.

The first step is upper and lower bounds for deterministic algorithms. The analysis shows a large improvement over the "know nothing" case.

Theorem 2.9. *(Know R_{max} and R_{min}, deterministic upper bound) There exists a deterministic algorithm with competitive ratio $\frac{R_{max}}{R_{min}}$.*

Proof. Consider the algorithm which always exploits as soon as it sees an exploitation value $R \ge R_{\min}$. This algorithm exploits for $t \ge t_{\text{opt}}$ rounds since the optimal algorithm must explore for at least as many rounds. The competitive ratio is $\frac{R_{\text{opt}} t_{\text{opt}}}{Rt} \le \frac{R_{\max} t_{\text{opt}}}{R_{\min} t_{\text{opt}}} = \frac{R_{\max}}{R_{\min}}$. □

This upper bound is tight up to a constant of 2.

Corollary 2.10. *(Know R_{max} and R_{min}, deterministic lower bound) Given R_{min} and R_{max} s.t. $R_{min} \le R_{opt} \le R_{max}$, all deterministic algorithms have a competitive ratio greater than $\frac{R_{max}}{2R_{min}}$.*

Proof. Optimization of the deterministic full knowledge lower bound (Theorem 2.16). □

Once again, randomization allows us to reduce the upper bound from a linear to a logarithmic factor.

Theorem 2.11. *(Know R_{min}, R_{max}, randomized upper bound) If R_{min} and R_{max} are lower and upper bounds on R_{opt}, then there exists a randomized* *algorithm with knowledge of R_{min} and R_{max} which achieves a competitive ratio of* $1+\ln\frac{R_{max}}{R_{min}}$.

Proof. Let us choose a reward threshold G at random, where we choose to exploit if our exploitation value is ever greater than G. Specifically, for all $x \in [R_{\min}, R_{\max}]$, let

$$\Pr[G \le x] = \frac{1+\ln\frac{x}{R_{\min}}}{1+\ln\frac{R_{\max}}{R_{\min}}}$$

Note that this cumulative distribution function implies a point mass at $G = R_{\min}$ given by $\Pr[G = R_{\min}] = \frac{1}{1+\ln\frac{R_{\max}}{R_{\min}}}$. This distribution is normalized because the cumulative distribution is 1 at $R_{\max}$ and the probability of $G = R_{\min}$ is equal to the cumulative probability at $G = R_{\min}$.

Suppose that the optimal algorithm exploits when the exploitation value is R_{opt}. Then, if we set G below R_{opt}, we receive a reward for at least as many rounds, but may only receive G each of those rounds. If we set G above R_{opt}, we might not receive anything. Let us only consider the expected reward of the algorithm($E_r C_r$) in a round where the optimal algorithm receives a reward. Let the instantaneous reward our algorithm receives be denoted by R_A

$$\begin{array}{rc} & E_r R_A \\ = & R_{\min}\Pr_r[G = R_{\min}] + \int_{R_{\min}}^{R_{\text{opt}}}\Pr[G=y]y dy \\ = & R_{\min}\frac{1}{1+\ln\frac{R_{\max}}{R_{\min}}} + \int_{R_{\min}}^{R_{\text{opt}}}\frac{1}{y(1+\ln\frac{R_{\max}}{R_{\min}})}y dy \\ = & \frac{R_{\min}}{1+\ln\frac{R_{\max}}{R_{\min}}} + \frac{R_{\text{opt}}-R_{\min}}{1+\ln\frac{R_{\max}}{R_{\min}}} \\ = & \frac{R_{\text{opt}}}{1+\ln\frac{R_{\max}}{R_{\min}}} \end{array}$$

Therefore, the cumulative return is at least $\frac{R_{\text{opt}}t_{\text{opt}}}{1+\ln\frac{R_{\max}}{R_{\min}}}$ which implies the desired competitive ratio. □

Corollary 2.12. *(Know R_{max} and R_{min}, randomized lower). For all randomized algorithms, the competitive ratio is* $\Omega\left(\frac{\log\frac{R_{max}}{R_{min}}}{\log\log\frac{R_{max}}{R_{min}}}\right)$.

Proof. The corollary is implied by optimization of the full knowledge lower bound, theorem 2.17. □

2.5 Knowledge of $R_{\min}$, $R_{\max}$, and T

The conjunction of knowledge about the optimal algorithms exploitation value and the number of rounds yields straightforward results. In essence, we can always do at least as well as either piece of knowledge allows us to do individually.

Lemma 2.13. *(Randomized and deterministic full knowledge upper bounds) Given any algorithm A_m with a computable competitive ratio $V_m(R_{max}, R_{min})$ using R_{max} and R_{min} and any other algorithm A_T with a computable competitive ratio $V_T(T)$ using knowledge of T, there exists an algorithm with competitive ratio* $\min(V_m(R_{max}, R_{min}), V_T(T))$.

Proof. Consider the algorithm which evaluates $\operatorname{argmin}(V_{\text{m}}(R_{\max}, R_{\min}), V_T(T))$ and then simply uses the algorithm which achieves the minimum. □

Our lower bounds have essentially the same functional form as the upper bound. Before stating lower bounds, we first first show that it is necessary only to consider "monotonic" algorithms.

Definition 2.14. An algorithm is *monotonic* if it never switches from exploiting to exploring.

Lemma 2.15. *For all algorithms A there exists a monotonic algorithm A' (using T and A) such that for all sequences of length T, A' achieves greater reward than A.*

Proof. From Lemma 2.1, we know that the algorithm which does n explorations followed by m exploitations achieves the largest possible reward amongst all algorithms which do n explorations and m exploitations in any order. We construct an algorithm A' which uses A as an oracle in order to determine the number n without engaging in any exploitation steps.

The essential technique is simulation. We specify exactly how algorithm A' acts on some *particular* sequence using algorithm A and T. Algorithm A' always outputs "explore" when algorithm A outputs "explore" and it simulates a round for algorithm A whenever algorithm A outputs "exploit". Algorithm A' can perfectly simulate an "exploit" round for algorithm A because algorithm A receives no new information whenever it exploits. The process of simulation continues until the number of simulated rounds plus the number of "explore" actions reaches T. Then, algorithm A' outputs "exploit" for all remaining rounds. By construction, algorithm A' does all "explores" before "exploits". Also by construction, algorithm A' does exactly as many "explores" as algorithm A and so the total cumulative reward of algorithm A' is larger than or equal to the total cumulative reward of algorithm A. □

Now, we are ready to state the full knowledge lower bound theorem for deterministic algorithms.

Theorem 2.16. *(Know R_{max}, R_{min}, and T, deterministic lower bound). No algorithm given only knowledge of R_{max}, R_{min}, and T can achieve a competitive ratio less than* $\min\left(\frac{R_{max}}{2R_{min}}, T\right)$.

Proof. Consider all sequences of length T beginning with $R_{\min}$ and ending with 0 to T exploitation values of $R_{\max}$. These sequences have the form

$$R_{\min} \ldots R_{\min} R_{\max} \ldots R_{\max}$$

. There are two classes of deterministic algorithms: (1) Algorithms which for all sequences exploit at $R_{\max}$ or (2) algorithms for which there exists a sequence and round $t+1$ (dependent on the algorithm) where the deterministic algorithm exploits a value of $R_{\min}$.

Algorithms in class (1) have an infinite competitive ratio on the sequence $R_{\min} \ldots R_{\min}$.

For algorithms in class (2) consider the sequence which consists of t exploitation values with $R_{\mathrm{t}} = R_{\min}$ followed by $T-t-1$ exploitation values of $R_{\max}$. The deterministic algorithm achieves cumulative reward $C = R_{\min}(T-t)$ while the optimal algorithm achieves competitive reward $C_{\mathrm{opt}} = \max(R_{\max}(T-t-1), R_{\min}T)$. This implies a competitive ratio of

$$\frac{C_{\mathrm{opt}}}{C} = \frac{\max(R_{\max}(T-t-1), R_{\min}T)}{R_{\min}(T-t)}$$

For $t = T-1$, the competitive ratio is T, and for $t < T-1$, the competitive ratio is at least $\frac{R_{\max}}{2R_{\min}}$. $\square$

The full knowledge randomized lower bound is the most difficult proof we present. It gives intuition for the innate difficulties in optimizing the explore/exploit tradeoff.

Theorem 2.17. *(Know R_{max}, R_{min}, and T, randomized lower bound) Let $f(x) = \frac{\log_2 x}{\log_2 \log_2 x}$. All algorithms given only knowledge of R_{max}, R_{min}, and T have a competitive ratio of $\Omega\left(f(\min(\frac{R_{max}}{R_{min}}, T))\right)$.*

Proof. Pick the largest l for which $\min(\frac{R_{\max}}{R_{\min}} - l^{2l}, T - l^l) > 0$. We define a set of l exploitation value sequences and show that all algorithms (randomized or deterministic) perform poorly on these sequences. Let R_{kj} denote the jth exploitation value of the kth sequence. For all $i \in \{1,\ldots,l\}$ let $T_i = T - l^{l-i}$. For all $k \in \{1,\ldots,l\}$, and $j \in \{1,\ldots,T\}$ define

$$\begin{aligned} R_{kj} &= 0 \text{ for all } j < T_1 \\ R_{kj} &= R_{\min} l^{2i} \text{ for } j \text{ satisfying } T_i \le j < T_{i+1} < T_k \\ R_{kj} &= R_{\min} l^{2k} \text{ for all } j \ge T_k \end{aligned}$$

Since the exploitation values do not change between round T_i and round T_{i+1}, it is sometimes convenient to index the exploitation value by $R'_{ki} \equiv R_{kj}$ for all $j \in \{T_i, T_{i+1} - 1\}$.

Consider the sequence $k = l$. Every algorithm has some probability of switching from exploration to exploitation at each round on this sequence. Fix some algorithm A and let p_{kj} be the probability of switching from exploration to exploitation after exactly j rounds of exploration on sequence k. Let C_r be the total cumulative reward achieved by algorithm A using random bits r. Since the exploitation value is the same for all $j \in \{T_i, T_{i+1} - 1\}$, we know that for all l sequences:

$$\begin{aligned} E_r C_r &= \sum_{j=1}^{T} p_{kj} R_{kj} (T-j) \\ &\le \sum_{i=1}^{l} p'_{ki} R'_{ki} l^{l-i} \end{aligned}$$

where $p'_{ki} = \sum_{j=T_i}^{j=T_{i+1}-1} p_{kj}$.

Since the kth sequence is indistinguishable from the lth sequence until round T_{k+1}, all algorithms must have $p'_{ki} = p'_{li}$ for all $i \le k$. Also, we have $p'_{k(k+1)} \le \sum_{i=k+1}^{l} p'_{li}$. Since we have $R'_{ki} = R'_{kk}$ for all $i > k$, we get:

$$\begin{aligned} & E_r C_r \\ \le\ & \sum_{i=1}^{k} p'_{li} R'_{ki} l^{l-i} + R'_{kk} l^{l-k-1} \sum_{i=k+1}^{l} p'_{li} \\ =\ & R_{\min}[p'_{lk} l^{l+k} + l^{l+k-1} \sum_{i=k+1}^{l} p'_{li} + \sum_{i=1}^{k-1} p'_{li} l^{l+i}] \\ \le\ & R_{\min}[p'_{lk} l^{l+k} + l^{l+k-1}] \\ =\ & R_{\min}(p'_{lk} + \tfrac{1}{l}) l^{l+k} \end{aligned}$$

Since $\sum_i p'_{li} = 1$ there exists an i for which $p'_{li} \le \frac{1}{l}$. On sequence i, we have an expected reward of at most $E_r C_r \le 2R_{\min} l^{l+i-1}$. On sequence i the optimal algorithm always chooses to exploit at round i and receive reward:

$$C_{\mathrm{opt}} = R_{\min} l^{2i} (T - T_i) = R_{\min} l^{l+i}$$

Consequently, the competitive ratio is at least $\frac{C_{\mathrm{opt}}}{E_r C_r} \ge \frac{l}{2}$ establishing a lower bound of $\Omega(l)$. To achieve the theorem, note that $f(x)^{f(x)} \le x$. $\square$

3 Application

3.1 Application to Markov Decision Processes

The model in this paper in which an agent can either do pure information gathering or pure exploitation does not quite fit the standard Markov Decision Process used to formalize the reinforcement learning setting. We now outline and discuss where specific aspects do not hold.

The four assumptions implicitly made in defining our abstract algorithm are:

(1) The value of the exploitation policy (given the exploration done) is known and a "sub-algorithm" exists which monotonically increases the exploitation policy with more exploration.
(2) The sequence of exploitation values is fixed.
(3) Exploration results in zero reward while exploitation results in zero exploration.
(4) Any information gained through exploitation does not result in policy improvement.

Assumption 1 does not hold in stochastic policies, though it might still hold with high probability in some situations.

Realizing assumption 2 is difficult in a multi-state world. The potential benefit of exploration in general depends on the state the agent is in, so the decision to explore or exploit should be state dependent. However, if each round of the algorithm represents a game which ends in some finite time, then the agent might decide whether it explores or exploits during the course of each game. The criterion is then to maximize the cumulative reward in the T games being played.

The impact of assumption 3 is unclear, in the more general framework. Exploration in general results in some reward, otherwise the exploitation policy could not improve. However, the impact this has on the current situation is not clear.

The impact of assumption 4 is in general be problem dependent.

Fully analyzing the exact connections between this abstract analysis and the reinforcement learning is an open problem and important for practical algorithms.

3.2 Application to computational search problems

There are many examples of planning and search problems where the computational difficulty of finding a good solution is much greater than the computational difficulty of verifying a good solution. This class of problems includes the NP-complete problems as well as others. Many of these problems have known methods for approximation—in particular it may be possible to find an approximate solution quickly and the quality of the solution can be refined with additional computation. For a discussion of some of these situations see [7].

When the utility of an approximate solution decreases with time used to find such a solution, these problems have an inherent exploration/exploitation tradeoff. A meta-algorithm can either decide to "explore" (and thus find a better solution) or "exploit" and execute the solution locking in the value of the solution. At any round, it is often easy to calculate the optimal "exploitation" value. Furthermore, our assumption that exploitation does not provide exploration is often satisfied since using an approximate solution often does not aid in the calculation of a better approximate solution.

4 Conclusion

We analyzed the explore/exploit tradeoff with an abstract model and show that a small amount of extra information (knowledge of the number of timesteps T or upper and lower bounds on R_{opt}) can greatly improve the competitive ratio. The lower bounds and the upper bounds in the analysis are tight up to $\log\log$ factors.

There are two undesirable properties of our analysis which can be addressed with a refined analysis. Although we show that we can achieve something near optimal in expectation, the variance on the reward achieved can be quite large. This problem has cropped up elsewhere (see [8] for an example) in competitive analysis and been successfully addressed.

Another undesirable property of our analysis is that every upper bound is instantiated with an algorithm that does all exploration before exploitation. An alteration of the model removes this unintuitive result. In particular, if the algorithm knows the *distribution* of ending times, there are examples where it is desirable to switch from exploitation back to exploration.

Acknowledgments

Thanks to Avrim Blum for many helpful suggestions and good advice.

References

[1] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. "Gambling in a rigged casino: the adversarial multi-armed bandit problem", Proc. of the 36th Annual Symposium on Foundations of Computer Science, 1995.

[2] A. Blum, T. Sandholm, M. Zinkevich. "Online Algorithms for Market Clearing", Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2002

[3] Allan Borodin and Ran El-Yaniv. "Online Computation and Competitive Analysis", Cambridge University Press, 1998.

[4] Michael Duff, "Q-Learning for Bandit Problems", Proceedings of Machine Learning 1995.

[5] R. El-Yaniv, A. Fiat, R. M. Karp, and G. Turpin, "Competitive Analysis of Financial Games", FOCS92, pages 327-333.

[6] M. Kearns and S. Singh. "Near-optimal reinforcement learning in polynomial time", Proc. 15th International Conf. on Machine Learning, 1998.

[7] K. Larson and T. Sandholm. "Bargaining with limited computation: Deliberation equilibrium" Artificial Intelligence 2001, 132(2): 183–217.

[8] Stefano Leonardi, Alberto Marchetti-Spaccamela, Alessio Presciutti, Adi Ros'en, "On-line Randomized Call Control Revisited", SODA (Symposium on Discrete Algorithms) 1998.

[9] S. D. Whitehead, "Complexity and cooperation in Q-learning", Proc. 8th International Conf. on Machine Learning, pages 363-367, 1991