Scaling Gaussian Process Optimization by Evaluating a Few Unique Candidates Multiple Times

Daniele Calandriello [1] **Luigi Carratino** [2] **Alessandro Lazaric** [3] **Michal Valko** [1] **Lorenzo Rosasco** [2 4 5]

Abstract

Computing a Gaussian process (GP) posterior has a computational cost cubical in the number of historical points. A reformulation of the same GP posterior highlights that this complexity mainly depends on how many <u>unique</u> historical points are considered. This can have important implication in active learning settings, where the set of historical points is constructed sequentially by the learner. We show that sequential black-box optimization based on GPs (GP-Opt) can be made efficient by sticking to a candidate solution for multiple evaluation steps and switch only when necessary. Limiting the number of switches also limits the number of unique points in the history of the GP. Thus, the efficient GP reformulation can be used to exactly and cheaply compute the posteriors required to run the GP-Opt algorithms. This approach is especially useful in real-world applications of GP-Opt with high switch costs (e.g. switching chemicals in wet labs, data/model loading in hyperparameter optimization). As examples of this meta-approach, we modify two well-established GP-Opt algorithms, GP-UCB and GP-EI, to switch candidates as infrequently as possible adapting rules from batched GP-Opt. These versions preserve all the theoretical no-regret guarantees while improving practical aspects of the algorithms such as runtime, memory complexity, and the ability of batching candidates and evaluating them in parallel.

[1]DeepMind [2]MaLGa, DIBRIS, University of Genova, Italy [3]Meta AI [4]Massachusetts Institute of Technology [5]Istituto Italiano di Tecnologia. Correspondence to: Daniele Calandriello <dcalandriello@google.com>.

1. Introduction

Bayesian and bandit optimization (Mockus, 1989; Lattimore & Szepesvári, 2020) are two well established frameworks for black-box function optimization under uncertainty. They model the optimization process as a sequential learning problem. For an unknown function f and a decision set $\mathcal{A}$ (e.g., $\mathcal{A} \subseteq \mathbb{R}^d$) at each step t the learner:
(1) selects candidate $\mathbf{x}_t \in \mathcal{A}$ by maximizing an acquisition function u_t as a surrogate of f;
(2) receives a noisy feedback [1] $y_t \stackrel{\text{def}}{=} f(\mathbf{x}_t) + \eta_t$ from the environment, where $\eta_t \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, \xi^2)$;
(3) uses y_t to improve u_t's approximation of f.

A common approach to measure the learner's convergence is to study the *cumulative regret* $R_T \stackrel{\text{def}}{=} \sum_{t=1}^{T} f^\star - f(\mathbf{x}_t)$ of the learner's choice of candidates compared to the optimum of the function $f^\star = \max_{\mathbf{x}} f(\mathbf{x})$. A learner is deemed no-regret when it converges on average (i.e., $\lim_{T \to \infty} R_T/T \to 0$). Thanks to their flexibility and capability of modeling uncertainty, Gaussian processes (GP) (Rasmussen & Williams, 2006) have emerged as an effective choice for regret minimization, jump-starting the field of GP optimization (GP-Opt). Decision rules that leverage GPs to estimate upper confidence bounds (GP-UCB (Srinivas et al., 2010; Chowdhury & Gopalan, 2017)) or expected improvement (GP-EI (Wang & de Freitas, 2014)) provably achieve a regret[2] $\widetilde{\mathcal{O}}(\gamma_T \sqrt{T})$ over T steps. Here, γ_T is the so-called maximum information gain associated with $\mathcal{A}$ and the GP. It captures in a data-adaptive way the implicit number of parameters of the non-parametric GP model, quantifying the effective complexity of the optimization problem. Under appropriate assumptions γ_T grows slowly in T making GP based methods no-regret (Srinivas et al., 2010; Scarlett et al., 2017). However, as most non-parametric approaches, GP-Opt algorithms face scalability issues when T grows large, severely limiting their applicability to large scale settings. For example, in some hyper-parameter optimization problems the solutions can still improve even after tens of thousands of candidate evaluations (e.g. the NAS-bench search (Ying et al., 2019) evaluated 5 million candidates).

---

[1]Actions or arms are also used for $\mathbf{x}_t$; and rewards for y_t.
[2]The notation $\widetilde{\mathcal{O}}$ ignores $\mathrm{polylog}$ terms.

Even more, for recommender systems that continuously adapt to the users' behaviour, the horizon T is unbounded and the optimization process never ends (e.g. (Saito et al., 2021; Li et al., 2010) record tens of millions of interactions with users). Many approaches have been proposed to extend GP-Opt algorithms to these settings.

1.1. Approaches to scalable no-regret GP optimization

GP's most commonly considered limitation is the **computational bottleneck**, which stems from the high cost of quantifying uncertainty using exact inference on the GP model. In particular, in the traditional i.i.d. learning setting the complexity of computing a GP posterior over n training points is usually quantified as $\widetilde{\mathcal{O}}(n^3)$ time and $\widetilde{\mathcal{O}}(n^2)$ space (Rasmussen & Williams, 2006), which makes training GPs on large dataset unfeasible in general. A similar complexity analysis in the GP-Opt settings replaces n with the T candidates chosen by the algorithm, giving us a baseline complexity of $\widetilde{\mathcal{O}}(T^3)$ time and $\widetilde{\mathcal{O}}(T^2)$ space complexity. Two main approaches have emerged to reduce this complexity. The most common can provably preserve no-regret guarantees, and is based on approximating the GP model, using either inducing points (Quinonero-Candela et al., 2007; Calandriello et al., 2019) or quadrature Fourier features (Mutny & Krause, 2018). Another approach is to use approximate posterior inference, e.g., using iterative solvers (Gardner et al., 2018), but without no-regret guarantees.

Beyond computations, vanilla GP-Opt algorithms suffer from a **sequential experimentation bottleneck**, which stems from having to wait for feedback at every step of the sequential interaction protocol. Whenever each function evaluation is time consuming this bottleneck can actually dominate the computational one, and even computationally efficient GP-Opt algorithms cannot scale as they spend most of their time waiting for feedback. However, even in this slow evaluation setting scalability can be achieved by selecting batches of candidates at the same time and evaluating them in parallel. Experimental parallelism greatly reduces total evaluation time, and millions of evaluations are possible resulting in much better final performance (Ying et al., 2019). The key trade-off is between batch sizes and regret, with larger batches increasing the potential for parallelism but also increase feedback delay and regret. Several approaches have been proposed over the year to manage this trade-off (Desautels et al., 2014; Kathuria et al., 2016; Daxberger & Low, 2017).

Overall, to match existing state of the art GP-Opt algorithms in guarantees for regret, computational cost and batching capability, an algorithm should achieve $\widetilde{\mathcal{O}}(\gamma_T \sqrt{T})$ regret with a $\widetilde{\mathcal{O}}(T\gamma_T^3)$ time and $\widetilde{\mathcal{O}}(T\gamma_T + \gamma_T^2)$ space complexity, and only $\widetilde{\mathcal{O}}(\gamma_T)$ rounds of interactions (i.e., batches).

1.2. Contributions

In this paper we propose a meta-algorithm, MINI-META, that we use to generate scalable versions of many popular GP-Opt algorithm. Applying MINI-META to GP-UCB and GP-EI we obtain two new MINI-variants, MINI-GP-UCB and MINI-GP-EI, that achieve performance comparable to existing state-of-the-art scalable GP-Opt algorithms, but rely on very different underlying principles to achieve scalability.

There are two main building blocks in MINI-META. The first is a reformulation of the posterior of a GP that highlights that scalability in a GP does not depend on the number of points in the GP, but rather on the number of unique (i.e., different from all others) points. Indicating with q_t the number of unique points up to time t, the time complexity of computing a GP can then be reduced from $\widetilde{\mathcal{O}}(T^3)$ to $\widetilde{\mathcal{O}}(q_T^3)$. This is an elementary property of GPs, but its applicability greatly depends on the underlying task. It has little impact and is not used in i.i.d. learning settings where duplicates in the training data are exceedingly rare. It can bring more benefits in settings were multiple evaluations of the same candidate are naturally part of the solution, and it has been leveraged when possible in stochastic optimization (Picheny et al., 2013) and active learning (Binois et al., 2019). In this paper we show that this approach can be taken a step further in GP-Opt tasks. In particular, the second building block of MINI-META is a meta-strategy that given a GP-Opt algorithm as input, modifies the candidate selection strategy of the input learner to explicitly minimize q_t to improve performance while preserving the same regret rate guarantees as the original candidate selection strategy.

Out of the many meta-strategies that one can consider, a simple yet effective approach that we advocate is to limit the number of times the candidate selection strategy is allowed to switch candidates, since the number of candidate switches clearly bounds q_t. This can be done transparently for many input learners by using the selection strategy, and then sticking to the same candidate for multiple steps before invoking the selection strategy again. Furthermore, the rule we derive to decide when to switch candidate does not depend on the function feedback. Therefore the algorithm can first select a candidate, then choose how many steps it should stick to it, and finally evaluate the candidate multiple times in parallel. In other words, our meta-approach can not only retro-fit existing GP-Opt algorithms to be computationally scalable, but also experimentally scalable thanks to batching.

Theoretically, we study the impact of this generic modification both under the lenses of model freezing in the linear bandits (Abbasi-Yadkori et al., 2011) and delayed feedback in batch GP-Opt (Desautels et al., 2014; Calandriello et al., 2020). Using these tools we derive strong guarantees for two new variants of GP-UCB and GP-EI that balance between minimizing unique actions and achieving low regret,

hence dubbed minimize-unique-GP-UCB (MINI-GP-UCB) and minimize-unique-GP-EI (MINI-GP-EI). Thanks to a careful balancing of the trade-off, both algorithms match the $\widetilde{\mathcal{O}}(\gamma_T \sqrt{T})$ regret of their traditional counterparts, while evaluating a number of unique candidates $q_t \leq \widetilde{\mathcal{O}}(\gamma_t)$ that scales only with the maximum information gain. Moreover, these algorithms are extremely scalable, requiring only $\widetilde{\mathcal{O}}(T + Aq_T^3) \leq \widetilde{\mathcal{O}}(T + A\gamma_T^3)$ time, where A is the time required to optimize q_t, $\widetilde{\mathcal{O}}(T\gamma_T + \gamma_T^2)$ space, and $\widetilde{\mathcal{O}}(\gamma_T)$ rounds of interaction.

Our approach also highlights an even less studied family of scalability bottlenecks for GPs, the **candidate switch bottlenecks**. Similar to the sequential experimentation bottleneck, this is an experimental bottleneck that stems from the fact that in some cases, changing the candidate being evaluated can be time or cost expensive (e.g., even if testing a chemical might be fast, re-calibrating instruments for different chemicals might be expensive).

Finally, our approach is unique in the scalable GP literature in the sense that it completely side-steps the necessity of approximating inference or the GP model; avoiding the need of inducing points, randomized sketching, or iterative solvers. Instead, algorithms based on MINI-META can use exact inference on an exact GP model thanks to the efficient posterior reformulation and the different optimization path chosen to minimize unique candidates. This greatly simplifies the algorithm, making it more broadly applicable.

2. Preliminaries

Notation. We denote with $[t] = \{1, \ldots, t\}$ the set of integers up to t. At step t, the matrix $\mathbf{X}_t \overset{\text{def}}{=} [\mathbf{x}_1, \ldots, \mathbf{x}_t]^\top \in \mathbb{R}^{t \times d}$ contains all the candidates selected so far, and $\mathbf{y}_t \overset{\text{def}}{=} [y_1, \ldots, y_t]^\top$ their corresponding feedbacks. We define a batch as a sequence of steps between $t \leq t'$ such that the learner must select all candidates $\mathbf{x}_t, \ldots \mathbf{x}_{t'}$ before observing the corresponding feedback $y_t, \ldots, y_{t'}$. To connect steps and batches we adopt a standard notation (Desautels et al., 2014; Calandriello et al., 2020) where $\mathtt{fb}(t)$ indicates the last step of the previous batch, i.e., at step t the learner has access only to feedback $\mathbf{y}_{\mathtt{fb}(t)}$ up to step $\mathtt{fb}(t)$.

Gaussian processes. GPs (Rasmussen & Williams, 2006) are usually described using a mean function μ, which we assume to be zero, and a (bounded) covariance or kernel function $\mathrm{k} : \mathcal{A} \times \mathcal{A} \to [0, \kappa^2]$. Based on the prior μ and k, the posterior of the GP conditioned on some data $\mathbf{X}_t, \mathbf{y}_t$ is traditionally formulated as

$$
\begin{aligned}
\mu_t(\mathbf{x}_i) &= \mathrm{k}(\mathbf{x}_i, \mathbf{X}_t)(\mathbf{K}_t + \lambda \mathbf{I})^{-1}\mathbf{K}_t \mathbf{y}_t, \\
\sigma_t^2(\mathbf{x}_i) &= \mathrm{k}(\mathbf{x}_i, \mathbf{x}_i) - \mathrm{k}(\mathbf{x}_i, \mathbf{X}_t)(\mathbf{K}_t + \lambda \mathbf{I})^{-1}\mathrm{k}(\mathbf{X}_t, \mathbf{x}_i),
\end{aligned}
\tag{1}
$$

Algorithm 1 Optimistic GP optimization

Require: Set of candidates $\mathcal{A}$, acq. func. u_0
1: Initialize $t = 0$
2: **for** $t = \{1, \ldots, T\}$ **do**
3:     Select $\mathbf{x}_{t+1} = \arg\max_{\mathbf{x} \in \mathcal{A}} u_t(\mathbf{x})$
4:     Get feedback y_{t+1}
5:     Update μ_t, σ_t, β_t and u_t
6: **end for**

---

where we shortened $\mathbf{K}_t = \mathrm{k}(\mathbf{X}_t, \mathbf{X}_t)$, and use only the subscript t to indicate that μ_t and σ_t are conditioned on all data up to time t. Note also that μ_t and σ_t are only proper Bayesian GP posterior when $\lambda = \xi^2$, but we leave here λ as a free parameter as it will be useful in frequentist regret analysis. Given prior and posterior of the GP, we can also define the information gain between them as

$$
\gamma(\mathbf{X}, \mathbf{y}) \overset{\text{def}}{=} \tfrac{1}{2} \log \det(\mathbf{I} + \xi^{-2}\mathrm{k}(\mathbf{X}, \mathbf{X})).
$$

Based on $\gamma(\mathbf{X}, \mathbf{y})$ we can also define the maximum information gain $\gamma_t \overset{\text{def}}{=} \max_{\mathbf{X}:|\mathbf{X}|=t} \gamma(\mathbf{X}, \mathbf{y})$ at step t as a worst-case bound on the complexity of the optimization process. Large amounts of research have been dedicated to characterizing how γ_T behaves under different assumptions on $\mathcal{A}$ and k (Srinivas et al., 2010; Scarlett et al., 2017). This includes sufficient conditions for γ_T to be sublinear, both with polynomial and logarithmic rates in T. We leave the reader to recent surveys (Vakili et al., 2021) for more results, and treat here γ_T as a desirable data-adaptive measure of the intrinsic complexity of the problem.

Optimistic Gaussian process optimization. Given a GP posterior, we can construct an *acquisition function* $u_t(\cdot) : \mathcal{A} \to \mathbb{R}$ to guide us in the candidate selection. We focus specifically on acquisition functions based on upper confidence bounds (GP-UCB (Srinivas et al., 2010)) and expected improvement (GP-EI (Mockus, 1989))

$$
u_t^{\text{GP-UCB}}(\mathbf{x}) = \mu_t(\mathbf{x}) + \beta_t^{\text{GP-UCB}}\sigma_t(\mathbf{x}), \tag{2}
$$

$$
u_t^{\text{GP-EI}}(\mathbf{x}) = \Big(\beta_t^{\text{GP-EI}}\sigma_t(\mathbf{x}) \tag{3}
$$

$$
\cdot \Big[\big(\tfrac{z}{\beta_t^{\text{GP-EI}}}\big)\mathrm{CDF}_{\mathcal{N}}\big(\tfrac{z}{\beta_t^{\text{GP-EI}}}\big) + \mathrm{PDF}_{\mathcal{N}}\big(\tfrac{z}{\beta_t^{\text{GP-EI}}}\big) \Big] \Big),
$$

where $z = \frac{\mu_t(\mathbf{x}) - \max_{\mathbf{x}'} \mu_t(\mathbf{x}')}{\sigma_t(\mathbf{x})}$, $\mathrm{CDF}_{\mathcal{N}}(\cdot)$ and $\mathrm{PDF}_{\mathcal{N}}(\cdot)$ are the cumulative and probability density functions of a standard Gaussian, and $\beta_t \in \mathbb{R}_+$ must be appropriately chosen.

Given an acquisition function u_t, a standard way to obtain a GP-Opt algorithm based on it is to apply the meta-algorithm reported in Algorithm 1. In particular, the acquisition function is used by the meta-algorithm to choose candidates optimistically/greedily w.r.t. u_t. Applying Algorithm 1 to the GP-UCB and GP-EI acquisition function we obtain exactly the GP-UCB and GP-EI algorithms (Wang & de Freitas, 2014; Chowdhury & Gopalan, 2017), which are guaranteed

Algorithm 2 Meta-algorithm to minimize unique candidates (MINI-META)

Require: Set of candidates $\mathcal{A}$, acq. func. u_0, threshold C
 1: Initialize $t = 0$
 2: **for** $h = \{1, \dots\}$ **do**
 3:      Select $\mathbf{x}_{t+1} = \arg\max_{\mathbf{x} \in \mathcal{A}} u_t(\mathbf{x})$
 4:      Evaluate $\mathbf{x}_{t+1}$ for $B_h = \lfloor (C^2 - 1)/\sigma_t^2(\mathbf{x}_{t+1}) \rfloor$ times
        (i.e., $\mathbf{x}_s = \mathbf{x}_{t+1}$ for all $s \in [t+1, t+B_h]$)
 5:      Get feedback $\{y_s\}_{s=t+1}^{t+B_h}$
 6:      Set $t = t + B_h$, update posteriors μ_t, σ_t and acquisition function u_t
 7: **end for**
---

to achieve low regret. Unfortunately, optimizing u_t or even evaluating it is computationally inefficient, as it involves evaluating $\sigma_t(\mathbf{x})$ which requires at least $\mathcal{O}(t^2)$ time due to the multiplication $(\mathbf{K}_t + \lambda \mathbf{I})^{-1} \mathrm{k}(\mathbf{X}_t, \mathbf{x}_j)$. Moreover, the sequential nature of the optimization protocol precludes the possibility of running experiments in parallel, which is also necessary to achieve true scalability. We propose now a new approach to address both bottlenecks.

3. Efficient GP Optimization With Few Unique Candidates

Our approach to achieve scalability will be composed by two main ingredients. The first ingredient is a reformulation of the GP posterior that can be efficiently computed when the GP is supported on a small number q_t of unique candidates (i.e., the history $\mathbf{X}_t$ contains t rows with repetitions, of which only q_t are unique). To leverage this first ingredient we then need a second ingredient, a generic approach to guarantee that q_t stays small. This takes the form of a modification of the usual GP-Opt loop of Algorithm 1 to reduce the number of candidate switches, which acts as an upper bound on q_t. Since our switching rule is agnostic to u_t, our approach remains a flexible meta-algorithm that can be applied to different acquisition functions to obtain different efficient GP-Opt algorithms. As an added bonus, we also show how these choices allow to seamlessly incorporate candidate batching in the resulting algorithm, which improves experimental scalability.

3.1. A meta-algorithm to select few unique candidates

To control the number of candidate switches we generalize and improve the rarely switching OFUL (RS-OFUL) algorithm (Abbasi-Yadkori et al., 2011). In particular, we extend RS-OFUL from euclidean to Hilbert spaces, modify it to operate with acquisition functions different than the GP-UCB one, and replace RS-OFUL's switching rule with an improved criteria recently proposed for batched GP

optimization (Calandriello et al., 2020). Our final meta-algorithm MINI-META is reported in Algorithm 2.

We refer to it as a meta-algorithm because it can generate different GP-Opt algorithms using different acquisition functions $u_0(\cdot)$ as input. As an example, if we use $u_t^{\text{GP-UCB}}(\cdot)$ (Equation 2) we obtain an algorithm similar to GP-UCB that we call minimize-unique-GP-UCB or MINI-GP-UCB for short. Similarly, we can use Algorithm 2 to convert GP-EI into MINI-GP-EI. Moreover, all of the algorithms generated by MINI-META can compute posteriors efficiently using a reformulation of μ_t and σ_t introduced later in Equation 4 rather than the standard formulation of Equation 1.

The meta-algorithm operates in epochs/batches indexed by h, where each epoch is delimited by a candidate switch. Inside each epoch, the meta-algorithm selects the next candidate $\mathbf{x}_{t+1}$ to be evaluated by maximizing the acquisition function u_t, and we discuss more in detail later how easy or hard this inner optimization problem can be based on properties of u_t and $\mathcal{A}$. Then, after selecting the epoch's candidate the meta-algorithm must choose the length of the epoch. Since we do not switch candidate until the end of the epoch, this amounts to selecting how many times the candidate $\mathbf{x}_{t+1}$ will be repeatedly evaluated. The main trade-off here is between long epochs that improve scalability and short epochs that make it easier to switch often and explore/exploit efficiently. Specializing an approach used in the BBKB algorithm (Calandriello et al., 2020) to our setting, we propose to select $B_h = \lfloor (C^2 - 1)/\sigma_t(\mathbf{x}_{t+1}) \rfloor$, which as we prove achieves both goals. Finally, at the end of the epoch we collect the feedback, and use it to update the GP posterior and the chosen acquisition function. The loop is then repeated forever or until a desired number of epochs/steps is reached.

We also highlight that not all epoch-based GP-Opt algorithms are also batched GP-Opt algorithms. In particular, in batched GP algorithms candidates in a batch can be evaluated in parallel before feedback of previous candidates is available. Since there is no dependency on the feedback neither in our candidate selection (it is always the same candidate) nor in the epoch termination rule, our meta-algorithm can transform a sequential GP optimization algorithm to a batch variant, e.g., MINI-GP-EI is a batch algorithm even though GP-EI is not.

3.2. GPs supported on few unique candidates.

We can now show that if h is small, all the relevant quantities from Algorithm 2 can be computed efficiently. In particular, after running Algorithm 2 for h epochs let $\mathbf{X}_h$ indicate the $h \times d$ matrix containing the h candidates selected so far. To simplify exposition, we will assume all h candidates are distinct, and comment in the appendix how removing this assumption just involves slightly harder book-keeping

to merge feedbacks coming from two identical candidates selected in different epochs.

Given $\mathbf{X}_h$, let $\mathbf{W}_h \in \mathbb{R}^{h \times h}$ be a diagonal matrix; where $[\mathbf{W}_h]_{i,i} = B_i$ contains the number of times the candidate of the i-th epoch is contained in $\mathbf{X}_t$ (i.e., number of times it is selected), and let t_i denote the starting time-step of the i-th epoch. Then, we have

$$
\begin{aligned}
\mu_{t_h}(\mathbf{x}) &= \mathrm{k}(\mathbf{x}, \mathbf{X}_h)(\mathbf{K}_h + \lambda \mathbf{W}_h^{-1})^{-1}\mathbf{y}_h, \\
\sigma_{t_h}^2(\mathbf{x}) &= \mathrm{k}(\mathbf{x}, \mathbf{x}) - \mathrm{k}(\mathbf{x}, \mathbf{X}_h)(\mathbf{K}_h + \lambda \mathbf{W}_h^{-1})^{-1}\mathrm{k}(\mathbf{X}_h, \mathbf{x})
\end{aligned}
\tag{4}
$$

where $\mathbf{K}_h = \mathrm{k}(\mathbf{X}_h, \mathbf{X}_h) \in \mathbb{R}^{h \times h}$, and $\mathbf{y}_h \in \mathbb{R}^h$ is such that $[\mathbf{y}_h]_i = \sum_{s=t_i}^{t_i + B_i} y_s$, Using the feature-space view of GPs, it is straightforward to prove using basic linear algebra identities that the posterior Equation 4 is equal to Equation 1, i.e., that these are not approximations to the posterior but reformulations of the posterior (see Appendix A). However, when $h \ll t$, Eq. 4 can be computed efficiently in $\mathcal{O}(h^3)$, since it only involves the inversion of an $h \times h$ matrix $\mathbf{K}_h$.

The reformulations of Equation 4 are not completely new, e.g., Picheny et al. (2013) presented similar reformulations based on averaged feedbacks. At a higher level, they can be seen as a special case of the more generic framework of reformulating GPs posteriors as distribution conditioned on a set of inducing points, also known as sparse GP approximation (Quinonero-Candela et al., 2007). Traditionally, this framework has been designed to identify a small number of inducing variables that could act as a bottleneck for the GP model, reducing its number of parameters and accuracy, but also the computational cost of inference. However, unlike previous approaches we propose to utilize the whole optimization history as inducing variables, removing this bottleneck. Therefore we are not approximating a full GP with a factorized GP, or a dense GP with a sparse GP, but simply exploiting the intrinsic sparseness and low number of implicit parameters that arise when GPs are defined on few unique candidates. Note that enforcing a candidate selection strategy that keeps q_t under control is crucial in creating a gap between Equation 1 and Equation 4. This explains why historically this reformulation has not seen success in fields such as the standard supervised GP regression setting, where the selection process is not under the control of the learner. In particular, when $\mathbf{X}_t$ is sampled i.i.d. from some distribution the rows are all different w.p. 1, and our method offers no scalability improvement. Looking instead at the history of GP optimization methods, this reformulation has been largely ignored because most approaches try to evaluate a very diverse sets of candidates to improve initial exploration. This increases the number of unique historical candidates, and makes the impact of the reformulation negligible or even detrimental since it prevents other kinds of efficient incremental updates.

4. Regret and computational meta-analysis

Rather than designing a specialized analysis for every MINI-variant, we propose instead a meta-analysis that cover a generic u_t acquisition function, and then will instantiate this analysis at the end of the section with $u_t^{\mathrm{GP\text{-}UCB}}$ and $u_t^{\mathrm{GP\text{-}EI}}$.

While the computational meta-analysis holds for any u_t, the regret meta-analysis requires u_t to satisfy a simple condition. Given feedback up to step $t - 1$, candidates selected optimistically using the acquisition function u_t at step t must satisfy w.h.p. this bound on the instantaneous regret r_t

$$
r_t \overset{\text{def}}{=} f^\star - f(\mathbf{x}_t) \leq G_t \sigma_{t-1}(\mathbf{x}_t), \tag{5}
$$

for some non-decreasing sequence G_t. This condition is satisfied for example by GP-UCB (Srinivas et al., 2010; Abbasi-Yadkori et al., 2011; Chowdhury & Gopalan, 2017) and GP-EI (Wang & de Freitas, 2014). Based on this condition, our paper's main result is the following meta-theorem.

Theorem 4.1. *For any $1 < C$, set $\mathcal{A}$ and acquisition function u_0, Algorithm 2 runs in $\widetilde{\mathcal{O}}(T + \gamma_T \cdot (A + \gamma_T^3))$ time, $\widetilde{\mathcal{O}}(T\gamma_T)$ space, and $\widetilde{\mathcal{O}}(\gamma_T)$ epochs/batches, where A indicates the complexity of solving the inner optimization problem (i.e., $\arg\max_{\mathbf{x} \in \mathcal{A}} u_t(\mathbf{x})$).*

Moreover, if u_t satisfies w.p. $1 - \delta$ the condition of Equation 5, then for all steps t the instantaneous regret $r_t^{\mathrm{MINI\text{-}META}}$ of the candidates selected by Algorithm 1 satisfies $r_t^{\mathrm{MINI\text{-}META}} \leq CG_t \sigma_{t-1}(\mathbf{x}_t)$ for the same sequence of G_t and with the same probability.

Focusing on regret, Theorem 4.1 shows that restricting the number of candidate switches does not greatly impact regret, just as in (Abbasi-Yadkori et al., 2011). Moreover MINI-META's improved switching rule reduces the regret amplification factor from $\mathcal{O}(2^C)$ of RS-OFUL to $\mathcal{O}(C)$. Additionally, our result also highlights for the first time that approximating the GP model is not necessary to achieve scalability and no-regret, and simply using a different candidate selection strategy brings great computational savings. Moreover, compared to existing methods based on efficient reformulations of GPs (Picheny et al., 2013; Binois et al., 2019) we explicitly target minimizing unique candidates as a goal of our algorithm, and can rigorously quantify the computational improvement that we are guaranteed to achieve compared to a vanilla GP inference approach.

On the computational side, we bound the number of switches and therefore of unique candidates using the information gain $\widetilde{\mathcal{O}}(\gamma_T)$. This, allows us to bound both the number of batches and the cost of evaluating the GP posterior, but unfortunately without knowing how expensive it is to optimize the acquisition function it is not possible to fully characterize the runtime. We postpone this to a later discussion. Nonetheless, an important aspect that can be already

included in the meta-theorem is that this maximization only needs to be performed once per batch. Previous approaches tried to promote diverse batches, and therefore this optimization had to be repeatedly performed with a runtime of at least $\widetilde{\mathcal{O}}(TA)$, while for the first time we achieve a runtime $\widetilde{\mathcal{O}}(T + A)$ linear in T. However, this crucially relies on being able to repeatedly evaluate the same candidate, which is not possible in all optimization settings.

4.1. Regret meta-analysis details

Starting from the condition in Equation 5, the meta-analysis is based on well-established tools. First we have to deal with the fact that Algorithm 1 only updates the posterior and receives feedback at the end of the batches. In particular, this means that Equation 5 only applies for the first evaluation of the candidate, and not across the batch. Tools developed in the original RS-OFUL analysis (Abbasi-Yadkori et al., 2011) and refined in the context of batch GP optimization (Calandriello et al., 2020) can be used to control this.

Lemma 4.2. *Let u_t be an acquisition function that satisfies Equation 5 at the beginning of each batch. Then running Algorithm 2 with parameter C and the same u_t guarantees that $r_t \leq CG_t\sigma_{t-1}(\mathbf{x}_t)$ w.p. $1 - \delta$ for all steps t (i.e., not only at the beginning of each batch).*

Sketch of proof. Let t be the first step of epoch/batch h (i.e., we received feedback up to $t - 1$). Then

$$r_{t'} \leq G_{t-1}\sigma_{t-1}(\mathbf{x}_{t'}) = G_{t-1}\frac{\sigma_{t-1}(\mathbf{x}_{t'})}{\sigma_{t'-1}(\mathbf{x}_{t'})} \cdot \sigma_{t'-1}(\mathbf{x}_{t'})$$

$$\leq G_{t-1}C \cdot \sigma_{t'-1}(\mathbf{x}_{t'}),$$

where the first inequality comes from applying Equation 5 to step t' since $\mathbf{x}_t = \mathbf{x}_{t'}$ (all candidates in the batch are the same), and the second inequality comes from the fact that the length of each batch B_h is designed to guarantee that for any $t' > t$ inside the batch we can bound the ratio $\sigma_{t-1}(\mathbf{x})/\sigma_{t'-1}(\mathbf{x}) \leq \mathcal{O}(C)$ (details in the appendix). Therefore, simply by losing a constant factor C we can recover the regret $r_{t'}$ as if we applied Equation 5 with all the feedback up to step $t' - 1$. $\square$

Established the $G_tC\sigma_{t-1}(\mathbf{x}_t)$ bound, the rest of each regret proofs slightly diverge depending on the acquisition function, differing mostly on the specific sequence of G_t, and we discuss in Section 4.3 where we also discuss how the final regret of each of the alternatives scales. For now, we would like to highlight that the $1 - \delta$ success probability of Lemma 4.2, and therefore the probability of low regret, depends only on the randomness of the candidate selection, i.e., the same randomness present in the original variants. This is in stark contrast with existing results for scalable GP optimization where to provide guarantees on the regret the GP was approximated using technique relying on some

form of additional randomness (e.g., random projections). In other words, our μ-variants are much more <u>deterministic</u> compared to other scalable methods.

4.2. Computational meta-analysis.

Similarly to regret, we provide here a meta-analysis for a generic u_t, parametrizing the analysis in terms of a generic A computational cost of optimizing u_t (i.e., computing $\arg\max_{\mathbf{x}\in\mathcal{A}} u_t(\mathbf{x})$)). With this notation, the computational complexity of MINI-META is $\mathcal{O}(T + h \cdot A + h \cdot h^3))$. The $\mathcal{O}(T)$ term covers the time necessary to store/load the T feedbacks. The hA term comes from the h optimizations of u_t, once for each loop. Finally the $\mathcal{O}(h \cdot h^3)$ term is due to the computation of the length B_h of each batch, which requires at each loop to computing the posterior σ_t using the $\mathcal{O}(h^3)$ efficient reformulation. We can refine this analysis along two axes: bounding h, and bounding A.

Bounding the number of switches h is fully part of the meta-analysis, since our bound on the number of epochs does not directly depend on the specific acquisition function u_t.

Lemma 4.3. *After T steps, MINI-META performs at most $h \leq \mathcal{O}\left(\frac{C^2}{C^2-1}\gamma_T\right)$ switches.*

Notice that once again this is a deterministic statement, since unlike existing methods our approach does not depend on randomized methods to increase scalability. Combining this result in the meta-analysis we obtain an overall runtime of $\mathcal{O}(T + \gamma_T \cdot (A + \gamma_T^3))$.

Bounding the optimization cost A can only be done by making assumptions on u_t and $\mathcal{A}$, since in general maximizing the acquisition function is a non-convex optimization problem, often NP-hard. In the simpler case where $\mathcal{A}$ is finite with cardinality $|\mathcal{A}|$, and u_t is based only on μ_t and σ_t of each candidate, then the runtime becomes $\mathcal{O}(T + h \cdot (|\mathcal{A}|h^2 + h^3)$. Further combining this with Lemma 4.3 we obtain an overall runtime of $\mathcal{O}(T + |\mathcal{A}|\gamma_T^3 + \gamma_T^4)$. We can compare this result with the $\widetilde{\mathcal{O}}(T|\mathcal{A}|\gamma_T^2)$ runtime complexity of the currently fastest no-regret algorithm, BBKB (Calandriello et al., 2020) to see that the potential runtime improvement is large (i.e., from $\mathcal{O}(T|\mathcal{A}|)$ to $\mathcal{O}(T + |\mathcal{A}|h)$. This is because MINI-META only needs to maximize u_t once per batch, while existing batched algorithms need multiple maximization per batch to promote in-batch diversity, using different strategies such as feedback hallucination (Desautels et al., 2014). Even if we ignored the cost of optimizing u_t, our result would still be the first to decouple the the $\mathcal{O}(T)$ component from γ_T. This is because for existing batching rule, even computing the batch length required to evaluate at least one σ_t at each step, with each evaluation costing at least $\mathcal{O}(\gamma_T^2)$. Instead, our batching rule is based only on σ_t once at the beginning of the batch.

4.3. Instantiating the meta-analysis

Instantiating the results from the previous section we can now provide bounds for two popular acquisition functions. For simplicity and to guarantee that all steps of the algorithm can be implementable in accord with the regret analysis, we restrict ourselves to the assumption of finite $\mathcal{A}$ where the acquisition functions can be exactly maximized. For all of these variants, the run-time is bounded as $\mathcal{O}(T + |\mathcal{A}|\gamma_T^3 + \gamma_T^4)$, so we will mainly discuss regret in this section.

MINI-GP-UCB We consider frequentist ($\|f\| \leq F$) and Bayesian ($f \sim GP$) settings.

Theorem 4.4. *Assume $\|f\| \leq F$. For any $1 < C$, $\delta \in [0, 1]$, run Algorithm 1 with $u_t = u_t^{\text{GP-UCB}}$, $\lambda = \xi^2$, and*

$$\beta_h = \Theta\left(\sqrt{\log\det\left(\frac{\mathbf{W}_h^{1/2}\mathbf{K}_h\mathbf{W}_h^{1/2}}{\lambda} + \mathbf{I}\right) + \log(\tfrac{1}{\delta})} + F\right).$$

Then w.p. $1 - \delta$, $R_T \leq \widetilde{\mathcal{O}}\left((\sqrt{\gamma_T} + F)C\sqrt{\gamma_T T}\right)$.

This result is a combination of Theorem 4.1 with (Chowdhury & Gopalan, 2017, Thm. 1). The proof of this result is straightforward using Theorem 4.1. In particular, the original GP-UCB provides a bound on the regret of the form $R_T \leq \widetilde{\mathcal{O}}(\beta_T \sum_{t=1}^T \sigma_{t-1}(\mathbf{x}_t))$, which using Lemma 4.2 easily becomes $R_T \leq \widetilde{\mathcal{O}}(\beta_T C \sum_{t=1}^T \sigma_{t-1}(\mathbf{x}_t))$, only a C factor worse but with a much lower computational complexity. Compared to other approximations of GP-UCB, MINI-GP-UCB achieve logarithmic improvements due to a tighter β_h. In particular, all previous approximate GP-UCB variants had to approximate $\log\det(\mathbf{K}_t/\lambda + \mathbf{I})$ with a more or less loose upper bound, which resulted in excessive exploration and worse regret. Instead, MINI-GP-UCB uses the exact log-determinant of the GP, since it only needs to be defined on the unique candidates.

Theorem 4.5. *Assume $f \sim GP$. For any $1 < C$, $\delta \in [0, 1]$, run Algorithm 1 with $u_t = u_t^{\text{GP-UCB}}$, setting $\lambda = \xi^2$ and $\beta_t = \sqrt{2\log(|\mathcal{A}|t^2\pi^2/(6\delta))}$. Then w.p. $1 - \delta$,*

$$R_T \leq \widetilde{\mathcal{O}}(C\sqrt{T\gamma_T}).$$

This result is a combination of Theorem 4.1 and (Srinivas et al., 2010, Thm. 1). The main advantage of the Bayesian tuning of MINI-GP-UCB is that computing β_t does not require to know a bound on the norm of the function F (which is often infinite under the GP prior). However MINI-GP-UCB still needs access to the noise level ξ to tune λ.

MINI-GP-EI We can combine Theorem 4.1 with (Wang & de Freitas, 2014, Thm. 1) under frequentist assumptions.

Theorem 4.6. *Assume $\|f\| \leq F$. For any $1 < C$ and $\delta \in [0, 1]$, run Algorithm 1 with $u_t = u_t^{\text{GP-EI}}$,*

$\lambda = \xi^2$, *and* $\beta_h = \big(\log\det(\mathbf{W}_h^{1/2}\mathbf{K}_h\mathbf{W}_h^{1/2}/\lambda + \mathbf{I}) + \sqrt{\log\det(\mathbf{W}_h^{1/2}\mathbf{K}_h\mathbf{W}_h^{1/2}/\lambda + \mathbf{I})\log(\tfrac{t}{\delta}) + \log(\tfrac{t}{\delta})}\big)^{1/2}$. *Then w.p. $1 - \delta$, $R_T \leq \widetilde{\mathcal{O}}\left((\sqrt{\gamma_T} + F)C\sqrt{\gamma_T T}\right)$.*

Again we can easily integrate Theorem 4.1 in the original proof. In particular, the original GP-EI analysis provides a bound on the regret of the form $R_T \leq \widetilde{\mathcal{O}}\left(\left(\sqrt{\lambda}F + \beta_T\right)\sum_{t=1}^T \sigma_{t-1}(\mathbf{x}_t)\right)$, which again using Lemma 4.2 easily becomes $R_T \leq \widetilde{\mathcal{O}}\left(\left(\sqrt{\lambda}F + \beta_T\right)C\sum_{t=1}^T \sigma_{t-1}(\mathbf{x}_t)\right)$ recovering the original regret up to constants. Although the regret of MINI-GP-UCB and MINI-GP-EI are comparable, the underlying algorithms have important differences. In particular, tuning β_h in MINI-GP-EI does not require knowing a bound F on the norm of the function, which is hard to obtain in many cases. However the analysis of MINI-GP-EI requires to set $\lambda = \xi^2$, which might be as hard to estimate.
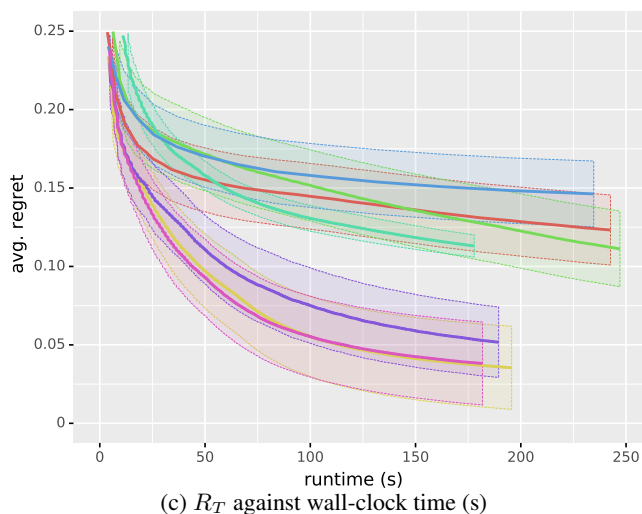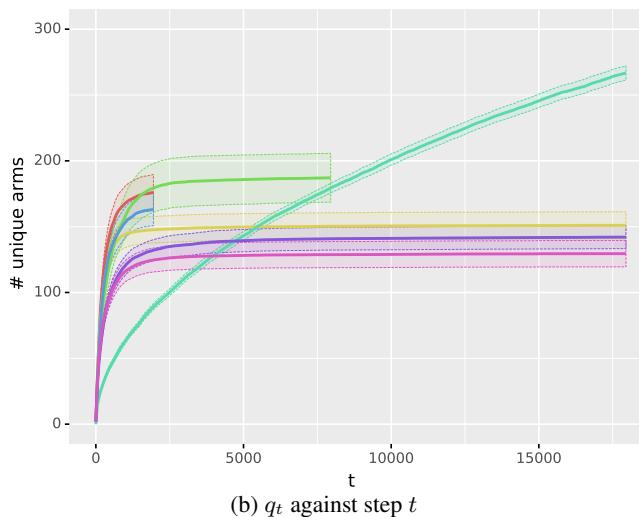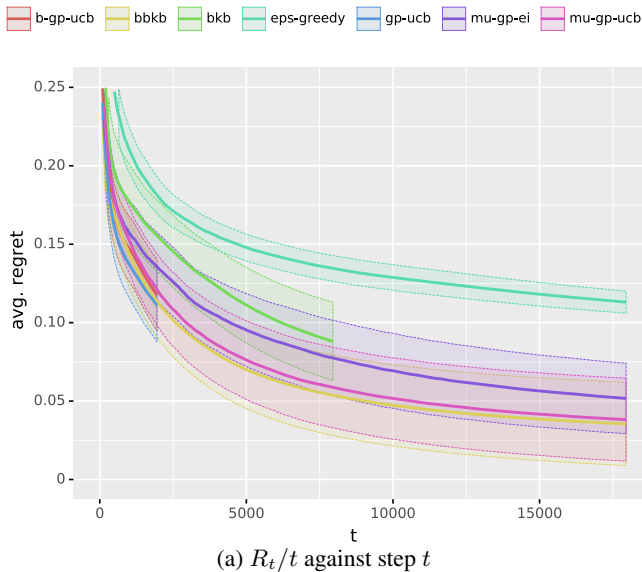
4.4. Extension to continuous spaces

The computational and regret analysis in this section rely on the ability of the algorithm to optimize the acquisition function exactly (i.e., find the true minimizer of $\arg\max_{\mathbf{x}\in\mathcal{A}} u_t(\mathbf{x})$), to avoid selecting sub-optimal actions, and efficiently (i.e., the A term in Section 4.2 should not be too large), to avoid dominating the runtime. Both of these requirements are easy to satisfy on discrete $\mathcal{A}$ (where u_t can be optimized by enumeration), which is why we focus on these decision sets. However, when optimizing continous u_t we can easily extend the MINI-META framework to replace enumeration with methods based on continuous solvers (Mutny & Krause, 2018; Jones et al., 1998) or sampling (Balandat et al., 2019). This does not come with theoretical guarantees (i.e., without exact and efficient acquisition optimization both regret and runtime can potentially be large), but the minimizing the number of unique candidates by infrequent switches preserves important practical advantages also when the number of candidates is large or infinite. In particular MINI-META optimizes the continuous acquisition function only h times, once per batch, and much less frequently than other methods that to promote diversity need to re-optimize u_t at each step. When $h \ll T$ this saves a large number of expensive invocations of the optimizer. Moreover, most optimizers/samplers look for a good candidate by repeated evaluation of the posterior. Even in continuous settings MINI-META's posterior remains sparse, and efficient to evaluate further compounding the computational gains.

5. Experiments

We now evaluate our proposed approach empirically, focusing here on real-world data and on synthetic data in the
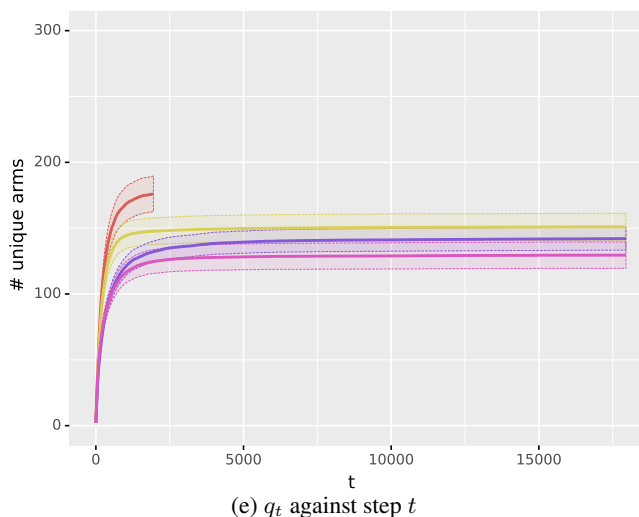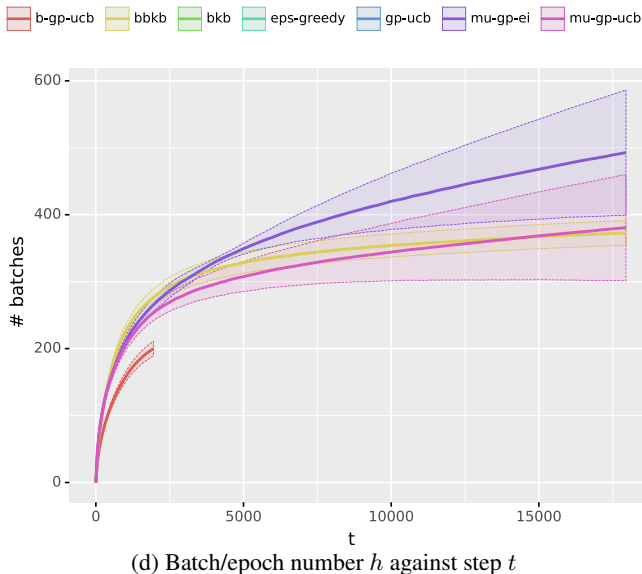
(a) R_t/t against step t

(b) q_t against step t

(c) R_T against wall-clock time (s)

appendix. In particular, following the approach taken in (Calandriello et al., 2020) we evaluate MINI-GP-UCB and MINI-GP-EI on NAS-bench (Ying et al., 2019). We compare MINI-GP-UCB and MINI-GP-EI with BBKB (Calandriello et al., 2020), GP-UCB (Srinivas et al., 2010), GP-BUCB (Desautels et al., 2014), GP-BTS (Kandasamy et al., 2018), and an epsilon greedy strategy. For MINI-GP-UCB, MINI-GP-EI, and BBKB hyperparameters that need to be tuned are C, the bandwidth of the Gaussian kernel σ, while λ is set to the 90-th percentile of the standard deviation of the target as an oracle for the noise level. For epsilon greedy epsilon is set as a/t^b and $a \in \{0.1, 1, 10\}$ and $b \in \{1/3, 1/2, 1, 2\}$ is tuned via grid search. For each experiment we repeat the run 40 times with different seeds, and report mean and confidence intervals for the hyperparameter configuration (searched in a grid) that achieves lowest average regret (specific values reported in the appendix). All experiments are ran using a single, recent generation CPU core to avoid inconsistencies between some implementations using paralallel BLAS and some not. The neural architecture search setting is the same as (Calandriello et al., 2020). In particular, the search space is the discrete combinatorial space of possible 1-hidden layer 4 node networks using either convolutions or max-pooling, that is then used as a module in an inception-like architecture. The final space $\mathcal{A}$ has $|\mathcal{A}| = 12416$ candidates in $d = 19$ dimensions.

In Fig. (a) we report average regret vs steps (i.e., t). To compensate scale effect in the rewards, the average regret is normalized by the average regret achieved by a completely exploratory policy that selects candidates uniformly at random. Both MINI-variants are comparable to the current state of the art BBKB, and better than a tuned epsilon greedy. In Fig. (b) we report the number of unique candidates (note, not of switches) selected by each algorithm. We note that despite not explicitly being designed for this, BBKB does not select a very large number of unique arms. However, MINI-GP-UCB and MINI-GP-EI still select an even smaller number of unique candidates. In Fig. (c) we report average regret vs wallclock runtime rather than steps. Therefore a more efficient algorithm will terminate faster and achieve a lower average regret in the same span of time. We see that both MINI-GP-UCB and MINI-GP-EI are faster than BBKB, terminating earlier. However the empirical runtime gap does not seem to match the large theoretical runtime gap (i.e., $\mathcal{O}(T + A)$ vs $\mathcal{O}(TA)$). In particular, the actual empirical diversity in BBKB's batches seem to be limited, and allows for a large degree of lazy updates, which are not accounted for in the worst-case complexity.

Another interesting figure to empirically measure is the difference between the number h of batches/switches at step T and the number of unique candidates q_T selected. As we know, h is an upper bound for q_T, but looking at Fig. (d) and Fig. (e) we see that especially in the later stages of

(d) Batch/epoch number h against step t

(e) q_t against step t

the optimization these two quantities have a significant gap between them. This can be explained by noticing that as the optimization progresses the algorithm tend to focus on a few good candidates, switching back and forth between them which increases h but not q_T.

6. Conclusions, limitations and open questions

Our paper highlighted how existing tools can be combined in a new effective approach, capable of positively impacting the GP optimization setting both theoretically and empirically. Theoretically, as to the best of our knowledge, our proposed MINI-variants achieve the tightest guarantees both in regret and runtime among scalable GP optimization methods, using a very different approach that does not require approximating the GP or other randomized approximations. Empirically, because our method comes with a number of

practical properties, including scalability, adaptability to different acquisition functions, and being suitable to batched settings as well as settings where switching costs are be significant. However, there remain several limitations, which brings with them open questions.

Our MINI-variants inherit the limitations of the original methods and while some (e.g., scalability) are removed, others remain. In particular, MINI-GP-UCB and MINI-GP-EI still rely on quantities that are hard to estimate, such as the function norm, noise level, kernel choice, or kernel bandwidth. It is unclear how an on-line tuning of these quantities might be done without losing the no-regret guarantees. It would be interesting to see if recent approaches to tune these quantities for traditional GP optimization (Berkenkamp et al., 2019; Durand et al., 2018) can be transferred to MINI-variants, leveraging their unique property of being based on exact GP inference. Furthermore, while MINI-META could be applied to other acquisition functions, not all would result in a scalable algorithm. For example GP-TS (Chowdhury & Gopalan, 2017) also satisfies Equation 5. However, sampling a TS posterior is not scalable (i.e., $\mathcal{O}(|\mathcal{A}|^2)$ for finite $\mathcal{A}$) even for sparse posteriors, and therefore a hypotetical MINI-GP-TS would remain not scalable.

From a complexity perspective, optimizing the acquisition function exactly remains one of the hardest obstacles in GP optimization, and it is still not clear how the no-regret guarantees can be extended to approximate maximization. Generic non-convex optimizers such as DiRECT (Mutny & Krause, 2018; Jones, 2001) have an exponential complexity, and even considering the effective dimensionality reduction induced by Equation 4, which reduces the parameter space from $\mathcal{O}(t)$ to $\mathcal{O}(q_t)$, they might remain infeasible.

Despite not being explicitly optimized for this task, we empirically observe that other approximate GP-Opt methods, such as BBKB, also tend to achieve a small q_t. Indeed, any algorithm that quickly converges to a small set of good candidates would evaluate a small number of unique candidates. Therefore, it is an important open question to try and bound the number of candidates evaluated by a generic GP optimization algorithm without low-switching enforcement, as it might show that the reformulations of Equation 4 might be more broadly applicable than expected.

Finally, looking at the applicability of our method, our whole approach is based on the possibility of evaluating multiple times the same candidate without affecting the outcome. While this is a strength in settings with high switching costs, it also makes it unsuitable in settings where this is impossible (e.g., repeated medical testing of a single patient), or limited (e.g., news change over time and cannot be recommended over and over). More subtly, it also makes it poorly suited to low or noiseless settings, where multiple evaluations of the same candidate would not be very useful.

References

Abbasi-Yadkori, Y., Pál, D., and Szepesvári, C. Improved algorithms for linear stochastic bandits. In Advances in Neural Information Processing Systems, pp. 2312–2320, 2011.

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. Botorch: Programmable bayesian optimization in pytorch. arXiv preprint arXiv:1910.06403, 2019.

Berkenkamp, F., Schoellig, A. P., and Krause, A. No-regret bayesian optimization with unknown hyperparameters. Journal of Machine Learning Research, 20(50):1–24, 2019. URL http://jmlr.org/papers/v20/18-213.html.

Binois, M., Huang, J., Gramacy, R. B., and Ludkovski, M. Replication or exploration? sequential design for stochastic simulation experiments. Technometrics, 61(1):7–23, 2019.

Calandriello, D., Carratino, L., Lazaric, A., Valko, M., and Rosasco, L. Gaussian process optimization with adaptive sketching: Scalable and no regret. In Conference on Learning Theory, 2019.

Calandriello, D., Carratino, L., Valko, M., Lazaric, A., and Rosasco, L. Near-linear time gaussian process optimization with adaptive batching and resparsification. In International Conference on Machine Learning, 2020.

Chowdhury, S. R. and Gopalan, A. On kernelized multi-armed bandits. In International Conference on Machine Learning, pp. 844–853, 2017.

Daxberger, E. A. and Low, B. K. H. Distributed batch Gaussian process optimization. In Precup, D. and Teh, Y. W. (eds.), Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pp. 951–960, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Desautels, T., Krause, A., and Burdick, J. W. Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization. The Journal of Machine Learning Research, 15(1):3873–3923, 2014.

Durand, A., Maillard, O.-A., and Pineau, J. Streaming kernel regression with provably adaptive mean, variance, and regularization. The Journal of Machine Learning Research, 19(1):650–683, 2018.

Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q., and Wilson, A. G. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. Advances in Neural Information Processing Systems, 2018:7576–7586, 2018.

Jones, D. R. Direct global optimization algorithmDirect Global Optimization Algorithm, pp. 431–440. Springer US, Boston, MA, 2001. ISBN 978-0-306-48332-5. doi: 10.1007/0-306-48332-7_93. URL https://doi.org/10.1007/0-306-48332-7_93.

Jones, D. R., Schonlau, M., and Welch, W. J. Efficient global optimization of expensive black-box functions. Journal of Global optimization, 13(4):455–492, 1998.

Kandasamy, K., Krishnamurthy, A., Schneider, J., and Póczos, B. Parallelised bayesian optimisation via thompson sampling. In International Conference on Artificial Intelligence and Statistics, pp. 133–142, 2018.

Kathuria, T., Deshpande, A., and Kohli, P. Batched gaussian process bandit optimization via determinantal point processes. In Advances in Neural Information Processing Systems, pp. 4206–4214, 2016.

Lattimore, T. and Szepesvári, C. Bandit algorithms. Cambridge University Press, 2020.

Li, L., Chu, W., Langford, J., and Schapire, R. E. A contextual-bandit approach to personalized news article recommendation. International World Wide Web Conference, 2010. URL http://rob.schapire.net/papers/www10.pdf.

Mockus, J. Bayesian approach to global optimization: theory and applications. 1989.

Mutny, M. and Krause, A. Efficient High Dimensional Bayesian Optimization with Additivity and Quadrature Fourier Features. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), Advances in Neural Information Processing Systems 31, pp. 9019–9030. Curran Associates, Inc., 2018.

Picheny, V., Ginsbourger, D., Richet, Y., and Caplin, G. Quantile-based optimization of noisy computer experiments with tunable precision. Technometrics, 55(1):2–13, 2013.

Quinonero-Candela, J., Rasmussen, C. E., and Williams, C. K. Approximation methods for gaussian process regression. Large-scale kernel machines, pp. 203–224, 2007.

Rasmussen, C. E. and Williams, C. K. I. Gaussian processes for machine learning. Adaptive computation and machine learning. MIT Press, Cambridge, Mass, 2006. ISBN 978-0-262-18253-9. OCLC: ocm61285753.

Saito, Y., Shunsuke, A., Megumi, M., and Yusuke, N. Open bandit dataset and pipeline: Towards realistic and reproducible off-policy evaluation. NeurIPS2021 Datasets and Benchmarks Track, 2021.

Scarlett, J., Bogunovic, I., and Cevher, V. Lower bounds on regret for noisy gaussian process bandit optimization. In Conference on Learning Theory, pp. 1723–1742, 2017.

Srinivas, N., Krause, A., Seeger, M., and Kakade, S. M. Gaussian process optimization in the bandit setting: No regret and experimental design. In International Conference on Machine Learning, pp. 1015–1022, 2010.

Vakili, S., Khezeli, K., and Picheny, V. On information gain and regret bounds in gaussian process bandits. In Banerjee, A. and Fukumizu, K. (eds.), Proceedings of The 24th International Conference on Artificial Intelligence and Statistics, volume 130 of Proceedings of Machine Learning Research, pp. 82–90. PMLR, 13–15 Apr 2021.

Wang, Z. and de Freitas, N. Theoretical analysis of bayesian optimisation with unknown gaussian process hyperparameters. arXiv preprint arXiv:1406.7758, 2014.

Ying, C., Klein, A., Real, E., Christiansen, E., Murphy, K., and Hutter, F. Nas-bench-101: Towards reproducible neural architecture search. arXiv preprint arXiv:1902.09635, 2019.

A. Proofs of Section 3

For several of the proofs in this section it will be useful to introduce the so-called feature space formulation of a GP posterior (Rasmussen & Williams, 2006). In particular, to every kernel function k we can associate a feature map $\varphi(\cdot) : \mathcal{A} \to \mathcal{H}$ where $\mathcal{H}$ is the reproducing kernel Hilbert space associated with k and the GP. The main property of $\varphi(\cdot)$ is that for any $\mathbf{x}$ and $\mathbf{x}'$ we have $k(\mathbf{x}, \mathbf{x}') = \varphi(\mathbf{x})^\mathsf{T}\varphi(\mathbf{x}')$. With a slight abuse of notation, we will also indicate with $\varphi(\mathbf{X}) = [\varphi(\mathbf{x}_1), \ldots, \varphi(\mathbf{x}_t)]^\mathsf{T}$ the linear operator obtained by stacking together the various $\varphi(\cdot)$, such that $\mathbf{K}_t = \varphi(\mathbf{X}_t)\varphi(\mathbf{X}_t)^\mathsf{T}$. Note that this also allows us to define the equivalent of $\mathbf{K}_t$ in $\mathcal{H}$ as $\varphi(\mathbf{X}_t)^\mathsf{T}\varphi(\mathbf{X}_t) : \mathcal{H} \to \mathcal{H}$. Finally, to connect $\mathbf{K}_t$ and $\varphi(\mathbf{X}_t)^\mathsf{T}\varphi(\mathbf{X}_t)$ we will heavily use this fundamental linear algebra equality

$$\mathbf{B}^\mathsf{T}(\mathbf{BB}^\mathsf{T} + \lambda\mathbf{I})^{-1}\mathbf{B} = \mathbf{B}^\mathsf{T}\mathbf{B}(\mathbf{B}^\mathsf{T}\mathbf{B} + \lambda\mathbf{I})^{-1} \tag{6}$$

which can be easily shown to be valid for any linear operator $\mathbf{B}$ using its singular value decomposition.

A.1. Proof of equivalence between Equation 1 and Equation 4

Assume for now that t is the step at the end (i.e., $\mathtt{fb}(t) = t$) of batch h. We will relax this assumption at the end of the section to discuss how this can be extended to intermediate steps. Using the feature-space representation of a GP (see e.g., (Rasmussen & Williams, 2006, Eq. (2.11))), and defining $\mathbf{V}_t \overset{\mathrm{def}}{=} \varphi(\mathbf{X}_t)^\mathsf{T}\varphi(\mathbf{X}_t) + \lambda\mathbf{I}$, we can rewrite the posterior variance as

$$\begin{aligned}
\sigma_t^2(\mathbf{x}_i) &= k(\mathbf{x}_i, \mathbf{x}_i) - k(\mathbf{x}_i, \mathbf{X}_t)(\mathbf{K}_t + \lambda\mathbf{I})^{-1}k(\mathbf{X}_t, \mathbf{x}_i) \\
&\overset{a}{=} \varphi(\mathbf{x}_i)^\mathsf{T}\varphi(\mathbf{x}_i) - \varphi(\mathbf{x}_i)^\mathsf{T}\varphi(\mathbf{X}_t)^\mathsf{T}(\varphi(\mathbf{X}_t)\varphi(\mathbf{X}_t)^\mathsf{T} + \lambda\mathbf{I})^{-1}\varphi(\mathbf{X}_t)\varphi(\mathbf{x}_i) \\
&\overset{b}{=} \varphi(\mathbf{x}_i)^\mathsf{T}\varphi(\mathbf{x}_i) - \varphi(\mathbf{x}_i)^\mathsf{T}\varphi(\mathbf{X}_t)^\mathsf{T}\varphi(\mathbf{X}_t)(\varphi(\mathbf{X}_t)^\mathsf{T}\varphi(\mathbf{X}_t) + \lambda\mathbf{I})^{-1}\varphi(\mathbf{x}_i) \\
&\overset{c}{=} \varphi(\mathbf{x}_i)^\mathsf{T}\varphi(\mathbf{x}_i) - \varphi(\mathbf{x}_i)^\mathsf{T}\underbrace{(\varphi(\mathbf{X}_t)^\mathsf{T}\varphi(\mathbf{X}_t) + \lambda\mathbf{I} - \lambda\mathbf{I})}_{\mathbf{V}_t}\underbrace{(\varphi(\mathbf{X}_t)^\mathsf{T}\varphi(\mathbf{X}_t) + \lambda\mathbf{I})^{-1}}_{\mathbf{V}_t^{-1}}\varphi(\mathbf{x}_i) \\
&\overset{d}{=} \varphi(\mathbf{x}_i)^\mathsf{T}\varphi(\mathbf{x}_i) - \varphi(\mathbf{x}_i)^\mathsf{T}(\mathbf{I} - \lambda\mathbf{V}_t^{-1})\varphi(\mathbf{x}_i) = \lambda\varphi(\mathbf{x}_i)^\mathsf{T}\mathbf{V}_t^{-1}\varphi(\mathbf{x}_i) = \varphi(\mathbf{x}_i)^\mathsf{T}\mathbf{A}_t^{-1}\varphi(\mathbf{x}_i),
\end{aligned}$$

where in each passage

a) we simply apply the definition of k and φ;

b) we apply Equation 6 with $\varphi(\mathbf{X})$ as $\mathbf{B}$;

c) we add and subtract $\lambda\mathbf{I}$ to highlight the presence of $\mathbf{V}_t$ in the reformulation;

d) we collect λ to replace $\mathbf{V}_t$ with $\mathbf{A}_t \overset{\mathrm{def}}{=} \varphi(\mathbf{X})^\mathsf{T}\varphi(\mathbf{X})/\lambda + \mathbf{I}$ as in (Rasmussen & Williams, 2006, Eq. (2.11)).

Exploiting the fact that all candidate in a batch are identical (i.e., $\mathbf{x}_{\mathtt{fb}(s)+1} = \mathbf{x}_s$), and denoting with $\{\mathbf{x}_j\}_{j=1}^h$ the candidate in each batch we can rewrite $\mathbf{A}_t$ as

$$\begin{aligned}
\mathbf{A}_t &= \mathbf{I} + \lambda^{-1}\varphi(\mathbf{X})^\mathsf{T}\varphi(\mathbf{X}) = \mathbf{I} + \lambda^{-1}\sum_{s=1}^t \varphi(\mathbf{x}_s)\varphi(\mathbf{x}_s)^\mathsf{T} \\
&= \mathbf{I} + \lambda^{-1}\sum_{s=1}^t \varphi(\mathbf{x}_{\mathtt{fb}(s)+1})\varphi(\mathbf{x}_{\mathtt{fb}(s)+1})^\mathsf{T} = \mathbf{I} + \lambda^{-1}\sum_{j=1}^h B_j\varphi(\mathbf{x}_j)\varphi(\mathbf{x}_j)^\mathsf{T} \\
&= \mathbf{I} + \lambda^{-1}\sum_{j=1}^h [\mathbf{W}_h]_{j,j}\varphi(\mathbf{x}_j)\varphi(\mathbf{x}_j)^\mathsf{T} = \mathbf{I} + \lambda^{-1}\varphi(\mathbf{X}_h)^\mathsf{T}\mathbf{W}_h\varphi(\mathbf{X}_h),
\end{aligned}$$

where $\mathbf{W}_h$ is defined as described in Equation 4. Applying now steps $a - d$ in reverse, with the only difference being the application of Equation 6 to $\mathbf{W}_h^{1/2}\varphi(\mathbf{X}_h)$ rather than $\varphi(\mathbf{X}_t)$ in step b, we obtain

$$
\begin{aligned}
\sigma_t^2(\mathbf{x}_i) = \varphi(\mathbf{x}_i)^\intercal \mathbf{A}_t^{-1}\varphi(\mathbf{x}_i) &= \varphi(\mathbf{x}_i)^\intercal(\varphi(\mathbf{X}_h)^\intercal\mathbf{W}_h\varphi(\mathbf{X}_h)/\lambda + \mathbf{I})^{-1}\varphi(\mathbf{x}_i) \\
&\overset{d,c}{=} \varphi(\mathbf{x}_i)^\intercal\varphi(\mathbf{x}_i) - \varphi(\mathbf{x}_i)^\intercal\varphi(\mathbf{X}_t)^\intercal\mathbf{W}_h^{1/2}\mathbf{W}_h^{1/2}\varphi(\mathbf{X}_t)(\varphi(\mathbf{X}_t)^\intercal\mathbf{W}_h^{1/2}\mathbf{W}_h^{1/2}\varphi(\mathbf{X}_t) + \lambda\mathbf{I})^{-1}\varphi(\mathbf{x}_i) \\
&\overset{b}{=} \varphi(\mathbf{x}_i)^\intercal\varphi(\mathbf{x}_i) - \varphi(\mathbf{x}_i)^\intercal\varphi(\mathbf{X}_t)^\intercal\mathbf{W}_h^{1/2}(\mathbf{W}_h^{1/2}\varphi(\mathbf{X}_t)\varphi(\mathbf{X}_t)^\intercal\mathbf{W}_h^{1/2} + \lambda\mathbf{I})^{-1}\mathbf{W}_h^{1/2}\varphi(\mathbf{X}_t)\varphi(\mathbf{x}_i) \\
&\overset{a}{=} \mathrm{k}(\mathbf{x}_i, \mathbf{x}_i) - \mathrm{k}(\mathbf{x}_i, \mathbf{X}_t)\mathbf{W}_h^{1/2}(\mathbf{W}_h^{1/2}\mathbf{K}_t\mathbf{W}_h^{1/2} + \lambda\mathbf{I})^{-1}\mathbf{W}_h^{1/2}\mathrm{k}(\mathbf{X}_t, \mathbf{x}_i) \\
&= \mathrm{k}(\mathbf{x}_i, \mathbf{x}_i) - \mathrm{k}(\mathbf{x}_i, \mathbf{X}_t)(\mathbf{K}_t + \lambda\mathbf{W}_h^{-1})^{-1}\mathrm{k}(\mathbf{X}_t, \mathbf{x}_i),
\end{aligned}
$$

where in the last equality we simply collected $\mathbf{W}_h$ to obtain the formulation of Equation 4. The reasoning for the mean is identical, with one minor difference. After rewriting μ_t in its feature-space view, and applying the fundamental equality

$$
\begin{aligned}
\mu_t(\mathbf{x}_i) = \mathrm{k}(\mathbf{x}_i, \mathbf{X}_t)(\mathbf{K}_t + \lambda\mathbf{I})^{-1}\mathbf{K}_t\mathbf{y}_t \\
= \varphi(\mathbf{x}_i)^\intercal\varphi(\mathbf{X}_t)^\intercal(\varphi(\mathbf{X}_t)\varphi(\mathbf{X}_t)^\intercal + \lambda\mathbf{I})^{-1}\varphi(\mathbf{X}_t)\varphi(\mathbf{X}_t)^\intercal\mathbf{y}_t \\
\overset{a}{=} \varphi(\mathbf{x}_i)^\intercal\varphi(\mathbf{X}_t)^\intercal\varphi(\mathbf{X}_t)(\varphi(\mathbf{X}_t)^\intercal\varphi(\mathbf{X}_t) + \lambda\mathbf{I})^{-1}\varphi(\mathbf{X}_t)^\intercal\mathbf{y}_t \\
\overset{b}{=} \varphi(\mathbf{x}_i)^\intercal\varphi(\mathbf{X}_h)^\intercal\mathbf{W}_h\varphi(\mathbf{X}_h)(\varphi(\mathbf{X}_h)^\intercal\mathbf{W}_h\varphi(\mathbf{X}_h) + \lambda\mathbf{I})^{-1}\varphi(\mathbf{X}_t)^\intercal\mathbf{y}_t,
\end{aligned}
$$

where equality a is once again due to Equation 6, and b is due to the already proven equality $\varphi(\mathbf{X}_t)^\intercal\varphi(\mathbf{X}_t) = \varphi(\mathbf{X}_h)^\intercal\mathbf{W}_h\varphi(\mathbf{X}_h)$. To handle the last remaining term $\varphi(\mathbf{X}_t)^\intercal\mathbf{y}_t$ we can rewrite

$$
\begin{aligned}
\varphi(\mathbf{X}_t)^\intercal\mathbf{y}_t = \sum_{s=1}^{t}\varphi(\mathbf{x}_s)y_s &= \sum_{s=1}^{t}\varphi(\mathbf{x}_{\mathrm{fb}(s)+1})y_s \\
&= \sum_{j=1}^{h}\varphi(\mathbf{x}_j)\sum_{s=t_j+1}^{t_j+B_h}y_s = \sum_{j=1}^{h}\varphi(\mathbf{x}_j)[\mathbf{y}_h]_j = \varphi(\mathbf{X}_h)^\intercal\mathbf{y}_h,
\end{aligned}
$$

where once again t_j is the step before the beginning of the j-th epoch, that is $\mathrm{fb}(t) = t_j$ for all steps in the j-th epoch and the candidate $\mathbf{x}_{t_j+1}$ is the one evaluated multiple times in the j-th epoch. Putting it all together, and re-applying Equation 6 we obtain

$$
\begin{aligned}
\mu_t(\mathbf{x}_i) &= \varphi(\mathbf{x}_i)^\intercal\varphi(\mathbf{X}_h)^\intercal\mathbf{W}_h\varphi(\mathbf{X}_h)(\varphi(\mathbf{X}_h)^\intercal\mathbf{W}_h\varphi(\mathbf{X}_h) + \lambda\mathbf{I})^{-1}\varphi(\mathbf{X}_h)^\intercal\mathbf{y}_h \\
&= \varphi(\mathbf{x}_i)^\intercal\varphi(\mathbf{X}_h)^\intercal\mathbf{W}_h^{1/2}(\mathbf{W}_h^{1/2}\varphi(\mathbf{X}_h)\varphi(\mathbf{X}_h)^\intercal\mathbf{W}_h^{1/2} + \lambda\mathbf{I})^{-1}\mathbf{W}_h^{1/2}\varphi(\mathbf{X}_h)\varphi(\mathbf{X}_h)^\intercal\mathbf{y}_h \\
&= \mathrm{k}(\mathbf{x}_i, \mathbf{X}_h)(\mathbf{K}_h + \lambda\mathbf{W}_h^{-1})^{-1}\mathbf{K}_h\mathbf{y}_h,
\end{aligned}
$$

which concludes the proof of the equivalence between Equation 1 and Equation 4.

In this analysis we made two simplifications: that the step t was at the end of a batch, and that no two candidates were the same in different batches. To relax the first we can just consider extending $\mathbf{W}_h$ to contain not only all past B_j, but also a partial count of the current batch. Similarly $\mathbf{y}_h$ has to be extended to include the partial feedback received during the epoch. Similarly, if the same candidate was selected in two batches j and j', we simply have to merge their contributions in the sum $\sum_{j=1}^{h} B_j\varphi(\mathbf{x}_j)\varphi(\mathbf{x}_j)^\intercal$ e.g., by removing the j-th term and account the j'-th term with $B_j + B_{j'}$ multiplicity.

A.2. Proof of Lemma 4.3

We begin with the following result from (Calandriello et al., 2020).

Proposition A.1 ((Calandriello et al., 2020, Lem. 4)). *For any kernel* k, *set of points* $\mathbf{X}_t$, $\mathbf{x} \in \mathcal{A}$ *and* $t < t'$

$$
1 \le \frac{\sigma_t^2(\mathbf{x})}{\sigma_{t'}^2(\mathbf{x})} \le 1 + \sum_{s=t+1}^{t'}\sigma_t^2(\mathbf{x}_s).
$$

Using Proposition A.1 it is also very easy to show the following known property of the one-step ratio between posteriors (i.e., $t' = t + 1$)

$$1 \le \frac{\sigma_t^2(\mathbf{x})}{\sigma_{t+1}^2(\mathbf{x})} \le 1 + \sigma_t^2(\mathbf{x}_{t+1}) \le 1 + \sigma_0^2(\mathbf{x}_{t+1}) = 1 + \mathrm{k}(\mathbf{x}_{t+1}, \mathbf{x}_{t+1})/\lambda \le 1 + \kappa/\lambda, \qquad (7)$$

where the first inequality is due to Proposition A.1, the second due to the monotonicity of the posterior in t, and the third due to our assumption $\mathrm{k}(\mathbf{x}, \mathbf{x}) \le \kappa^2$.

Applying Proposition A.1 to our setting, where $\mathbf{x}_s$ does not change for the whole batch, and our epoch termination rule from Algorithm 2 we obtain the following corollary

Corollary A.2. *During epoch h, let t_h be the step before the beginning of the batch, and let $\mathbf{x}_{t_h+1}$ be the candidate selected for the whole batch at step t_h+1. Then for any kernel k, set of points $\mathbf{X}_{t_h}$, $\mathbf{x} \in \mathcal{A}$, and $t_h+1 = \mathrm{fb}(t)+1 \le t \le \mathrm{fb}(t)+B_h$ we have*

$$1 \le \frac{\sigma_{\mathrm{fb}(t)}^2(\mathbf{x})}{\sigma_t^2(\mathbf{x})} \le 1 + B_h \sigma_{\mathrm{fb}(t)}^2(\mathbf{x}_{\mathrm{fb}(t)+1}).$$

Moreover, selecting $B_h = \lfloor (C^2 - 1)/\sigma_{t_h}^2(\mathbf{x}_{t_h+1}) \rfloor$ we have $\frac{\sigma_{\mathrm{fb}(t)}(\mathbf{x})}{\sigma_t(\mathbf{x})} \le C$.

To bound the number of epochs and prove Lemma 4.3 we follow a blueprint from (Calandriello et al., 2020).

Proof of Lemma 4.3. To bound the number of epoch we start from the fundamental inequality based on the choice of B_h and the properties of the floor function. Consider an arbitrary B_j,

$$B_j \ge \frac{C^2 - 1}{\sigma_{t_j}^2(\mathbf{x}_{t_j+1})} - 1$$

$$\Rightarrow \sigma_{t_j}^2(\mathbf{x}_{t_j+1})(B_j + 1) \ge C^2 - 1$$

$$\Rightarrow 2\sigma_{t_j}^2(\mathbf{x}_{t_j+1})B_j \ge C^2 - 1.$$

Note that due to the construction of the batch $\sigma_{t_j}^2(\mathbf{x}_{t_j+1}) = \sigma_{t_j}^2(\mathbf{x}_{t_j+2}) = \cdots = \sigma_{t_j}^2(\mathbf{x}_{t_j+B_j})$, and therefore $\sigma_{t_j}^2(\mathbf{x}_{t_j+1})B_j = \sum_{s=t_j+1}^{t_j+B_j} \sigma_{t_j}^2(\mathbf{x}_s)$. Summing across batches up to batch h we have

$$h(C^2 - 1) \le 2\sum_{j=1}^h \sigma_{t_j}^2(\mathbf{x}_{t_j+1})B_j = 2\sum_{j=1}^h \sum_{s=t_j+1}^{t_j+B_j} \sigma_{t_j}^2(\mathbf{x}_s)$$

$$= 2\sum_{j=1}^h \sum_{s=t_j+1}^{t_j+B_j} \frac{\sigma_{t_j}^2(\mathbf{x}_s)}{\sigma_{s-1}^2(\mathbf{x}_s)}\sigma_{s-1}^2(\mathbf{x}_s) \overset{\text{Corollary A.2}}{\le} 2\sum_{j=1}^h \sum_{s=t_j+1}^{t_j+B_j} C^2\sigma_{s-1}^2(\mathbf{x}_s)$$

$$= 2C^2\sum_{s=1}^T \sigma_{s-1}^2(\mathbf{x}_s).$$

Now that we obtain the sum of posterior variances we need to connect this quantity to γ_T. In order to do this we will use a standard bound (see e.g., (Hazan et al., 2006)) on the summation

$$\sum_{s=1}^T \sigma_s^2(\mathbf{x}_s) \le \log\det(\mathbf{I} + \mathrm{k}(\mathbf{X}_t, \mathbf{X}_t)/\lambda) = 2\gamma(\mathbf{X}_t, \mathbf{y}_t) \le 2\gamma_T.$$

All that is left is to use Equation 7 to bound $\sigma_{s-1}^2(\mathbf{x}_s)$ in terms of $\sigma_s^2(\mathbf{x}_s)$ and rearrange appropriately all the derived results to obtain

$$h \le \frac{2C^2}{C^2 - 1}\sum_{s=1}^T \sigma_{s-1}^2(\mathbf{x}_s) \le \frac{2C^2}{C^2 - 1}(1 + \kappa^2/\lambda)\sum_{s=1}^T \sigma_s^2(\mathbf{x}_s) \le \frac{4C^2}{C^2 - 1}(1 + \kappa^2/\lambda)\gamma_T \le \mathcal{O}\left(\frac{C^2}{C^2 - 1}\gamma_T\right)$$

$$\square$$

A.3. Proof of Lemma 4.2

The proof of this result follows directly from the sketch of proof in Section 4.1, combined with Corollary A.2 here in the appendix.

A.4. Proofs of Section 4.3

We indicate here how each original regret proofs can be modified to obtain Theorems 4.4 to 4.6. All proof will depend on a standard blueprint (Srinivas et al., 2010; Abbasi-Yadkori et al., 2011; Chowdhury & Gopalan, 2017; Calandriello et al., 2020) that we present first. For simplicity we also assume again that step T is exactly at the end of batch h. This is without loss of generality as we can always artificially truncate the current batch at step T for the sake of the analysis without increasing the regret. (i.e., Corollary A.2 and all other results will still hold).

First we leverage results from the original analyses to show that the instantaneous regret r_{t_j+1} at the beginning of batch j is bounded (i.e., Equation 5) as

$$r_{t_j+1} \leq G_{t_j} \sigma_{t_j}(\mathbf{x}_{t_j+1}).$$

Moreover, the same candidate is selected at all steps in the batch, so it holds as well that for all $t' \in [t_j+1, t_j+B_j]$ we have

$$r_{t'} = r_{t_j+1} \leq G_{t_j} \sigma_{t_j}(\mathbf{x}_{t_j+1}) = G_{t_j} \sigma_{t_j}(\mathbf{x}_{t'})$$

Then, leveraging again Corollary A.2 to bound the posterior ratios we obtain

$$\sum_{s=t_j+1}^{t_j+B_j} r_s \leq G_{t_j} \sum_{s=t_j+1}^{t_j+B_j} \sigma_{t_j}(\mathbf{x}_s) \overset{\text{Corollary A.2}}{\leq} G_{t_j} C \sum_{s=t_j+1}^{t_j+B_j} \sigma_{s-1}(\mathbf{x}_s).$$

Finally, using the fact that G_t is non-decreasing, and summing across batches

$$
\begin{aligned}
R_T = \sum_{t=1}^{T} r_t = \sum_{j=1}^{h} \sum_{s=t_j+1}^{t_j+B_j} r_s \\
\leq \sum_{j=1}^{h} G_{t_j} \sum_{s=t_j+1}^{t_j+B_j} \sigma_{t_j}(\mathbf{x}_s) \leq \sum_{j=1}^{h} G_{t_j} C \sum_{s=t_j+1}^{t_j+B_j} \sigma_{s-1}(\mathbf{x}_s) \\
\leq G_T C \sum_{j=1}^{h} \sum_{s=t_j+1}^{t_j+B_j} \sigma_{s-1}(\mathbf{x}_s) \leq G_T C \sum_{s=1}^{T} \sigma_{s-1}(\mathbf{x}_s).
\end{aligned}
$$

Putting it all together and expressing everything in terms of γ_T we derive

$$
\begin{aligned}
R_T &\leq G_T C \sum_{s=1}^{T} \sigma_{s-1}(\mathbf{x}_s) \\
&\overset{a}{\leq} G_T C \sqrt{T} \sqrt{\sum_{s=1}^{T} \sigma_{s-1}^2(\mathbf{x}_s)} \\
&\overset{b}{\leq} G_T C \sqrt{T} \sqrt{\left(1 + \frac{\kappa^2}{\lambda}\right) \sum_{s=1}^{T} \sigma_s^2(\mathbf{x}_s)} \\
&\overset{c}{\leq} G_T C \sqrt{T} \sqrt{\left(1 + \frac{\kappa^2}{\lambda}\right) \gamma_T},
\end{aligned}
$$

where a is due to Cauchy-Schwarz inequality, b due to Equation 7, and c due to the usual bounding of posterior variances with the $\log \det$ and information gain (Hazan et al., 2006). Now that we have this blueprint, all that is left is to look at results in the literature on how G_t can be bounded under different assumptions and acquisition functions.

MINI-GP-UCB	$\sigma = 455.56$, $C = 1.1$
MINI-GP-EI	$\sigma = 455.56$, $C = 1.1$
BBKB	$\sigma = 277.78$, $C = 1.1$
BKB	$\sigma = 455.56$
GP-UCB	$\sigma = 500.00$
GP-BUCB	$\sigma = 455.56$, $C = 1.1$
ε-GREEDY	$a = 1$, $b = 0.5$

Table 1: Optimal hyper-parameters found for the NAS-bench experiment.

Proof of Theorem 4.4. For this theorem we leverage the assumptions $\|f\| \leq F$ with $u_t^{\text{MINI-GP-UCB}}$ as acquisition function. Already Theorem 1 from (Chowdhury & Gopalan, 2017) showed that if $\lambda = \xi^2$ and β_t is tuned as $\beta_t = \Theta(\sqrt{\log \det(\mathbf{K}_t/\xi^2 + \mathbf{I}) + \log(1/\delta)} + F)$ we obtain that $G_t \leq \beta_t$ suffices to guarantee Equation 5. Note however that in our case we can efficiently compute $\log \det(\mathbf{K}_t/\xi^2 + \mathbf{I})$ leveraging the few unique candidates. In particular, since we only need to compute β_{t_h} at the beginning of a batch, we can rewrite

$$
\begin{aligned}
\log \det(\mathbf{K}_t/\xi^2 + \mathbf{I}) &= \log \det(\varphi(\mathbf{X}_t)\varphi(\mathbf{X}_t)^\top/\xi^2 + \mathbf{I}) \\
&\overset{a}{=} \log \det(\varphi(\mathbf{X}_t)^\top\varphi(\mathbf{X}_t)/\xi^2 + \mathbf{I}) \\
&\overset{b}{=} \log \det(\varphi(\mathbf{X}_h)^\top\mathbf{W}_h\varphi(\mathbf{X}_h)/\xi^2 + \mathbf{I}) \\
&\overset{c}{=} \log \det(\mathbf{W}_h^{1/2}\varphi(\mathbf{X}_h)^\top\varphi(\mathbf{X}_h)\mathbf{W}_h^{1/2}/\xi^2 + \mathbf{I}) \\
&= \log \det(\mathbf{W}_h^{1/2}\mathbf{K}_h\mathbf{W}_h^{1/2}/\xi^2 + \mathbf{I}),
\end{aligned}
$$

where a is due to Sylvester's determinant identity, b is our usual re-writing, and c is Sylvester's determinant identity again. Not that since this a strict equality, we still have that at the last batch h

$$
\log \det(\mathbf{W}_h^{1/2}\mathbf{K}_h\mathbf{W}_h^{1/2}/\xi^2 + \mathbf{I}) = \log \det(\mathbf{K}_T/\xi^2 + \mathbf{I}) \leq \gamma_T,
$$

which gives the second half of the theorem. $\qquad\square$

Proof of Theorem 4.5. This is a direct consequence of Theorem 1 from (Srinivas et al., 2010). In particular, they once again show that $G_t \leq \beta_t$ suffices to guarantee Equation 5. $\qquad\square$

Proof of Theorem 4.6. For this combination our starting point is Equation (38) in (Wang & de Freitas, 2014), which states that for

$$
\nu_t = \sqrt{\log \det(\mathbf{K}_t/\lambda + \mathbf{I}) + \sqrt{\log \det(\mathbf{K}_t/\lambda + \mathbf{I}) \log(t/\delta)} + \log(t/\delta)}, \qquad \lambda = \xi^2,
$$

running a standard GP-Opt loop (i.e., Algorithm 1) with $u_t = u_t^{\text{GP-EI}}$ guarantees

$$
r_t \leq \tilde{\mathcal{O}}\left(\left(\sqrt{F^2 + \gamma_t} + \nu_t\right)\sigma_{t-1}(\mathbf{x}_t)\right),
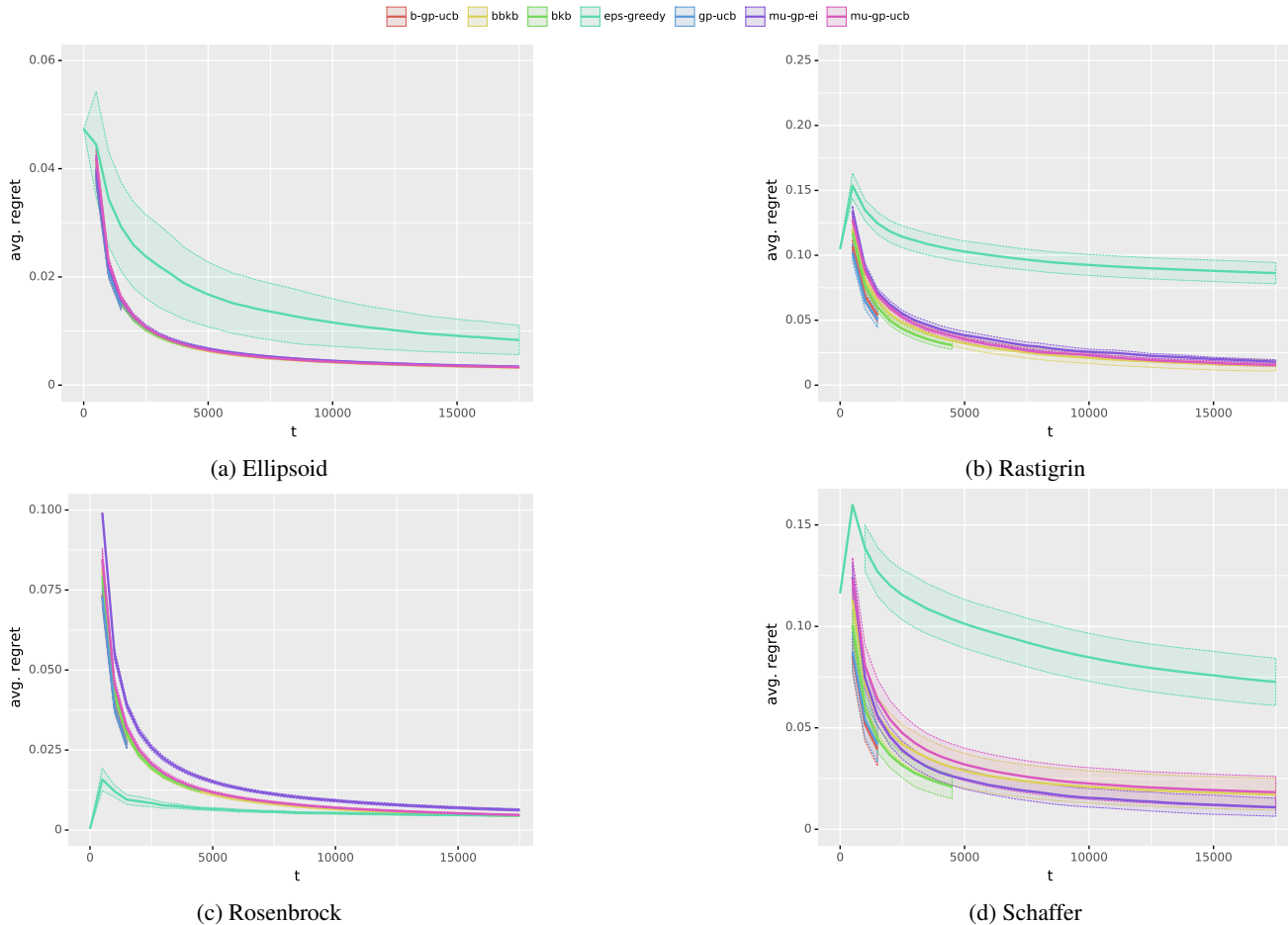$$

where we greatly simplified their notation by ignoring constant and logarithmic terms. Noticing now that at each batch j the values of β_j and ν_{t_j} are equal thanks again to the equivalence $\log \det(\mathbf{K}_t/\lambda + \mathbf{I}) = \log \det(\mathbf{W}_h^{1/2}\mathbf{K}_h\mathbf{W}_h^{1/2}/\lambda + \mathbf{I})$, we can obtain a bound on G_t that satisfies Equation 5 as

$$
G_t \leq \tilde{\mathcal{O}}\left(\left(\sqrt{F^2 + \gamma_t} + \beta_t\right)\right) \leq \tilde{\mathcal{O}}\left(\left(\sqrt{F^2 + \gamma_t} + \sqrt{\gamma_t}\right)\right) \leq \tilde{\mathcal{O}}\left((F + \sqrt{\gamma_t})\right).
$$

Putting this together with the usual blueprint to bound the error incurred by batching and the bound on the sum of posterior variances with γ_T we obtain our result. $\qquad\square$

B. Extended experimental results

We report here additional details on the NAS-bench experiments, as well as an evaluation on synthetic functions.

(a) Ellipsoid

(b) Rastigrin

(c) Rosenbrock

(d) Schaffer

Figure 3: R_t/t against step t

B.1. Additional details on NAS-bench experiments

All the implementations are based on the code released by (Calandriello et al., 2020) for BBKB and BKB, available at https://github.com/luigicarratino/batch-bkb. For each algorithm we run the experiment 40 times with different seeds, and select the hyper-parameters that achieves the lowest average regret. Hyper-parameters for all algorithms based on GPs are searched among

$$\sigma^2 = \{100.00, 144.45, 188.89, 233.33, 277.78, 322.22, 366.67, 411.11, 455.56, 500.00\}$$
$$C = \{1.1, 1.2\}$$

where C is the threshold used for batching, and σ is the bandwidth of the Gaussian kernel. For ε-GREEDY the exploration rate is is optimized as $\varepsilon = a/t^b$ over $a \in \{0.1, 1, 10\}$ and $b \in \{1/3, 1/2, 1, 2\}$. The empirically optimal values are reported in Table 1.

B.2. Evaluation on synthetic functions

We also evaluate our method on common benchmark functions from the noisy BBOB benchmark suite (Hansen et al., 2010). In particular we focus on the Rosenbrock (f_{104}), ellipsoid (f_{116}), and Schaffer (f_{122}) functions, all with moderate Gaussian noise. We also include an extra function, the separable Rastigrin function, to add another more complex separable function with the ellipsoid.

All functions are defined on $\mathbb{R}^3$, with each coordinate split into 21 sections with the same length by 22 evenly spaced points placed between $[-5, 5]$. The resulting discrete grid of points represent our candidate set and contains $|\mathcal{A}| = 22^3 = 10648$

unique candidates. Similarly to the NAS-bench experiments, σ of the Gaussian kernel, C and the ε-GREEDY parameters are selected optimally over a grid search.

The results on regret against steps are reported in Figures 3a to 3d and on regret against time in Figures 4a to 4d. As we can see, our approach achieve comparable runtime and regret to other state of the art GP-Opt methods. However, on more complex problem like the Rosenbreck function, even the flexibility of a GP is not capable of capturing the underlying shape of the optimization problem, and all GP-Opt methods, including ours, only perform roughly as well as a tuned ε-GREEDY exploration.
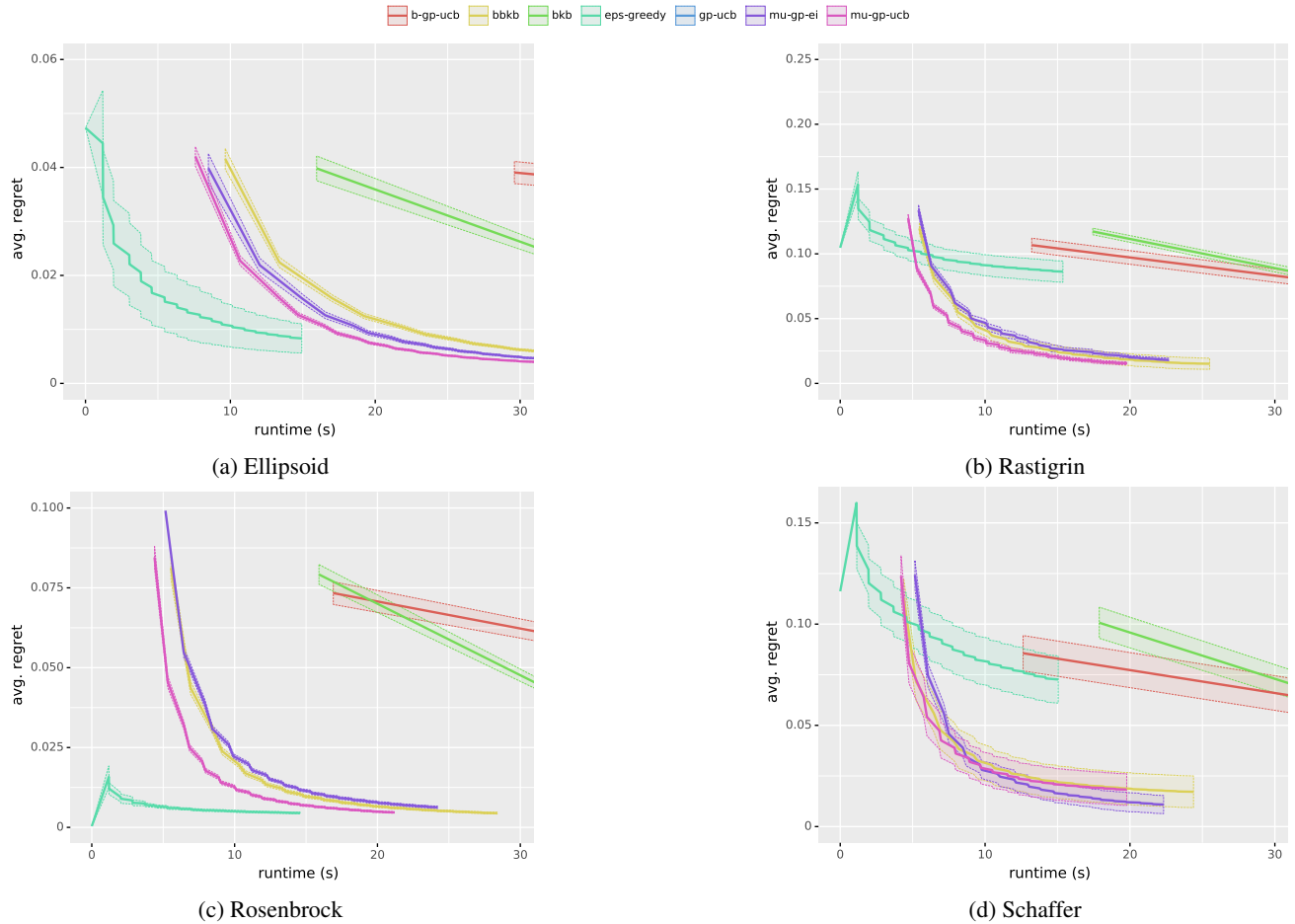

Figure 4: R_t/t against time (in seconds)

References

Elad Hazan, Adam Kalai, Satyen Kale, and Amit Agarwal. Logarithmic regret algorithms for online convex optimization. In Conference on Learning Theory. Springer, 2006.

Hansen, Nikolaus and Auger, Anne and Finck, Steffen and Ros, Raymond. Real-parameter black-box optimization benchmarking 2010: Experimental setup. INRIA, 2010.