Anytime Multi-Armed Bandits Algorithms

Alexander Berenbeim

Today

What Are Anytime Algorithms

What Are Anytime Algorithms

- Algorithms which generate imprecise but increasingly better approximate solutions as run time T increases are anytime algorithms, i.e. offer a trade off between solution *quality* and *computation time*

What Are Anytime Algorithms

- Algorithms which generate imprecise but increasingly better approximate solutions as run time T increases are anytime algorithms, i.e. offer a trade off between solution *quality* and *computation time*
- The distinguishing feature of these algorithms is that we can interrupt the algorithm at *any* time and receive an *acceptable* approximate solution, e.g. the `Newton-Raphson` algorithm

What Are Anytime Algorithms

- Algorithms which generate imprecise but increasingly better approximate solutions as run time T increases are anytime algorithms, i.e. offer a trade off between solution *quality* and *computation time*
- The distinguishing feature of these algorithms is that we can interrupt the algorithm at *any* time and receive an *acceptable* approximate solution, e.g. the `Newton-Raphson` algorithm
- Arise when making time dependent decisions: for example, rerouting flights to minimize revenue loss given a storm.

What Are Anytime Algorithms

- Algorithms which generate imprecise but increasingly better approximate solutions as run time T increases are anytime algorithms, i.e. offer a trade off between solution *quality* and *computation time*
- The distinguishing feature of these algorithms is that we can interrupt the algorithm at *any* time and receive an *acceptable* approximate solution, e.g. the `Newton-Raphson` algorithm
- Arise when making time dependent decisions: for example, rerouting flights to minimize revenue loss given a storm.
- Can be constructed as an algorithm with a parameter that influences the running time

What Are Anytime Algorithms

- Algorithms which generate imprecise but increasingly better approximate solutions as run time T increases are anytime algorithms, i.e. offer a trade off between solution *quality* and *computation time*
- The distinguishing feature of these algorithms is that we can interrupt the algorithm at *any* time and receive an *acceptable* approximate solution, e.g. the `Newton-Raphson` algorithm
- Arise when making time dependent decisions: for example, rerouting flights to minimize revenue loss given a storm.
- Can be constructed as an algorithm with a parameter that influences the running time

What Should Anytime Algorithms Require

What Should Anytime Algorithms Require

1. Interruptability: The algorithm can be stopped at *any time* and provide an *approximate solution*

What Should Anytime Algorithms Require

1. Interruptability: The algorithm can be stopped at *any time* and provide an *approximate solution*
2. Preemptability: we can suspend the algorithm and resume it with minimal overhead.

What Should Anytime Algorithms Require

1. Interruptability: The algorithm can be stopped at *any time* and provide an *approximate solution*
2. Preemptability: we can suspend the algorithm and resume it with minimal overhead.
3. Measurable quality: qualities of approximate result have precise measurements,

What Should Anytime Algorithms Require

1. Interruptability: The algorithm can be stopped at *any time* and provide an *approximate solution*
2. Preemptability: we can suspend the algorithm and resume it with minimal overhead.
3. Measurable quality: qualities of approximate result have precise measurements,e.g.
    - Certainty: probability of correctness

What Should Anytime Algorithms Require

1. Interruptability: The algorithm can be stopped at *any time* and provide an *approximate solution*
2. Preemptability: we can suspend the algorithm and resume it with minimal overhead.
3. Measurable quality: qualities of approximate result have precise measurements,e.g.
   - Certainty: probability of correctness
   - Accuracy: error bounds

What Should Anytime Algorithms Require

1. Interruptability: The algorithm can be stopped at *any time* and provide an *approximate solution*
2. Preemptability: we can suspend the algorithm and resume it with minimal overhead.
3. Measurable quality: qualities of approximate result have precise measurements,e.g.
   - Certainty: probability of correctness
   - Accuracy: error bounds
   - Specificity: amount of particulars

What Should Anytime Algorithms Require

1. Interruptability: The algorithm can be stopped at *any time* and provide an *approximate solution*
2. Preemptability: we can suspend the algorithm and resume it with minimal overhead.
3. Measurable quality: qualities of approximate result have precise measurements,e.g.
   - Certainty: probability of correctness
   - Accuracy: error bounds
   - Specificity: amount of particulars
4. Recognizable quality: the quality of an output can be determined by runtime

What Should Anytime Algorithms Require

1. Interruptability: The algorithm can be stopped at *any time* and provide an *approximate solution*
2. Preemptability: we can suspend the algorithm and resume it with minimal overhead.
3. Measurable quality: qualities of approximate result have precise measurements,e.g.
   - Certainty: probability of correctness
   - Accuracy: error bounds
   - Specificity: amount of particulars
4. Recognizable quality: the quality of an output can be determined by runtime
5. Monotonicity: quality is non-decreasing function of run time

What Should Anytime Algorithms Require

1. Interruptability: The algorithm can be stopped at *any time* and provide an *approximate solution*
2. Preemptability: we can suspend the algorithm and resume it with minimal overhead.
3. Measurable quality: qualities of approximate result have precise measurements,e.g.
   - Certainty: probability of correctness
   - Accuracy: error bounds
   - Specificity: amount of particulars
4. Recognizable quality: the quality of an output can be determined by runtime
5. Monotonicity: quality is non-decreasing function of run time
6. Consistency: quality is correlated with computation time and input quality

What Should Anytime Algorithms Require

1. Interruptability: The algorithm can be stopped at *any time* and provide an *approximate solution*
2. Preemptability: we can suspend the algorithm and resume it with minimal overhead.
3. Measurable quality: qualities of approximate result have precise measurements,e.g.
   - Certainty: probability of correctness
   - Accuracy: error bounds
   - Specificity: amount of particulars
4. Recognizable quality: the quality of an output can be determined by runtime
5. Monotonicity: quality is non-decreasing function of run time
6. Consistency: quality is correlated with computation time and input quality
7. Diminishing Returns: solution quality improvement diminishes over time

What Should Anytime Algorithms Require

1. Interruptability: The algorithm can be stopped at *any time* and provide an *approximate solution*
2. Preemptability: we can suspend the algorithm and resume it with minimal overhead.
3. Measurable quality: qualities of approximate result have precise measurements,e.g.
   - Certainty: probability of correctness
   - Accuracy: error bounds
   - Specificity: amount of particulars
4. Recognizable quality: the quality of an output can be determined by runtime
5. Monotonicity: quality is non-decreasing function of run time
6. Consistency: quality is correlated with computation time and input quality
7. Diminishing Returns: solution quality improvement diminishes over time

EXAMPLE: TSP

- Any iterative improvement algorithm for TSP can be viewed as an anytime algorithm

EXAMPLE: TSP

- Any iterative improvement algorithm for TSP can be viewed as an anytime algorithm
- Specifically, finding a tour with minimum cost by *randomized tour improvement* where r edges in one feasible tour are exchanged for r edges not in the present solution such that the new solution is a tour of cost less than the previous tour would be an anytime algorithm.

EXAMPLE: TSP

- Any iterative improvement algorithm for TSP can be viewed as an anytime algorithm
- Specifically, finding a tour with minimum cost by *randomized tour improvement* where r edges in one feasible tour are exchanged for r edges not in the present solution such that the new solution is a tour of cost less than the previous tour would be an anytime algorithm.

What Are The Multiarmed Bandit Problems

- Conceptually: A gambler needs to decide which *arms* of a K-slot machine should be pulled to maximize total reward over a series of trials.

What Are The Multiarmed Bandit Problems

- Conceptually: A gambler needs to decide which *arms* of a K-slot machine should be pulled to maximize total reward over a series of trials.
- Strategic Tradeoffs: Exploitation versus Exploration

What Are The Multiarmed Bandit Problems

- Conceptually: A gambler needs to decide which *arms* of a K-slot machine should be pulled to maximize total reward over a series of trials.
- Strategic Tradeoffs: Exploitation versus Exploration
- Formally: `MBP` are described by pairing $(\mathcal{S}, \Gamma)$, where $\mathcal{S}$ is a set whose elements are strategies and Γ, a family of functions $c : \mathcal{S} \to \mathbb{R}$, called cost functions

What Are The Multiarmed Bandit Problems

- Conceptually: A gambler needs to decide which *arms* of a K-slot machine should be pulled to maximize total reward over a series of trials.
- Strategic Tradeoffs: Exploitation versus Exploration
- Formally: `MBP` are described by pairing $(\mathcal{S}, \Gamma)$, where $\mathcal{S}$ is a set whose elements are strategies and Γ, a family of functions $c : \mathcal{S} \to \mathbb{R}$, called cost functions
- An `MBA` is a randomized online algorithm specified by $(\Omega_{alg}, \{X_t : \Omega_{alg} \times \mathbb{R}^{t-1} \to \mathcal{S}\}_{t \leq T})$

What Are The Multiarmed Bandit Problems

- Conceptually: A gambler needs to decide which *arms* of a K-slot machine should be pulled to maximize total reward over a series of trials.
- Strategic Tradeoffs: Exploitation versus Exploration
- Formally: `MBP` are described by pairing $(\mathcal{S}, \Gamma)$, where $\mathcal{S}$ is a set whose elements are strategies and Γ, a family of functions $c : \mathcal{S} \to \mathbb{R}$, called cost functions
- An `MBA` is a randomized online algorithm specified by $(\Omega_{alg}, \{X_t : \Omega_{alg} \times \mathbb{R}^{t-1} \to \mathcal{S}\}_{t \leq T})$
  - Ω_{alg} is a probability space, T is our run time

What Are The Multiarmed Bandit Problems

- Conceptually: A gambler needs to decide which *arms* of a K-slot machine should be pulled to maximize total reward over a series of trials.
- Strategic Tradeoffs: Exploitation versus Exploration
- Formally: `MBP` are described by pairing $(\mathcal{S}, \Gamma)$, where $\mathcal{S}$ is a set whose elements are strategies and Γ, a family of functions $c : \mathcal{S} \to \mathbb{R}$, called cost functions
- An `MBA` is a randomized online algorithm specified by $(\Omega_{alg}, \{X_t : \Omega_{alg} \times \mathbb{R}^{t-1} \to \mathcal{S}\}_{t \leq T})$
    - Ω_{alg} is a probability space, T is our run time
    - $X_t(r, y_1, \ldots, y_{t-1}) = x$ is read as *choosing* strategy x at time t if the *random seed* is r and the costs observed for trials 1 to t-1 are y_1 to y_{t-1} respectively.

What Are The Multiarmed Bandit Problems

- Conceptually: A gambler needs to decide which *arms* of a K-slot machine should be pulled to maximize total reward over a series of trials.
- Strategic Tradeoffs: Exploitation versus Exploration
- Formally: `MBP` are described by pairing $(\mathcal{S}, \Gamma)$, where $\mathcal{S}$ is a set whose elements are strategies and Γ, a family of functions $c : \mathcal{S} \to \mathbb{R}$, called cost functions
- An `MBA` is a randomized online algorithm specified by $(\Omega_{alg}, \{X_t : \Omega_{alg} \times \mathbb{R}^{t-1} \to \mathcal{S}\}_{t \leq T})$
  - Ω_{alg} is a probability space, T is our run time
  - $X_t(r, y_1, \ldots, y_{t-1}) = x$ is read as *choosing* strategy x at time t if the *random seed* is r and the costs observed for trials 1 to t-1 are y_1 to y_{t-1} respectively.
- Performance is measured by regret, the difference between the expected return of an optimal strategy and the gambler's expected return.

What Are The Multiarmed Bandit Problems

- Conceptually: A gambler needs to decide which *arms* of a K-slot machine should be pulled to maximize total reward over a series of trials.
- Strategic Tradeoffs: Exploitation versus Exploration
- Formally: `MBP` are described by pairing $(\mathcal{S}, \Gamma)$, where $\mathcal{S}$ is a set whose elements are strategies and Γ, a family of functions $c : \mathcal{S} \to \mathbb{R}$, called cost functions
- An `MBA` is a randomized online algorithm specified by $(\Omega_{alg}, \{X_t : \Omega_{alg} \times \mathbb{R}^{t-1} \to \mathcal{S}\}_{t \leq T})$
  - Ω_{alg} is a probability space, T is our run time
  - $X_t(r, y_1, \ldots, y_{t-1}) = x$ is read as *choosing* strategy x at time t if the *random seed* is r and the costs observed for trials 1 to t-1 are y_1 to y_{t-1} respectively.
- Performance is measured by regret, the difference between the expected return of an optimal strategy and the gambler's expected return.

Analyzing an Adversarial Bandit Problem

- An adversary for `MBA` is specified by Ω_{adv} and $\{C_t : \Omega_{adv} \times \mathcal{S}^{t-1} \to \Gamma\}_{t \leq T}$

Analyzing an Adversarial Bandit Problem

- An adversary for `MBA` is specified by Ω_{adv} and $\{C_t : \Omega_{adv} \times \mathcal{S}^{t-1} \to \Gamma\}_{t \leq T}$
- We read $C_t(r', x_1, \ldots, x_{t-1}) = c$ as the choice of cost function that an *adversary* makes given random seed and previous strategies employed by the algorithm.

Analyzing an Adversarial Bandit Problem

- An adversary for `MBA` is specified by Ω_{adv} and $\{C_t : \Omega_{adv} \times \mathcal{S}^{t-1} \to \Gamma\}_{t \leq T}$
- We read $C_t(r', x_1, \ldots, x_{t-1}) = c$ as the choice of cost function that an *adversary* makes given random seed and previous strategies employed by the algorithm.
- Adversaries can be oblivious or adaptive, and deterministic or probabilistic, where the choice for oblivious adversaries is either constant mapping on the domain, or dependent only on r (e.g. C_t is a random variable)

Analyzing an Adversarial Bandit Problem

- An adversary for MBA is specified by Ω_{adv} and $\{C_t : \Omega_{adv} \times \mathcal{S}^{t-1} \to \Gamma\}_{t \leq T}$
- We read $C_t(r', x_1, \ldots, x_{t-1}) = c$ as the choice of cost function that an *adversary* makes given random seed and previous strategies employed by the algorithm.
- Adversaries can be oblivious or adaptive, and deterministic or probabilistic, where the choice for oblivious adversaries is either constant mapping on the domain, or dependent only on r (e.g. C_t is a random variable)
- Given ALG and ADV, the transcript of play is the tuple $(\Omega, (x_t), (c_t), (y_t))$ where $\Omega = \Omega_{alg} \times \Omega_{adv}$

Analyzing an Adversarial Bandit Problem

- An adversary for MBA is specified by Ω_{adv} and $\{C_t : \Omega_{adv} \times \mathcal{S}^{t-1} \to \Gamma\}_{t \leq T}$
- We read $C_t(r', x_1, \ldots, x_{t-1}) = c$ as the choice of cost function that an *adversary* makes given random seed and previous strategies employed by the algorithm.
- Adversaries can be oblivious or adaptive, and deterministic or probabilistic, where the choice for oblivious adversaries is either constant mapping on the domain, or dependent only on r (e.g. C_t is a random variable)
- Given ALG and ADV, the transcript of play is the tuple $(\Omega, (x_t), (c_t), (y_t))$ where $\Omega = \Omega_{alg} \times \Omega_{adv}$ and
  - $x_t(r, r') = X_t(r, y_1(r, r'), \ldots, y_{t-1}(r, r'))$

Analyzing an Adversarial Bandit Problem

- An adversary for MBA is specified by Ω_{adv} and $\{C_t : \Omega_{adv} \times \mathcal{S}^{t-1} \to \Gamma\}_{t \leq T}$
- We read $C_t(r', x_1, \ldots, x_{t-1}) = c$ as the choice of cost function that an *adversary* makes given random seed and previous strategies employed by the algorithm.
- Adversaries can be oblivious or adaptive, and deterministic or probabilistic, where the choice for oblivious adversaries is either constant mapping on the domain, or dependent only on r (e.g. C_t is a random variable)
- Given ALG and ADV, the transcript of play is the tuple $(\Omega, (x_t), (c_t), (y_t))$ where $\Omega = \Omega_{alg} \times \Omega_{adv}$ and
    - $x_t(r, r') = X_t(r, y_1(r, r'), \ldots, y_{t-1}(r, r'))$
    - $c_t(r, r') = C_t(r', x_1(r, r'), \ldots, x_{t-1}(r, r'))$

Analyzing an Adversarial Bandit Problem

- An adversary for `MBA` is specified by Ω_{adv} and $\{C_t : \Omega_{adv} \times \mathcal{S}^{t-1} \to \Gamma\}_{t \leq T}$
- We read $C_t(r', x_1, \ldots, x_{t-1}) = c$ as the choice of cost function that an *adversary* makes given random seed and previous strategies employed by the algorithm.
- Adversaries can be oblivious or adaptive, and deterministic or probabilistic, where the choice for oblivious adversaries is either constant mapping on the domain, or dependent only on r (e.g. C_t is a random variable)
- Given `ALG` and `ADV`, the transcript of play is the tuple $(\Omega, (x_t), (c_t), (y_t))$ where $\Omega = \Omega_{alg} \times \Omega_{adv}$ and
  - $x_t(r, r') = X_t(r, y_1(r, r'), \ldots, y_{t-1}(r, r'))$
  - $c_t(r, r') = C_t(r', x_1(r, r'), \ldots, x_{t-1}(r, r'))$
  - $y_t(r, r') = c_t(r, r')$

Analyzing an Adversarial Bandit Problem

- An adversary for MBA is specified by Ω_{adv} and $\{C_t : \Omega_{adv} \times \mathcal{S}^{t-1} \to \Gamma\}_{t \leq T}$
- We read $C_t(r', x_1, \ldots, x_{t-1}) = c$ as the choice of cost function that an *adversary* makes given random seed and previous strategies employed by the algorithm.
- Adversaries can be oblivious or adaptive, and deterministic or probabilistic, where the choice for oblivious adversaries is either constant mapping on the domain, or dependent only on r (e.g. C_t is a random variable)
- Given ALG and ADV, the transcript of play is the tuple $(\Omega, (x_t), (c_t), (y_t))$ where $\Omega = \Omega_{alg} \times \Omega_{adv}$ and
  - $x_t(r, r') = X_t(r, y_1(r, r'), \ldots, y_{t-1}(r, r'))$
  - $c_t(r, r') = C_t(r', x_1(r, r'), \ldots, x_{t-1}(r, r'))$
  - $y_t(r, r') = c_t(r, r')$
- Oblivious adversary algorithms have a regret lower bound of $\Omega(\sqrt{KT})$.

Analyzing an Adversarial Bandit Problem

- An adversary for MBA is specified by Ω_{adv} and $\{C_t : \Omega_{adv} \times \mathcal{S}^{t-1} \to \Gamma\}_{t \leq T}$
- We read $C_t(r', x_1, \ldots, x_{t-1}) = c$ as the choice of cost function that an *adversary* makes given random seed and previous strategies employed by the algorithm.
- Adversaries can be oblivious or adaptive, and deterministic or probabilistic, where the choice for oblivious adversaries is either constant mapping on the domain, or dependent only on r (e.g. C_t is a random variable)
- Given ALG and ADV, the transcript of play is the tuple $(\Omega, (x_t), (c_t), (y_t))$ where $\Omega = \Omega_{alg} \times \Omega_{adv}$ and
  - $x_t(r, r') = X_t(r, y_1(r, r'), \ldots, y_{t-1}(r, r'))$
  - $c_t(r, r') = C_t(r', x_1(r, r'), \ldots, x_{t-1}(r, r'))$
  - $y_t(r, r') = c_t(r, r')$
- Oblivious adversary algorithms have a regret lower bound of $\Omega(\sqrt{KT})$.

Measuring Regret

- Given $\mathtt{ALG}$ and $\mathtt{ADV}$ on $(\mathcal{S}, \Gamma)$, the regret of $\mathtt{ALG}$ relative to strategy x is

$$R(\mathtt{ALG}, \mathtt{ADV}; x, T) = \mathbb{E}\left[\sum_{t \le T} c_t(x_t) - c_t(x)\right]$$

Measuring Regret

- Given $\texttt{ALG}$ and $\texttt{ADV}$ on $(\mathcal{S}, \Gamma)$, the regret of $\texttt{ALG}$ relative to strategy x is

$$R(\texttt{ALG}, \texttt{ADV}; x, T) = \mathbb{E}\left[\sum_{t \leq T} c_t(x_t) - c_t(x)\right]$$

- The normalized regret is defined

$$\bar{R}(\texttt{ALG}, \texttt{ADV}; x, T) = \frac{1}{T} R(\texttt{ALG}, \texttt{ADV}; x, T)$$

Measuring Regret

- Given $\mathtt{ALG}$ and $\mathtt{ADV}$ on $(\mathcal{S}, \Gamma)$, the regret of $\mathtt{ALG}$ relative to strategy x is

$$R(\mathtt{ALG}, \mathtt{ADV}; x, T) = \mathbb{E}\left[\sum_{t \leq T} c_t(x_t) - c_t(x)\right]$$

- The normalized regret is defined

$$\bar{R}(\mathtt{ALG}, \mathtt{ADV}; x, T) = \frac{1}{T} R(\mathtt{ALG}, \mathtt{ADV}; x, T)$$

- If $\mathcal{A}$ is a set of adversaries, and $U \subseteq \mathcal{S}$, the normalized U-regret of $\mathtt{ALG}$ against $\mathcal{A}$ is

$$\bar{R}(\mathtt{ALG}, \mathcal{A}; U, T) := \max_{\mathtt{ADV} \in \mathcal{A}} \max_{x \in U} \bar{R}(\mathtt{ALG}, \mathtt{ADV}; x, T)$$

Measuring Regret

- Given $\texttt{ALG}$ and $\texttt{ADV}$ on $(\mathcal{S}, \Gamma)$, the regret of $\texttt{ALG}$ relative to strategy x is

$$R(\texttt{ALG}, \texttt{ADV}; x, T) = \mathbb{E}\left[\sum_{t \leq T} c_t(x_t) - c_t(x)\right]$$

- The normalized regret is defined

$$\bar{R}(\texttt{ALG}, \texttt{ADV}; x, T) = \frac{1}{T} R(\texttt{ALG}, \texttt{ADV}; x, T)$$

- If $\mathcal{A}$ is a set of adversaries, and $U \subseteq \mathcal{S}$, the normalized U-regret of $\texttt{ALG}$ against $\mathcal{A}$ is

$$\bar{R}(\texttt{ALG}, \mathcal{A}; U, T) := \max_{\texttt{ADV} \in \mathcal{A}} \max_{x \in U} \bar{R}(\texttt{ALG}, \texttt{ADV}; x, T)$$

Good multiarmed algorithm: EXP3

- (weak regret) $\left(\max_j \sum_{t=1}^{T} c_j(t)\right) - \mathbb{E}[G_A(T)]$, where $G_A(t) = \sum_{s=1}^{t} c_{i_s}(s)$ the sum of observed rewards.

Good multiarmed algorithm: EXP3

- (weak regret) $\left(\max_j \sum_{t=1}^{T} c_j(t)\right) - \mathbb{E}[G_A(T)]$, where $G_A(t) = \sum_{s=1}^{t} c_{i_s}(s)$ the sum of observed rewards.
- Exponential Weight Algorithm For Exploration and Exploitation (EXP3) is a good algorithm for dealing with adaptive adversaries

Good multiarmed algorithm: EXP3

- (weak regret) $\left(\max_j \sum_{t=1}^{T} c_j(t)\right) - \mathbb{E}[G_A(T)]$, where $G_A(t) = \sum_{s=1}^{t} c_{i_s}(s)$ the sum of observed rewards.
- Exponential Weight Algorithm For Exploration and Exploitation (EXP3) is a good algorithm for dealing with adaptive adversaries
- In EXP3, the probability of choosing the k^{th} lever at round t is defined by

Good multiarmed algorithm: EXP3

- (weak regret) $\left(\max_j \sum_{t=1}^{T} c_j(t)\right) - \mathbb{E}[G_A(T)]$, where $G_A(t) = \sum_{s=1}^{t} c_{i_s}(s)$ the sum of observed rewards.
- Exponential Weight Algorithm For Exploration and Exploitation (EXP3) is a good algorithm for dealing with adaptive adversaries
- In EXP3, the probability of choosing the k^{th} lever at round t is defined by
  - $p_k(t) = (1-\gamma)\frac{w_k(t)}{\sum_{j\in[K]} w_j(t)} + \frac{\gamma}{K}$

Good multiarmed algorithm: EXP3

- (weak regret) $\left(\max_j \sum_{t=1}^{T} c_j(t)\right) - \mathbb{E}[G_A(T)]$, where $G_A(t) = \sum_{s=1}^{t} c_{i_s}(s)$ the sum of observed rewards.
- Exponential Weight Algorithm For Exploration and Exploitation (EXP3) is a good algorithm for dealing with adaptive adversaries
- In EXP3, the probability of choosing the k^{th} lever at round t is defined by
  - $p_k(t) = (1-\gamma)\frac{w_k(t)}{\sum_{j\in[K]} w_j(t)} + \frac{\gamma}{K}$
  - with the weight for the k^{th} level updated as $w_k(t+1) = w_k(t)\exp\left(\gamma\frac{c_j(t)}{p_j(t)K}\right)$ and otherwise $w_j(t+1) := w_j(t)$.

Good multiarmed algorithm: EXP3

- (weak regret) $\left(\max_j \sum_{t=1}^{T} c_j(t)\right) - \mathbb{E}[G_A(T)]$, where $G_A(t) = \sum_{s=1}^{t} c_{i_s}(s)$ the sum of observed rewards.
- Exponential Weight Algorithm For Exploration and Exploitation (EXP3) is a good algorithm for dealing with adaptive adversaries
- In EXP3, the probability of choosing the k^{th} lever at round t is defined by
  - $p_k(t) = (1-\gamma)\dfrac{w_k(t)}{\sum_{j\in[K]} w_j(t)} + \dfrac{\gamma}{K}$
  - with the weight for the k^{th} level updated as $w_k(t+1) = w_k(t)\exp\left(\gamma \dfrac{c_j(t)}{p_j(t)K}\right)$ and otherwise $w_j(t+1) := w_j(t)$.
- When pitted against an adaptive adversary, EXP3 achieves regret $O(\sqrt{TK\log(K)})$ on strategy set $\{1,\ldots,K\}$

- (weak regret) $\left(\max_j \sum_{t=1}^{T} c_j(t)\right) - \mathbb{E}[G_A(T)]$, where $G_A(t) = \sum_{s=1}^{t} c_{i_s}(s)$ the sum of observed rewards.
- Exponential Weight Algorithm For Exploration and Exploitation (EXP3) is a good algorithm for dealing with adaptive adversaries
- In EXP3, the probability of choosing the k^{th} lever at round t is defined by
  - $p_k(t) = (1-\gamma)\dfrac{w_k(t)}{\sum_{j\in[K]} w_j(t)} + \dfrac{\gamma}{K}$
  - with the weight for the k^{th} level updated as $w_k(t+1) = w_k(t)\exp\left(\gamma \dfrac{c_j(t)}{p_j(t)K}\right)$ and otherwise $w_j(t+1) := w_j(t)$.
- When pitted against an adaptive adversary, EXP3 achieves regret $O(\sqrt{TK\log(K)})$ on strategy set $\{1,\ldots,K\}$

Anytime Bandit Algorithms

- We say $\mathtt{ALG}$ on strategy set $\mathbb{N}$ is an anytime bandit algorithm on $\mathbb{N}$ if there exists $\tau : \mathbb{N} \times \mathbb{R}_{>0} \to \mathbb{N}$ such that $\bar{R}(\mathtt{ALG}, \mathcal{A}_{obl}; \{1, \ldots, j\}, T) < \delta$ for all $T > \tau(j, \delta)$.

- We say ALG on strategy set $\mathbb{N}$ is an anytime bandit algorithm on $\mathbb{N}$ if there exists $\tau : \mathbb{N} \times \mathbb{R}_{>0} \to \mathbb{N}$ such that $\bar{R}(\mathtt{ALG}, \mathcal{A}_{obl}; \{1, \ldots, j\}, T) < \delta$ for all $T > \tau(j, \delta)$.
- $\tau(j, \delta)$ is called the convergence time for ALG

- We say ALG on strategy set $\mathbb{N}$ is an anytime bandit algorithm on $\mathbb{N}$ if there exists $\tau : \mathbb{N} \times \mathbb{R}_{>0} \to \mathbb{N}$ such that $\bar{R}(\texttt{ALG}, \mathcal{A}_{obl}; \{1, \ldots, j\}, T) < \delta$ for all $T > \tau(j, \delta)$.
- $\tau(j, \delta)$ is called the convergence time for ALG
- If such an algorithm exists, then for all probability spaces $(\mathcal{S}, \mu)$, there is an anytime algorithm for $(\mathcal{S}, \mu)$, such that $\tau : \mathbb{R}^2_{>0} \to \mathbb{N}$ such that for all $(\varepsilon, \delta) \in \mathbb{R}^2_{>0}$ and randomized oblivious adversaries ADV, there is $U \subseteq \mathcal{S}$ such that $\mu(\mathcal{S} \setminus U) \leq \varepsilon$ and $\bar{R}(\texttt{ALG}, \texttt{ADV}; U, T) < \delta$ for all $T > \tau(\varepsilon, \delta)$.

- We say $\mathtt{ALG}$ on strategy set $\mathbb{N}$ is an anytime bandit algorithm on $\mathbb{N}$ if there exists $\tau : \mathbb{N} \times \mathbb{R}_{>0} \to \mathbb{N}$ such that $\bar{R}(\mathtt{ALG}, \mathcal{A}_{obl}; \{1, \ldots, j\}, T) < \delta$ for all $T > \tau(j, \delta)$.
- $\tau(j, \delta)$ is called the convergence time for $\mathtt{ALG}$
- If such an algorithm exists, then for all probability spaces $(\mathcal{S}, \mu)$, there is an anytime algorithm for $(\mathcal{S}, \mu)$, such that $\tau : \mathbb{R}^2_{>0} \to \mathbb{N}$ such that for all $(\varepsilon, \delta) \in \mathbb{R}^2_{>0}$ and randomized oblivious adversaries $\mathtt{ADV}$, there is $U \subseteq \mathcal{S}$ such that $\mu(\mathcal{S} \backslash U) \leq \varepsilon$ and $\bar{R}(\mathtt{ALG}, \mathtt{ADV}; U, T) < \delta$ for all $T > \tau(\varepsilon, \delta)$.

Constructing An Anytime Algorithm

- Given any monotonic $F : \mathbb{N} \to \mathbb{N}$, we define `ABA(F)` as follows:

Constructing An Anytime Algorithm

- Given any monotonic $F : \mathbb{N} \to \mathbb{N}$, we define `ABA(F)` as follows:
  - For each $k \in \mathbb{N}$, at time F(k), `ALG` initializes `EXP3` on strategy set $\{1, 2, \ldots, 2^k\}$.

Constructing An Anytime Algorithm

- Given any monotonic $F : \mathbb{N} \to \mathbb{N}$, we define `ABA(F)` as follows:
  - For each $k \in \mathbb{N}$, at time F(k), `ALG` initializes `EXP3` on strategy set $\{1, 2, \ldots, 2^k\}$.
  - From $F(k)$ to $F(k+1) - 1$, it uses this instance of `EXP3` to select strategies in $\mathbb{N}$.

Constructing An Anytime Algorithm

- Given any monotonic $F : \mathbb{N} \rightarrow \mathbb{N}$, we define `ABA(F)` as follows:
  - For each $k \in \mathbb{N}$, at time F(k), `ALG` initializes `EXP3` on strategy set $\{1, 2, \ldots, 2^k\}$.
  - From $F(k)$ to $F(k+1) - 1$, it uses this instance of `EXP3` to select strategies in $\mathbb{N}$.
  - At the end of each trial, the cost of the chosen strategy is fed back into `EXP3`.

Constructing An Anytime Algorithm

- Given any monotonic $F : \mathbb{N} \to \mathbb{N}$, we define ABA(F) as follows:
  - For each $k \in \mathbb{N}$, at time F(k), ALG initializes EXP3 on strategy set $\{1, 2, \ldots, 2^k\}$.
  - From $F(k)$ to $F(k+1) - 1$, it uses this instance of EXP3 to select strategies in $\mathbb{N}$.
  - At the end of each trial, the cost of the chosen strategy is fed back into EXP3.
- The rest of the talk consists of showing for the class of adaptive adversaries $\mathcal{A}$ on strategy set $\mathbb{N}$ and cost function class $\Gamma = [0, 1]^{\mathbb{N}}$, the regret of ABA (F) satisfies

$$\bar{R}(\mathtt{ABA}(F), \mathcal{A}; [j]_+, T) = O\left(F\left(\frac{\lceil \log_2 j \rceil}{T}\right) + \sqrt{\frac{k2^k}{T}}\right)$$

Constructing a General Anytime Algorithm

- Assume that $A_{\mathbb{N}}$ is an anytime algorithm for $\mathbb{N}$ with convergence time $\tau(j, \delta)$.

Constructing a General Anytime Algorithm

- Assume that $A_{\mathbb{N}}$ is an anytime algorithm for $\mathbb{N}$ with convergence time $\tau(j, \delta)$.For $(\mathcal{S}, \mu)$, we make an anytime algorithm as follows

Constructing a General Anytime Algorithm

- Assume that $A_{\mathbb{N}}$ is an anytime algorithm for $\mathbb{N}$ with convergence time $\tau(j, \delta)$.For $(\mathcal{S}, \mu)$, we make an anytime algorithm as follows
  1. Sample infinite sequence $(x_i)_{\mathbb{N}} \subseteq \mathcal{S}$

  drawn iid with distribution \$$\mu$\$

- Assume that $A_{\mathbb{N}}$ is an anytime algorithm for $\mathbb{N}$ with convergence time $\tau(j, \delta)$.For $(\mathcal{S}, \mu)$, we make an anytime algorithm as follows
  1. Sample infinite sequence $(x_i)_{\mathbb{N}} \subseteq \mathcal{S}$

  drawn iid with distribution \$$\mu$\$
  1. A_μ simulates $A_{\mathbb{N}}$ choosing x_j everytime $A_{\mathbb{N}}$ chooses $j \in \mathbb{N}$

Constructing a General Anytime Algorithm

- Assume that $A_{\mathbb{N}}$ is an anytime algorithm for $\mathbb{N}$ with convergence time $\tau(j, \delta)$.For $(\mathcal{S}, \mu)$, we make an anytime algorithm as follows
  1. Sample infinite sequence $(x_i)_{\mathbb{N}} \subseteq \mathcal{S}$

  drawn iid with distribution \$$\mu$\$
  1. A_μ simulates $A_{\mathbb{N}}$ choosing x_j everytime $A_{\mathbb{N}}$ chooses $j \in \mathbb{N}$
  2. Using lazy evaluation, when $A_{\mathbb{N}}$ chooses a new j, A_μ draws \$$\mathrm{x_j}$\$

Constructing a General Anytime Algorithm

- Assume that $A_{\mathbb{N}}$ is an anytime algorithm for $\mathbb{N}$ with convergence time $\tau(j, \delta)$.For $(\mathcal{S}, \mu)$, we make an anytime algorithm as follows
    1. Sample infinite sequence $(x_i)_{\mathbb{N}} \subseteq \mathcal{S}$

drawn iid with distribution \$$\mu$\$

1. A_μ simulates $A_{\mathbb{N}}$ choosing x_j everytime $A_{\mathbb{N}}$ chooses $j \in \mathbb{N}$
2. Using lazy evaluation, when $A_{\mathbb{N}}$ chooses a new j, A_μ draws \$x$_j$\$

- A_μ has convergence time $\tau^*(\varepsilon, \delta) = \tau\left(\left\lceil \frac{1}{\varepsilon} \log\left(\frac{2}{\delta}\right)\right\rceil\right)$

Outline Of Proof of Convergence

- Let $T \geq \tau^*(\varepsilon, \delta)$ and for $\theta \in [0, 1]$, let U_θ denote the strategies of $\mathcal{S}$ with average cost greater than θ (this is measurable).

Outline Of Proof of Convergence

- Let $T \geq \tau^*(\varepsilon, \delta)$ and for $\theta \in [0, 1]$, let U_θ denote the strategies of $\mathcal{S}$ with average cost greater than θ (this is measurable).
- Since measure we have

Outline Of Proof of Convergence

- Let $T \geq \tau^*(\varepsilon, \delta)$ and for $\theta \in [0, 1]$, let U_θ denote the strategies of $\mathcal{S}$ with average cost greater than θ (this is measurable).
- Since measure we have
  - $\theta^* = \inf\{\theta : \mu(U_\theta) < 1 - \varepsilon\}$

Outline Of Proof of Convergence

- Let $T \geq \tau^*(\varepsilon, \delta)$ and for $\theta \in [0,1]$, let U_θ denote the strategies of $\mathcal{S}$ with average cost greater than θ (this is measurable).
- Since measure we have
  - $\theta^* = \inf\{\theta : \mu(U_\theta) < 1 - \varepsilon\}$
  - $U = \bigcap_{\theta < \theta^*} U_\theta$

Outline Of Proof of Convergence

- Let $T \geq \tau^*(\varepsilon, \delta)$ and for $\theta \in [0,1]$, let U_θ denote the strategies of $\mathcal{S}$ with average cost greater than θ (this is measurable).
- Since measure we have
  - $\theta^* = \inf\{\theta : \mu(U_\theta) < 1 - \varepsilon\}$
  - $U = \bigcap_{\theta < \theta^*} U_\theta$
  - $V = U_{\theta^*} = \bigcup_{\theta > \theta^*} U_\theta$

Outline Of Proof of Convergence

- Let $T \geq \tau^*(\varepsilon, \delta)$ and for $\theta \in [0, 1]$, let U_θ denote the strategies of $\mathcal{S}$ with average cost greater than θ (this is measurable).
- Since measure we have
  - $\theta^* = \inf\{\theta : \mu(U_\theta) < 1 - \varepsilon\}$
  - $U = \bigcap_{\theta < \theta^*} U_\theta$
  - $V = U_{\theta^*} = \bigcup_{\theta > \theta^*} U_\theta$
- Since $V \subseteq U$ and $\mu(V) \leq 1 - \varepsilon \leq \mu(U)$, if we set $j = \lceil (1/\varepsilon) \log(2/\delta) \rceil$, and let $\mathcal{E}$ denote the event that $\{x_1, \ldots, x_k j\} \subseteq V$, then for any $x \in U$ we find that

Outline Of Proof of Convergence

- Let $T \geq \tau^*(\varepsilon, \delta)$ and for $\theta \in [0, 1]$, let U_θ denote the strategies of $\mathcal{S}$ with average cost greater than θ (this is measurable).
- Since measure we have
  - $\theta^* = \inf\{\theta : \mu(U_\theta) < 1 - \varepsilon\}$
  - $U = \bigcap_{\theta < \theta^*} U_\theta$
  - $V = U_{\theta^*} = \bigcup_{\theta > \theta^*} U_\theta$
- Since $V \subseteq U$ and $\mu(V) \leq 1 - \varepsilon \leq \mu(U)$, if we set $j = \lceil (1/\varepsilon) \log(2/\delta) \rceil$, and let $\mathcal{E}$ denote the event that $\{x_1, \ldots, x_k j\} \subseteq V$, then for any $x \in U$ we find that

$$\mathbb{E}[\frac{1}{T} \sum_{[T]} c_t(x_t) - c - t(x)]$$

Outline Of Proof of Convergence

- Let $T \geq \tau^*(\varepsilon, \delta)$ and for $\theta \in [0,1]$, let U_θ denote the strategies of $\mathcal{S}$ with average cost greater than θ (this is measurable).
- Since measure we have
  - $\theta^* = \inf\{\theta : \mu(U_\theta) < 1 - \varepsilon\}$
  - $U = \bigcap_{\theta < \theta^*} U_\theta$
  - $V = U_{\theta^*} = \bigcup_{\theta > \theta^*} U_\theta$
- Since $V \subseteq U$ and $\mu(V) \leq 1 - \varepsilon \leq \mu(U)$, if we set $j = \lceil (1/\varepsilon) \log(2/\delta) \rceil$, and let $\mathcal{E}$ denote the event that $\{x_1, \ldots, x_k j\} \subseteq V$, then for any $x \in U$ we find that

$$\begin{aligned}
\mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x_t) - c - t(x)] &= \mathbb{P}[\mathcal{E}]\mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x_t) - c_t(x)||\mathcal{E}] \\
&+ (1 - \mathbb{P}[\mathcal{E}])\mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x_t) - c_t(x)||\bar{\mathcal{E}}]
\end{aligned}$$

Outline Of Proof of Convergence

- Let $T \geq \tau^*(\varepsilon, \delta)$ and for $\theta \in [0, 1]$, let U_θ denote the strategies of $\mathcal{S}$ with average cost greater than θ (this is measurable).
- Since measure we have
  - $\theta^* = \inf\{\theta : \mu(U_\theta) < 1 - \varepsilon\}$
  - $U = \bigcap_{\theta < \theta^*} U_\theta$
  - $V = U_{\theta^*} = \bigcup_{\theta > \theta^*} U_\theta$
- Since $V \subseteq U$ and $\mu(V) \leq 1 - \varepsilon \leq \mu(U)$, if we set $j = \lceil (1/\varepsilon) \log(2/\delta) \rceil$, and let $\mathcal{E}$ denote the event that $\{x_1, \ldots, x_k j\} \subseteq V$, then for any $x \in U$ we find that

$$
\begin{aligned}
\mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x_t) - c - t(x)] &= \mathbb{P}[\mathcal{E}]\mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x_t) - c_t(x)||\mathcal{E}] \\
&+ (1 - \mathbb{P}[\mathcal{E}])\mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x_t) - c_t(x)||\bar{\mathcal{E}}] \\
&\leq \mathbb{P}[\mathcal{E}] + \mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x_t) - c_t(x)||\bar{\mathcal{E}}]
\end{aligned}
$$

Outline Of Proof of Convergence

- Let $T \geq \tau^*(\varepsilon, \delta)$ and for $\theta \in [0,1]$, let U_θ denote the strategies of $\mathcal{S}$ with average cost greater than θ (this is measurable).
- Since measure we have
  - $\theta^* = \inf\{\theta : \mu(U_\theta) < 1 - \varepsilon\}$
  - $U = \bigcap_{\theta < \theta^*} U_\theta$
  - $V = U_{\theta^*} = \bigcup_{\theta > \theta^*} U_\theta$
- Since $V \subseteq U$ and $\mu(V) \leq 1 - \varepsilon \leq \mu(U)$, if we set $j = \lceil (1/\varepsilon) \log(2/\delta) \rceil$, and let $\mathcal{E}$ denote the event that $\{x_1, \ldots, x_k j\} \subseteq V$, then for any $x \in U$ we find that

$$\begin{aligned}
\mathbb{E}[\frac{1}{T} \sum_{[T]} c_t(x_t) - c - t(x)] &= \mathbb{P}[\mathcal{E}]\mathbb{E}[\frac{1}{T} \sum_{[T]} c_t(x_t) - c_t(x) || \mathcal{E}] \\
&+ (1 - \mathbb{P}[\mathcal{E}])\mathbb{E}[\frac{1}{T} \sum_{[T]} c_t(x_t) - c_t(x) || \bar{\mathcal{E}}] \\
&\leq \mathbb{P}[\mathcal{E}] + \mathbb{E}[\frac{1}{T} \sum_{[T]} c_t(x_t) - c_t(x) || \bar{\mathcal{E}}]
\end{aligned}$$

- To see that each term is less than $\delta/2$, we can show that

Outline of Proof of Convergence

- To see that each term is less than $\delta/2$, we can show that
  - $\mathbb{P}[\mathcal{E}] \leq (1-\varepsilon)^j < e^{-j\varepsilon} \leq \delta/2$

Outline of Proof of Convergence

- To see that each term is less than $\delta/2$, we can show that
  - $\mathbb{P}[\mathcal{E}] \leq (1-\varepsilon)^j < e^{-j\varepsilon} \leq \delta/2$
  - The definition of $\tau(\varepsilon, \delta)$ implies that
    - $\mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x_t)||x_1, x_2, \ldots, x_j] <$

      $\frac{\delta}{2} + \min_i \mathbb{E}[\frac{1}{T}\sum_{i=1}^{T} c_t(x_i)||x_1, \ldots, x_j]$

Outline of Proof of Convergence

- To see that each term is less than $\delta/2$, we can show that
  - $\mathbb{P}[\mathcal{E}] \leq (1-\varepsilon)^j < e^{-j\varepsilon} \leq \delta/2$
  - The definition of $\tau(\varepsilon, \delta)$ implies that
    - $\mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x_t) || x_1, x_2, \ldots, x_j] <$
      $\frac{\delta}{2} + \min_i \mathbb{E}[\frac{1}{T}\sum_{i=1}^{T} c_t(x_i) || x_1, \ldots, x_j]$
    - $\mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x_i) || x_1, \ldots x_j] = \mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x_i)]$ (if the adversary is oblivious)

Outline of Proof of Convergence

- To see that each term is less than $\delta/2$, we can show that
  - $\mathbb{P}[\mathcal{E}] \leq (1-\varepsilon)^j < e^{-j\varepsilon} \leq \delta/2$
  - The definition of $\tau(\varepsilon, \delta)$ implies that
    - $\mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x_t)||x_1, x_2, \ldots, x_j] <$ $\frac{\delta}{2} + \min_i \mathbb{E}[\frac{1}{T}\sum_{i=1}^{T} c_t(x_i)||x_1, \ldots, x_j]$
    - $\mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x_i)||x_1, \ldots x_j] = \mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x_i)]$ (if the adversary is oblivious)
    - $\mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x_t)||\bar{\mathcal{E}}] < \frac{\delta}{2} + \mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x)||\bar{\mathcal{E}}]$

Outline of Proof of Convergence

- To see that each term is less than $\delta/2$, we can show that
  - $\mathbb{P}[\mathcal{E}] \leq (1-\varepsilon)^j < e^{-j\varepsilon} \leq \delta/2$
  - The definition of $\tau(\varepsilon, \delta)$ implies that
    - $\mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x_t)||x_1, x_2, \ldots, x_j] <$
      $\frac{\delta}{2} + \min_i \mathbb{E}[\frac{1}{T}\sum_{i=1}^{T} c_t(x_i)||x_1, \ldots, x_j]$
    - $\mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x_i)||x_1, \ldots x_j] = \mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x_i)]$ (if the adversary is oblivious)
    - $\mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x_t)||\bar{\mathcal{E}}] < \frac{\delta}{2} + \mathbb{E}[\frac{1}{T}\sum_{[T]} c_t(x)||\bar{\mathcal{E}}]$
- Moreover, the inequality $\tau(j, \delta) \leq j poly(\log(1/\varepsilon), 1/\delta)$ implies that $\tau^*(\varepsilon, \delta) \leq \frac{1}{\varepsilon} poly(\log(1/\varepsilon), 1/\delta)$.

ATA Performance

- For $i \geq \lceil \log_2 j \rceil$, in the $\texttt{EXP3}$ subroutine from $t_0 = F(i)$ to $t_i - 1 := \min(T, F(i+1) - 1)$, the strategy x belongs to $[K] = [2^i]$

ATA Performance

- For $i \geq \lceil \log_2 j \rceil$, in the $\texttt{EXP3}$ subroutine from $t_0 = F(i)$ to $t_i - 1 := \min(T, F(i+1) - 1)$, the strategy x belongs to $[K] = [2^i]$
- The regret bound for $\texttt{EXP3}$ guarantees

$$\mathbf{E}\left[\sum_{t=t_0}^{t_1 - 1} c_t(x_t) - c_t(x)\right] = O(\sqrt{K \log(K)(t_1 - t_0)} = O(\sqrt{i 2^i T})$$

ATA Performance

- For $i \geq \lceil \log_2 j \rceil$, in the EXP3 subroutine from $t_0 = F(i)$ to $t_i - 1 := \min(T, F(i+1) - 1)$, the strategy x belongs to $[K] = [2^i]$
- The regret bound for EXP3 guarantees

$$\mathbf{E}\left[\sum_{t=t_0}^{t_1-1} c_t(x_t) - c_t(x)\right] = O(\sqrt{K \log(K)(t_1 - t_0)} = O(\sqrt{i 2^i T})$$

ATA Performance (Ctd)

$$\mathbb{E}\left[\sum_{t=1}^{T} c_t(x_t) - c_t(x)\right]$$

ATA Performance (Ctd)

$$\mathbb{E}\left[\sum_{t=1}^{T} c_t(x_t) - c_t(x)\right] = \sum_{i=1}^{k-1} \mathbb{E}\left[\sum_{t=F(i)}^{\min(T,F(i+1)-1)} c_t(x_t) - c_t(x)\right]$$

ATA Performance (Ctd)

$$\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{T} c_t(x_t) - c_t(x)\right] &= \sum_{i=1}^{k-1} \mathbb{E}\left[\sum_{t=F(i)}^{\min(T,F(i+1)-1)} c_t(x_t) - c_t(x)\right] \\
&\leq \sum_{i<\lceil \log_2 j\rceil} \sum_{t=F(i)}^{\min(T,F(i+1)-1)} 1 \\
+ \sum_{\lceil \log_2 j\rceil \leq i<k} & \mathbb{E}\left[\sum_{t=F(i)}^{\min(T,F(i+1)-1)} (c_t(x_t) - c_t(x))\right]
\end{aligned}$$

ATA Performance (Ctd)

$$
\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{T} c_t(x_t) - c_t(x)\right] &= \sum_{i=1}^{k-1} \mathbb{E}\left[\sum_{t=F(i)}^{\min(T,F(i+1)-1)} c_t(x_t) - c_t(x)\right] \\
&\leq \sum_{i<\lceil \log_2 j \rceil} \sum_{t=F(i)}^{\min(T,F(i+1)-1)} 1 \\
+ \sum_{\lceil \log_2 j \rceil \leq i < k} &\mathbb{E}\left[\sum_{t=F(i)}^{\min(T,F(i+1)-1)} (c_t(x_t) - c_t(x))\right] \\
&\leq F(\lceil \log_2 j \rceil) + \sum_{i=1}^{k-1} O(\log i 2^i T)
\end{aligned}
$$

ATA Performance (Ctd)

$$
\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{T} c_t(x_t) - c_t(x)\right] &= \sum_{i=1}^{k-1} \mathbb{E}\left[\sum_{t=F(i)}^{\min(T,F(i+1)-1)} c_t(x_t) - c_t(x)\right] \\
&\leq \sum_{i<\lceil \log_2 j \rceil} \sum_{t=F(i)}^{\min(T,F(i+1)-1)} 1 \\
+ \sum_{\lceil \log_2 j \rceil \leq i < k} &\mathbb{E}\left[\sum_{t=F(i)}^{\min(T,F(i+1)-1)} (c_t(x_t) - c_t(x))\right] \\
&\leq F(\lceil \log_2 j \rceil) + \sum_{i=1}^{k-1} O(\log i 2^i T) \\
&= F(\lceil \log_2 j \rceil) + O(\sqrt{k 2^k T})
\end{aligned}
$$

Questions?