A. Properties of the Scaling Function λ

We will prove and discuss some basic properties of λ that will provide intuition for the setting and also be used in the later proofs.

Lemma 4. *Let $\alpha \in \mathbb{R}_+$. If $\alpha \leq 1$, then $\lambda(\alpha m) \geq \alpha\lambda(m)$. If $\alpha \geq 1$, then $\lambda(\alpha m) \leq \alpha\lambda(m)$.*

Proof. By concavity,

$$\lambda(\alpha m) = \lambda(\alpha m + (1-\alpha)0) \geq \alpha\lambda(m)$$

To prove the second part, observe that:

$$\lambda(m) = \lambda\left(\frac{1}{\alpha}\alpha m\right) \geq \frac{1}{\alpha}\lambda(\alpha m) \implies \lambda(\alpha m) \leq \alpha\lambda(m)$$

The first part is applied above since $\alpha \geq 1$. □

If we divide both sides of each inequality by αm, then each side represents the average time per arm pull when the system executes αm and m arm pulls, respectively. Intuitively, the average time per arm pull should decrease as the number of parallel arm pulls increases.

Lemma 5. *For all $m_1, m_2 > 0$,*

$$\lambda\left(m_1 + m_2\right) \leq \lambda\left(m_1\right) + \lambda\left(m_2\right),$$

Proof. We can apply Inequality 2:

$$\frac{\lambda(m_1+m_2)-\lambda(m_1)}{m_2} \leq \frac{\lambda(m_2)-\lambda(0)}{m_2} \implies \lambda\left(m_1 + m_2\right) \leq \lambda\left(m_1\right) + \lambda\left(m_2\right)$$

□

This property implies that the fastest way to schedule m arm pulls is to run them all in parallel instead of executing m_1 and then $m - m_1$.

B. Proof of the Upper Bound in the Fixed Confidence Setting

We will upper bound the time complexity of Algorithm 1 and show that it outputs the top arm with probability at least $1-\delta$. Bounding the time complexity is challenging, as we must show that the algorithm does not waste too much time as it ramps up its parallelism during execution. If it takes too long to find an appropriate level of parallelism, then it may be significantly longer than $T^\star$. However, if it ramps up too fast, it may *overshoot* by unnecessarily pulling arms that should have been eliminated much sooner. For example, if arm i could safely be eliminated after 10 more pulls, but the algorithm pulls each surviving arm 1000 times in parallel in a round, then the algorithm has wasted time pulling arm i more than necessary. As the pulls are executed in parallel, any information is only acquired after they all finish (at the same time). This behavior can be costly if a particular round of Algorithm 1 pulls many arms that could be eliminated soon excessively. We will use an analysis that carefully defines these overshooting events and shows that the extremely costly events do not occur often, and the less costly ones cannot significantly impact the overall runtime.

B.1. Confidence Intervals

In this section, we will state some results from Jun et al. (2016) to bound the error probability. We will first define some notation. Let $\mathcal{E}_i(\delta) = \{\forall r \geq 1, L_i(r,\delta) \leq \mu_i \leq U_i(r,\delta)\}$, which is the event that the confidence bounds capture the true mean of arm i at all rounds of the algorithm. Let $\omega = \sqrt{\delta/(6n)}$. Define $\bar{N}_i := 1 + \left\lfloor 64\Delta_i^{-2}\log((2/\omega)\log_2(192\Delta_i^{-2}/\omega))\right\rfloor$. We will use the following results from Jun et al. (2016) to show that the confidence intervals in (5) trap the true means.

Lemma 6. *[Jun et al. (2016), Lemma 1] (non-asymptotic law of the iterated logarithm). Let $X_1, X_2, \ldots$ be i.i.d. zero-mean sub-Gaussian random variables with scale $\sigma > 0$; i.e. $\mathbb{E}\exp(\lambda X_i) \leq \exp\left(\frac{\lambda^2\sigma^2}{2}\right)$. Let $\omega \in (0, \sqrt{1/6})$. Then,*

$$\Pr\left[\forall \tau \geq 1, \left|\sum_{s=1}^{\tau} X_s\right| \leq 4\sigma\sqrt{\tau \log\log_2 2\tau/\omega}\right] \geq 1 - 6\omega^2$$

This lemma is a simplification of Lemma 1 from Jamieson et al. (2014). Observe that a random variable bounded $a.s.$ in $[0,1]$ is sub-Gaussian with scale $1/2$. By Lemma 6, $\Pr\left[\cap_{i=1}^n \mathcal{E}_i(\delta)\right] \geq 1-\delta$. Hereafter, we will assume $\cap_{i=1}^n \mathcal{E}_i(\delta)$ and show that the algorithm always outputs the top arm in this event and bound its runtime. Furthermore, we introduce the following result from Jun et al. (2016).

Lemma 7. *[Jun et al. (2016), Lemma 2] Assume $\cap_{i=1}^n \mathcal{E}_i(\delta)$. In Algorithm 2, let $N'(r) = \min_{i \in S_r} N_i(r)$. Then,*

$$\forall r, \forall i > 1, \left(N'(r) \geq \bar{N}_i \implies U_i(r,\delta) < \max_{j \in S_r} L_j(r,\delta)\right) \tag{8}$$

$$\forall r, i = 1, \left(N'(r) \geq \bar{N}_i \implies L_i(r,\delta) > \overset{(2)}{\max_{j \in S_r}} U_j(r,\delta)\right) \tag{9}$$

So, as long as $\mathbb{P}(\cap_{i=1}^n \mathcal{E}_i) \geq 1-\delta$, the algorithm will output the correct set of arms after each surviving arm has been pulled $\bar{N}_i$ times with probability at least $1-\delta$.

B.2. Proof of Theorem 1

In this proof, we will assume $\cap_{i=1}^n \mathcal{E}_i(\delta)$, which we know by Lemma 6 to have probability at least $1-\delta$.

Correctness: If an arm i is returned by the algorithm as the best arm, then the other surviving arms have true means less than μ_i, because $\cap_{i=1}^n \mathcal{E}_i(\delta)$ implies that the confidence bounds capture the true means, and the algorithm will only accept arm i if its lower confidence bound (LCB) is greater than the other arms' upper confidence bounds (UCB). The same argument follows for arms that are rejected. So, when $\cap_{i=1}^n \mathcal{E}_i(\delta)$ occurs, arms cannot be erroneously returned or rejected. So, the algorithm outputs the correct arm with probability at least $1-\delta$.

Time Complexity: Recall that the dynamic program $\mathcal{T}_2\left(\{\bar{N}_i\}_{i=2}^n\}\right)$ is the runtime of an algorithm that operates in stages to eliminate arms. This algorithm knows the arm gaps, but not the arm orderings. We will refer to this algorithm hereafter as the *oracle algorithm*. The oracle algorithm consists of $\widetilde{r} \leq n-1$ stages, and eliminates $\widetilde{n}_i$ arms in stage i, which is possible with probability at least $1-\delta$. Define $\bar{N}_{n+1} = 0$. Also define $\widetilde{n}_{i,\text{tot}} = \sum_{j=1}^{i} \widetilde{n}_j$ (number of arms eliminated by round i, with $\widetilde{n}_{0,\text{tot}} = 0$), $\Delta\widetilde{N}_i = \bar{N}_{n-\widetilde{n}_{i,\text{tot}}+1} - \bar{N}_{n-\widetilde{n}_{i-1,\text{tot}}+1}$ (pulls per surviving arm in oracle stage i), and $\widetilde{T}_i^\star = \lambda\left((n-\widetilde{n}_{i-1,\text{tot}})\Delta\widetilde{N}_i\right)$ (time for oracle stage i). Then,

$$\begin{aligned}
\widetilde{T}^\star &= \mathcal{T}_2\left(\{\bar{N}_i\}_{i=2}^n\}\right) \\
&= \sum_{i=1}^{\widetilde{r}} \lambda\left((n-\widetilde{n}_{i-1,\text{tot}})(\bar{N}_{n-\widetilde{n}_{i,\text{tot}}+1} - \bar{N}_{n-\widetilde{n}_{i-1,\text{tot}}+1})\right) \\
&= \sum_{i=1}^{\widetilde{r}} \lambda\left((n-\widetilde{n}_{i-1,\text{tot}})\Delta\widetilde{N}_i\right) \\
&= \sum_{i=1}^{\widetilde{r}} \widetilde{T}_i^\star
\end{aligned}$$

Now, assign each pull in oracle stage 1 a unique index from $\{1, \ldots, \Delta\widetilde{N}_1\}$. For each pull in oracle stage 2, assign a unique index from $\{\Delta\widetilde{N}_1 + 1, \ldots \Delta, \widetilde{N}_2\}$. Assign indices for the remaining pulls accordingly. Now, define the function $\widetilde{R}$ that outputs which oracle stage a pull index corresponds to. That is,

$$\widetilde{R}(q) = \begin{cases} \max_{i \in [\widetilde{r}]} i \text{ s.t. } q \leq \bar{N}_{n-\widetilde{n}_{i,\text{tot}}+1} & \text{if } q \leq \bar{N}_2 \\ \widetilde{r}+1 & \text{if } q > \bar{N}_2 \end{cases}$$

In general, symbols with a tilde are associated with the oracle algorithm in this proof.

Define stage 1 of Algorithm 1 (and not the oracle algorithm) to be the rounds of the algorithm up to and including the round that results in the first arm elimination. Define stage 2 to be from the next round until the next eliminations unless multiple arms were eliminated at the same time as the first elimination. If the second arm was eliminated at the same time as the first arm, define a dummy stage 2 that takes 0 rounds. Define additional dummy stages for each additional arm that was eliminated in the first elimination round. Define stage i similarly. Observe that there are always $n-1$ stages now, so during stage i, $n-i+1$ arms remain.

Let ℓ_i be the number of rounds in stage i and let $\bar{\ell}_i = \sum_{j=1}^{i} \ell_j$ (rounds in first j stages). Similarly, let $\bar{q}_j = \sum_{i=1}^{j} q_i$ (pulls per surviving arm after round j) and $\bar{t}_j = \sum_{i=1}^{j} t_i$ (sum of time allocations in first j rounds) where $t_i = \beta^{i-1} t_1$. Define $T_i = \sum_{j=\bar{\ell}_{i-1}+1}^{\bar{\ell}_i} \lambda(|S_j| q_j) \leq \sum_{j=\bar{\ell}_{i-1}+1}^{\bar{\ell}_i} t_j$ (time for stage i).

Let us discuss a few easily-verified facts about the algorithm before proceeding.

Fact 8. *Suppose Algorithm 1 takes l rounds. Then its runtime is at most $\frac{\beta^l}{\beta-1} t_1$.*

The above property suggests that adding an additional round to the algorithm will increase its runtime by a factor of β.

Fact 9. $\lambda(q_j|S_j|) = \lambda\left(\left\lfloor \frac{\lambda^{-1}(t_j)}{|S_j|} \right\rfloor |S_j|\right) \leq t_j = \beta^{j-1} t_1 \leq \lambda(2q_j|S_j|) \leq 2\lambda(q_j|S_j|)$

The second to last inequality follows because $q_j \geq 1$ for all $j \geq 1$.

Stage i could fall into one of 5 cases. We will first handle 3 cases which are easier to analyze, and then turn to two more challenging cases.

Case 1: Suppose $\ell_i \geq 2$. Define $\widetilde{s}_0 = \widetilde{R}(\bar{q}_{\bar{\ell}_i - 2} + 1)$ and $\widetilde{s}_f = \widetilde{R}(\bar{q}_{\bar{\ell}_i - 1})$. This means that stage i took at least 2 rounds and the pull indices in its second to last round of stage i fall between the oracle algorithm's stage $\widetilde{s}_0$ and $\widetilde{s}_f$, respectively. Because the pull indices of round $\bar{\ell}_i - 1$ fall within the oracle algorithm's stages $\widetilde{s}_0, \ldots, \widetilde{s}_f$, the oracle algorithm must have more pulls per arm across these stages than round $\bar{\ell}_i - 1$ of Algorithm 1 does. That is,

$$q_{\bar{\ell}_i - 1} \leq \sum_{k=\widetilde{s}_0}^{\widetilde{s}_f} \Delta \widetilde{N}_k$$

This implies that:

$$\begin{aligned}(n-i+1) q_{\bar{\ell}_i - 1} &\leq \sum_{k=\widetilde{s}_0}^{\widetilde{s}_f} (n-i+1) \Delta \widetilde{N}_k \\ &\leq \sum_{k=\widetilde{s}_0}^{\widetilde{s}_f} (n - \widetilde{n}_{k-1,\mathrm{tot}}) \Delta \widetilde{N}_k \end{aligned} \tag{10}$$

The second inequality uses the fact that $\widetilde{n}_{\widetilde{s}_0 - 1,\mathrm{tot}} \leq \widetilde{n}_{\widetilde{s}_f - 1,\mathrm{tot}} \leq i-1$. Each surviving arm after round $\bar{q}_{\bar{\ell}_i - 1}$ of Algorithm 1 has been pulled at least $\bar{N}_{n - \widetilde{n}_{\widetilde{s}_f - 1} + 1}$ times (because $\widetilde{s}_f = \widetilde{R}(\bar{q}_{\bar{\ell}_i - 1})$), guaranteeing that at least $\widetilde{n}_{\widetilde{s}_f - 1}$ arms were eliminated after this round (Lemma 7).

Let us now use these facts to bound the runtime of the first i stages of Algorithm 1.

$$\begin{aligned}\sum_{k=1}^{i} T_k &\leq \sum_{r=1}^{\bar{\ell}_i} t_r \\ &= \sum_{r=1}^{\bar{\ell}_i} \beta^{r-1} t_1 \\ &\leq \frac{\beta^2}{\beta - 1} \beta^{\bar{\ell}_i - 2} t_{\bar{\ell}_1}\end{aligned}$$

$$
\begin{aligned}
&\leq 2\frac{\beta^2}{\beta-1}\lambda((n-i+1)q_{\bar{\ell}_i-1}) \\
&\leq 2\frac{\beta^2}{\beta-1}\lambda\left(\sum_{k=\widetilde{s}_0}^{\widetilde{s}_f}(n-\widetilde{n}_{k-1,\mathrm{tot}})\Delta\widetilde{N}_k\right) \\
&\leq 2\frac{\beta^2}{\beta-1}\sum_{k=\widetilde{s}_0}^{\widetilde{s}_f}\lambda((n-\widetilde{n}_{k-1,\mathrm{tot}})\Delta\widetilde{N}_k) \\
&= 2\frac{\beta^2}{\beta-1}\sum_{k=\widetilde{s}_0}^{\widetilde{s}_f}\widetilde{T}_k^\star
\end{aligned}
$$

The first inequality uses Fact 9. The second inequality uses Fact 8. The third inequality again uses Fact 9. The fourth inequality uses Inequality 10, and the fifth inequality applies Lemma 5. The last equality uses the definition of $\widetilde{T}_k^\star$.

Case 2: Suppose $\ell_i = 1$ and suppose that $\widetilde{R}(\bar{q}_{\bar{\ell}_{i-1}}+1) = \widetilde{R}(\bar{q}_{\bar{\ell}_i}) = \widetilde{s}$. That is, stage i takes a single round and has pull indices that are completely contained in oracle stage k. By almost the same argument as above:

$$q_{\bar{\ell}_i} \leq \Delta\widetilde{N}_{\widetilde{s}}$$

As before, this implies

$$
\begin{aligned}
(n-i+1)q_{\bar{\ell}_i} &\leq (n-i+1)\Delta\widetilde{N}_{\widetilde{s}} \\
&\leq (n-\widetilde{n}_{\widetilde{s}-1,\mathrm{tot}})\Delta\widetilde{N}_{\widetilde{s}} \qquad (11)
\end{aligned}
$$

The second inequality uses the fact that $\widetilde{n}_{\widetilde{s}-1,\mathrm{tot}} \leq i-1$. Each surviving arm after round $\bar{q}_{\bar{\ell}_1-1}$ of Algorithm 1 has been pulled at least $\bar{N}_{n-\widetilde{n}_{\widetilde{s}-1}+1}$ times (because $\widetilde{s} = \widetilde{R}(\bar{q}_{\bar{\ell}_i-1}+1)$), guaranteeing that at least $\widetilde{n}_{\widetilde{s}-1}$ arms were eliminated after this round (Lemma 7). Note that this implies that overshooting did not occur. Again, as before

$$
\begin{aligned}
\sum_{k=1}^{i}T_k &\leq \sum_{r=1}^{\bar{\ell}_i}t_r \\
&= \sum_{r=1}^{\bar{\ell}_i}\beta^{r-1}t_1 \\
&\leq \frac{\beta}{\beta-1}\beta^{\bar{\ell}_i-1}t_{\bar{\ell}_1} \\
&\leq 2\frac{\beta}{\beta-1}\lambda((n-i+1)q_{\bar{\ell}_i}) \\
&\leq 2\frac{\beta}{\beta-1}\lambda\left((n-\widetilde{n}_{k-1,\mathrm{tot}})\Delta\widetilde{N}_{\widetilde{s}}\right) \\
&= 2\frac{\beta}{\beta-1}\widetilde{T}_{\widetilde{s}}^\star
\end{aligned}
$$

This result is missing a factor of β compared to Case 1, since in the prior case we used the second to last round in stage i to achieve the bound on the first i stages, whereas here we can use the last (and only) round.

Case 3: Suppose $\ell_i = 0$. This means that the i-th elimination occurred in the same round as the $i-1$-th elimination. The time taken for these stages is 0.

Discussion of Cases 1-3: Let stage i_f be the last stage of Algorithm 1 that falls into either cases 1 or 2 above. Then, stages $1,\ldots,i_f$ take at most $2\frac{\beta^2}{\beta-1}\widetilde{T}^\star$ time. The remaining stages all consist of a single round, and they may potentially pull arms

excessively due to overshooting (excessively pulling arms that would have been eliminated after a few pulls in the round). We must now argue that overshooting does not result in a significant increase in runtime. Observe that naively bounding the up to $n-1$ remaining stages by a factor of β^{n-1} results in a very expensive runtime. However, we will use a more aggressive analysis by defining events where excessive overshooting occurs, and bounding the number of these events.

We will discuss 2 cases for the remaining stages now. If some stage $i \in [i_f]$ falls in cases 4 and 5, we already have a runtime bound for the first i_f cases and therefore do not need to bound the runtime of these stages.

Let $f = \beta^{-\sqrt{\log_\beta(n)}}$. Ignore the very last stage for now, which can only add a factor of β to the total runtime. Each of the remaining stages fall into one of two cases:

Case 4: Define $\widetilde{s}_0 = \widetilde{R}(\bar{q}_{\bar{\ell}_{i-1}} + 1)$ and $\widetilde{s}_f = \widetilde{R}(\bar{q}_{\bar{\ell}_i})$. Suppose stage i eliminates at least $1-f$ fraction of the surviving arms, $\ell_i = 1$, and $\widetilde{s}_0 < \widetilde{s}_f$. That is, stage i of Algorithm 1 takes a single round, $S_{\bar{\ell}_i+1} \le f S_{\bar{\ell}_i}$, and has pull indices that span multiple oracle stages. This can occur at most $\left\lceil \sqrt{\log_\beta(n)} \right\rceil \le 2\sqrt{\log_\beta(n)}$ times, because $\beta \le n$. Each one of these occurrences adds a factor of β to the runtime of the stages prior to it. The worst increase occurs if this case happens at the very last stages. So, even if this case occurred the maximum number of times after all other stages, this would add a factor of $\beta^{2\sqrt{\log_\beta(n)}}$ to the runtime by Fact 8.

Case 5: Define $\widetilde{s}_0 = \widetilde{R}(\bar{q}_{\bar{\ell}_{i-1}} + 1)$ and $\widetilde{s}_f = \widetilde{R}(\bar{q}_{\bar{\ell}_i})$. Suppose stage i eliminates less than $1-f$ fraction of the arms, $\ell_i = 1$ and $\widetilde{s}_0 < \widetilde{s}_f$. That is, stage i of Algorithm 1 takes a single round, $S_{\bar{\ell}_i+1} > f S_{\bar{\ell}_i}$, and has pull indices that span multiple oracle stages (which could mean overshooting). This can occur up to $n-1$ times, and at the end of stage i, at least $f(n-i+1) \le (n - \widetilde{n}_{y-1,\mathrm{tot}}) \le (n - \widetilde{n}_{x-1,\mathrm{tot}})$ arms remain (Lemma 7). Therefore,

$$(n-i+1)q_{\bar{\ell}_i} \le \sum_{k=x}^{y} (n-i+1)\Delta\widetilde{N}_k \tag{12}$$

We can use this fact as before to show the following:

$$\begin{aligned}
T_i &= \lambda((n-i+1)q_{\bar{\ell}_i}) \\
&\le \sum_{k=\widetilde{s}_0}^{\widetilde{s}_f} \lambda((n-i+1)\Delta\widetilde{N}_k) \\
&\le \sum_{k=\widetilde{s}_0}^{\widetilde{s}_f} \lambda\left(\frac{1}{f}(n - \widetilde{n}_{k-1,\mathrm{tot}})\Delta\widetilde{N}_k\right) \\
&\le \frac{1}{f}\sum_{k=\widetilde{s}_0}^{\widetilde{s}_f} \lambda\left((n - \widetilde{n}_{k-1,\mathrm{tot}})\Delta\widetilde{N}_k\right) \\
&\le \frac{1}{f}\sum_{k=\widetilde{s}_0}^{\widetilde{s}_f} \widetilde{T}^\star_k
\end{aligned}$$

The first inequality follows by applying Inequality 12 followed by Lemma 5. The third inequality applies Lemma 4, since $\beta \le n \implies 1/f \ge 1$.

While this case can happen up to $n-1$ times, each stage in the oracle algorithm can only be covered by at most 2 such stages from Algorithm 1, because each stage in this case has pull indices intersecting with at least 2 oracle stages. So, even if all stages fell in this case, this would add $\frac{2}{f}\widetilde{T}^\star = 2\beta^{2\sqrt{\log_\beta(n)}}\widetilde{T}^\star$ to the runtime.

Putting together the pieces: The runtime up to stage i_f is at most $2\frac{\beta^2}{\beta-1}\widetilde{T}^\star$. Case 4 can add up to a factor of $\beta^{2\sqrt{\log_\beta(n)}}$ to the runtime and case 5 can add up to $2\beta^{2\sqrt{\log_\beta(n)}}\widetilde{T}^\star$ to the runtime. It is possible that cases 4 and 5 will interchange. Since $\frac{1}{f} \ge 1$ and $\beta \ge 1$, the sequence with the worst runtime has Case 5 occurring maximally before Case 4 occurs. The final

stage, which we ignored earlier, can add an additional factor of β. So,

$$\begin{aligned}
T &\leq 2\left(\frac{\beta^2}{\beta-1}+2\beta^{2\sqrt{\log_\beta(n)}}\right)\beta^{2\sqrt{\log_\beta(n)}}\beta\widetilde{T}^\star \\
&\leq 2\left(\frac{\beta^2}{\beta-1}+2\right)\beta^{4\sqrt{\log_\beta(n)}}\beta\widetilde{T}^\star \\
&\leq 4\frac{\beta^3}{\beta-1}\beta^{4\sqrt{\log_\beta(n)}}\widetilde{T}^\star
\end{aligned}$$

This completes the proof of the first claim of the theorem. Now, let us relate $\widetilde{T}^\star$ to $T^\star$ whose proof is given below.

Lemma 10. $\widetilde{T}^\star = \mathcal{T}_2\left(\{\bar{N}_i\}_{i=2}^n\}\right) \leq (128\log((2/\omega)\log_2(192\Delta_2^{-2}/\omega)\vee 2))T^\star$

By applying Lemma 10, we have that

$$\begin{aligned}
T &\leq 4\frac{\beta^3}{\beta-1}\beta^{4\sqrt{\log_\beta(n)}}\widetilde{T}^\star \\
&\leq 4\frac{\beta^{3+4\sqrt{\log_\beta(n)}}}{\beta-1}(128\log((2/\omega)\log_2(192\Delta_2^{-2}/\omega))\vee 2)T^\star \\
&\leq 512\frac{\beta^{3+4\sqrt{\log_\beta(n)}}}{\beta-1}\left(\log((2/\omega)\log_2(192\Delta_2^{-2}/\omega))\vee 1/64\right)T^\star
\end{aligned}$$

□

Proof of Lemma 10:

In this proof alone, we will abuse notation by letting $\widetilde{r}$ and $\widetilde{n}_i$ be a feasible point for the dynamic program minimization problem and not an optimal solution that achieves $\widetilde{T}^\star$ runtime. By first applying the definition of $\widetilde{T}^\star$, we have

$$\begin{aligned}
\widetilde{T}^\star &= \min_{\widetilde{r},\widetilde{n}_i}\sum_{i=1}^{\widetilde{r}}\lambda\left((n-\widetilde{n}_{i-1,\text{tot}})(\bar{N}_{n-\widetilde{n}_{i,\text{tot}}+1}-\bar{N}_{n-\widetilde{n}_{i-1,\text{tot}}+1})\right) \\
&= \min_{\widetilde{r},\widetilde{n}_i}\lambda\left(n(\bar{N}_{n-\widetilde{n}_{1,\text{tot}}+1})\right)+\sum_{i=2}^{\widetilde{r}}\lambda\left((n-\widetilde{n}_{i-1,\text{tot}})(\bar{N}_{n-\widetilde{n}_{i,\text{tot}}+1}-\bar{N}_{n-\widetilde{n}_{i-1,\text{tot}}+1})\right) \\
&= \min_{\widetilde{r},\widetilde{n}_i}\lambda\left(n\left(1+64\Delta^{-2}_{n-\widetilde{n}_{1,\text{tot}}+1}\log((2/\omega)\log_2(192\Delta^{-2}_{n-\widetilde{n}_{1,\text{tot}}+1}/\omega))\right)\right) \\
&+\sum_{i=2}^{\widetilde{r}}\lambda((n-\widetilde{n}_{i-1,\text{tot}})(64\Delta^{-2}_{n-\widetilde{n}_{i,\text{tot}}+1}\log((2/\omega)\log_2(192\Delta^{-2}_{n-\widetilde{n}_{i,\text{tot}}+1}/\omega)) \\
&-64\Delta^{-2}_{n-\widetilde{n}_{i-1,\text{tot}}+1}\log((2/\omega)\log_2(192\Delta^{-2}_{n-\widetilde{n}_{i-1,\text{tot}}+1}/\omega)))) \\
&\leq \lambda(n)+\min_{\widetilde{r},\widetilde{n}_i}\lambda\left(64n\Delta^{-2}_{n-\widetilde{n}_{1,\text{tot}}+1}\log((2/\omega)\log_2(192\Delta_2^{-2}/\omega))\right) \\
&+\sum_{i=2}^{\widetilde{r}}\lambda(64\log((2/\omega)\log_2(192\Delta_2^{-2}/\omega))(n-\widetilde{n}_{i-1,\text{tot}})(\Delta^{-2}_{n-\widetilde{n}_{i,\text{tot}}+1}-\Delta^{-2}_{n-\widetilde{n}_{i-1,\text{tot}}+1})) \\
&= \lambda(n)+\min_{\widetilde{r},\widetilde{n}_i}\sum_{i=1}^{\widetilde{r}}\lambda(64\log((2/\omega)\log_2(192\Delta_2^{-2}/\omega))(n-\widetilde{n}_{i-1,\text{tot}})(\Delta^{-2}_{n-\widetilde{n}_{i,\text{tot}}+1}-\Delta^{-2}_{n-\widetilde{n}_{i-1,\text{tot}}+1})) \\
&\leq \lambda(n)+(64\log((2/\omega)\log_2(192\Delta_2^{-2}/\omega))\vee 1)\min_{\widetilde{r},\widetilde{n}_i}\sum_{i=1}^{\widetilde{r}}\lambda((n-\widetilde{n}_{i-1,\text{tot}})(\Delta^{-2}_{n-\widetilde{n}_{i,\text{tot}}+1}-\Delta^{-2}_{n-\widetilde{n}_{i-1,\text{tot}}+1})) \\
&= \lambda(n)+(64\log((2/\omega)\log_2(192\Delta_2^{-2}/\omega))\vee 1)T^\star
\end{aligned}$$

$$\leq (1 + (64 \log((2/\omega) \log_2(192\Delta_2^{-2}/\omega)) \vee 1))T^\star$$
$$\leq (128 \log((2/\omega) \log_2(192\Delta_2^{-2}/\omega)) \vee 2)T^\star$$

The second equation uses $\bar{N}_{n+1} = 0$ and $\widetilde{n}_0 = 0$. The third equation plugs in $\bar{N}_i$. The fourth equation applies Lemma 5 to the first term before pulling it out of the optimization problem and uses $\Delta_2 \leq \Delta_i$ in the remaining terms. The fifth equation rearranges terms by placing the second term back in the sum, using $\widetilde{n}_0 = 0$. The sixth equation uses Lemma 4. The seventh equation uses the definition of $T^\star$. The eighth equation uses the fact that each arm must be pulled at least once before elimination, so $\lambda(n) \leq T^\star$. The final equation bounds the sum by a factor of 2. □

Fact 11. *The factor $\beta^{4\sqrt{\log_\beta(n)}}$ grows slower than n^α for any $\alpha > 0$. To view this, observe that*

$$\begin{aligned}
\lim_{n\to\infty} \frac{\beta^{4\sqrt{\log_\beta(n)}}}{n^\alpha} &= \beta^{\lim_{n\to\infty} \log_\beta\left(\frac{\beta^{4\sqrt{\log_\beta(n)}}}{n^\alpha}\right)} \\
&= \beta^{\lim_{n\to\infty} 4\sqrt{\log_\beta(n)} - \alpha \log_\beta(n)} \\
&= \beta^{\lim_{n\to\infty} \log_\beta(n)\left(\frac{4}{\sqrt{\log_\beta(n)}} - \alpha\right)} \\
&= \beta^{-\infty} \\
&= 0
\end{aligned}$$

By a similar proof, it can be shown that the function grows faster than any polylogarithmic function.

Remark 1. *APR can be easily modified for top-k arm identification as in (Jun et al., 2016) by eliminating arms when their UCB is below the k-th highest LCB. Arms are additionally thrown into an acceptance set (and not sampled anymore) if their LCB is higher than the $k+1$-th highest UCB. When k arms have been accepted, the algorithm terminates. By redefining the Δ_i values in terms of the means of the k-th and $k+1$-th best arms, we can obtain an identical runtime bound with the same error probability.*

C. Proof of the Lower Bound in the Fixed Confidence Setting

In this section, we will prove the lower bound (Theorem 2) in the fixed confidence setting. Throughout this section, we will assume, without loss of generality, that the samples for each arm $k \in [n]$ are generated upfront and each time we pull an arm, these observations are revealed in order.

Our first result below shows that it is sufficient to consider algorithms which do not wait for some time before executing arm pulls (thereby resulting in idle resources), unless it waits for some arm pulls to complete at some point in the future (possibly to incorporate that information before planning future pulls). For this, recall from Section 3.3 that $\mathcal{A}_{\delta,\lambda}$ is the class of algorithms that which can identify the best arm with probability at least $1-\delta$ for the given scaling function λ on any set of distributions $\theta \in \Theta$. Let $\widetilde{\mathcal{A}}_{\delta,\lambda}$ denote the subset of algorithms in $\mathcal{A}_{\delta,\lambda}$, where the algorithm starts a new set of arm pulls at time 0 or precisely at the same time when a previous set of pulls are completed; i.e for an algorithm in $\widetilde{\mathcal{A}}_{\delta,\lambda}$, if $t_{s1} \leq t_{s2} \leq \ldots$ denote the times at which new arm pulls are started, and if $0 < t_{e1} \leq t_{e2} \leq \ldots$ denote the times at which previous pulls ended, then, $t_{s1} = 0$ and for all $i \geq 2$, $t_{si} = t_{ej}$ for some t_{ej}. We have the following result.

Lemma 12. *Let $A \in \mathcal{A}_{\delta,\lambda}$. Then, there exists $\widetilde{A} \in \widetilde{\mathcal{A}}_{\delta,\lambda}$ such that $\widetilde{A}$ outputs the same arm as A and $T(\widetilde{A},\theta) \leq T(A,\theta)$ a.s.*

Proof. For the given $A \in \mathcal{A}_{\delta,\lambda}$, let t_1' denote the first time instant at which some arm pulls are started in A and when a previous set of arm pulls were not completed. Let A_1 be the algorithm that shifts the arm pulls at t_1' forward to t_1'' defined as follows,

$$t_1'' = \max\{t \leq t_1'; \text{some pulls were completed at time } t \text{ in } A\}.$$

As A has the same information at t_1'' as it has at t_1', and since the resources used for the pulls at t_1' are also available at t_1'', A_1 can execute the same decisions, and moreover by the assumptions at the beginning of this section, it will observe the same

rewards. Hence, it will output the same arm as A. Moreover, since none of the arm pulls in A_1 are started later than in A, it will not complete later than A.

The above transformation of the algorithm A, achieves the same output as A and finishes no later than A. We can now keep repeating this transformation. For example, t_2' can denote the first time instant in which some arm pulls are started in A_1 and when a previous set of arm pulls were not completed. Let A_2 be the resulting algorithm after shifting it forward as described above. Since there are at most finitely many arm pulls in A, the above procedure should terminate in some algorithm $\widetilde{A}$, which achieves the same output as A and finishes no later than A. Moreover, this algorithm, by definition, lies in $\mathcal{A}_{\delta,\lambda}$. □

Lemma 12 establishes that the algorithm which achieves the lowest expected time is in $\widetilde{\mathcal{A}}_{\delta,\lambda}$, and therefore, we can restrict our attention to this subclass. For such an algorithm, we can index the times at which decisions were made by $r \in \mathbb{N}$ and let $\{t_r\}_{r\in\mathbb{N}}$ denote the corresponding times so that $0 = t_0 < t_1 < t_2 < \dots$. For $r \geq 0$, let a_r denote the action taken by the algorithm in round r, where each action is a multiset of (i, α) pairs for each arm pull started by the algorithm at time t_r; here, $i \in [n]$ is an arm that was chosen for evaluation and $\alpha \in [0, 1]$ is the fraction of the resource assigned to that arm pull. Similarly, for $r \geq 1$, let O_r denote the observations received at time t_r, where O_r is a multiset of (i, Y) tuples with $i \in [n]$ being an arm that was chosen for evaluation at some previous time and Y is the observed reward from this pull. An algorithm $A \in \widetilde{\mathcal{A}}_{\delta,\lambda}$ can be completely characterized by the action rule, the stopping rule, and the recommendation rule, defined below.

1. The *action rule* is a method to select new arms to evaluate and the amount of resources to assign to each of them. a_r is $\mathcal{F}_r$ measurable, where $\mathcal{F}_r = \sigma(\{O_{r'}\}_{r'=1}^{r})$ is the sigma-algebra generated by past observations.
2. The *stopping rule* ρ, is when the algorithm decides to stop executing more arm pulls. It is a stopping time with respect to $\{\mathcal{F}_r\}_{r\in\mathbb{N}}$ satisfying $\mathbb{P}(\rho < \infty) = \mathbb{P}(t_\rho < \infty) = 1$.
3. The *recommendation rule* is the arm in $[n]$ chosen by the algorithm as the best arm. It is a $\mathcal{F}_\rho$-measurable.

Next, recall, our notation from Section 3.3, where parenthesized subscripts $\theta_{(k)}$ index the arms while subscripts without parentheses θ_k are the distributions ordered in decreasing order of mean value. Similarly let $\mu_{(k)}, \mu_k$ denote the means of $\theta_{(k)}, \theta_k$ respectively and recall that $\mu_1 < \mu_2 \leq \mu_3 \cdots \leq \mu_n$. For $r \in \mathbb{N}$ and $k \in [n]$, we will let $N_{(k)}(r)$ denote the number of completed arm pulls for arm (k) from time 0 to time t_r. Let $N(r) = \sum_{k\in[n]} N_{(k)}(r)$ denote the number of completed arm pulls of all the arms from time 0 to time t_r. Finally, we may define the gaps for a set of n distributions $\theta \in \Theta$ as follows. This is consistent with our previous definition (1).

$$\Delta_{(k)} = \mu_1 - \mu_2 \quad \text{if } \theta_{(k)} = \theta_1, \qquad\qquad \Delta_{(k)} = \mu_1 - \mu_{(k)} \quad \text{otherwise.}$$

Our proof of the lower bound first establishes lower bounds on the sample complexity for BAI and then translates these lower bounds to a lower bound on the runtime for an algorithm in our setting. We will prove this hardness result for problems in Θ with Gaussian reward distributions, although it is straightforward to generalise it to distributions for which the KL divergence satisfies $\mathrm{KL}(\nu\|\nu') \asymp (\mu - \mu')^2$ where μ, μ' are respectively the means of ν, ν'. The following result is an adaptation of a well-known hardness result for BAI to algorithms in in $\widetilde{\mathcal{A}}_{\delta,\lambda}$. We have given its proof in Appendix C.2.

Lemma 13. *Let $\theta \in \Theta$ such that the reward distribution $\theta_{(k)}$ is Gaussian with unit variance for all $k \in [n]$. Then, any $A \in \widetilde{\mathcal{A}}_{\delta,\lambda}$ satisfies,*

$$\mathbb{E}[N_{(k)}(\rho)] \geq \frac{2}{\Delta_{(k)}^2} \log\left(\frac{1}{2.4\delta}\right).$$

We will prove the lower bound for scaling functions λ which satisfy, for some $\alpha_1, \alpha_2 > 0$,

$$\alpha_1 m \leq \lambda(m) \leq \alpha_2 m \qquad \text{for all } m \geq 1. \tag{13}$$

If $\lambda(n) = \beta_1 m + \beta_2 \lambda'(m)$, where $\lambda'(m) \in o(m)$ and is concave with $\lambda'(0) = 0$, then λ satisfies all the properties in Section 3 and satisfies the above conditions for $\alpha_1 = \beta_1$ and $\alpha_2 = \beta_1 + \beta_2\lambda'(1)$. Note that since we are assuming λ is increasing and concave, the upper bound in (13) is already implied by Lemma 4; in particular, $\lambda(m) \leq \lambda(1)m$ for all $m \geq 1$. While the $\lambda(m) \geq \alpha_1 m$ condition means that the proposed algorithm is optimal only for a limited class of scaling functions,

it is worth noting that it also captures a practically meaningful use case. To see this, note that the above conditions imply $\lim_{m\to\infty} \frac{\lambda(m)}{m} \geq \alpha_1$. This states that even when the resource is used most efficiently as possible (i.e. at the maximum possible throughput), a minimum amount of 'work' needs to be done to execute one arm pull.

We shall now prove the lower bound for scaling functions which satisfy (13) and for reward distributions which satisfy the conditions in Lemma 13.

C.1. Proof of Theorem 2

In this proof, without loss of generality, we will assume that the arms are ordered so that $\mu_{(1)} > \mu_{(2)} \geq \mu_{(3)} \geq \dots \mu_{(n)}$. Let $A \in \widetilde{\mathcal{A}}_{\delta,\lambda}$ be arbitrary and let ρ denote the round at which the algorithm A decided to stop and recommend an arm. The minimum time taken to execute these arm pulls can be upper bound as follows.

$$T(A,\theta) \geq \lambda\left(\sum_{k=1}^{n} N_{(k)}(\rho)\right) \geq \alpha_1\left(\sum_{k=1}^{n} N_{(k)}(\rho)\right).$$

Here, the first step simply assumes that all arms were pulled exactly $N_{(k)}(\rho)$ times in the first set of pulls and the second uses (13). Now define $\widetilde{N}_{(k)}(r)$ recursively as follows for all $r \in \mathbb{N}$,

$$\widetilde{N}_{(1)}(r) = N_{(1)}(r), \qquad \widetilde{N}_{(k)}(r) = \min\left(\widetilde{N}_{(k-1)}(r), N_{(k)}(r)\right) \text{ for } k \geq 2.$$

Next, denote $\mathcal{K}(A) = \{2\} \cup \{3 \leq k \leq n; \widetilde{N}_{(k)}(\rho) = N_{(k)}(\rho)\} \cup \{n+1\}$. Write $\mathcal{K}(A) = \{k_1, k_2, \dots, k_{\tilde{p}}, k_{\tilde{p}+1}\}$ so that the k_i's are in ascending order, i.e. $2 = k_1 < k_2 < \dots < k_{\tilde{p}} < k_{\tilde{p}+1} = n+1$. Observe that, by definition, we have $N_{(k)}(\rho) \geq N_{(k_i)}(\rho)$ for $k = k_i, \dots, k_{i+1} - 1$. We can now upper bound $\mathbb{E}[T(A,\theta)]$ as follows.

$$\begin{aligned}
\mathbb{E}[T(A,\theta)] &\geq \alpha_1\left(\mathbb{E}N_{(1)}(\rho) + \sum_{k=k_1}^{k_2-1}\mathbb{E}N_{(k)}(\rho) + \sum_{k=k_2}^{k_3-1}\mathbb{E}N_{(k)}(\rho) + \dots \sum_{k=k_{\tilde{p}}}^{k_{\tilde{p}+1}-1}\mathbb{E}N_{(k)}(\rho)\right) \\
&\geq \alpha_1\left(\mathbb{E}N_{(1)}(\rho) + (k_2-k_1)\mathbb{E}N_{(2)}(\rho) + (k_3-k_2)\mathbb{E}N_{(k_2)}(\rho) + \dots + (k_{\tilde{p}+1}-k_{\tilde{p}})\mathbb{E}N_{(k_{\tilde{p}})}(\rho)\right) \\
&\geq 2\alpha_1 \log\left(\frac{1}{2.4\delta}\right)\left((k_2-1)\frac{1}{\Delta_2^2} + (k_3-k_2)\frac{1}{\Delta_{k_2}^2} + (k_4-k_3)\frac{1}{\Delta_{k_3}^2} + \dots (k_{\tilde{p}+1}-k_{\tilde{p}})\frac{1}{\Delta_{k_{\tilde{p}}}^2}\right) \qquad (14) \\
&\geq 2\alpha_1 \log\left(\frac{1}{2.4\delta}\right)\Bigg((k_2-1)\left(\frac{1}{\Delta_2^2} - \frac{1}{\Delta_{k_2}^2}\right) + (k_3-1)\left(\frac{1}{\Delta_{k_2}^2} - \frac{1}{\Delta_{k_3}^2}\right) + \dots \\
&\qquad\qquad + (k_{\tilde{p}}-1)\left(\frac{1}{\Delta_{k_{\tilde{p}-1}}^2} - \frac{1}{\Delta_{k_{\tilde{p}}}^2}\right) + n\frac{1}{\Delta_{k_{\tilde{p}}}^2}\Bigg) \\
&\geq \frac{2\alpha_1}{\alpha_2}\log\left(\frac{1}{2.4\delta}\right)\Bigg(\lambda\left((k_2-1)\left(\frac{1}{\Delta_2^2} - \frac{1}{\Delta_{k_2}^2}\right)\right) + \lambda\left((k_3-1)\left(\frac{1}{\Delta_{k_2}^2} - \frac{1}{\Delta_{k_3}^2}\right)\right) + \dots \\
&\qquad\qquad + \lambda\left((k_{\tilde{p}}-1)\left(\frac{1}{\Delta_{k_{\tilde{p}-1}}^2} - \frac{1}{\Delta_{k_{\tilde{p}}}^2}\right)\right) + \lambda\left(n\frac{1}{\Delta_{k_{\tilde{p}}}^2}\right)\Bigg)
\end{aligned}$$

Here, the second step uses the definition of $\mathcal{K}(A) = \{k_1, k_2, \dots, k_{\tilde{p}}\}$. In the third step, first we have used Lemma 13, while observing that $\Delta_1 = \Delta_2$; then, we have used the fact that $k_1 = 2$, and therefore $k_2 - k_1 + 1 = k_2 - 1$. The third step can be obtained via the following series of manipulations. First observe that we can write

$$(k_2-1)\frac{1}{\Delta_2^2} + (k_3-k_2)\frac{1}{\Delta_{k_2}^2} = (k_2-1)\left(\frac{1}{\Delta_2^2} - \frac{1}{\Delta_{k_2}^2}\right) + (k_3-1)\frac{1}{\Delta_{k_2}^2}.$$

We then apply the same manipulation to $(k_3-1)\frac{1}{\Delta_{k_2}^2}$ and the next term in the summation in (14). Proceeding in this fashion, the coefficient for the last term will be $k_{\tilde{p}+1} - k_{\tilde{p}} + (k_{\tilde{p}} - 1) = (n+1) - 1 = n$. Finally, the last step uses (13).

The above bound is a lower bound on the run time of algorithm A in terms of $\mathcal{K}(A)$. To lower bound the run time of any algorithm, it is sufficient to lower bound the above expression over all values of $(k_2, \dots, k_{\tilde{p}})$, where $\tilde{p} \leq n-1$ and

$3 \le k_2 < \ldots k_{\tilde{p}} \le n$. Writing $\alpha_1/\alpha_2 = c_\lambda$, we have,

$$\inf_{A \in \widetilde{\mathcal{A}}_{\delta,\lambda}} \mathbb{E}[T(A,\theta)] \ge 2c_\lambda \log\left(\frac{1}{2.4\delta}\right) \min_{(k_2,\ldots,k_{\tilde{p}})} \left(\lambda\left((k_2-1)\left(\frac{1}{\Delta_2^2} - \frac{1}{\Delta_{k_2}^2}\right)\right) + \lambda\left((k_3-1)\left(\frac{1}{\Delta_{k_2}^2} - \frac{1}{\Delta_{k_3}^2}\right)\right) + \right.$$
$$\left. \cdots + \lambda\left((k_{\tilde{p}}-1)\left(\frac{1}{\Delta_{k_{\tilde{p}-1}}^2} - \frac{1}{\Delta_{k_{\tilde{p}}}^2}\right)\right) + \lambda\left(n\frac{1}{\Delta_{k_{\tilde{p}}}^2}\right)\right)$$
$$= 2c_\lambda \log\left(\frac{1}{2.4\delta}\right) T^\star.$$

Here, we have observed that the expression in the RHS inside the minimum is simply the expansion of the dynamic program $\mathcal{T}_2\left(\{\Delta_i^{-2}\}_{i=2}^n\right)$, see (3) and (4). The proof is completed by Lemma 12, which implies that $\inf_{A \in \widetilde{\mathcal{A}}_{\delta,\lambda}} \mathbb{E}[T(A,\theta)] = \inf_{A \in \mathcal{A}_{\delta,\lambda}} \mathbb{E}[T(A,\theta)]$. □

C.2. Proof of Lemma 13

Our proof of Lemma 13 uses essentially the same argument as Kaufmann et al. (2016). However, due to the differences in our set up, we need to verify that the same proof carries through. For this, consider two sets of distributions $\theta, \theta' \in \Theta$ such that for all $k \in [n]$, $\theta_{(k)}$ and $\theta'_{(k)}$ are absolutely continuous with respect to each other. This means, for each $k \in [n]$, there exists a measure such that $\theta_{(k)}$ and $\theta'_{(k)}$ have densities f_k, f'_k with respect to this measure. Let $\{(K_s, Z_s)\}_{s=1}^N$ be a sequence of arm-observation pairs received in order when executing algorithm $A \in \widetilde{\mathcal{A}}_{\delta,\lambda}$. Define the log-likelihood ratio L_N after the first N observations as shown below. Let $L_r^\dagger$ denote the log likelihood ratio using the observations up to round r, i.e the first $N(r)$ observations. We have:

$$L_N = L_N\left(\{(K_s,Z_s)\}_{s=1}^N\right) = \sum_{k=1}^n \sum_{s=1}^N \mathbb{1}(K_s = k) \log\left(\frac{f_k(Z_s)}{f'_k(Z_s)}\right), \qquad L_r^\dagger = L_{N(r)}\left(\{(K_s,Z_s)\}_{s=1}^{N(r)}\right). \tag{15}$$

The following result, from Kaufmann et al. (2016) upper bounds the log likelihood ratio at the stopping time.

Lemma 14 (Lemma 19 (Kaufmann et al., 2016)). *Let $\theta, \theta' \in \Theta$ be sets of distributions such that $\theta_{(k)}$ and $\theta'_{(k)}$ are absolutely continuous with respect to each other for all $k \in [n]$. Let $A \in \widetilde{\mathcal{A}}_{\delta,\lambda}$ and ρ be any almost surely finite stopping time with respect to $\{\mathcal{F}_r\}_{r\in\mathbb{N}}$. Let $L_r^\dagger$ be as defined in (15). For every event $E \in \mathcal{F}_\rho$ (i.e. E such that $E \cap (\rho = r) \in \mathcal{F}_r$),*

$$\mathbb{E}_\theta[L_\rho^\dagger] \ge d(\mathbb{P}_\theta(E), \mathbb{P}_{\theta'}(E)).$$

Here, $d(x,y) = x\log(x/y) + (1-x)\log((1-x)/(1-y))$.

Proof. The first challenge in adapting the above result from Kaufmann et al. (2016) for the sequential setting to ours is to show the following technical result: *Let Z_s, L_N be as defined above. Then, for all $N \ge 0$, and for all measurable $g : \mathbb{R}^N \to \mathbb{R}$, $\mathbb{E}_{\theta'}\left[g\left(\{Z_s\}_{s=1}^N\right)\right] = \mathbb{E}_\theta\left[g\left(\{Z_s\}_{s=1}^N\right)\exp\left(-L_N\right)\right]$.* This claim can be proved using the same induction argument in Lemma 18 of Kaufmann et al. (2016), where the base case uses the fact that the algorithm's initial action does not depend on any observations and the induction step notes that $\mathbb{E}_{\theta'}\left[g\left(\{Z_s\}_{s=1}^{N+1}\right) \mid \left(\{(K_s,Z_s)\}_{s=1}^N\right)\right]$ is a measurable function mapping $\mathbb{R}^N$ to $\mathbb{R}$.

Now, let $\mathcal{F}'_N = \sigma(\{(K_s,Z_s)\}_{s=1}^N))$. The above result implies, in particular, that for all $N \ge 1$ and any event $E' \in \mathcal{F}'_N$, we have $\mathbb{E}_{\theta'}[\mathbb{1}_{E'}] = \mathbb{E}_\theta[\mathbb{1}_{E'}\exp(-L_N)]$. Now, let E and ρ be as given in the theorem statement.

$$\mathbb{P}_{\theta'}(E) = \mathbb{E}_{\theta'}[\mathbb{1}_E] = \sum_{r=1}^\infty \mathbb{E}_{\theta'}[\underbrace{\mathbb{1}(E \cap \{\rho = r\})}_{\in \mathcal{F}_r = \mathcal{F}'_{N(r)}}] = \sum_{r=1}^\infty \mathbb{E}_{\theta'}[\mathbb{1}(E \cap \{\rho = r\})\exp(-L_{N(r)})] = \mathbb{E}_\theta[\mathbb{1}_E \exp(-L_\rho^\dagger)].$$

Given this result, the remainder of the proof is identical to the proof of their Lemma 19. □

We are now ready to prove Lemma 13.

Proof of Lemma 13. Recall from the beginning of this section that we assume the samples for each arm are generated upfront and then revealed in order; let $\{Y_{k,s}\}_{s\geq 1}$ denote the sequence of these observations for arm k. We can write the log-likelihood ration $L_r^\dagger$ as follows,

$$L_r^\dagger = \sum_{k=1}^{n} \sum_{s=1}^{N_{(k)}(r)} \log\left(\frac{f_k(Y_{k,s})}{f'_k(Y_{k,s})}\right).$$

We can now conclude, for any event $E \in \mathcal{F}_\rho$,

$$d(\mathbb{P}_\theta(E), \mathbb{P}_{\theta'}(E)) \leq \mathbb{E}_\theta[L_\rho^\dagger] = \mathbb{E}_\theta\left[\sum_{k=1}^{n} \sum_{s=1}^{N_{(k)}(r)} \log\left(\frac{f_k(Y_{k,s})}{f'_k(Y_{k,s})}\right)\right] = \sum_{k=1}^{n} \mathbb{E}_\theta[N_{(k)}(r)]\mathrm{KL}(\theta_k \| \theta'_{(k)}) \tag{16}$$

Here, the first step uses Lemma 14, the second step uses the expression for $L_\rho^\dagger$ above, and the last step uses Wald's identity.

Let $k \in [n]$ be given and assume $\theta_{(k)}$ is not the best arm, i.e. $\mu_{(k)} \neq \mu_1$; we will handle the case where $\theta_{(k)}$ is the best arm later. Now fix $\alpha > 0$ and let θ' be an alternative model where $\theta'_{(j)} = \theta_j$ for $j \neq k$ and $\theta'_{(k)} = \mathcal{N}(\mu_1 + \alpha, 1)$. Let E denote the event that the recommended arm is k. Since $A \in \mathcal{A}_{\delta,\lambda}$, we have $\mathbb{P}_{\theta'}(E) \geq 1 - \delta$ and $\mathbb{P}_\theta(E) \leq 1 - \delta$. Therefore,

$$\log\left(\frac{1}{2.4\delta}\right) \leq d(\delta, 1-\delta) \leq d(\mathbb{P}_\theta(E), \mathbb{P}_{\theta'}(E)) \leq \sum_{j=1}^{n} \mathbb{E}_\theta[N_{(j)}(r)]\mathrm{KL}(\theta_j \| \theta'_{(j)}) = \mathbb{E}_\theta[N_{(k)}(r)]\frac{(\Delta_{(k)} + \alpha)^2}{2}.$$

Here, the first step uses the algebraic identity $d(\delta, 1-\delta) \geq \log(1/(2.4\delta))$, the second step uses the fact that $d(x, y)$ is increasing in x for $x > y$ and decreasing when $x < y$, and the third step uses (16). The last step first observes that the KL divergence between all arms is zero except between $\theta_{(k)}$ and $\theta'_{(k)}$; moreover, as the KL divergence between two Gaussians can be expressed as $\mathrm{KL}(\mathcal{N}(\mu_1, 1) \| \mathcal{N}(\mu_2, 1)) = (\mu_1 - \mu_2)^2/2$, we have $\mathrm{KL}(\theta_k \| \theta'_{(k)}) = (\mu_1 + \alpha - \mu_{(k)})^2/2 = (\Delta_{(k)} + \alpha)^2/2$. Therefore, $\mathbb{E}[N_{(k)}(\rho)] \geq \frac{2}{(\Delta_{(k)}+\alpha)^2} \log\left(\frac{1}{2.4\delta}\right)$. The statement is true for all $\alpha > 0$, so letting $\alpha \to 0$ yields the claim for all $\theta_{(k)}$ that are not the best arm.

The proof for when $\theta_{(k)}$ is the best arm can be obtained by considering a model θ' where $\theta'_{(j)} = \theta_{(j)}$ for $j \neq k$ and $\theta'_{(k)} = \mathcal{N}(\mu_2 - \alpha, 1)$ and following a similar argument. □

D. Proof of Upper Bound in the Fixed Deadline Setting

D.1. Proof of Theorem 3

In this section, we will prove Theorem 3. The analysis will borrow techniques from the proof for sequential halving Karnin et al. (2013). We will utilize the following lemmas in the proof. Let $\hat{p}_i^r$ be the empirical mean of arm i at round r. The following lemma is a direct application of Hoeffding's inequality on the difference of empirical means $\hat{p}_1^r - \hat{p}_i^r$.

Lemma 15. *Assume that the best arm was not eliminated prior to round r. Then, for any arm $i \in S_r$,*

$$\Pr[\hat{p}_1^r < \hat{p}_i^r] \leq \exp\left(-\frac{1}{2} t_r \Delta_i^2\right)$$

The following lemma is a version of Lemma 4.3 from Karnin et al. (2013). We will use this lemma with the union bound to bound the probability of eliminating an arm at a given stage.

Lemma 16. *Suppose S arms remain and each arm was pulled t times. The probability that $\left\lceil \frac{|S|}{2} \right\rceil$ of the arms have greater empirical mean than the best arm is at most:*

$$3\exp\left(-\frac{1}{2} t \Delta_{i_S}^2\right)$$

where $i_S = \left\lceil \frac{1}{2} \left\lceil \frac{|S|}{2} \right\rceil \right\rceil$

Proof. Define S' as the set of arms in S minus the arms with the top $\left\lceil \frac{1}{2}\left\lceil \frac{|S|}{2}\right\rceil\right\rceil$ true means. In order for half of the arms to have higher empirical mean than the best arm, at least $\left\lceil \frac{|S|}{2}\right\rceil - \left\lceil \frac{1}{2}\left\lceil \frac{|S|}{2}\right\rceil\right\rceil + 1 = \left\lceil \frac{|S|}{2}\right\rceil - \left\lceil \frac{1}{2}\left\lfloor \frac{|S|}{2}\right\rfloor\right\rceil \geq \frac{|S|}{4}$ arms in S' must have greater empirical mean than the best arm. Since S' has size $|S| - \left\lceil \frac{1}{2}\left\lceil \frac{|S|}{2}\right\rceil\right\rceil \leq \frac{3}{4}|S|$, at least $\frac{1}{3}$-rd of the arms in S' need to have higher empirical mean than the best arm's empirical mean. Let us first upper bound N, the number of arms in S' that have higher empirical mean than the top arm.

$$\begin{aligned}
\mathbb{E}[N] &= \sum_{i\in S'} \Pr[\hat{p}_1 < \hat{p}_i] \\
&\leq \sum_{i\in S'} \exp\left(-\frac{1}{2}t\Delta_i^2\right) \\
&\leq |S'| \max_{i\in S'} \exp\left(-\frac{1}{2}t\Delta_i^2\right) \\
&\leq |S'| \exp\left(-\frac{1}{2}t\Delta_{i_S}^2\right)
\end{aligned}$$

Now, let's use Markov's inequality to bound the desired probability:

$$\begin{aligned}
\Pr\left[N > \frac{1}{3}S'\right] &\leq 3\frac{\mathbb{E}[N]}{S'} \\
&\leq 3\exp\left(-\frac{1}{2}t\Delta_{i_S}^2\right)
\end{aligned}$$

□

In the following lemma, we will lower bound the number of pulls each surviving arm receives, and show that it increases by a factor of 2^k each stage.

Lemma 17. *The number of pulls per arm at stage r of Algorithm 2, t_r, is greater than:*

$$2^{rk}(2^k-1)x(k)$$

Proof. We will show this by induction. Let $r = 0$.

$$\begin{aligned}
(2^k-1)x(k) &= (2^k-1)\left\lfloor \frac{\lambda^{-1}\left(\frac{T}{\lceil\log_{2^k}(n)\rceil}\right)}{2^{k\lceil\log_{2^k}(n)\rceil}(2^k-1)}\right\rfloor \\
&\leq (2^k-1)\left\lfloor \frac{\lambda^{-1}\left(\frac{T}{\lceil\log_{2^k}(n)\rceil}\right)}{n(2^k-1)}\right\rfloor \\
&\leq \left\lfloor \frac{\lambda^{-1}\left(\frac{T}{\lceil\log_{2^k}(n)\rceil}\right)}{n}\right\rfloor \\
&= t_0
\end{aligned}$$

Suppose this lemma holds for some r. Then,

$$2^{(r+1)k}(2^k-1)x(k) = 2^{(r+1)k}(2^k-1)\left\lfloor \frac{\lambda^{-1}\left(\frac{T}{\lceil\log_{2^k}(n)\rceil}\right)}{2^{k\lceil\log_{2^k}(n)\rceil}(2^k-1)}\right\rfloor$$

$$\leq 2^{(r+1)k} \left\lfloor \frac{\lambda^{-1}\left(\frac{T}{\lceil \log_{2^k}(n)\rceil}\right)}{2^{k\lceil \log_{2^k}(n)\rceil}} \right\rfloor$$
$$\leq \left\lfloor \frac{2^{(r+1)k}\lambda^{-1}\left(\frac{T}{\lceil \log_{2^k}(n)\rceil}\right)}{2^{k\lceil \log_{2^k}(n)\rceil}} \right\rfloor$$

Suppose n is a factor of 2^k. Then $|S_r| = \frac{n}{2^{rk}}$.

$$\left\lfloor \frac{2^{(r+1)k}\lambda^{-1}\left(\frac{T}{\lceil \log_{2^k}(n)\rceil}\right)}{2^{k\lceil \log_{2^k}(n)\rceil}} \right\rfloor = \left\lfloor \frac{2^{(r+1)k}\lambda^{-1}\left(\frac{T}{\lceil \log_{2^k}(n)\rceil}\right)}{n} \right\rfloor$$
$$= \left\lfloor \frac{\lambda^{-1}\left(\frac{T}{\lceil \log_{2^k}(n)\rceil}\right)}{|S_{r+1}|} \right\rfloor$$
$$= t_r$$

Suppose n is not a factor of 2^k. Then, $|S_r| \leq \frac{n}{2^{(r-1)k}}$.

$$\left\lfloor \frac{2^{(r+1)k}\lambda^{-1}\left(\frac{T}{\lceil \log_{2^k}(n)\rceil}\right)}{2^{k\lceil \log_{2^k}(n)\rceil}} \right\rfloor \leq \left\lfloor \frac{2^{rk}\lambda^{-1}\left(\frac{T}{\lceil \log_{2^k}(n)\rceil}\right)}{n} \right\rfloor$$
$$\leq \left\lfloor \frac{\lambda^{-1}\left(\frac{T}{\lceil \log_{2^k}(n)\rceil}\right)}{|S_{r+1}|} \right\rfloor$$
$$= t_r$$

□

Now, we will use the previous lemmas to prove a lemma that bounds the probability of elimination in a given stage.

Lemma 18. *Assume the best arm has not been eliminated by round r. If $|S_r| \geq 2^k$, the probability that the best arm is eliminated in round r is at most:*

$$3k \exp\left(-\frac{nx(k)}{8H_2}\right)$$

Otherwise, the probability that the best arm is eliminated is at most

$$3\lceil \log_2(|S_r|)\rceil \exp\left(-\frac{nx(k)}{8H_2}\right)$$

Proof. Let's assume $|S_r| \geq 2^k$. In round r, each arm is sampled $t_r \geq 2^{rk}(2^k - 1)x(k)$ times. We only keep the top $\frac{1}{2^k}$ (by their empirical means) fraction of the arms in S_r. Suppose we eliminate arms in the following fashion: first we eliminate the lowest $\left\lfloor \frac{|S_r|}{2} \right\rfloor$ arms, then the next $\left\lfloor \frac{|S_r|}{4} \right\rfloor$, and so on ($k$ times). Call the event that the best arm is eliminated in the i-th such elimination procedure B_i. Call the set of arms that survived until the i-th event $S_{r,i}$ We thus want to upper bound:

$$P\left(\bigcup_{j=1}^{k} B_j\right) \leq \sum_{j=1}^{k} P(B_j)$$

$$\leq 3\sum_{j=1}^{k}\exp\left(-\frac{1}{2}t_r\Delta^2_{i_{S_{r,j}}}\right)$$
$$\leq 3\sum_{j=1}^{k}\exp\left(-\frac{1}{2}2^{rk}(2^k-1)x(k)\Delta^2_{i_{S_{r,j}}}\right)$$
$$\leq 3\sum_{j=1}^{k}\exp\left(-\frac{1}{2}2^{rk+j-1}x(k)\Delta^2_{i_{S_{r,j}}}\right)$$

The first line follows from the union bound. The second line follows via application of Lemma 16. The third step follows since $t_r \geq 2^{rk}(2^k-1)x(k)$ by Lemma 17. The last step applies $2^{j-1} \leq 2^k - 1$.

Observe that $i_{S_{r,i}} \geq \frac{n}{2^{rk+i+1}}$. So,

$$3\sum_{j=1}^{k}\exp\left(-\frac{1}{2}2^{rk+j-1}X_k\Delta^2_{i_{S_{r,j}}}\right) \leq 3\sum_{j=1}^{k}\exp\left(-\frac{1}{8}2^{rk+j+1}x(k)\Delta^2_{i_{S_{r,j}}}\right)$$
$$\leq 3\sum_{j=1}^{k}\exp\left(-\frac{n}{8}\frac{\Delta^2_{i_{S_{r,j}}}}{i_{S_{r,i}}}\right)$$
$$\leq 3k\exp\left(-\frac{nx(k)}{8H_2}\right)$$

The second line uses the bound on $i_{S_{r,i}}$, and the third line maximizes the expression in the sum over i (recall that $H_2 = \max_{i\neq 1} i\Delta_i^{-2}$).

The proof of the second half of the lemma follows by replacing k with $\lceil\log_2(|S|)\rceil$ in the previous steps.

□

We are now ready to prove Theorem 3.

Proof of Theorem 3:

Proof. The algorithm always terminates in T time, as it takes $\lceil\log_{2^k}(n)\rceil$ stages that each take at most $T/\lceil\log_{2^k}(n)\rceil$ time. So, let's upper bound the probability of error.

To select the best arm, it must survive all $\left\lceil\frac{\log_2(n)}{k}\right\rceil$ rounds, after which it will be the last arm remaining. Using the union bound with Lemma 18, we can upper bound the probability of elimination of the best arm by:

$$\sum_{r=0}^{\lfloor\log_2(n)/k\rfloor} 3k\exp\left(-\frac{nx(k)}{8H_2}\right) + 3\lceil\log_2(|S_{r_f-1}|)\rceil\exp\left(-\frac{nx(k)}{8H_2}\right)$$
$$= 3\lceil\log_2(n)\rceil\exp\left(-\frac{nx(k)}{8H_2}\right) \tag{17}$$

□

Remark 2. *Applying Theorem 3 with $k = 1$, and $\lambda(m) = m^{-1}$, assuming n is a power of 2 (for simplicity), and that 1 is evenly divisible by $\frac{n\log_2(n)}{T}$, we recover the error probability upper bound of sequential halving (Karnin et al., 2013):*

$$3\log_2(n)\exp\left(-\frac{nT}{8H_2\log_2(n)}\right)$$

Remark 3. *The naive extension of sequential halving (which is Algorithm 2 with $k = 1$) has a higher bound on the error probability than Algorithm 2 with $k = k^\star$. This is because $x(k^\star) \geq x(1)$ and the error probability upper bound (Equation 17) is decreasing in $x(k)$.*

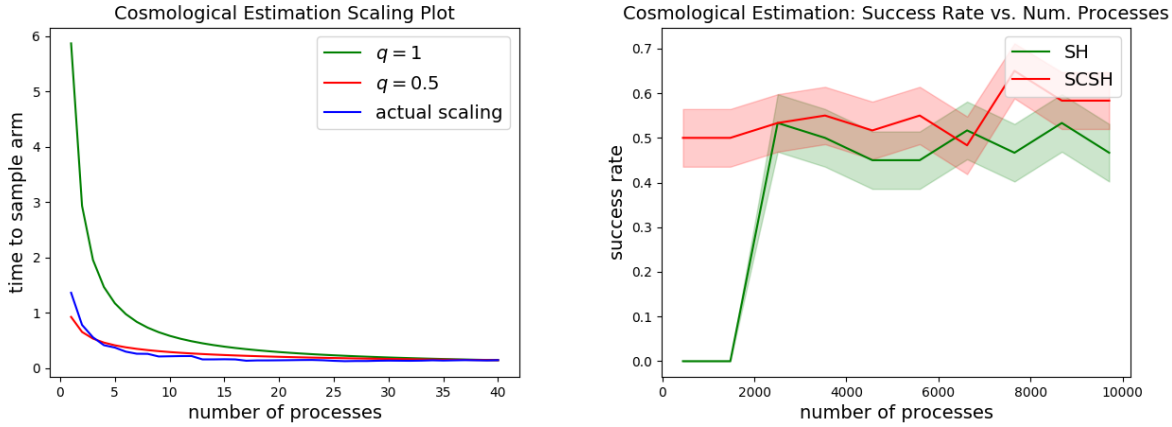

Figure 5. **Cosmological Parameter Estimation Experiment: Left:** From a total pool of 40 processes allowed per arm pull, we plot the empirical scaling function averaged over 10 runs of the likelihood computation procedure and plot $\lambda(m) \propto m^{-q}$ for $q \in \{0.5, 1\}$ after calibrating $\lambda(1)$ (which corresponds to using 40 processes). The behavior of the empirical scaling function is initially linear but becomes close to inverse square-root after 7 processes. **Right:** We evaluate SSH on a cosmological parameter estimation task with 64 arms, $T = 0.56$, and vary the total number of processes available. We present the mean and standard error across 60 runs. Although the scaling function is good ($q \approx 0.5$), SSH appears to consistently do as well or better than SH, especially when resources are very limited.

D.2. Lower bounding $x(k)$ for $\lambda(m) = m^q$

In this section, we will lower bound $x(k)$, which will allow us to upper bound the error probability of Algorithm 2 when contextualizing the results for different values of k.

Assuming $x(k) \geq 1$ and $2^k \leq n$:

$$\begin{aligned}
x(k) &= \left\lfloor \frac{\lambda^{-1}\left(\frac{T}{\lceil \log_{2^k}(n) \rceil}\right)}{2^{k\lceil \log_{2^k}(n) \rceil}(2^k - 1)} \right\rfloor \\
&= \frac{\lambda^{-1}\left(\frac{T}{\lceil \log_{2^k}(n) \rceil}\right)}{2^k n(2^k - 1)} \\
&\geq \frac{\lambda^{-1}\left(\frac{T}{\lceil \log_{2^k}(n) \rceil}\right)}{4^k n} \\
&\geq \frac{1}{4^k n}\left(\frac{T}{\lceil \log_{2^k}(n) \rceil}\right)^{1/q} \\
&\geq \frac{1}{4^k n}\left(\frac{Tk}{2\log_2(n)}\right)^{1/q}
\end{aligned}$$

In the second line, we upper bound the first term in the denominator, which may be greater by a factor of 2^k. In the fourth line, we plug in $\lambda^{-1}(t) = t^{1/q}$. In the last step, we upper bound the ceil using $2^k \leq n$.

E. Physical experiment: Estimation of cosmological parameters

We evaluate Algorithm 2 on a variant of a problem in Kandasamy et al. (2019), where the goal is to estimate the Hubble constant, dark matter fraction, and dark energy fraction from data on Type Ia supernova via maximum likelihood estimation, using data from Davis et al. (2007). We focus on the fixed budget setting in this experiment as algorithms in the fixed budget settings are more practical and easier to adapt to real problems (Karnin et al., 2013). The likelihood is computed using the method from Shchigolev (2017), and involves numerically evaluating integrals for each data point before a serial portion. We discretize the continuous parameter space into 64 candidates (arms). We treat the likelihood of a parameter when computed using all 192 points in the dataset as the mean of an arm. When sampling an arm, we compute the likelihood using 50 points

sampled with replacement from the dataset. Each arm pull can be allocated a number of processes to parallelize the set of numerical integrals that must be computed. First, we approximately compute the scaling function of the arm sampling as a function of its number of processes. In this particular scenario, the communication and synchronization costs are very limited, which results in fairly good scaling. Nevertheless, we plot the empirically estimated scaling function (Figure 5(a)) and observe that the scaling function is close to $\lambda(m) = m^{-0.5}$. When computing $k^\star$, we approximately compute $\lambda^{-1}(t)$ by finding the smallest m such that $\lambda(m) \leq t$. Additional experimental details for this experiment can be found in Section E. In this experiment, we fix the time budget to be 0.56 seconds. In this setting, $k^\star = 2$, so each stage takes 0.28 seconds, which requires at least 7 processes per pull to complete pulls in time. Sequential halving, which has 0.14 seconds/stage, requires 25 processes per pull to complete pulls in time. We observe that Algorithm 2 consistently matches or outperforms sequential halving in this setting (Figure 5(b)).

F. Baseline hyperparameters

APR and BatchRacing: Empirically, we found that the deviation function used in the confidence intervals of Algorithm 1 and the baselines is very conservative, so we scale the deviation by 0.2 for all algorithms. Scaling confidence intervals this way is a common technique in applied UCB-style bandit work (Kandasamy et al., 2015; Wang et al., 2017). Despite this scaling, all algorithms almost always identify the best arm in the synthetic experiments.

UCB-E: UCB-E (Audibert & Bubeck, 2010) requires a scale hyperparameter for the confidence bounds. Specifically, the confidence bounds are computed as $(\hat{\mu} - \frac{a}{\sqrt{n}}, \hat{\mu} + \frac{a}{\sqrt{n}})$, where $\hat{\mu}$ is the empirical mean, n is the number of times the arm has been pulled, and a is the scale parameter. We tune a via grid search, and find that when $q = 0.9$, $a = 1$ performs best empirically. For all other experiments, no value of a helps, as the time budgets are too small for a sequential algorithm to make sufficient progress.