Supplementary Material

In the supplementary material, we have provided the following. Additional related works are discussed in Appendix A. Further details on the GP-ThreDS algorithm are presented in Appendix B. Proof of Theorem 1 and all the lemmas are provided in Appendix C. More details on the experiments, as well as additional experiments are given in Appendix D.

A Additional Related Work

Following the work by [1] on GP-UCB, several extensions have been proposed based on combining GP with bandit techniques. Representative results include extensions to arbitrary compact metric spaces [2], contextual bandits [3, 4], parallel observations [5, 6], ordinal models [7], robust optimization [8], and multi-fidelity observations [9]. In [10], the authors proposed an improved version of GP-UCB with an improved confidence interval based on a self-normalized concentration inequality that was inspired by similar results derived in [11] for linear bandits. The query point selection strategy in all these approaches involves optimizing the UCB over the entire domain through an exhaustive search over a grid of $O(t^{2d})$ points at time instant t.

There is a growing body of work in the literature addressing the high cost associated with computing the posterior distribution (the first computational bottleneck as discussed in the main text). Such approaches usually involve approximating the GP posterior by using techniques such as adaptive matrix sketching [12], sparse variational inference [13, 14, 15, 16], random Fourier features [17], linearization [18] and additivity [19].

As pointed out in the main text, the dominating source of the computational cost is in finding the maximizer of the UCB score. This issue has not received much attention except in a couple of recent studies. Mutný et al. [20] considered a problem where the kernel can be approximated with Quadratic Fourier Features. This additional assumption results in a linear model where the UCB proxy can be optimized using an efficient global optimizer. However, this assumption practically limits the GP model to squared exponential kernels. In contrast, the computationally efficient approach proposed in this work is generally applicable. In [21], the authors proposed an adaptive discretization approach similar to that of [22, 23]. The key idea is to replace the uniform discretization in GP-UCB with a non-uniform discretization that adapts to the observed function values so that regions with higher function values enjoy a finer discretization. Nevertheless, the discretization is still carried over the entire function domain throughout the learning process, and a global maximization of the UCB score needs to be carried out at each time instant over a linearly growing set of discrete points. The proposed GP-ThreDS, however, continuously shrinks the function domain and evaluates the UCB score always on a *bounded* set of discrete points. The global maximization objective is also relaxed to determining the existence of threshold-exceeding points.

Using a tree structure to represent successive partitions of the search domain is a classical approach and has seen its use in the bandit literature [22, 24, 25]. Such methods are characterized by growing the tree at nodes with high UCB without pruning the nodes with low values of UCB. This is fundamentally different from the domain shrinking approach of GP-ThreDS. A different tree-based method was considered in [26] where the tree structure is dynamic and may not be computationally efficient. Furthermore, they did not provide any theoretical results.

In contrast to our agnostic regularity assumption on f (being fixed and belonging to an RKHS), a Bayesian setting was also considered in [1] where f is assumed to be a sample from a GP. The regret bounds were then provided in high probability with respect to both noise and the randomness in f.

For GP-UCB algorithm, [1] proved a tighter $O(\sqrt{T\gamma_T})$ bound under the Bayesian setting. Under the Bayesian setting, [27] built on ideas from [28, 29] to show that GP-TS achieves the same order of regret as GP-UCB. A Bayesian optimization algorithm for max-value entropy search was shown to enjoy the same regret order as GP-UCB and GP-TS in [30]. An $\Omega(\sqrt{T})$ lower bound on regret was proven under the Bayesian setting [31, 32]

Several works consider a noise-free setting ($\epsilon_t = 0, \forall t$) which results in tighter regret bounds. In particular [33] and [34] studied the noise-free Bayesian optimization for Matérn family of kernels, under simple and cumulative regret settings, respectively. The simple regret problem can be addressed using pure exploration algorithms such as epsilon greedy. Under a Bayesian and noise free setting the regret bounds can further improve to exponential rates [35, 36, 37] .

Practitioners' approach to Bayesian optimization generally consists of selecting the observation points based on optimizing the so called acquisition functions (such as UCB in GP-UCB and Thompson sample in GP-TS). Other notable acquisition functions are GP-EI (that stands for expected improvement) and GP-PI (that stands for probability of improvement) [38], which are shown to enjoy the same regret guarantees as GP-UCB [39, 40, 41]. When implementing Bayesian optimization algorithms which are based on an acquisition function (GP-UCB, TS, PI and EI), a practical idea is to use an off-the-shelf optimizer to solve the optimization of the acquisition function at each iteration. This method although can lead to significant gains in computational complexity, invalidates the existing regret bounds. Our focus in this work has been to introduce a practical algorithm with provable regret guarantees.

B GP-ThreDS Algorithm

In this section, we provide a pseudo code for GP-ThreDS as well as additional details on its implementation.

B.1 GP-ThreDS Pseudo Code

A pseudo code for GP-ThreDS is given in Algorithm 1 below. In the pseudo-code,`getHighPerformingNodes` is the routine identifying the high-performing nodes on a tree of depth d that is the routine described in Section 3.2 of the main paper. $\mathcal{L}_v$ denotes the set of high-performing nodes returned by `getHighPerformingNodes` corresponding to the node v in $\mathcal{D}_k$.

B.2 Random-walk based search for high-performing nodes

We provide additional details of the random walk based search described in Sec. 3.2.1 in the main paper. We first provide a pseudo code for the random-walk based search strategy for the case of identifying a single high-performing node with confidence level δ_{RW} in Alg. 2.
As the names suggest, the function `root` returns the root node of the tree, `parent`, `leftChild` and `rightChild` return the parent node, the left child and the right child, respectively, of the node in the argument. `SequentialTest` is the sequential test routine described in Section 3.2.2 of the main paper. In addition to the node and threshold, it takes two confidence parameters as input. If only one is provided, then a sequential test with symmetric confidence levels is carried out as described in the main paper. If two arguments are provided, then the routine carries out the sequential test with asymmetric confidence levels as described in Appendix B.3. In this case, the former parameter is considered to be a bound on the probability of a false negative and the latter parameter a bound on the probability of a false positive. Lastly, $p \in (0, 1/2)$ is the confidence parameter that is associated with the bias of the random walk, $\hat{\delta}$ is the confidence parameter determined by δ_{RW}, whose exact value is given below.
This strategy can be extended for the case of an unknown number of high-performing leaf nodes as follows. First, each high-performing node is identified using a separate iteration of the routine described in Alg. 2. Thus, several runs of the random walk are carried out. Secondly, in order to address the issue of unknown number of high-performing nodes, a termination test at the root node is carried out to determine whether there is any unidentified high-performing leaf node left. A pseudo code is described in Alg. 3.

In Alg. 3 a high-performing node is identified, it is not considered in future iterations, in order to avoid redetection. Specifically, while carrying out the sequential test on any node, during the $(r+1)^{\text{th}}$ iteration, on any ancestor of the r identified nodes, the points belonging to the identified high-performing nodes are not considered. This is reflected in updating $\mathcal{T}$ to $\mathcal{T} \setminus$ retNode. The parameter δ_{RW} is set to $\delta_0/4T$

Algorithm 1 GP-ThreDS

Input: $\mathcal{D}_0 = \{\mathcal{X}\}$, $[a_1, b_1] = [0, 1]$, $\delta_0 \in (0, 1)$
// The refining stage in epoch 0 is completed and we have a tree with 2^d leaves with root at $\mathcal{D}_0$.
Set $k \leftarrow 1$, $\rho_1 \leftarrow d$, $\tau_1 = (a_1 + b_1)2$
repeat
  Set $\mathcal{D}_k \leftarrow \emptyset$
  // Start Pruning Stage
  for υ in $\mathcal{D}_{k-1}$ **do**
    Set $\mathcal{T}_\upsilon$ to be the tree of depth d rooted at the node $\upsilon \in \mathcal{D}_{k-1}$
    $\mathcal{L}_\upsilon \leftarrow$ getHighPerformingNodes($\mathcal{T}_\upsilon$, τ_k, $\delta_0/4T$)
    $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \mathcal{L}_\upsilon$
  end for
  if $\mathcal{D}_k = \emptyset$ **then**
    // No refining
    $\mathcal{D}_k \leftarrow \mathcal{D}_{k-1}$, $\rho_{k+1} \leftarrow \rho_k$
    $a_{k+1} \leftarrow a_k - \frac{(b_k - a_k)}{2}$ and $b_{k+1} \leftarrow b_k - \frac{(b_k - a_k)}{2}$
  else
    //Carry refining stage by growing subtrees rooted at υ for all $\upsilon \in \mathcal{D}_k$
    $a_{k+1} \leftarrow \tau_k - c2^{-\alpha\rho_k/d+1}$, $b_{k+1} \leftarrow b_k$, $\rho_{k+1} \leftarrow \rho_k + d$
  end if
  // Update the threshold
  $\tau_{k+1} = (a_{k+1} + b_{k+1})/2$
  $k \leftarrow k + 1$
until query budget is exhausted

Algorithm 2 Random-walk based strategy for one high-performing node

Input: Binary tree $\mathcal{T}$ of depth d, threshold τ, confidence level δ_{RW}.
Set currNode $\leftarrow$ root($\mathcal{T}$), terminate $\leftarrow 0$
while terminate $\neq 1$ **do**
  if depth(currNode) $== d$ **then**
    retLeaf $\leftarrow$ SequentialTest(currNode,$\tau, p, \hat{\delta}$)
    if retLeaf $== 1$ **then**
      terminate $\leftarrow 1$
      retNode $\leftarrow$ currNode
    else
      currNode $\leftarrow$ parent(currNode)
    end if
  else
    retLeft $\leftarrow$ SequentialTest(leftChild(currNode),τ, p)
    if retLeft $== 1$ **then**
      currNode $\leftarrow$ leftChild(currNode)
    else
      retRight $\leftarrow$ SequentialTest(rightChild(currNode),τ, p)
      if retRight $== 1$ **then**
        currNode $\leftarrow$ rightChild(currNode)
      else
        currNode $\leftarrow$ parent(currNode)
      end if
    end if
  end if
end while
return retNode

and $\hat{\delta}^{(r)} = \frac{\delta_0}{8Tr(r+1)(p-1/2)^2} \log\left(\frac{4dT}{\delta_0}\right)$.

In the experiments, for a simpler implementation, we have devised the search for high-performing nodes at depth d, for a given input domain D, by directly searching among the leaf nodes. To be complete, a pseudo-code is given in Alg. 4 which is slightly different from the pseudo-code given in Alg. 3 regarding this step.

Algorithm 3 Random-walk based strategy for multiple high-performing node

Input: Binary tree $\mathcal{T}$ of depth d, threshold τ, confidence level δ_{RW}.
Set currNode $\leftarrow$ `root`($\mathcal{T}$), terminate $\leftarrow 0$, $r \leftarrow 1$, HPNodes $\leftarrow \emptyset$, retNode = NULL
while terminate $\neq 1$ **do**
  if currNode == `root`($\mathcal{T}$) **then**
    retRoot $\leftarrow$ `SequentialTest`(currNode,$\tau, \hat{\delta}^{(r)}, p$)
    if retRoot $== -1$ **then**
      terminate $\leftarrow 1$
    end if
  else
    if `depth`(currNode) $== d$ **then**
      retLeaf $\leftarrow$ `SequentialTest`(currNode,$\tau, p, \hat{\delta}^{(r)}$)
      if retLeaf $== 1$ **then**
        retNode $\leftarrow$ currNode
      else
        currNode $\leftarrow$ `parent`(currNode)
      end if
    else
      retLeft $\leftarrow$ `SequentialTest`(`leftChild`(currNode),τ, p)
      if retLeft $== 1$ **then**
        currNode $\leftarrow$ `leftChild`(currNode)
      else
        retRight $\leftarrow$ `SequentialTest`(`rightChild`(currNode),τ, p)
        if retRight $== 1$ **then**
          currNode $\leftarrow$ `rightChild`(currNode)
        else
          currNode $\leftarrow$ `parent`(currNode)
        end if
      end if
    end if
    if retNode ! = NULL **then**
      HPNodes $\leftarrow$ HPNodes $\cup$ retNode
      $\mathcal{T} \leftarrow \mathcal{T} \setminus$ retNode
      $r \leftarrow r + 1$
      retNode = NULL
    end if
  end if
end while
return HPNodes

B.3 Sequential test with asymmetric confidence levels

In this section, we describe the sequential test with asymmetric confidence levels. Recall the sequential test with symmetric confidence level η, given a node/region D and a threshold τ described in Sec. 3.2.2 of the main paper (see Fig. 1). In the asymmetric case, similarly, we assume a threshold τ and a node/region D is given. We consider two separate cases based on whether false positive rate is higher than false negative rate or vice versa.

In the first case, let $1-\hat{\delta}$ denote the confidence level for declaring whether node D is high-performing (i.e., it contains a point with function value exceeding τ). Let $1-p$ denote the confidence level for declaring the

Algorithm 4 Heuristic for Updating the domain

Input: Input domain D, discretization D_g, $\mathcal{C}$, confidence parameter δ, $x_1 \in D_g$, threshold τ
Set HPNodes $\leftarrow \{\}$, terminate $\leftarrow 0$, $t \leftarrow 1$, $t_{loc} \leftarrow 1$
while terminate $\neq 1$ **do**
  if $\max_{x \in D_g} \mu_{t-1}(x) + \beta_t(\delta)\sigma_{t-1}(x) \leq \tau - L\Delta^{\alpha}$ **then**
    terminate $\leftarrow 1$ // Stop the search
  else if $\max_{x \in D_g} \mu_{t-1}(x) - \beta_t(\delta)\sigma_{t-1}(x) \geq \tau$ **then**
    $c^* \leftarrow \{c \in \mathcal{C} : \arg\max_{x \in D_g} \mu_{t-1}(x) - \beta_t(\delta)\sigma_{t-1}(x) \in c\}$
    HPNodes $\leftarrow$ HPNodes $\cup$ c^* // Add the child node to the list of high-performing nodes
    $\mathcal{C} \leftarrow \mathcal{C} \setminus c^*$ // Update the set of children
    $D_g \leftarrow D_g \setminus \{x : x \in c^*\}$ // Update the set of search points
    $t_{loc} \leftarrow 0$
  else if $t_{loc} == t_{\text{term}}$ **then**
    $c^* \leftarrow \{c \in \mathcal{C} : \arg\max_{x \in D_g} \mu_{t-1}(x) - \beta_t(\delta)\sigma_{t-1}(x) \in c\}$
    HPNodes $\leftarrow$ HPNodes $\cup$ c^*
    $\mathcal{C} \leftarrow \mathcal{C} \setminus c^*$
    $D_g \leftarrow D_g \setminus \{x : x \in c^*\}$
    $t_{loc} \leftarrow 0$
  end if
  $x_t \leftarrow \arg\max_{x \in D_g} \mu_{t-1}(x) + \beta_t(\delta)\sigma_{t-1}(x)$
  Observe $y_t = f(x_t) + \epsilon_t$
  Update $t \leftarrow t+1$, $t_{loc} \leftarrow t_{loc} + 1$ and use the update equations to obtain μ_t and σ_t
end while
return HPNodes

node is not high-performing (i.e., it does not contain a point with function value exceeding τ). We assume that the false positive rate $\hat{\delta}$ is lower than the false negative rate p. In this case the lower confidence bound and the upper confidence bound used in the test will initially be set to $\max_{x \in D_g} \mu_{s-1}(x) - \beta_s(\hat{\delta})\sigma_{s-1}(x) \geq \tau$ and $\max_{x \in D_g} \mu_{s-1}(x) + \beta_s(p)\sigma_{s-1}(x) \leq \tau - L\Delta^{\alpha}$, respectively. After $\bar{S}(p)$ steps, the upper confidence bound used in the test will be set to $\max_{x \in D_g} \mu_{s-1}(x) + \beta_s(\hat{\delta})\sigma_{s-1}(x) \leq \tau - L\Delta^{\alpha}$. The number of samples is capped to $\bar{S}(\hat{\delta})$.

If the sequential test outputs a negative value, we have $f(x^*_{D_g}) < \tau - L\Delta^{\alpha}$ with probability at least $1-p$, (the probability will be higher, at least $1-\hat{\delta}$, if the sequential test outputs a negative value after $\bar{S}(p)$ steps). If the sequential test outputs a positive value, we have $f(x^*_{D_g}) > \tau_k$ with a confidence of $1-\hat{\delta}$. A pseudo code is provided in Alg. 5.

In the case where $\hat{\delta}$ is the false negative rate and p is false positive rate ($\hat{\delta} < p$), the upper and lower confidence bounds used in the test are set to $\max_{x \in D_g} \mu_{s-1}(x) + \beta_s(\hat{\delta})\sigma_{s-1}(x) \leq \tau - L\Delta^{\alpha}$ and $\max_{x \in D_g} \mu_{s-1}(x) - \beta_s(p)\sigma_{s-1}(x) \geq \tau$, respectively. The cap on the number of samples is set to $S(\hat{\delta})$.

In this case, if the sequential test outputs a positive value, we have $f(x^*_{D_g}) > \tau_k$ with probability of at least $1-p$. If the sequential test outputs a negative value, we have $f(x^*_{D_g}) < \tau - L\Delta^{\alpha}$ with probability at least $1-\hat{\delta}$. The cap on the number of samples is set to $S(\hat{\delta})$ which helps us to focus our attention on the high confidence result of -1 rather than on the low confidence result of $+1$ resulting from the cap on the number of samples. The test may terminate (after $\bar{S}(\hat{\delta})$ steps) and output a potentially erroneous positive value that is addressed in the analysis by the update rule of the thresholds τ_k over epochs k. A pseudo code is provided in Alg. 6.

C Detailed Proofs

In this section, we provide the proof of Theorem 1 in the main paper, as well as all the lemmas that are used in the proof of theorems. We first state all the lemmas . We then provide the proof of Theorem 1 followed by the proof of the lemmas.

- If $\max_{x\in D_g} \mu_{s-1}(x) - \beta_s(\eta)\sigma_{s-1}(x) \geq \tau$, terminate and output $+1$.
- If $\max_{x\in D_g} \mu_{s-1}(x) + \beta_s(\eta)\sigma_{s-1}(x) \leq \tau - L\Delta^\alpha$, terminate and output -1.
- Otherwise, query $x_s = \arg\max_{x\in D_g} \mu_{s-1}(x) + \beta_s(\frac{\delta_0}{4T})\sigma_{s-1}(x)$
- Observe $y_s = f(x_s) + \epsilon_s$ and use it to obtain μ_s and σ_s. Increment s by 1.
- Repeat until $s < \bar{S}(\eta, L\Delta^\alpha)$.

If $s = \bar{S}(\eta, L\Delta^\alpha)$, terminate and output $+1$.

Figure 1: The local sequential test for the decision problem of finding a τ-exceeding point.

Algorithm 5 Sequential Test with Asymmetric Confidence (low false positive rate)

Input: Discretization D_g, confidence parameters $\hat{\delta}$ and p, $x_1 \in D_g$, threshold τ
terminate $\leftarrow 0$, $s \leftarrow 1$
while terminate $\neq 1$ **do**
  if $s < \bar{S}(p)$ **then**
    if $\max_{x\in D_g} \mu_{s-1}(x) - \beta_s(\hat{\delta})\sigma_{s-1}(x) \geq \tau$ **then**
      terminate $\leftarrow 1$, retVal $\leftarrow +1$
    else if $\max_{x\in D_g} \mu_{s-1}(x) + \beta_s(p)\sigma_{s-1}(x) \leq \tau - L\Delta^\alpha$ **then**
      terminate $\leftarrow 1$, retVal $\leftarrow -1$
    else
      Query $x_s = \arg\max_{x\in D_g} \mu_{s-1}(x) + \beta_s(\frac{\delta_0}{4T})\sigma_{s-1}(x)$
      Observe $y_s = f(x_s) + \epsilon_s$ and use it to obtain μ_s and σ_s
      $s \leftarrow s + 1$
    end if
  end if
  if $s \geq \bar{S}(p)$ and $s < \bar{S}(\hat{\delta})$ **then**
    if $\max_{x\in D_g} \mu_{s-1}(x) - \beta_s(\hat{\delta})\sigma_{s-1}(x) \geq \tau$ **then**
      terminate $\leftarrow 1$, retVal $\leftarrow +1$
    else if $\max_{x\in D_g} \mu_{s-1}(x) + \beta_s(\hat{\delta})\sigma_{s-1}(x) \leq \tau - L\Delta^\alpha$ **then**
      terminate $\leftarrow 1$, retVal $\leftarrow -1$
    else
      Query $x_s = \arg\max_{x\in D_g} \mu_{s-1}(x) + \beta_s(\frac{\delta_0}{4T})\sigma_{s-1}(x)$
      Observe $y_s = f(x_s) + \epsilon_s$ and use it to obtain μ_s and σ_s
      $s \leftarrow s + 1$
    end if
  end if
  if $s == \bar{S}(\hat{\delta})$ **then**
    terminate $\leftarrow 1$, retVal $\leftarrow +1$
  end if
end while
return retVal

Lemma 1. *For any set of sampling points $\{x_1, x_2, \ldots, x_t\}$ chosen from D_g (under any choice of algorithm), the following holds: $\sum_{s=1}^{t} \sigma_{s-1}(x_s) \leq (1 + 2\lambda)\sqrt{|D_g|t}$.*

Lemma 2. *If the local test is terminated by the termination condition at instant $\bar{S}(\delta_2, \Delta_f)$ defined as $\bar{S}(\delta_2, \Delta_f) = \min\left\{t \in \mathbb{N} : 2(1 + 2\lambda)\beta_t(\delta_2)\sqrt{\frac{|D_g|}{t}} \leq \Delta_f\right\} + 1$, then with probability at least $1 - \delta_2$, we have $\tau - L\Delta^\alpha - \Delta_f \leq f(x^*_{D_g}) \leq \tau + \Delta_f$.*

Lemma 3. *Consider the random walk based routine described in Section B.2 with a local confidence parameter $p \in (0, 1/2)$. Then with probability at least $1 - \delta_1$, one iteration of RWT visits less than $\frac{\log(d/\delta_1)}{2(p-1/2)^2}$ nodes before termination.*

Lemma 4. *The number of points in the discretization, $|D_g|$, for any node D, is upper bounded by a constant, independent of time. i.e., $|D_g| = O(1)$, $\forall\ t \leq T$.*

Lemma 5. *Consider the local test module carried out on a domain D, with a threshold τ and a confidence parameter $p \in (0, 1/2)$, during an epoch $k \geq 1$ of GP-ThreDS. If D contains a τ-exceeding point, then*

Algorithm 6 Sequential Test with Asymmetric Confidence (low false negative rate)

Input: Discretization D_g, confidence parameters $\hat{\delta}$ and p, $x_1 \in D_g$, threshold τ
terminate $\leftarrow 0$, $s \leftarrow 1$
while terminate $\neq 1$ **do**
  if $s < \bar{S}(\hat{\delta})$ **then**
    if $\max_{x \in D_g} \mu_{s-1}(x) - \beta_s(p)\sigma_{s-1}(x) \geq \tau$ **then**
      terminate $\leftarrow 1$, retVal $\leftarrow +1$
    else if $\max_{x \in D_g} \mu_{s-1}(x) + \beta_s(\hat{\delta})\sigma_{s-1}(x) \leq \tau - L\Delta^{\alpha}$ **then**
      terminate $\leftarrow 1$, retVal $\leftarrow -1$
    else
      Query $x_s = \arg\max_{x \in D_g} \mu_{s-1}(x) + \beta_s(\frac{\delta_0}{4T})\sigma_{s-1}(x)$
      Observe $y_s = f(x_s) + \epsilon_s$ and use it to obtain μ_s and σ_s
      $s \leftarrow s + 1$
    end if
  end if
  if $s == \bar{S}(\hat{\delta})$ **then**
    terminate $\leftarrow 1$, retVal $\leftarrow +1$
  end if
end while
return retVal

the local test module outputs $+1$ with probability at least $1 - p$. If the local test outputs -1, then with probability at least $1 - p$, D does not contain a τ-exceeding point.

Lemma 6. *Let the interval in which $f(x^*)$ lies, as maintained by the algorithm at the beginning of epoch k, be denoted by $[a_k, b_k]$. Then $|b_k - a_k| \leq (1 + 2c(\rho_k/d - 1))2^{-\alpha(\rho_k/d-1)}$.*

C.1 Proof of Theorem 1

For the regret analysis of GP-ThreDS, we write the overall regret as a sum of two terms, R_1 and R_2. R_1 is the regret incurred by the algorithm until the end of the epoch k_0, and R_2 is the regret incurred by the algorithm after k_0 epochs are completed, where $k_0 = \max\{k : \rho_k \leq \frac{d}{2\alpha} \log T\}$. All the following regret calculations are conditioned on the event that throughout the time horizon, all the random walk modules identify all the target nodes always correctly. We later show that this event occurs with a high probability.

We begin with the analysis of R_1. To obtain an upper bound on R_1, we first obtain the regret incurred at each node and sum that over the different nodes visited by the algorithm in the first k_0 epochs. Since the sampling of the algorithm is independent across different nodes, we can bound the regret incurred at any node D visited by the algorithm during an epoch $k \leq k_0$ independent of others. We denote the discretized version of the domain by D_g and x_D^* and $x_{D_g}^*$ are defined as follows: $x_D^* = \arg\max_{x \in D} f(x)$ and $x_{D_g}^* = \arg\max_{x \in D_g} f(x)$. Recall that the cap on the number of samples in epoch k is defined as

$$\bar{S}^{(k)}(p) = \min\left\{ t \in \mathbb{N} : 2\left(B + R\sqrt{2(\gamma_{t-1} + 1 + \log(1/p))}\right)(1 + 2\lambda)\sqrt{\frac{|D_g|}{t}} \leq L\Delta_k^{\alpha} \right\} + 1.$$

We focus our attention on any arbitrary node visited during the k^{th} epoch. Let N denote the random number of queries issued at that node and $\bar{R}(N)$ denote the regret incurred at that node. By the

definition of regret, we have,

$$\begin{aligned}\bar{R}(N) &= \sum_{n=1}^{N} f(x^*) - f(x_n) \\ &= \sum_{n=1}^{N} f(x^*) - \tau_k + L\Delta_k^\alpha + \tau_k - f(x^*_{D_g}) - L\Delta_k^\alpha + f(x^*_{D_g}) - f(x_n) \\ &= \underbrace{\left[\sum_{n=1}^{N} f(x^*) - \tau_k + L\Delta_k^\alpha\right]}_{R^{(1)}(N)} + \underbrace{\left[\sum_{n=1}^{N} \tau_k - L\Delta_k^\alpha - f(x^*_{D_g})\right]}_{R^{(2)}(N)} + \underbrace{\left[\sum_{n=1}^{N} f(x^*_{D_g}) - f(x_n)\right]}_{R^{(3)}(N)}.\end{aligned}$$

where x_n is the point sampled by the algorithm at the n^{th} time instant spent at the node. We will bound each of the three terms separately as outlined in the main text.

We begin with bounding the third term, $R^{(3)}(N)$. Notice that it can be bounded in the same way as the regret for IGP-UCB since the sampling is always carried out on the grid by maximizing the UCB score over it. Since $x_n = \arg\max_{x \in D_g} \mu_{n-1}(x) + \beta_n(\delta_0/4T)\sigma_{n-1}(x)$, therefore, with probability at least $1 - \delta_0/4T$, we have

$$\begin{aligned}f(x^*_{D_g}) - f(x_n) &\leq \mu_{n-1}(x^*_{D_g}) + \beta_n(\delta_0/4T)\sigma_{n-1}(x^*_{D_g}) - (\mu_n(x_n) - \beta_n(\delta_0/4T)\sigma_{n-1}(x_n)) \\ &\leq \mu_{n-1}(x_n) + \beta_n(\delta_0/4T)\sigma_{n-1}(x_n) - \mu_n(x_n) + \beta_n(\delta_0/4T)\sigma_{n-1}(x_n) \\ &\leq 2\beta_n(\delta_0/4T)\sigma_{n-1}(x_n).\end{aligned}$$

From Lemma 1, we can conclude that $\sum_{n=1}^{N} \sigma_{n-1}(x_n) \leq (1+2\lambda)\sqrt{|D_g|N}$. Using this result along with the bound on $f(x^*_{D_g}) - f(x_n)$, we obtain

$$\begin{aligned}R^{(3)}(N) &= \sum_{n=1}^{N} f(x^*_{D_g}) - f(x_n) \\ &\leq \sum_{n=1}^{N} 2\beta_n(\delta_0/4T)\sigma_{n-1}(x_n) \\ &\leq 2\beta_N(\delta_0/4T)\sum_{n=1}^{N} \sigma_{n-1}(x_n) \\ &\leq 2\left(B + R\sqrt{2(\gamma_{N-1} + 1 + \log(4T/\delta_0))}\right)(1+2\lambda)\sqrt{|D_g|N}.\end{aligned}$$

To bound the first term, $R^{(1)}(N)$, we relate the maximum number of samples taken at the node to $f(x^*) - \tau_k + L\Delta_k^\alpha$ using Lemmas 2 and 6. Recall the definition of $\bar{S}^{(k)}(p)$. It is defined as

$$\bar{S}^{(k)}(p) = \min\left\{t \in \mathbb{N} : 2\left(B + R\sqrt{2(\gamma_{t-1} + 1 + \log(1/p))}\right)(1+2\lambda)\sqrt{\frac{|D_g|}{t}} \leq L\Delta_k^\alpha\right\} + 1.$$

This implies that,

$$\begin{aligned}L\Delta_k^\alpha &\leq 2\left(B + R\sqrt{2(\gamma_{\bar{S}^{(k)}-3} + 1 + \log(1/p))}\right)(1+2\lambda)\sqrt{\frac{|D_g|}{\bar{S}^{(k)} - 2}} \\ \implies 2^{-\alpha\rho_k/d} &\leq \frac{2}{c}\left(B + R\sqrt{2(\gamma_{\bar{S}^{(k)}} + 1 + \log(1/p))}\right)(1+2\lambda)\sqrt{\frac{3|D_g|}{\bar{S}^{(k)}}}\end{aligned}$$

Notice that $f(x^*)$ lies in $[a_k, b_k]$ (under the high probability event on which the analysis is conditioned). Since $\tau_k = (a_k + b_k)/2$, therefore $|f(x^*) - \tau_k| \leq |b_k - a_k|/2$. Using this, we can write $R^{(1)}(N)$ as,

$$
\begin{aligned}
R^{(1)}(N) &= \sum_{n=1}^{N} f(x^*) - \tau_k + L\Delta_k^\alpha \\
&\leq \left(|f(x^*) - \tau_k| + L\Delta_k^\alpha\right) N \\
&\leq \left(\frac{|b_k - a_k|}{2} + c2^{-\alpha\rho_k/d}\right) N \\
&\leq \left((1 + 2c(\rho_k/d - 1))2^{-\alpha(\rho_k/d-1)-1} + c2^{-\alpha\rho_k/d}\right) N \\
&\leq \left((1 + 2c(\rho_k/d - 1))2^{-\alpha\rho_k/d} + c2^{-\alpha\rho_k/d}\right) N \\
&\leq \frac{2N}{c}(1 + c + 2c(\rho_k/d - 1)) \left(B + R\sqrt{2(\gamma_{\bar{S}^{(k)}} + 1 + \log(1/p))}\right)(1 + 2\lambda)\sqrt{\frac{3|D_g|}{\bar{S}^{(k)}}} \\
&\leq \frac{2N}{c}(1 + c + \frac{c}{\alpha}\log_2 T) \left(B + R\sqrt{2(\gamma_N + 1 + \log(1/p))}\right)(1 + 2\lambda)\sqrt{\frac{3|D_g|}{N}} \\
&\leq \frac{2}{c}\left(2 + \frac{c}{\alpha}\log_2 T\right) \left(B + R\sqrt{2(\gamma_N + 1 + \log(1/p))}\right)(1 + 2\lambda)\sqrt{3|D_g|N},
\end{aligned}
$$

where we use Lemma 6 in line 4, definition of k_0 in line 7 and the fact that $N \leq \bar{S}$. Lastly, we consider the second term, $R^{(2)}(N)$. Note that it is trivially upper bounded by zero if $f(x^*_{D_g}) > \tau_k - L\Delta_k^\alpha$. For the case when $f(x^*_{D_g}) < \tau_k - L\Delta_k^\alpha$, we analyze it like $R^{(1)}(N)$ with a different time instant instead of $\bar{S}^{(k)}(p)$. Define t_1 as

$$
t_1 = \min\left\{t \in \mathbb{N} : 2\left(B + R\sqrt{2(\gamma_{t-1} + 1 + \log(4T/\delta_0))}\right)(1 + 2\lambda)\sqrt{\frac{|D_g|}{t}} \leq \tau_k - L\Delta_k^\alpha - f(x^*_{D_g})\right\}.
$$

From Lemma 2, we know that $\Pr(N > t_1) \leq \frac{\delta_0}{4T}$. Therefore, with probability at least $1 - \frac{\delta_0}{4T}$, we have $N \leq t_1$. Conditioning on this event and using a similar sequence of arguments as used in proof of $R^{(1)}(N)$, we can write

$$
\tau_k - L\Delta_k^\alpha - f(x^*_{D_g}) \leq 2\left(B + R\sqrt{2(\gamma_{t_1} + 1 + \log(4T/\delta_0))}\right)(1 + 2\sigma)\sqrt{\frac{2|D_g|}{t_1}}.
$$

Thus with probability at least $1 - \frac{\delta_0}{4T}$, we have,

$$
\begin{aligned}
R^{(2)}(N) &= \sum_{n=1}^{N} \tau_k - L\Delta_k^\alpha - f(x^*_{D_g}) \\
&\leq \left(\tau_k - L\Delta_k^\alpha - f(x^*_{D_g})\right) N \\
&\leq 2N\left(B + R\sqrt{2(\gamma_{t_1} + 1 + \log(4T/\delta_0))}\right)(1 + 2\lambda)\sqrt{\frac{2|D_g|}{t_1}} \\
&\leq 2\left(B + R\sqrt{2(\gamma_N + 1 + \log(4T/\delta_0))}\right)(1 + 2\lambda)\sqrt{2|D_g|N}.
\end{aligned}
$$

On combining all the terms, we can conclude that $\bar{R}(N)$ is $O(\log T\sqrt{N(\gamma_N + \log(T/\delta_0))})$. To compute R_1, we just need to evaluate the total number of nodes visited by the algorithm in the first k_0 epochs. Using Lemma 3, we can conclude that with probability at least $1 - \delta_0/4T$, one iteration of random walk would have visited less than $\frac{1}{2(p-1/2)^2}\log\left(\frac{4dT}{\delta_0}\right)$ nodes. Therefore, throughout the algorithm, all iterations of random walks would have visited less than $\frac{1}{2(p-1/2)^2}\log\left(\frac{4dT}{\delta_0}\right)$ nodes with probability at least $1 - \delta_0/4$.

Let L_m denote the number of nodes at depth md of tree $\mathcal{T}_0$ for $m = 1, 2, \ldots, k_0$ that contain a point x such that $f(x) \geq \tau_m - c2^{-\alpha\rho_m/d+1}$. Therefore L_m denotes an upper bound on the number of target nodes for epoch m. Let $L_0 = \max_{1\leq i\leq k_0} L_i$. Using the upper bound on the number of nodes visited during on iteration of RWT, we can conclude that the algorithm would have visited less than $K = \dfrac{k_0 L_0}{(p-1/2)^2} \log\left(\dfrac{4dT}{\delta_0}\right)$ nodes in the first k_0 epochs with probability at least $1 - \delta_0/4$. To bound k_0, note that the update scheme of the interval $[a_k, b_k]$ (and consequently τ_k) ensures that the algorithm does not spend more than 2 epochs at any specific depth of the tree. This implies that $k \leq 2\rho_k/d$. Thus, $k_0 \leq \frac{1}{\alpha}\log_2 T$.

Let N_j denote the random number of queries at node j visited during the algorithm and $\bar{R}_j(N_j)$ denote the associated regret for $j = 1, 2, \ldots, K$. Therefore, for some constant C_0, independent of T, we have,

$$\begin{aligned}
R_1 &\leq \sum_{j=1}^{K} \bar{R}_j(N_j) \\
&\leq C_0 \log T \sum_{j=1}^{K} \sqrt{N_j(\gamma_{N_j} + \log(T/\delta_0))} \\
&\leq C_0 \log T \sqrt{\gamma_T + \log(T/\delta_0)} \sum_{j=1}^{K} \sqrt{N_j} \\
&\leq C_0 \log T \sqrt{\gamma_T + \log(T/\delta_0)} \cdot \sqrt{K \sum_{j=1}^{K} N_j} \\
&\leq C_0 \log T \sqrt{\gamma_T + \log(T/\delta_0)} \cdot \sqrt{KT} \\
&\leq C_0 \sqrt{\frac{T\log(T) L_0}{2(p-1/2)^2} \log\left(\frac{4dT}{\delta_0}\right)} \cdot \sqrt{\gamma_T + \log(T/\delta_0)} \cdot \log T.
\end{aligned}$$

Therefore, R_1 is $O(\sqrt{T\gamma_T}\log T\sqrt{\log T \cdot \log(T/\delta_0)})$.

We now focus on bounding R_2. Let D represent a node being visited after k_0 epochs and $x_D^* = \arg\max_{x\in D} f(x)$. The instantaneous regret at time instant t can be written as

$$\begin{aligned}
r_t &= f(x^*) - f(x_t) \\
&= [f(x^*) - f(x_D^*)] + [f(x_D^*) - f(x_t)].
\end{aligned}$$

We bound both the expressions, $f(x^*) - f(x_D^*)$ and $f(x_D^*) - f(x_t)$ separately for any such node. We begin with the second expression. After k_0 epochs, all the high-performing nodes being considered by the algorithm would be at a depth of at least $\frac{d}{2\alpha}\log_2 T$ in the original infinite binary tree. This implies that the length of the edges of the cuboid corresponding to the nodes would be smaller than $T^{-1/(2\alpha)}$. Consequently, no two points in any such node would be more than $\sqrt{d}T^{-1/(2\alpha)}$ apart. Therefore, $f(x_D^*) - f(x_t) \leq L\sqrt{d^\alpha/T}$, where x_t is a point sampled at time instant t after k_0 epochs have been completed. To bound the first expression, notice that $f(x_D^*) \in [a_{k_0+1}, b_{k_0+1}]$ for all nodes visited after k_0 epochs have been completed. This follows from the construction of intervals $[a_k, b_k]$. Since $f(x^*)$ also lies in $[a_{k_0+1}, b_{k_0+1}]$ (under the high probability event), we have, $f(x^*) - f(x_D^*) \leq |b_{k_0+1} - a_{k_0+1}| \leq (1 + 2c(\rho_{k_0+1}/d - 1))2^{-\alpha(\rho_{k_0+1}/d-1)}$ for any node visited after k_0 epochs. Therefore, we can bound the instantaneous regret as

$$\begin{aligned}
r_t &= [f(x^*) - f(x_D^*)] + [f(x_D^*) - f(x_t)] \\
&\leq (1 + 2c(\rho_{k_0+1}/d - 1))2^{-\alpha(\rho_{k_0+1}/d-1)} + L\sqrt{\frac{d^\alpha}{T}} \\
&\leq 2(2 + \frac{c}{2\alpha}\log_2 T)\sqrt{\frac{1}{T}} + L\sqrt{\frac{d^\alpha}{T}}.
\end{aligned}$$

If T_{R_2} denotes the samples taken by the algorithm after completing k_0 epochs, then R_2 can be bounded as

$$R_2 \leq \frac{T_{R_2}}{\sqrt{T}}\left(2\left(2 + \frac{c}{2\alpha}\log_2 T\right) + L\sqrt{d^\alpha}\right).$$

Noting that $T_{R_2} \leq T$, we have that R_2 is $O(\sqrt{T}\log T)$. On adding the bounds on R_1 and R_2, we obtain that the regret incurred by the algorithm is $O(\sqrt{T\gamma_T}\log T\sqrt{\log T \cdot \log(T/\delta_0)})$, as required.

We now show that this bound holds with high probability. Firstly, we had obtained a bound on $R^{(3)}(N)$ for a node $D \subseteq \mathcal{X}$ by conditioning on the event that $|f(x) - \mu_{t-1}(x)| \leq \beta_t(\delta_0/4T)\sigma_{t-1}(x)$ holds for all $x \in D$ and $t \geq 1$. Since the probability that event occurs is at least $1 - \delta_0/4T$, the bound on $R^{(3)}(N)$ holds simultaneously for all nodes visited throughout the time horizon with a probability of at least $1 - \delta_0/4$. Similarly, to obtain a bound on $R^{(2)}(N)$ for any node $D \subseteq \mathcal{X}$, we had conditioned the analysis on another event which holds with a probability of at least $1 - \delta_0/4T$. Therefore, the bound on $R^{(2)}(N)$ holds simultaneously for all nodes visited by the algorithm with a probability of at least $1 - \delta_0/4$. We also note that while using Lemma 3 to bound the number of nodes visited by the algorithm, we had conditioned the analysis on another event (that bounded the number of nodes visited in an iteration of RWT) that holds with a probability of at least $1 - \delta_0/4$ (See Sec. C.4). Lastly, since we assume that the algorithm always identifies all the target nodes correctly, we also need to account for the probability that this is true. From the error analysis of RWT as described in Section C.4, we note that every target node is identified correctly with a probability of at least $1 - \delta_0/4T$. Therefore, using a probability union bound, the algorithm identifies all the target nodes correctly with a probability of no less than $1 - \delta_0/4$. Combining all the above observations, we can conclude that the above obtained regret bound holds with a probability of at least $1 - \delta_0$, as required.

C.2 Proof of Lemma 1

We consider a domain $D \subseteq \mathcal{X}$ and its discretization D_g that contains $|D_g|$ number of points. Let the points be indexed from 1 to $|D_g|$ and let n_i denote the number of times the i^{th} point was chosen in the set of sampled points $\{x_1, x_2, \ldots, x_t\}$. Let $\mathcal{I} = \{i : n_i > 0\}$ and $|\mathcal{I}|$ denote the number of elements in $\mathcal{I}$. Consider the i^{th} point, denoted by $x^{(i)}$, and let $1 \leq t_1 < t_2 < \cdots < t_{n_i} \leq t$ denote the time instances when the i^{th} point is sampled, that is, at time t_j, it is sampled for the j^{th} time, for $j = 1, 2, \ldots, n_i$. Clearly, we have $\sigma_{t_1-1}(x_{t_1}) = \sigma_{t_1-1}(x^{(i)}) \leq k(x^{(i)}, x^{(i)}) \leq 1$. For all $2 \leq j \leq n_i$, at time instant t_j, $x^{(i)}$ has been sampled for $j-1$ times before t_j. Using Proposition 3 from [21], we have $\sigma_{t_j-1}(x_{t_j}) = \sigma_{t_j-1}(x^{(i)}) \leq \dfrac{\lambda}{\sqrt{j-1}}$. This can be interpreted as bounding the standard deviation by only the contribution coming from the noisy observations. We would like to emphasize that we are using the Proposition 3 for the *surrogate* GP-model adopted for the optimization. While the actual noise is indeed R-sub-Gaussian, we are applying the Proposition 3 bearing in mind the *fictitious* Gaussian noise assumption for our *surrogate* model.

Thus for each point in $\mathcal{I}$, the contribution to the sum is upper bounded by $1 + \lambda \sum_{j=1}^{n_i-1} j^{-1/2}$. Thus, we have,

$$
\begin{aligned}
\sum_{s=1}^{t} \sigma_{s-1}(x_s) &\leq \sum_{i \in \mathcal{I}} \left(1 + \lambda \sum_{j=1}^{n_i-1} \frac{1}{\sqrt{j}}\right) \\
&\leq \sum_{i \in \mathcal{I}} \left(1 + \lambda \int_0^{n_i-1} \frac{1}{\sqrt{z}}\, dz\right) \\
&\leq \sum_{i \in \mathcal{I}} \left(1 + 2\lambda\sqrt{n_i - 1}\right) \\
&\leq (1 + 2\lambda) \sum_{i \in \mathcal{I}} \sqrt{n_i} \\
&\leq (1 + 2\lambda)|\mathcal{I}| \sqrt{\frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} n_i} \\
&\leq (1 + 2\lambda)\sqrt{|\mathcal{I}|t}.
\end{aligned}
$$

In the fifth step, we have used Jensen's Inequality. Noting that $|\mathcal{I}| \leq |D_g|$, we obtain the required result.

C.3 Proof of Lemma 2

Consider the performance of the local test on a domain $D \subseteq \mathcal{X}$ with a threshold τ. The discretized version of the domain is denoted by D_g. As before, we use the following notation throughout the proof of this lemma,. Let $x^*_D = \arg\max_{x\in D} f(x)$, $x^*_{D_g} = \arg\max_{x\in D_g} f(x)$, $\hat{x}_t = \arg\max_{x\in D_g} \mu_{t-1}(x) + \beta_t(p)\sigma_{t-1}(x)$ and let $\bar{x}_t = \arg\max_{x\in D_g} \mu_{t-1}(x) - \beta_t(p)\sigma_{t-1}(x)$. Lastly, recall that the termination time is defined as

$$\bar{S}(\delta_2, \Delta_f) = \min\left\{t \in \mathbb{N} : 2\left(B + R\sqrt{2(\gamma_{t-1} + 1 + \log(1/\delta_2))}\right)(1+2\lambda)\sqrt{\frac{|D_g|}{t}} \leq \Delta_f\right\} + 1.$$

Let us consider the case when $f(x^*_{D_g}) < \tau - L\Delta^\alpha - \Delta_f$ and let N denote the random number of samples taken in a sequential test without a cap on the total number of samples. We first make the following observation about the posterior variance at the point to be sampled at t, x_t, and $\hat{x}_t$. From the definitions of x_t and $\hat{x}_t$, we have,

$$\begin{aligned}\mu_{t-1}(x_t) + \beta_t(\delta_0/4T)\sigma_{t-1}(x_t) &\geq \mu_{t-1}(\hat{x}_t) + \beta_t(\delta_0/4T)\sigma_{t-1}(\hat{x}_t)\\ \mu_{t-1}(\hat{x}_t) + \beta_t(p)\sigma_{t-1}(\hat{x}_t) &\geq \mu_{t-1}(x_t) + \beta_t(p)\sigma_{t-1}(x_t)\end{aligned}$$

On adding the two, we obtain that $\sigma_{t-1}(\hat{x}_t) \leq \sigma_{t-1}(x_t)$. Note that this holds for all t. Next, we define the event E as $|f(x) - \mu_{t-1}(x)| \leq \beta_t(\delta_2)\sigma_{t-1}(x)$ being true for all $x \in D$ and $t \geq 1$. From Theorem 2 in [10], we know that the probability of E is at least $1 - \delta_2$. Let E^c denote the complement of the event E. Using the event E, we evaluate the probability that the local test queries more than n points. The probability that $N > n$ can be written as follows,

$$\begin{aligned}\Pr(N > n) &\leq \Pr\left(\{\forall t \leq n : \mu_{t-1}(\hat{x}_t) + \beta_t(p)\sigma_{t-1}(\hat{x}_t) \geq \tau - L\Delta^\alpha\}\right)\\
&\leq \Pr\left(\{\forall t \leq n : \mu_{t-1}(\hat{x}_t) + \beta_t(p)\sigma_{t-1}(\hat{x}_t) \geq \tau - L\Delta^\alpha\}\,|E\right)\Pr(E) +\\
&\quad \Pr\left(\{\forall t \leq n : \mu_{t-1}(\hat{x}_t) + \beta_t(p)\sigma_{t-1}(\hat{x}_t) \geq \tau - L\Delta^\alpha\}\,|E^c\right)\Pr(E^c)\\
&\leq \Pr\left(\sum_{t=1}^{n} \mu_{t-1}(\hat{x}_t) + \beta_t(p)\sigma_{t-1}(\hat{x}_t) \geq \sum_{t=1}^{n}(\tau - L\Delta^\alpha)\middle| E\right) + \Pr(E^c)\\
&\leq \Pr\left(\sum_{t=1}^{n} f(\hat{x}_t) + \beta_t(\delta_2)\sigma_{t-1}(\hat{x}_t) + \beta_t(p)\sigma_{t-1}(\hat{x}_t) \geq \sum_{t=1}^{n}(\tau - L\Delta^\alpha)\middle| E\right) + \delta_2\\
&\leq \Pr\left(\sum_{t=1}^{n} f(x^*_{D_g}) + 2\beta_t(\delta_2)\sigma_{t-1}(\hat{x}_t) \geq \sum_{t=1}^{n}(\tau - L\Delta^\alpha)\middle| E\right) + \delta_2\\
&\leq \Pr\left(\sum_{t=1}^{n} 2\beta_t(\delta_2)\sigma_{t-1}(x_t) \geq \sum_{t=1}^{n}(\tau - f(x^*_{D_g}) - L\Delta^\alpha)\middle| E\right) + \delta_2\end{aligned}$$

To bound the first term on the RHS, we make use Lemma 1.
Therefore, we have

$$\begin{aligned}\frac{1}{n}\sum_{t=1}^{n} 2\beta_t(\delta_2)\sigma_{t-1}(x_t) &\leq \frac{2\beta_n(\delta_2)}{n}\sum_{t=1}^{n}\sigma_{t-1}(x_t)\\
&\leq \frac{2\beta_n(\delta_2)}{n}(1+2\lambda)\sqrt{|D_g|n}\\
&\leq 2\beta_n(\delta_2)(1+2\lambda)\sqrt{\frac{|D_g|}{n}}\\
&\leq \Delta_f < \tau - f(x^*_{D_g}) - L\Delta^\alpha.\end{aligned}$$

This implies that the first term on RHS goes to zero for $n \geq \bar{S} - 1$ implying that the probability that the local test takes more than $\bar{S}$ samples when $f(x^*_{D_g}) < \tau - L\Delta^\alpha - \Delta_f$ is less than δ_2. This implies if the local test has reached the termination condition then with probability atleast $1 - \delta_2$, we have that $f(x^*_{D_g}) > \tau - L\Delta^\alpha - \Delta_f$. We can carry out a similar analysis for the case when $f(x^*_{D_g}) > \tau + \Delta_f$ to obtain the statement of the lemma.

C.4 Proof of Lemma 3

The proof of this lemma is mainly based on the analysis of random walk on a binary tree. This analysis is similar to the one described in [42]. We reproduce a slightly different version of the proof that is more focused on finding a high probability bound on the number of nodes visited in the random walk. In this proof, we consider a binary tree of depth d, denoted by $\hat{\mathcal{T}}$, to represent the tree considered in the random walk. We index the leaf nodes from 1 to n where $n = 2^d$. Throughout this proof, we refer to the high-performing nodes as target nodes. We begin with the case of a single target and then extend the proof for the case of multiple targets.

WLOG, we consider the single target node to be the leaf node indexed as 1. We divide the tree $\hat{\mathcal{T}}$ into a sequence of sub-trees denoted by $\hat{\mathcal{T}}_0, \hat{\mathcal{T}}_1, \dots$ for $i = 0, 1, 2, \dots d$ which are defined as follows. Consider the nodes on the path joining the root node to the target node. Such a path is unique as the underlying graph is a tree. Let v_i denote the node on this path that is at a distance of i from the target node. The distance between two nodes is defined as the length of the path connecting those two nodes. $\hat{\mathcal{T}}_i$ is defined to be tree that contains the node v_i along with the sub-tree rooted at the child that does not contain the target node.

This construction is similar to the one outlined in [42]. Also, $\hat{\mathcal{T}}_0$ corresponds to the target node. Since the random walk is biased towards the minimizer, given the construction of $\hat{\mathcal{T}}_i$, the probability that random walk is still in one of such sub-trees would decrease with time. To formalize this idea, we consider the last passage times of any sub-tree $\hat{\mathcal{T}}_i$ for $1 \leq i \leq d$. Let τ_i denote the last passage time to $\hat{\mathcal{T}}_i$.

We begin with the analysis for τ_d. This problem of random walk on $\hat{\mathcal{T}}_d$ can be mapped to the problem of a random walk on the set $S = \{-1, 0, 1, 2, \dots d\}$. If each non-negative integer is mapped to the subset of nodes at the corresponding depth in the sub-tree, then our random walk on $\mathcal{T}_d$ between different levels is equivalent to the random walk on these integers. Note that since the target node is not contained in this sub-tree, all nodes at the same depth are identical in terms of distance to the target node. In particular, they all are equally far away from exiting the tree and therefore can be abstracted into single node. This abstraction is precisely what leads to the equivalence between the two problems. Under this setup, the root node is mapped to 0 and the sub-tree containing the target node is mapped to -1, indicating an exit from the sub-tree $\hat{\mathcal{T}}_d$.

We begin the random walk at integer 0 where escaping the tree is equivalent to arriving on the integer -1. For the random walk to arrive on -1, it would have to take greater number of steps in the negative direction than it took in the positive one. Also, since the probability of moving along the negative direction is at least $1 - p$, we can write,

$$\mathbb{P}(\tau_d > n) \leq \mathbb{P}(Z \leq n/2),$$

where $Z \sim \text{Bin}(n, p)$ is a Binomial random variable. Therefore, we have

$$\mathbb{P}(\tau_d > n) \leq \exp(-2(p - 1/2)^2 n).$$

On account of the underlying symmetry, we can conclude that this bound holds for all i. Therefore, we have $\mathbb{P}(\tau_i > n) \leq \exp(-2(p - 1/2)^2 n)$ for all $i = 0, 1, \dots, d$.

For the case of multiple target nodes, we can construct a similar set of sub-graphs and conclude the same result for those sub-graphs. Note that we redefine these set for every different iteration of the random walk when it restarts after detecting a target node. Consider the case when there are L target nodes. We begin with considering the first iteration of the random walk. For each target node $j = 1, 2, \dots, L$, we define a sequence of sub-trees $\mathcal{T}_i^{(j)}$ for $i = \{0, 1, \dots, d\}$ exactly in the same manner as we did in the previous case. That is, $\mathcal{T}_i^{(j)}$ would be a tree consisting of the node that lies on the path between the target node j and the root node and is at a distance of i from the target node, along with child that does not contain the target node j. By definition, the sub-trees $\mathcal{T}_i^{(j)}$ are not disjoint for different values of j. Using these sub-trees, we define a partition of the binary tree denoted by the sub-graphs $\hat{\mathcal{T}}_i'$ for $i = \{0, 1, \dots, d\}$ as follows. If $\mathcal{V}$ denotes the set of all nodes on the binary tree, then for each $v \in \mathcal{V}$, we define $v(j) = \{i : v \in \mathcal{T}_i^{(j)}\}$. Therefore, $v(j)$ denotes the index of the sub-tree corresponding to the target node j to which the node v belongs. From the construction of $\mathcal{T}_i^{(j)}$, it follows that $v(j)$ is unique

for each $v \in \mathcal{V}$. Using this, we define

$$\hat{\mathcal{T}}_i' = \{v \in \mathcal{V} : \min_j v(j) = i\}$$

In other words, $\hat{\mathcal{T}}_i'$ consists of all the nodes such that there is at least one target node j for which it belongs to $\mathcal{T}_i^{(j)}$.

The motivation is that if the random walk escapes $\hat{\mathcal{T}}_i'$ in the correct direction then it has moved closer to at least one of the target nodes. It is not difficult to note that this is exactly how the sub-trees $\hat{\mathcal{T}}_i$ were designed in the previous proof. The only difference between the two cases is that $\hat{\mathcal{T}}_i'$ is designed to accommodate the presence of multiple target nodes where all the target nodes have the same level of preference for the random walk. In a similar vein to the case of a single target, we define τ_i' as the last passage time to $\hat{\mathcal{T}}_i'$ for $i = \{0, 1, \ldots, d\}$.

Leveraging the similarity of definitions of $\hat{\mathcal{T}}_i'$ and $\hat{\mathcal{T}}_i$ along with the agnosticism of the random walk to the target node, we can use exactly the same analysis as for the single target case to conclude that

$$\mathbb{P}(\tau_i' > n) \leq \exp(-2(p - 1/2)^2 n).$$

We let M denote the random number of steps taken by one iteration of random walk before termination. Therefore, we can write,

$$\begin{aligned}
\Pr(M > r) &\leq \Pr\left(\bigcup_{i=0}^{d} \{\tau_i' > r\}\right) \\
&\leq \sum_{i=0}^{d} \Pr\left(\tau_i' > r\right) \\
&\leq \sum_{i=0}^{d} \exp(-2(p - 1/2)^2 r) \\
&\leq d \exp(-2(p - 1/2)^2 r)
\end{aligned}$$

Using the above relation, we can conclude that one iteration of the random walk will take less than $\frac{1}{2(p-1/2)^2} \log\left(\frac{d}{\delta_1}\right)$ with probability at least $1 - \delta_1$, as required.

This also helps bound the probability of error in the random walk. If M_1 denotes the number of non-target leaf nodes visited in the random walk, then the probability that a target node is identified incorrectly is less than $M_1 \delta_2$, where δ_2 is the error probability for the leaf test. Using the above bound on M_1 with $\delta_1 = \delta_0/4T$ along with the value of δ_2 as specified by the algorithm (See Appendix B.2), we conclude that an iteration of random walk identifies a target node correctly with probability at least $1 - \delta_0/4T$.

C.5 Proof of Lemma 4

A key idea in the proof of the lemma is to establish that for the choice of parameters used in GP-ThreDS, the rate at which domain shrinks matches the rate at which the discretization gets finer. Let D be a node visited by the algorithm during epoch k and D_g be its associated discretization such that

$$\sup_{x \in D} \inf_{y \in D_g} \|x - y\| \leq \Delta_k.$$

More specifically, D refers to the subset of the domain corresponding to the node visited by the algorithm. Note that using the definition of a covering set, we can conclude that D_g is a Δ_k-cover of D. Then, using the bounds on the covering number of a hypercube in $\mathbb{R}^d$ [43], we have that $|D_g|$ is $O(\text{vol}(D)\Delta_k^{-d})$. Since D is a node visited during epoch k of the algorithm, it lies at a depth of at least $\rho_k - d$ on the infinite depth binary tree constructed on the domain. From the construction of the binary tree, we note that the lengths of nodes in all the dimensions get halved every d steps. Thus, the lengths of the edges of the cuboid corresponding to D are less than $2^{-\rho_k/d+1}$. Consequently, vol(D) is $O(2^{-\rho_k})$. On substituting this value in the bound for $|D_g|$ along with $\Delta_k = (c/L)^{1/\alpha} 2^{-\rho_k/d}$, we obtain $|D_g|$ is $O(1)$, independent of

k (and thus t). The exponential dependence of $|D_g|$ on d also immediately follows from the above analysis.

As mentioned in the main text, the proof of Theorem 2 follows from this lemma. Since only a constant number of UCB scores have to be evaluated at every time instant, the matrix inversion becomes the dominant cost resulting in a worst-case computational cost of $O(t^3)$ at time t. Consequently, this results in worst-case overall computational complexity of $O(T^4)$.

C.6 Proof of Lemma 5

For the analysis of the local test, we consider several cases based on the maximum value of the function on the grid and consider the results obtained in each one of them.

We consider the performance of the local test on a node corresponding to $D \subseteq \mathcal{X}$ visited by the random walk during epoch k. The discretized version of the domain is denoted by D_g. Recall that during epoch k, the closest point in D_g from any point $x \in D$ is at a distance less than Δ_k. We define $x^*_D, x^*_{D_g}, \hat{x}_t$ and $\bar{x}_t$ in the same way as in the proof of Lemma 2 (Appendix C.3).
The cap on the number of samples in epoch k is given as

$$\bar{S}^{(k)}(p) = \min\left\{t \in \mathbb{N} : 2\left(B + R\sqrt{2(\gamma_{t-1} + 1 + \log(1/p))}\right)(1+2\lambda)\sqrt{\frac{|D_g|}{t}} \leq L\Delta_k^\alpha\right\} + 1.$$

Similar to the proof of Lemma 2 (Appendix C.3), we define the event E as the inequality $|f(x) - \mu_{t-1}(x)| \leq \beta_t(p)\sigma_{t-1}(x)$ being true for all $x \in D$ and $t \geq 1$. We know this event occurs with a probability of at least $1-p$. For the following analysis, we assume that event E occurs. Consider the following scenarios based on the value of $f(x^*_{D_g})$.

- $f(x^*_{D_g}) > \tau_k + L\Delta_k^\alpha$:
  From the results obtained in the proof of Lemma 2, we know that the local test will not terminate. Also notice that the local test cannot return -1 as

  $$\begin{aligned}\mu_{t-1}(\hat{x}_t) + \beta_t(p)\sigma_{t-1}(\hat{x}_t) &\geq \mu_{t-1}(x^*_{D_g}) + \beta_t(p)\sigma_{t-1}(x^*_{D_g}) \\ &\geq f(x^*_{D_g}) \\ &> \tau_k - L\Delta_k^\alpha.\end{aligned}$$

  Therefore, the local test will always return $+1$.

- $\tau_k + L\Delta_k^\alpha \geq f(x^*_{D_g}) \geq \tau_k$:
  Similar to the previous case, we can conclude that the local test will never return -1. It may return $+1$ or terminate.

- $\tau_k > f(x^*_{D_g}) > \tau_k - L\Delta_k^\alpha$:
  Again, similar to the previous cases, the local test will never return -1. For this case, we also have,

  $$\begin{aligned}\mu_{t-1}(\bar{x}_t) - \beta_t(p)\sigma_{t-1}(\bar{x}_t) &\leq f(\bar{x}_t) \\ &\leq f(x^*_{D_g}) \\ &< \tau_k.\end{aligned}$$

  Therefore, the local test will also never return $+1$ (before termination) implying it will always terminate.

- $\tau_k - L\Delta_k^\alpha \geq f(x^*_{D_g}) \geq \tau_k - 2L\Delta_k^\alpha$:
  Similarly, the local test will not return $+1$ (before termination). It may return -1 or terminate.

- $\tau_k - 2L\Delta_k^\alpha > f(x^*_{D_g})$:
  From the results obtained in Sec. C.3 we can show the local will neither terminate nor return $+1$, implying that the local test will always return -1.

From the above analysis, one can directly obtain the statement of the lemma. If D is high-performing with respect to the threshold τ_k, then $f(x^*_D) > \tau_k$ implying that $f(x^*_{D_g}) > \tau_k - L\Delta_k^\alpha$. If $f(x^*_{D_g}) > \tau_k - L\Delta_k^\alpha$, then the local test will output $+1$ whenever event E occurs, i.e., with a probability of at least $1-p$.

Similarly, if the local test outputs -1 when E has occurred, we know that $f(x^*_{D_g}) < \tau_k - L\Delta_k^\alpha$, implying $f(x^*_D) < \tau_k$ and hence D is not high-performing w.r.t. the threshold τ_k, as required.

However, we would like to point out that when $\tau_k - L\Delta_k^\alpha > f(x^*_{D_g}) > \tau_k - 2L\Delta_k^\alpha$, the test may output -1 or terminate, in which case it returns a $+1$ and accept the current node. This happens because of the conservative nature of the local test. In order to avoid missing nodes that contain a point with a function value greater than τ_k, the local test sometimes accepts nodes nodes like these which have a point with a function value greater than $\tau_k - c2^{-\alpha\rho_k/d+1}$ but not greater than τ_k. This explains the reason behind the particular choice of values used in the update policy of τ_k.

C.7 Proof of Lemma 6

We prove the statement of the lemma using induction. Recall that $[a_k, b_k]$ denotes the interval to which $f(x^*)$ is likely to belong at the beginning of epoch k. For the base case, for the LHS we have $|b_1 - a_1| = |b - a| = 1$. Since $\rho_1 = d$, the RHS also evaluates to 1 verifying the base case. Let us assume that the relation $|b_k - a_k| \leq (1 + 2c(\rho_k - d)/d)2^{-\alpha(\rho_k/d-1)}$ is true for some $k \geq 1$.
In the event that the algorithm does not find any τ_k-exceeding point, we set $a_{k+1} = a_k - (b_k - a_k)/2$, $b_{k+1} = b_k - (b_k - a_k)/2$ and $\rho_{k+1} = \rho_k$. Thus, we have,

$$\begin{aligned}
|b_{k+1} - a_{k+1}| &= \left| b_k - \frac{b_k - a_k}{2} - a_k + \frac{b_k - a_k}{2} \right| \\
&= |b_k - a_k| \\
&\leq (1 + 2c(\rho_k - d)/d)2^{-\alpha(\rho_k/d-1)} \\
&\leq (1 + 2c(\rho_{k+1} - d)/d)2^{-\alpha(\rho_{k+1}/d-1)},
\end{aligned}$$

as required. Now, if the algorithm finds a τ_k-exceeding point, we set $a_{k+1} = \tau_k - c2^{-\alpha\rho_k/d+1}$, $b_{k+1} = b_k$ and $\rho_{k+1} = \rho_k + d$. Thus, we have,

$$\begin{aligned}
|b_{k+1} - a_{k+1}| &= |b_k - \tau_k + c2^{-\alpha\rho_k/d+1}| \\
&\leq \frac{1}{2}|b_k - a_k| + c2^{-\alpha\rho_k/d+1} \\
&\leq \frac{1}{2}\left(1 + \frac{2c(\rho_k - d)}{d}\right)2^{-\alpha(\rho_k/d-1)} + c2^{-\alpha\rho_k/d+1} \\
&\leq \left(1 + \frac{2c(\rho_k - d)}{d}\right)2^{-\alpha\rho_k/d} + 2c2^{-\alpha\rho_k/d} \\
&\leq \left(1 + 2c\left(\frac{(\rho_k + d) - d}{d}\right)\right)2^{-\alpha(\rho_{k+1}/d-1)} \\
&\leq \left(1 + 2c\left(\frac{\rho_{k+1} - d}{d}\right)\right)2^{-\alpha(\rho_{k+1}/d-1)} \\
&\leq \left(1 + \frac{2c(\rho_{k+1} - d)}{d}\right)2^{-\alpha(\rho_{k+1}/d-1)},
\end{aligned}$$

as required. This completes the proof.

D Supplemental Material on Experiments

In this section, we provide further details on the experiments, as well as additional experiments on a hyperparameter tuning problem.

D.1 Details of Benchmark Functions, Algorithms and their Parameters

We used two standard benchmark functions, Branin and Rosenbrock in our experiments. The analytical expression for these functions is given below [44, 45]

- *Branin*: $f(x, y) = -\frac{1}{51.95}\left(\left(v - \frac{5.1u^2}{4\pi^2} + \frac{5u}{\pi} - 6\right)^2 + \left(10 - \frac{10}{8\pi}\right)\cos(u) - 44.81\right)$, where $u = 15x - 5$ and $v = 15y$.

- *Rosenbrock*: $f(x, y) = 10 - 100(v - u)^2 - (1 - u)^2$, where $u = 0.3x + 0.8$ and $v = 0.3y + 0.8$.

The domain is set to $\mathcal{X} = [0, 1]^2$. The details of IGP-UCB, AD, PI and EI are provided below.

1. IGP-UCB: The algorithm is implemented exactly as outlined in [10] with B (in scaling parameter β_t) set to 0.5 and 2 for Branin and Rosenbrock, respectively. The parameters R and δ_0 are set to 10^{-2} and 10^{-3} in both experiments. γ_t was set to $\log t$. The size of discretization is increased over time, starting from 400 points at the beginning and capped at 6400 points.

2. Adapative Discretization (AD): The algorithm and its parameters are implemented exactly as described in [21].

3. Expected Improvement(EI)/Probability of Improvement (PI): Similar to IGP-UCB, EI and PI select the observation points based on maximizing an index often referred to as an acquisition function. The acquisition function of EI is $(\mu(x) - f^* - \varepsilon)\Phi(z) + \sigma(x)\phi(z)$, where $z = \frac{\mu(x) - f^* - \varepsilon}{\sigma(x)}$. The acquisition function of PI is $\Phi(z)$, where $z = \frac{\mu(x) - f^* - \xi}{\sigma(x)}$. Here, $\Phi(\cdot)$ and $\phi(\cdot)$ denote the CDF and PDF of a standard normal random variable. f^* is set to be the maximum value of $\mu(x)$ among the current observation points. The parameters ε and ξ are used to balance the exploration-exploitation trade-off. We follow [38] that showed the best choice of these parameters are small non-zero values. In particular, ε and ξ are both set to 0.01.

4. GP-ThreDS: A pseudo-code is given in Alg. 4. The parameter β_t is set exactly in the same way as in IGP-UCB. The initial interval $[a, b]$ is set to $[0.5, 1.2]$ for Branin and $[3, 12]$ for Rosenbrock. The parameter c is set to 0.2 for both functions.

D.2 Additional Experiments

We have re plotted the average regret against wall clock time for different algorithms on benchmark functions (the same as Fig. 4 in the main paper), with error bars in Fig. 2 and Fig. 3.

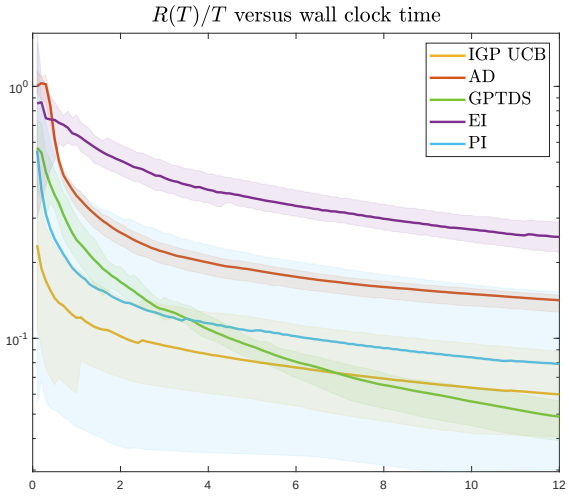


Figure 2: Average regret against wall clock time for Branin

D.3 Hyperparameter Tuning for a convolutional neural network

In this section, we provide additional experiments on using the Bayesian optimization algorithms for hyperparameter tuning for a convolutional neural network (CNN) on an image classification task. We have considered the task of digit classification on the MNIST dataset. For the experiments, we have considered a smaller training dataset that contains only 12000 images instead of 50000 images in the original dataset. This smaller training dataset is created by randomly sampling 1200 images corresponding to each digit, making a total of 12000 images. This dataset is split to training and validation sets of size 10000 and 2000, respectively. The split is done in a way that each label has equal numbers of images in the training

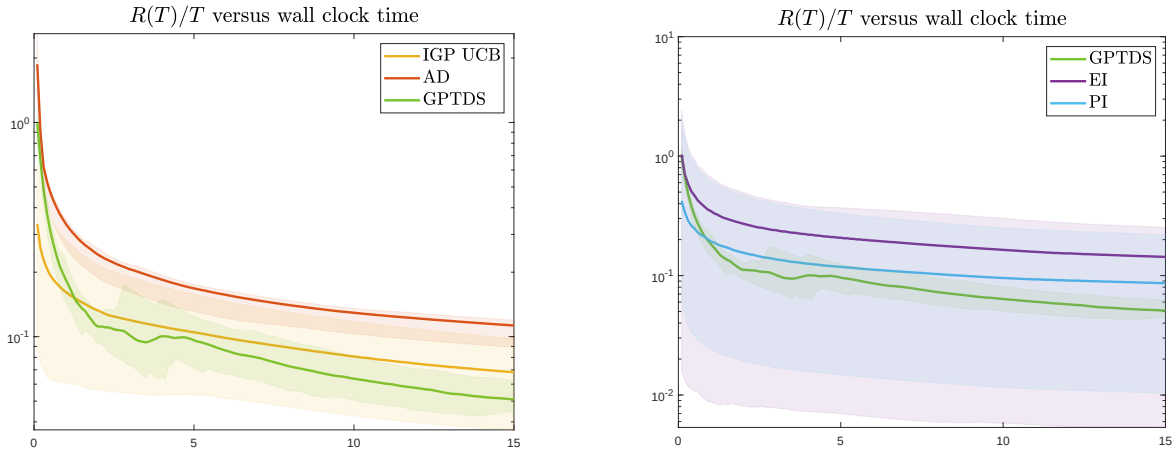


Figure 3: Average regret against wall clock time for Rosenbrock

and the validation set. We used the same test set of 10000 images as in the original MNIST data set.

We consider a simple CNN with 2 convolutional layers followed by two fully connected feedforward layers. We use the ReLU activation function and a max pooling layer with stride 2 after each convolutional layer. The performance of the algorithms is evaluated on the task of tuning the following five hyperparameters of this CNN.

- Batch size: We considered 8 possible values of the batch sizes given by $\{2^3, 2^4, \ldots, 2^{10}\}$.
- Kernel size of the first convolutional layer with possible values in $\{3, 5, 7, 9\}$.
- Kernel size of the second convolutional layer with possible values in $\{3, 5, 7, 9\}$.
- Number of (hidden) nodes in the first feedforward layer: The possible values for this hyperparameter belonged to $\{10, 11, 12, \ldots, 38, 39, 40\}$.
- Initial learning rate: We used stochastic gradient descent with momentum to optimize the loss function. This parameter defined the initial learning rate for the optimizer and it took values in $\{10^{-6}, 10^{-5}, \ldots, 10^{-1}\}$.

In the implementation, all these parameters were mapped to $[0, 1]$ with distinct intervals corresponding to each discrete value. The kernel sizes and the number of hidden nodes were mapped linearly to the interval while the other two parameters were mapped on a log scale, that is, $\log_2$(batch-size) and $\log_{10}$(learning-rate) were mapped uniformly to the interval $[0, 1]$.

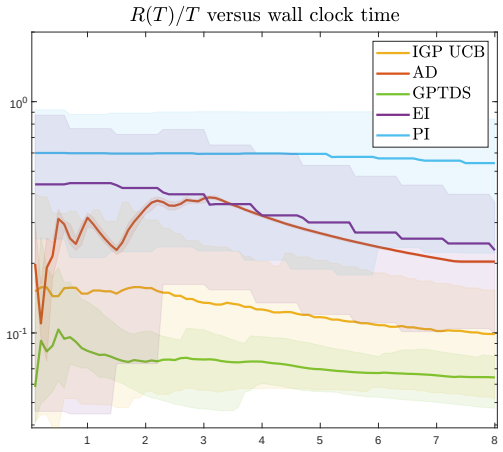
For this task, the underlying function was modelled using a Matérn kernel with smoothness parameter 2.5 and lengthscale $l = 0.2$. For this kernel γ_t was set to $\sqrt{t}$ and the noise variance was set to 0.0001. The implementation of all the algorithms is similar to the description in Sec. D.1.

For the exploration parameter β_t, B and R are set to 0.5 and 10^{-4}, respectively, for both IGP-UCB and GP-ThreDS. The confidence parameter δ_0 is set to 2.5×10^{-2} for IGP-UCB and 2×10^{-2} for GP-ThreDS. The slightly higher confidence for GP-ThreDS is chosen because it runs for a longer horizon to achieve the same compute time. In GP-ThreDS, the interval $[a, b]$ is set to $[0.3, 1.4]$ and c is set to 0.1.

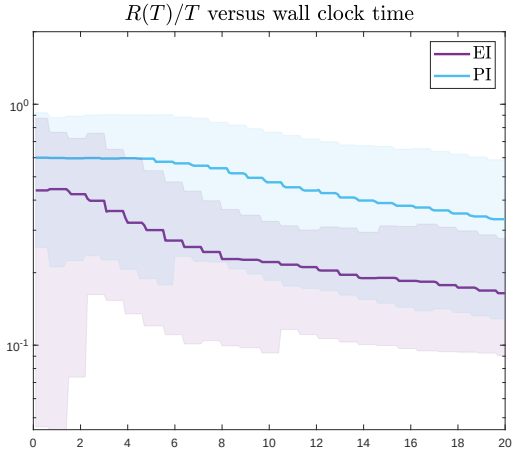
All the computations were carried out using MATLAB 2019a on a computer with 12 GB RAM and Intel i7 processor (3.4 GHz) with an overall compute time of around 200 hours.

The results for average regret against wall clock time, averaged over 10 Monte Carlo runs, are shown in Fig. 4a. As it can be seen from the figure, GP-ThreDS offers a better performance compared to all other algorithms. It is important to point out that the time on X-axis does not include the time spent

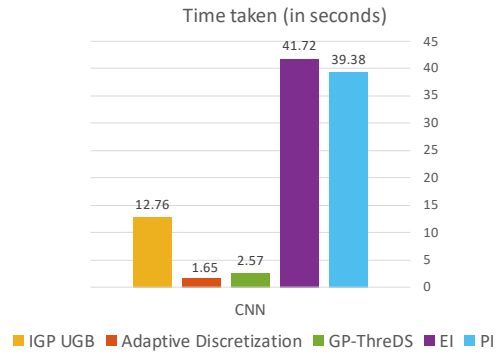
to train the CNN, it only includes the time spent on Bayesian optimization algorithms assuming the values of the objective function f (here the performance of CNN) are accessible in zero time. To be clear we refer to this time as *Optimization time*. The optimization time spent by different algorithms for $T = 50$ is shown in Fig. 4c. We also report the corresponding *total time* contrasting the optimization time that also includes the time spent to train the CNN, in Fig. 4d. It can be seen that the total time for GP-ThreDS is significantly lower than all other algorithms. Its optimization time however is comparable to AD, while significantly lower than IGP-UCB, EI, PI. It seems that, due to its exploration scheme, AD selects hyperparameters which lead to longer training times. PI and EI in comparison seem to take longer to converge as shown in Fig. 4b.



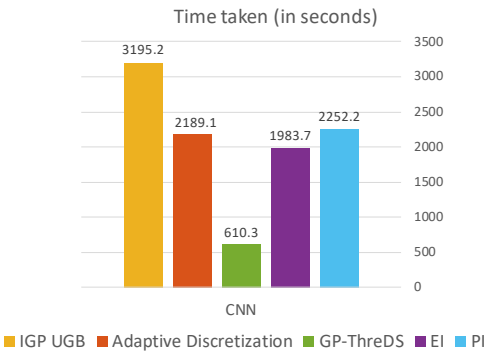
(a) Average regret against wall clock time for CNN.



(b) Average regret against wall clock time for EI and PI with a longer time horizon



(c) Optimization time spent by different algorithms for $T = 50$ samples



(d) Total time spent by different algorithms for $T = 50$ samples

References

[1] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, pages 1015–1022, 2010.

[2] Emile Contal and Nicolas Vayatis. Stochastic Process Bandits: Upper Confidence Bounds Algorithms via Generic Chaining. feb 2016.

[3] Andreas Krause and Cheng Soon Ong. Contextual Gaussian process bandit optimization. In *25th Annual Conference on Neural Information Processing Systems 2011, NIPS 2011*, pages 2447–2455, 2011.

[4] Michal Valko, Nathan Korda, Rémi Munos, Ilias Flaounas, and Nello Cristianini. Finite-time analysis of kernelised contextual bandits. In *Uncertainty in Artificial Intelligence - Proceedings of the 29th Conference, UAI 2013*, pages 654–663, 2013.

[5] Thomas Desautels, Andreas Krause, and Joel Burdick. Parallelizing exploration-exploitation tradeoffs with Gaussian process bandit optimization. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, volume 2, pages 1191–1198, 2012.

[6] Emile Contal, David Buffoni, Alexandre Robicquet, and Nicolas Vayatis. Parallel gaussian process optimization with upper confidence bound and pure exploration. In Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 225–240, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[7] Victor Picheny, Sattar Vakili, and Artem Artemev. Ordinal bayesian optimisation. *arXiv preprint arXiv:1912.02493*, 2019.

[8] Ilija Bogunovic, Stefanie Jegelka, Jonathan Scarlett, and Volkan Cevher. Adversarially robust optimization with Gaussian processes. In *Advances in Neural Information Processing Systems*, volume 2018-Decem, pages 5760–5770, 2018.

[9] Kirthevasan Kandasamy, Gautam Dasarathy, Junier Oliva, Jeff Schneider, and Barnabás Póczos. Multi-fidelity Gaussian process bandit optimisation. *Journal of Artificial Intelligence Research*, 66:151–196, 2019.

[10] Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. In *34th International Conference on Machine Learning, ICML 2017*, volume 2, pages 1397–1422, 2017.

[11] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *25th Annual Conference on Neural Information Processing Systems 2011, NIPS 2011*, 2011.

[12] Daniele Calandriello, Luigi Carratino, Alessandro Lazaric, Michal Valko, Lorenzo Rosasco, Lorenzo Rosasco Gaussian, Michal Valko MICHALVALKO, Alina Beygelzimer, and Daniel Hsu. Gaussian process optimization with adaptive sketching: Scalable and no regret. Technical report, 2019.

[13] Michalis K Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Journal of Machine Learning Research*, volume 5, pages 567–574, 2009.

[14] James Hensman, Nicolò Fusi, and Neil D. Lawrence. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence - Proceedings of the 29th Conference, UAI 2013*, pages 282–290, sep 2013.

[15] Sattar Vakili, Victor Picheny, and Artem Artemev. Scalable Thompson Sampling using Sparse Gaussian Process Models. 2020.

[16] Jonathan H Huggins, Trevor Campbell, Mikołaj Kasprzak, and Tamara Broderick. Scalable Gaussian process inference with finite-data mean and variance guarantees. In *AISTATS 2019 - 22nd International Conference on Artificial Intelligence and Statistics*, 2020.

[17] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20 - Proceedings of the 2007 Conference*, 2009.

[18] Ilja Kuzborskij, Leonardo Cella, and Nicolò Cesa-Bianchi. Efficient linear bandits through matrix sketching. In *AISTATS 2019 - 22nd International Conference on Artificial Intelligence and Statistics*, 2020.

[19] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional Bayesian Optimisation and bandits via additive models. In *32nd International Conference on Machine Learning, ICML 2015*, volume 1, pages 295–304, 2015.

[20] Mojmír Mutný and Andreas Krause. Efficient high dimensional Bayesian optimization with additivity and quadrature fourier features. In *Advances in Neural Information Processing Systems*, volume 2018-Decem, pages 9005–9016, 2018.

[21] Shubhanshu Shekhar and Tara Javidi. Gaussian process bandits with adaptive discretization. *Electronic Journal of Statistics*, 12(2):3829–3874, 2018.

[22] Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. X-Armed Bandits. *Journal of Machine Learning Research*, 12:1655–1695, 2011.

[23] Ziyu Wang, Babak Shakibi, Lin Jin, and Nando De Freitas. Bayesian multi-scale optimistic optimization. In *Journal of Machine Learning Research*, volume 33, pages 1005–1014, 2014.

[24] Rémi Munos. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *25th Annual Conference on Neural Information Processing Systems 2011, NIPS 2011*, 2011.

[25] Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 681–690, sep 2008.

[26] Linnan Wang, Rodrigo Fonseca, and Yuandong Tian. Learning Search Space Partition for Black-box Optimization using Monte Carlo Tree Search, jul 2020.

[27] Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabás Póczos. Parallelised Bayesian optimisation via Thompson sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 133–142, 2018.

[28] Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, nov 2014.

[29] D. Russo and B. Van Roy. An information-theoretic analysis of Thompson sampling. *The Journal of Machine Learning Research*, 17(1), 2016.

[30] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient Bayesian optimization. In *34th International Conference on Machine Learning, ICML 2017*, volume 7, pages 5530–5543, 2017.

[31] Jonathan Scarlett. Tight regret bounds for Bayesian optimization in one dimension. *arXiv preprint arXiv:1805.11792*, 2018.

[32] Shubhanshu Shekhar and Tara Javidi. Significance of gradient information in bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 2836–2844. PMLR, 2021.

[33] Adam D Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12:2879–2904, 2011.

[34] Sattar Vakili, Victor Picheny, and Nicolas Durrande. Regret bounds for noise-free bayesian optimization. *arXiv preprint arXiv:2002.05096*, 2020.

[35] Nando De Freitas, Alex J Smola, and Masrour Zoghi. Exponential regret bounds for Gaussian process bandits with deterministic observations. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, volume 2, pages 1743–1750, 2012.

[36] Kenji Kawaguchi, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Bayesian optimization with exponential convergence. In *Advances in Neural Information Processing Systems*, volume 2015-Janua, pages 2809–2817, 2015.

[37] Steffen Grünewälder, Jean Yves Audibert, Manfred Opper, and John Shawe-Taylor. Regret bounds for Gaussian process bandit problems. In *Journal of Machine Learning Research*, volume 9, pages 273–280, 2010.

[38] Matthew D Hoffman, Eric Brochu, and Nando de Freitas. Portfolio allocation for bayesian optimization. In *UAI*, pages 327–336. Citeseer, 2011.

[39] Zi Wang, Beomjoon Kim, and Leslie Pack Kaelbling. Regret bounds for meta bayesian optimization with an unknown gaussian process prior. *arXiv preprint arXiv:1811.09558*, 2018.

[40] Vu Nguyen, Sunil Gupta, Santu Rana, Cheng Li, and Svetha Venkatesh. Regret for expected improvement over the best-observed value and stopping condition. In *Asian Conference on Machine Learning*, pages 279–294. PMLR, 2017.

[41] Ziyu Wang and Nando de Freitas. Theoretical analysis of bayesian optimisation with unknown gaussian process hyper-parameters. *arXiv preprint arXiv:1406.7758*, 2014.

[42] Chao Wang, Qing Zhao, and Kobi Cohen. Dynamic Search on a Tree with Information-Directed Random Walk. In *IEEE Workshop on Signal Processing Advances in Wireless Communications, SPAWC*, volume 2018-June, pages 1–5. IEEE, jun 2018.

[43] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[44] Javad Azimi, Ali Jalali, and Xiaoli Z Fern. Hybrid batch bayesian optimization. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, volume 2, pages 1215–1222, 2012.

[45] Victor Picheny, Tobias Wagner, and David Ginsbourger. A benchmark of kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization*, 48(3):607–626, apr 2013.