Chapter 3

Theoretical Contributions

3.1 Introduction

Many problems require optimizing an unknown reward function, from which we can only obtain noisy observations. A central challenge is choosing actions that both explore (estimate) the function and exploit our knowledge about likely high reward regions. Carefully calibrating this exploration–exploitation tradeoff is especially important in cases where the experiments are costly in some sense, e.g., when each experiment takes a long time to perform and the time window available for experiments is short. In some such settings, it may be desirable to run several experiments in parallel. By parallelizing the experiments, substantially more information can be gathered in the same time-frame; however, future actions must be chosen without the benefit of intermediate results. One might conceptualize these problems as choosing "batches" of experiments to run simultaneously. The challenge is to assemble batches of experiments which both explore the function and exploit what are currently known to be high-performing regions.

Two key, interrelated questions arise: the *computational* question of how one should efficiently choose, out of the combinatorially large set of possible batches, those that are most effective; and the *statistical* question of how the algorithm's performance depends on the size of the batches (i.e., the degree of informational parallelism). In this chapter, we address these questions by presenting GP-BUCB and GP-AUCB; these are novel, efficient algorithms for selecting batches of experiments in the Bayesian optimization setting where the reward function is modeled as a sample from a Gaussian process prior (or has low norm in the associated Reproducing Kernel Hilbert Space).

In more detail, we provide the following main contributions:

- We introduce GP-BUCB, a novel algorithm for selecting actions to maximize reward in large-scale exploration-exploitation problems while accommodating parallel or batch execution of

The work in this chapter has been submitted to the Journal of Machine Learning Research as Desautels, Krause, and Burdick, "Parallelizing Exploration-Exploitation Tradeoffs In Bayesian Global Optimization", itself a substantial expansion of Desautels et al. (2012), published at ICML 2012.

the actions and the consequent observation of their reward. GP-BUCB may also be used in the setting of a bounded delay between initiation of an action and observation of its reward.

- We also introduce GP-AUCB, an algorithm which adaptively exploits parallelism to choose batches of actions, the sizes of which are limited by the conditional mutual information gained therein; this limit is such that the batch sizes are small when actions are selected for which the algorithm knows relatively little about the reward and batch sizes are large when the reward function is well known for the actions selected. We show that this adaptive parallelism is well-behaved and can easily be constrained using pre-defined limits.
- We prove sublinear bounds on the cumulative regret incurred by algorithms of a general class, including GP-BUCB and GP-AUCB, and provide corollary extensions to each of these two algorithms, thus bounding their rates of convergence.
- For some common kernels, we show that if the problem is initialized by making observations corresponding to an easily selected and provably bounded set of queries, the regret of GP-BUCB can be bounded to a constant multiplicative factor of the known regret bounds of the sequential GP-UCB algorithm. This asymptotic post-initialization guarantee is *independent* of batch size B so long as B grows at most polylogarithmically in T, the number of queries to be selected in total. This implies (near-)linear informational speedup through parallelism.
- We demonstrate how execution of many UCB algorithms, including the GP-UCB, GP-BUCB, and GP-AUCB algorithms, can be drastically accelerated by *lazily evaluating* the posterior variance. This technique does not result in any loss in accuracy.
- We evaluate GP-BUCB and GP-AUCB on several synthetic benchmark problems, as well as two real data sets, respectively related to automated vaccine design and therapeutic spinal cord stimulation. We show that GP-BUCB and GP-AUCB are competitive with state of the art heuristics for parallel Bayesian optimization, and that under certain circumstances, GP-BUCB is competitive with sequential action selection under GP-UCB, despite having the disadvantage of delayed feedback.
- We consider more complex notions of execution cost in the batch and delay settings and identify areas of this cost and performance space where our algorithms make favorable tradeoffs and are therefore especially suitable for employment.

In the remainder of the chapter, we begin by formally describing the problem setting (Section 3.2). In the next section, we describe the GP-BUCB algorithm, present the main regret bound, which applies to a general class of algorithms using an upper confidence bound decision rule, and present corollaries bounding the regret of GP-BUCB and initialized GP-BUCB (Section 3.3). We extend this analysis to GP-AUCB, providing a regret bound for that algorithm, discuss different possible stopping conditions for similar algorithms, and introduce the notion of lazy variance calculations (Section 3.4). We compare our algorithms' performance with each other and with several other

algorithms across a variety of problem instances, including two real data sets (Section 3.6). Finally, we present our conclusions (Section 3.7).

3.2 Problem Setting and Background

We wish to make a sequence of decisions (or equivalently, take actions) $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T \in D$, where D is the *decision set*, which is often (but not necessarily) a compact subset of $\mathbb{R}^d$, and the subscript denotes the *round* in which that decision was made; each round is an opportunity for the algorithm to make one decision. For each decision $\boldsymbol{x}_t$, we observe noisy scalar reward $y_t = f(\boldsymbol{x}_t) + \varepsilon_t$, where $f : D \to \mathbb{R}$ is an unknown function modeling the expected payoff $f(\boldsymbol{x})$ for each decision $\boldsymbol{x}$. For now we assume that the noise variables ε_t are i.i.d. Gaussian with known variance σ_n^2, i.e., $\varepsilon_t \sim \mathcal{N}(0, \sigma_n^2),\ \forall t \geq 1$. This assumption will be relaxed later. If the decisions $\boldsymbol{x}_t$ are made one at a time, each with the benefit of all observations $y_1, \ldots, y_{t-1}$ corresponding to previous actions $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}$, we shall refer to this case as the *strictly sequential* setting. In contrast, the main problem tackled in this chapter is the challenging setting where $\boldsymbol{x}_t$ may only depend on $y_1, \ldots, y_{t'}$, for some $t' < t-1$.

In selecting these decisions, we wish to maximize the cumulative reward $\sum_{t=1}^{T} f(\boldsymbol{x}_t)$, or equivalently minimize the cumulative *regret* $R_T = \sum_{t=1}^{T} r_t$, where $r_t = [f(\boldsymbol{x}^*) - f(\boldsymbol{x}_t)]$ and $\boldsymbol{x}^* \in \boldsymbol{X}^* = \operatorname{argmax}_{\boldsymbol{x} \in D} f(\boldsymbol{x})$ is an optimum decision (assumed to exist, but not necessarily to be unique). In experimental design, D might be the set of possible stimuli that can be applied, and $f(\boldsymbol{x})$ models the response to stimulus $\boldsymbol{x} \in D$. By minimizing the regret, we ensure progress towards the most effective stimulus uniformly over T. In fact, the average regret, R_T/T, is a natural upper bound on the suboptimality of the best stimulus considered so far, i.e., $R_T/T \geq \min_t [f(\boldsymbol{x}^*) - f(\boldsymbol{x}_t)]$ (often called the *simple regret*, Bubeck et al., 2009).

3.2.1 The Problem: Parallel or Delayed Selection

In many applications, at time τ, we wish to select a *batch* of decisions, e.g., $\boldsymbol{x}_\tau, ..., \boldsymbol{x}_{\tau+B-1}$, where B is the size of the batch, to be evaluated in parallel. One natural application is the design of *high-throughput* experiments, where several experiments are performed in parallel, but feedback is only received after the experiments have concluded. In other settings, feedback is delayed. In both situations, decisions are selected sequentially, but when making the decision $\boldsymbol{x}_t$ in round t, we can only make use of the feedback obtained in rounds $1, \ldots, t'$, for some $t' \leq t-1$. Formally, we assume there is some mapping $\text{fb} : \mathbb{N} \to \mathbb{N}_0$ such that $\text{fb}[t] \leq t-1,\ \forall t \in \mathbb{N}$, and when making a decision at time t, we can use feedback up to and including round $\text{fb}[t]$. If $\text{fb}[t] = 0$, no observation information is available.

This framework can model a variety of realistic scenarios. Setting $B = 1$ and $\text{fb}[t] = t-1$ corresponds to the non-delayed, strictly sequential setting in which a single action is selected and

the algorithm waits until the corresponding observation is returned before selecting the succeeding action. The "simple batch" setting in which we wish to select batches of size B can be captured by setting $\text{fb}[t]_{SB} = \lfloor (t-1)/B \rfloor B$, i.e.,

$$\text{fb}[t]_{SB} = \begin{cases} 0 & : t \in \{1, \ldots, B\} \\ B & : t \in \{B+1, \ldots, 2B\} \\ 2B & : t \in \{2B+1, \ldots, 3B\} \\ \vdots & \end{cases}.$$

Note that in the batch case, the time indexing within the batch is a matter of algorithmic construction, since the batch is built in a sequential fashion, but actions are initiated and observations are received simultaneously. If we wish to formalize the problem of selecting actions when feedback from those actions is delayed by exactly B rounds, the *simple delay* setting, we can simply define this feedback mapping as $\text{fb}[t]_{SD} = \max\{t - B, 0\}$. Note that in both the simple batch and delay cases, $B = 1$ is the strictly sequential case. In comparing these two simple cases for the same value of B, we observe that $\text{fb}[t]_{SB} \geq \text{fb}[t]_{SD}$, that is, the set of observations available in the simple batch case for making the tth decision is always at least as large as in the simple delay case, suggesting that the delay case is in some sense "harder" than the batch case. As we will see, however, the regret bounds for each algorithm presented in this chapter are established based on the maximal number of pending observations (i.e, those which have been initiated, but are still incomplete), which is $B - 1$ in both of these settings, resulting in unified proofs and regret bounds for the two cases.

More complex cases may also be handled by GP-BUCB. For example, we may also be interested in executing B experiments in parallel, but the duration of an experiment may be variable, and we can start a new experiment as soon as one finishes; this translates to having a queue of pending observations of some finite size B. Since we only select a new action when the queue is not full, there can be at most $B - 1$ actions waiting in the queue at the time any action is selected, as in the simple batch and delay cases. Again, the maximum number of pending observations is the key to bounding the regret, though the variable delay for individual observations requires some subtlety with the feedback mapping's specification.

Even in complex cases, one quantity intuitively used in describing the difficulty of the problem instance is the constant B. In this light, while developing GP-BUCB and initialized GP-BUCB in Sections 3.3.4 and 3.3.5, we only assume that the mapping $\text{fb}[t]$ is specified as part of the problem instance (possibly chosen adversarially) and $t - \text{fb}[t] \leq B$ for some known constant B.

3.2.2 Modeling f via Gaussian Processes (GPs)

Regardless of when feedback is obtained, if we are to turn finite numbers of observations into useful inference about the payoff function f, we will have to make assumptions about its structure. In particular, for large (possibly infinite) decision sets D there is no hope to do well, i.e., incur little regret or even simply converge to an optimal decision, if we do not make any assumptions. For good performance, we must choose a regression model which is both simple enough to be learned and expressive enough to capture the relevant behaviors of f. One effective formalism is to model f as a sample from a Gaussian process (GP) prior, as discussed in Section 2.4. Briefly recapitulated, a GP is a probability distribution across a class of – typically smooth – functions, which is parameterized by a kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$, which characterizes the smoothness of f, and a mean function $\mu(\boldsymbol{x})$. For notational convenience, we assume $\mu(\boldsymbol{x}) = 0$, without loss of generality, and we additionally assume that $k(\boldsymbol{x}, \boldsymbol{x}) \leq 1,\ \forall \boldsymbol{x} \in D$. We write $f \sim \mathcal{GP}(\mu, k)$ to denote that we model f as sampled from such a GP. If noise is i.i.d. Gaussian and the distribution of f is conditional on a vector of observations $\boldsymbol{y}_{1:t-1} = [y_1, ..., y_{t-1}]^T$ corresponding to decisions $\boldsymbol{X}_{t-1} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_{t-1}]^T$, one obtains a Gaussian posterior distribution $f(\boldsymbol{x})|\boldsymbol{y}_{1:t-1} \sim \mathcal{N}(\mu_{t-1}(\boldsymbol{x}), \sigma^2_{t-1}(\boldsymbol{x}))$ for each $\boldsymbol{x} \in D$, where

$$\mu_{t-1}(\boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{X}_{t-1})[K(\boldsymbol{X}_{t-1}, \boldsymbol{X}_{t-1}) + \sigma_n^2 I]^{-1}\boldsymbol{y}_{1:t-1} \text{ and} \tag{3.1}$$

$$\sigma^2_{t-1}(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) - K(\boldsymbol{x}, \boldsymbol{X}_{t-1})[K(\boldsymbol{X}_{t-1}, \boldsymbol{X}_{t-1}) + \sigma_n^2 I]^{-1}K(\boldsymbol{x}, \boldsymbol{X}_{t-1})^T, \tag{3.2}$$

where $K(\boldsymbol{x}, \boldsymbol{X}_{t-1})$ denotes the row vector of kernel evaluations between $\boldsymbol{x}$ and the elements of $\boldsymbol{X}_{t-1}$, the set of decisions taken in the past, and $K(\boldsymbol{X}_{t-1}, \boldsymbol{X}_{t-1})$ is the matrix of kernel evaluations, where $[K(\boldsymbol{X}_{t-1}, \boldsymbol{X}_{t-1})]_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j),\ \forall\, \boldsymbol{x}_i, \boldsymbol{x}_j \in \boldsymbol{X}_{t-1}$, i.e., the covariance matrix of the values of f over the set so far observed. Since Equations (3.1) and (3.2) can be computed efficiently, closed-form posterior inference is computationally tractable in a GP distribution via linear algebraic operations.

3.2.3 Conditional Mutual Information

A number of information theoretic quantities will be essential to the analysis of the algorithms presented in this chapter. In particular, the quantity

$$\gamma_T = \max_{A \subseteq D,\, |A| \leq T} I(f; \mathbf{y}_A) \tag{3.3}$$

is the maximum mutual information between the payoff function f and observations $\mathbf{y}_A$ of any set $A \subseteq D$ of the T decisions evaluated up until time T. For a GP, the mutual information $I(f; \mathbf{y}_A)$ is

$$I(f; \mathbf{y}_A) = H(\mathbf{y}_A) - H(\mathbf{y}_A \,|\, f) = \frac{1}{2} \log \left| \mathbf{I} + \sigma_n^{-2} K(A, A) \right|,$$

where $K(A, A)$ is the covariance matrix of the values of f at the elements of the set A, $H(\mathbf{y}_A)$ is the differential entropy of the probability distribution over the set of observations $\mathbf{y}_A$, and $H(\mathbf{y}_A \,|\, f)$ is the corresponding value when the distribution over $\mathbf{y}_A$ is conditioned on f, or equivalently, $f(A)$. The *conditional mutual information* for observations $\mathbf{y}_A$ given previous observations $\mathbf{y}_S$, is defined (for two finite sets $A, S \subseteq D$) as

$$I(f; \boldsymbol{y}_A \mid \boldsymbol{y}_S) = H(\boldsymbol{y}_A \mid \boldsymbol{y}_S) - H(\boldsymbol{y}_A \mid f).$$

The conditional mutual information gain from observations $\boldsymbol{y}_A$ of the set of actions A can also be calculated as a sum of the marginal information gains of each observation in the set; conditioned on $\boldsymbol{y}_S$, and for $A = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T\}$ this sum is

$$I(f; \boldsymbol{y}_A \mid \boldsymbol{y}_S) = \sum_{t'=1}^{T} \log\left(1 + \sigma_n^{-2} \sigma^2_{t'-1}(\boldsymbol{x}_{t'})\right), \tag{3.4}$$

where the term $\sigma^2_{t'-1}(\boldsymbol{x}_{t'})$ is the posterior variance over $f(\boldsymbol{x}_{t'})$, conditioned on $\boldsymbol{y}_S$ and $y_1, ..., y_{t'-1}$. It is important to note that $\sigma^2_{t'-1}(\boldsymbol{x}_{t'})$ is independent of the values of the observations. Since the sum's value can thus be calculated without making the observations (i.e., during the course of assembling a batch), it is possible to calculate the mutual information which will be gained from any hypothetical set of observations.

3.2.4 The GP-UCB approach

Modeling f as a sample from a GP has the major advantage that the predictive uncertainty can be used to guide exploration and exploitation. While several heuristics, such as Expected Improvement (Mockus et al., 1978) and Most Probable Improvement (Mockus, 1989) have been effectively employed in practice, nothing is known about their convergence properties in the case of noisy observations. Srinivas et al. (2010), guided by the success of upper-confidence based sampling approaches for multi-armed bandit problems (Auer, 2002), analyzed the *Gaussian process Upper Confidence Bound (GP-UCB)* selection rule,

$$\boldsymbol{x}_t = \underset{\boldsymbol{x} \in D}{\operatorname{argmax}} \left[\mu_{t-1}(\boldsymbol{x}) + \alpha_t^{1/2} \sigma_{t-1}(\boldsymbol{x})\right]. \tag{3.5}$$

This decision rule uses α_t, a domain-specific time-varying parameter, to trade off exploitation (sampling $\boldsymbol{x}$ with high mean) and exploration (sampling $\boldsymbol{x}$ with high standard deviation). Srinivas et al. (2010) showed that, with proper choice of α_t, the cumulative regret of GP-UCB grows sublinearly for many commonly used kernel functions. This algorithm is presented in simplified pseudocode as Algorithm 1.

Algorithm 1 GP-UCB

Input: Decision set D, GP prior μ_0, σ_0, kernel function $k(\cdot,\cdot)$
for $t = 1, 2, \ldots, T$ **do**
  Choose $\boldsymbol{x}_t = \text{argmax}_{\boldsymbol{x}\in D}[\mu_{t-1}(\boldsymbol{x}) + \alpha_t^{1/2}\sigma_{t-1}(\boldsymbol{x})]$
  Compute $\sigma_t(\cdot)$, e.g., via Equation (3.2)
  Obtain $y_t = f(\boldsymbol{x}_t) + \varepsilon_t$
  Perform Bayesian inference to obtain $\mu_t(\cdot)$, e.g., via Equation (3.1)
end for

Implicit in the definition of the GP-UCB decision rule, Equation (3.5), is the corresponding confidence interval,

$$C_t^{\text{seq}}(\boldsymbol{x}) \equiv \left[\mu_{t-1}(\boldsymbol{x}) \pm \alpha_t^{1/2}\sigma_{t-1}(\boldsymbol{x})\right], \tag{3.6}$$

where this confidence interval's upper confidence bound is the value of the argument of the decision rule. For this (or any) confidence interval, we will refer to the difference between the uppermost limit and the lowermost, here $w = 2\alpha_t^{1/2}\sigma_{t-1}(\boldsymbol{x})$, as the *width* w. This confidence interval is based on the posterior over f given $\boldsymbol{y}_{1:t-1}$; a new confidence interval is created for round $t+1$ after adding y_t to the set of observations. Srinivas et al. (2010) carefully select α_t such that a union bound over all $t \geq 1$ and $\boldsymbol{x} \in D$ yields a high-probability guarantee of confidence interval correctness; it is this guarantee which enables the construction of high-probability regret bounds. Using this guarantee, Srinivas et al. (2010) then prove that the cumulative regret of the GP-UCB algorithm can be bounded (up to logarithmic factors) as $R_T = O^*(\sqrt{T\alpha_T\gamma_T})$, where α_T is the confidence interval width multiplier described above and γ_T is the maximum mutual information between the payoff function f and the observations $\boldsymbol{y}_{1:T}$. For many commonly used kernel functions, Srinivas et al. (2010) show that γ_T grows sublinearly and α_T only needs to grow polylogarithmically in T, implying that R_T is also sublinear; thus $R_T/T \to 0$ as $T \to \infty$, i.e., GP-UCB is a no-regret algorithm.

Motivated by the strong theoretical and empirical performance of GP-UCB, we explore generalizations to batch and parallel selection (i.e., $B > 1$). One naïve approach would be to update the GP-UCB score, Equation (3.5), only once new feedback becomes available, but this algorithm would simply select the same action at each time step between acquisition of new observations, leading to limited exploration. To encourage more exploration, one may instead require that no decision is selected twice within a batch (i.e., simply rank decisions according to the GP-UCB score, and pick decisions in order of decreasing score, until new feedback is available). However, since f often varies smoothly, so does the GP-UCB score; under some circumstances, this algorithm would also suffer from limited exploration. Further, if the optimal set $\boldsymbol{X}^* \subseteq D$ is of size $|\boldsymbol{X}^*| < B$ and the regret of the every sub-optimal action is finite, the algorithm would be forced to suffer linear regret, since some $\boldsymbol{x} \notin \boldsymbol{X}^*$ must be included in every batch. In short, both of these naïve algorithms are flawed, in part because they fail to exploit knowledge of the redundancy in the observations which will be obtained as a result of their actions.

From the preceding discussion, choosing a diverse set of inputs and while relying upon only outdated feedback presents a significant challenge. In the following, we introduce the Gaussian process - Batch Upper Confidence Bound (GP-BUCB) algorithm, which encourages diversity in exploration, uses past information in a principled fashion, and yields strong performance guarantees. We also extend the GP-BUCB algorithm by the creation of the Gaussian process - Adaptive Upper Confidence Bound (GP-AUCB) algorithm, which retains the theoretical guarantees of the GP-BUCB algorithm, but creates batches of variable length in a data-driven manner.

3.3 GP-BUCB Algorithm and Regret Bounds

We introduce the GP-BUCB algorithm in Section 3.3.1. Section 3.3.2 states the chapter's major theorem, a bound on the cumulative regret of a general class of algorithms including GP-BUCB and GP-AUCB. This main result is in terms of a quantity C, a bound on information used within a batch; this quantity is examined in some detail in Section 3.3.3. Using this examination, Section 3.3.4 provides a corollary, bounding the regret of GP-BUCB specifically. Section 3.3.5 improves this regret bound by initializing GP-BUCB with a finite set of observations.

3.3.1 GP-BUCB: An Overview

A key property of GPs is that the predictive variance at time t, Equation (3.2), only depends on $\boldsymbol{X}_{t-1} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}\}$, i.e, *where* the observations are made, but not *which* values $\boldsymbol{y}_{1:t-1} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}]^T$ were actually observed. Thus, it is possible to compute the posterior variance used in the sequential GP-UCB decision rule, Equation (3.5), even while previous observations are not yet available. To do so, we *hallucinate* observations $\boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} = [\mu_{\mathrm{fb}[t]}(\boldsymbol{x}_{\mathrm{fb}[t]+1}), \ldots, \mu_{\mathrm{fb}[t]}(\boldsymbol{x}_{t-1})]$ for every observation not yet received. A natural approach towards parallel exploration is therefore to replace the sequential decision rule, Equation (3.5), with a decision rule which instead sequentially chooses decisions within the batch as

$$\boldsymbol{x}_t = \operatorname*{argmax}_{\boldsymbol{x} \in D} \left[\mu_{\mathrm{fb}[t]}(\boldsymbol{x}) + \beta_t^{1/2} \sigma_{t-1}(\boldsymbol{x}) \right]. \tag{3.7}$$

Here, the parameter β_t has a role analogous to the parameter α_t in the GP-UCB algorithm. The confidence intervals corresponding to this decision rule are of the form

$$C_t^{\mathrm{batch}}(\boldsymbol{x}) \equiv \left[\mu_{\mathrm{fb}[t]}(\boldsymbol{x}) \pm \beta_t^{1/2} \sigma_{t-1}(\boldsymbol{x}) \right]. \tag{3.8}$$

The resulting GP-BUCB algorithm is shown in pseudocode as Algorithm 2. This approach naturally encourages diversity in exploration by taking into account the change in predictive variance which

Algorithm 2 GP-BUCB

Input: Decision set D, GP prior μ_0, σ_0, kernel function $k(\cdot,\cdot)$
for $t = 1, 2, \ldots, T$ **do**
  Choose $\boldsymbol{x}_t = \operatorname{argmax}_{\boldsymbol{x}\in D}[\mu_{\mathrm{fb}[t]}(\boldsymbol{x}) + \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x})]$
  Compute $\sigma_t(\cdot)$
  if $t = \mathrm{fb}[t+1]$ **then**
    Obtain $y_{t'} = f(\boldsymbol{x}_{t'}) + \varepsilon_{t'}$ for $t' \in \{\mathrm{fb}[t]+1, \ldots, t\}$
    Perform Bayesian inference to obtain $\mu_t(\cdot)$
  end if
end for

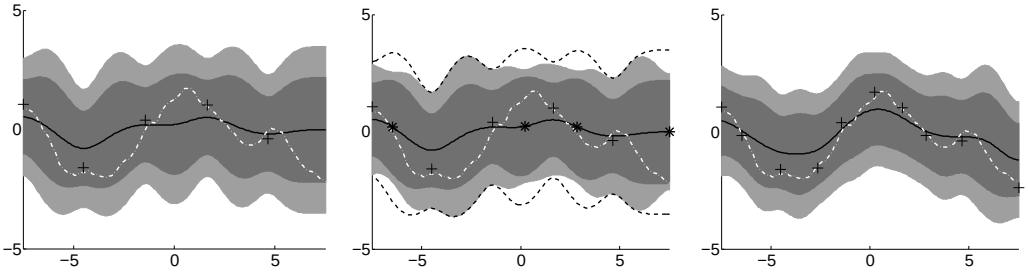
will eventually occur after pending observations: since the payoffs of "similar" decisions have similar predictive distributions, exploring one decision will automatically reduce the predictive variance of similar decisions, and thus their value in terms of exploration. This decision rule appropriately deprecates those observations which will be redundant with respect to pending observations, resulting in a more correct valuation of the action of exploring them.

The disadvantage of this approach appears as the algorithm progresses deeper into the batch. At each time t, the algorithm creates confidence intervals $C_t^{\mathrm{batch}}(\boldsymbol{x})$, the width of which is proportional to $\sigma_{t-1}(\boldsymbol{x})$. This standard deviation is used because it is the standard deviation of the posterior over the payoff f if all observations $\boldsymbol{y}_{1:t-1}$ are available, which enables GP-BUCB to avoid exploratory redundancy. However, doing so conflates the information which is actually available, gained via the observations $\boldsymbol{y}_{1:\mathrm{fb}[t]}$, with the hallucinated information corresponding to actions $\boldsymbol{x}_{\mathrm{fb}[t]+1}$ through $\boldsymbol{x}_{t-1}$. Thus, $\sigma_{t-1}(\boldsymbol{x})$ is "overconfident" about our knowledge of the function. The ratio of the width of the confidence interval derived from the set of actual observations $\boldsymbol{y}_{1:\mathrm{fb}[t]}$ to the width of the confidence interval derived from the partially hallucinated set of observations $\boldsymbol{y}_{1:t-1}$ is given by $\sigma_{\mathrm{fb}[t]}(\boldsymbol{x})/\sigma_{t-1}(\boldsymbol{x})$. This quantity is related to $I(f(\boldsymbol{x}); \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]})$, the hallucinated conditional mutual information with respect to $f(\boldsymbol{x})$, such that

$$\frac{\sigma_{\mathrm{fb}[t]}(\boldsymbol{x})}{\sigma_{t-1}(\boldsymbol{x})} = \exp\Big(I(f(\boldsymbol{x}); \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]})\Big). \tag{3.9}$$

This ratio quantifies the degree of "overconfidence" with respect to the posterior as of the beginning of the batch. Crucially, if there exists some constant C, such that $I(f(\boldsymbol{x}); \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) \leq C$, $\forall \boldsymbol{x} \in D$, the ratio $\sigma_{\mathrm{fb}[t]}(\boldsymbol{x})/\sigma_{t-1}(\boldsymbol{x})$ can also be bounded for every $\boldsymbol{x} \in D$. Bounding this ratio of confidence interval shrinkage due to hallucinated information links the hallucinated posterior at round t back to the posterior as of the last feedback, at round $\mathrm{fb}[t]$. Using this bound, the algorithm can be constructed to compensate for its overconfidence.

This compensation for overconfidence must strike a delicate balance between the algorithmic requirement of allowing the predictive variance over f to shrink as the algorithm hallucinates observations, thus allowing the algorithm to avoid redundancy, and maintaining the probabilistic inter-



(a) *Beginning of batch 2* (b) *Last decision of batch 2* (c) *Beginning of batch 3*

Figure 3.1: (a): The confidence intervals $C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x})$ (dark), computed from previous noisy observations $\boldsymbol{y}_{1:\text{fb}[t]}$ (crosses), are centered around the posterior mean (solid black) and contain $f(\boldsymbol{x})$ (white dashed) w.h.p. To avoid overconfidence, GP-BUCB chooses $C^{\text{batch}}_{\text{fb}[t]+1}(\boldsymbol{x})$ (light gray) such that even in the worst case, the succeeding confidence intervals in the batch, $C^{\text{batch}}_{\tau}(\boldsymbol{x})$, $\forall \tau : \text{fb}[\tau] = \text{fb}[t]$, will contain $C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x})$. (b): Due to the observations that GP-BUCB "hallucinates" (stars), the outer posterior confidence intervals $C^{\text{batch}}_{t}(\boldsymbol{x})$ shrink from their values at the start of the batch (black dashed), but still contain $C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x})$, as desired. (c): Upon selection of the last decision of the batch, the feedback for all decisions is obtained, and for the subsequent action selection in round t', new confidence intervals $C^{\text{seq}}_{\text{fb}[t']+1}(\boldsymbol{x})$ and $C^{\text{batch}}_{\text{fb}[t']+1}(\boldsymbol{x})$ are computed.

pretation of the confidence interval size, such that high-probability statements can be made about the regret. One satisfactory approach is to increase the width of the confidence intervals (through proper choice of the parameter β_t), such that the confidence intervals used by GP-BUCB are *conservative*, i.e., contain the true function $f(\boldsymbol{x})$ with high probability. More precisely, we require that for all $t \in \{1, 2, \ldots, T\}$ and $\boldsymbol{x} \in D$, $C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x}) \subseteq C^{\text{batch}}_{t}(\boldsymbol{x})$; that is, the batch algorithm's confidence intervals are sufficiently large to guarantee that even for the last action selection in the batch, they contain the confidence intervals which would be created by the GP-UCB algorithm, Equation (3.6), given $\boldsymbol{y}_{1:\text{fb}[t]}$. Srinivas et al. (2010) provide choices of α_t such that the GP-UCB confidence intervals have a high-probabilty guarantee of correctness $\forall t \geq 1, \boldsymbol{x} \in D$. If it can be shown that $C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x}) \subseteq C^{\text{batch}}_{t}(\boldsymbol{x})$, $\forall \boldsymbol{x} \in D, t \in \mathbb{N}$, the batch confidence intervals inherit the high-probability guarantee of correctness. By multiplicatively increasing the width of the hallucinated confidence intervals and using the same factor of width increase for all $\boldsymbol{x} \in D$ and $t \in \mathbb{N}$, the redundancy control of the hallucinated observations is maintained, and the required degree of conservatism can simultaneously be achieved. Figure 3.1 illustrates this idea. The main difficulty in using this approach is finding the degree of conservatism required to guarantee the containment of $C^{\text{seq}}_{\text{fb}[t]+1}(\boldsymbol{x})$ by $C^{\text{batch}}_{t}(\boldsymbol{x})$. Sections 3.3.2 and 3.3.4 show that by using a multiplicative factor $\exp(C)$ relative to $\alpha_{\text{fb}[t]}$, where C is a bound on the conditional mutual information which can be gained within a batch, β_t can be chosen such that containment of the sequential confidence intervals within the batch confidence intervals is achieved. This containment follows from Equation (3.14). We further show that, with appropriate initialization, the regret can be made to only mildly increase relative to GP-UCB, providing theoretical support for the potential for parallelizing GP optimization.

3.3.2 General Regret Bound

Our main theorem bounds the regret of GP-BUCB and related algorithms. This regret bound is formulated in terms of a bound C on the maximum conditional mutual information which can be hallucinated with respect to $f(\boldsymbol{x})$ for any $\boldsymbol{x}$ in D, which we assume to be known to the algorithm. We discuss methods of obtaining such a bound in Section 3.3.3. This bound is used to relate hallucinated confidence intervals, used to select actions, and the posterior confidence intervals as of the last feedback obtained, which contain the payoff function f with high probability. This theorem holds under any of three different assumptions about f, which may all be of practical interest. In particular, it holds even if the assumption that f is sampled from a GP is replaced by the assumption that f has low norm in the Reproducing Kernel Hilbert Space (RKHS) associated with the kernel function.

Theorem 1. *Let $\delta \in (0,1)$, γ_t be as defined in Equation (3.3), and α_t be a time-varying exploration-exploitation tradeoff parameter, as in Equation (3.5). Let the variance be bounded, such that $k(\boldsymbol{x},\boldsymbol{x}) \leq 1$, $\forall\, \boldsymbol{x} \in D$. Suppose one of the following sets of assumptions holds:*

1. *D is a finite set, the payoff function f is sampled from a known GP prior with known noise variance σ_n^2, and $\alpha_t = 2\log(|D|t^2\pi^2/6\delta)$.*
2. *$D \subseteq [0,l]^d$ is compact and convex, with $d \in \mathbb{N}$, $l > 0$, and the payoff function f is sampled from a known GP prior with known noise variance σ_n^2. Choose $\alpha_t = 2\log(t^2 2\pi^2/(3\delta)) + 2d\log\left(t^2 dbl\sqrt{\log(4da/\delta)}\right)$, where the constants $a, b > 0$ and $k(\boldsymbol{x},\boldsymbol{x}')$ are such that the following bound holds with high probability on the derivatives of GP sample paths f:*

$$\Pr\left\{\sup_{\boldsymbol{x}\in D} |\partial f/\partial x_j| > L\right\} \leq ae^{-(L/b)^2},\ j = 1,\ldots,d.$$

3. *D is arbitrary and the payoff function f has RKHS norm bounded as $||f||_k \leq M$ for some constant M. The noise variables ε_t form an* arbitrary *martingale difference sequence (meaning that $\mathbb{E}[\varepsilon_t \,|\, \varepsilon_1,\ldots,\varepsilon_{t-1}] = 0$ for all $t \in \mathbb{N}$), uniformly bounded by σ_n. Choose $\alpha_t = 2M^2 + 300\gamma_t \ln^3(t/\delta)$.*

Further suppose there exists a mapping $fb[t]$ which dictates at which rounds new feedback becomes available to the algorithm and a bound $C > 0$ such that, for all $t \in \mathbb{N}$ and all $\boldsymbol{x} \in D$,

$$I(f(\boldsymbol{x}); \boldsymbol{y}_{fb[t]+1:t-1} \mid \boldsymbol{y}_{1:fb[t]}) \leq C, \tag{3.10}$$

where $\boldsymbol{x}_t$ is selected according to Equation (3.7) using $\beta_t = \exp(2C)\alpha_{fb[t]}$ for all $t \in \{1,\ldots,T\}$. Then the cumulative regret is bounded by $O^*(\sqrt{T\gamma_T \exp(2C)\alpha_T})$ with high probability. Precisely,

$$\Pr\left\{R_T \leq \sqrt{C_1 T \exp(2C)\alpha_T\gamma_T}, +2 \quad \forall T \geq 1\right\} \geq 1-\delta$$

where $C_1 = 8/\log(1 + \sigma_n^{-2})$.

Proof. The proof of this result is presented in Appendix A.1. □

The key quantity that controls the regret in Theorem 1 is C, the bound in Equation (3.10) on the maximum conditional mutual information obtainable within a batch with respect to $f(\boldsymbol{x})$ for any $\boldsymbol{x} \in D$. In particular, the cumulative regret bound of Theorem 1 is a factor $\exp(C)$ larger than the regret bound for the sequential ($B = 1$) GP-UCB algorithm. Thus, for any practical meaningfulness of the bound, we must be able to define C. The various methods which can be used for selecting C are explored in the following sections.

3.3.3 Suitable Choices for C

The functional significance of a bound C on the information hallucinated with respect to any $f(\boldsymbol{x})$ arises through this quantity's ability to bound the degree of contamination of the GP-BUCB confidence intervals, given by Equation (3.8), with hallucinated information. As background for finding suitable values C, extension of the discussion of Section 3.2.3 on mutual information is required.

Two properties of the mutual information are particularly useful. These properties are monotonicity (adding an element $\boldsymbol{x}$ to the set A cannot decrease the mutual information between f and the corresponding set of observations $\boldsymbol{y}_A$) and submodularity (the increase in mutual information between f and $\boldsymbol{y}_A$ with the addition of an element $\boldsymbol{x}$ to set A cannot be greater than the corresponding increase in mutual information if $\boldsymbol{x}$ is added to $A', A' \subseteq A$) (Krause and Guestrin, 2005).

Using the time indexing notation developed in Section 3.2.1, the property of monotonicity allows the following series of inequalities:

$$I(f(\boldsymbol{x}); \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) \leq I(f; \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) \tag{3.11}$$

$$\leq \max_{A \subseteq D, |A| \leq B-1} I(f; \boldsymbol{y}_A \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) \tag{3.12}$$

$$\leq \max_{A \subseteq D, |A| \leq B-1} I(f; \boldsymbol{y}_A) = \gamma_{B-1}. \tag{3.13}$$

The first inequality follows from the monotonicity of mutual information, i.e., the information gained with respect to f as a whole must be at least as large as that gained with respect to $f(x)$. The second inequality holds because we specify the feedback mapping such that $t - \mathrm{fb}[t] \leq B$, and the third inequality holds due to the "information never hurts" bound (Cover and Thomas, 1991), which states that the conditional mutual information $I(f; \boldsymbol{y}_A \mid \boldsymbol{y}_S)$ is monotonically decreasing in S (i.e., as elements are added to set S).

Any bound C on the conditional mutual information hallucinated with respect to any $f(\boldsymbol{x})$ during

the selection of a batch can be combined with Equation (3.9) to produce the following statement:

$$\frac{\sigma_{\text{fb}[t]}(\boldsymbol{x})}{\sigma_{t-1}(\boldsymbol{x})} = \exp\left(I(f(\boldsymbol{x}); \boldsymbol{y}_{\text{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\text{fb}[t]})\right) \leq \exp(C). \tag{3.14}$$

A bound on $I(f(\boldsymbol{x}); \boldsymbol{y}_{\text{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\text{fb}[t]})$ itself or any bound on one of the terms on the right hand side of Equations (3.11), (3.12) or (3.13) is suitable for our purposes.

3.3.4 Corollary Regret Bound: GP-BUCB

The GP-BUCB algorithm requires that $t - \text{fb}[t] \leq B,\ \forall t \geq 1$, and uses a value C such that, for any $t \in \mathbb{N}$,

$$\max_{A \subseteq D, |A| \leq B-1} I(f; \boldsymbol{y}_A \mid \boldsymbol{y}_{1:\text{fb}[t]}) \leq C, \tag{3.15}$$

thus bounding $I(f(\boldsymbol{x}); \boldsymbol{y}_{\text{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\text{fb}[t]})$ for all $\boldsymbol{x} \in D$ and $t \in \mathbb{N}$ via Inequality (3.12). Otherwise stated, in GP-BUCB, the local information gain with respect to any $f(\boldsymbol{x}), \boldsymbol{x} \in D, t \in \mathbb{N}$ is bounded by fixing the feedback times and then bounding the maximum conditional mutual information with respect to the entire function f which can be acquired by *any* algorithm which chooses any set of $B-1$ or fewer observations. While this argument uses multiple upper bounds, any or all of which may be overly conservative, this approach is still sensible because such a bound C holds for any possible algorithm for constructing batches; it is otherwise quite difficult to disentangle the role of C in setting the exploration-exploitation tradeoff parameter β_t from its role as a bound on how much information is hallucinated by the algorithm, since a larger value of C (and thus β_t) typically *produces* more information gain by promoting exploration under the GP-BUCB decision rule, Equation (3.7).

Since the bound C is related to the maximum amount of conditional mutual information which could be acquired by a set of $B-1$ actions, one expects C to grow monotonically with B; with a larger set of pending actions, there is more potential for explorations which gain additional information. One easy upper bound for the information gained in any batch can be derived as follows. As noted in Section 3.3.3, mutual information is submodular, and thus the maximum conditional mutual information which can be gained by making any set of observations is maximized when the set of observations currently available, to which these new observations will be added, is empty. Letting the maximum mutual information with respect to f which can be obtained by any observation set of size $B-1$ be denoted γ_{B-1} and choosing $C = \gamma_{B-1}$ provides a bound on the possible local conditional mutual information gain for any $t \in \mathbb{N}$ and $\boldsymbol{x} \in D$, as in Inequality (3.13). This choice of bound yields the following Corollary, an extension to Theorem 1:

Corollary 2. *Assume the GP-BUCB algorithm is employed with a constant B such that $t - fb[t] \leq B$ for all $t \geq 1$. Let $\delta \in (0,1)$, and let one of the numbered conditions of Theorem 1 be met. If*

$\beta_t = \exp(2C)\alpha_{\mathrm{fb}[t]}$ for all $t \in \{1, \ldots, T\}$, the cumulative regret R_T of the GP-BUCB algorithm can be bounded with probability $1 - \delta$ as follows:

$$\Pr\left\{R_T \leq \sqrt{C_1 T \exp(2\gamma_{B-1})\alpha_T \gamma_T} + 2, \quad \forall T \geq 1\right\} \geq 1 - \delta,$$

where $C_1 = 8/\log(1 + \sigma_n^{-2})$ and γ_{B-1} and γ_T are as defined in Equation (3.3).

While the above result is useful, the choice $C = \gamma_{B-1}$ is not especially satisfying on its own. The maximum information gain γ_{B-1} usually grows at least as $\Omega(\log B)$, implying that $\exp(C)$ grows at least linearly in B, yielding a regret bound which is also at least linear in B. Section 3.3.5 shows that the GP-BUCB algorithm can be modified such that a constant choice of C *independent* of B suffices.

3.3.5 Better Bounds Through Initialization

To obtain regret bounds *independent* of batch size B, the monotonicity properties of conditional mutual information can again be exploited. This can be done by structuring GP-BUCB as a two-stage procedure. First, an *initialization set* D^{init} of size $|D^{\mathrm{init}}| = T^{\mathrm{init}}$ is selected nonadaptively (i.e., without any feedback); following the selection of this entire set, feedback $\boldsymbol{y}^{\mathrm{init}}$ for all decisions in $D^{\mathrm{init}} = \{\boldsymbol{x}_1^{\mathrm{init}}, \ldots, \boldsymbol{x}_{T^{\mathrm{init}}}^{\mathrm{init}}\}$ is obtained. In the second stage, GP-BUCB is applied to the posterior Gaussian process distribution, conditioned on $\boldsymbol{y}^{\mathrm{init}}$.

Notice that if we define

$$\gamma_T^{\mathrm{init}} = \max_{A \subseteq D, |A| \leq T} I(f; \boldsymbol{y}_A \mid \boldsymbol{y}^{\mathrm{init}}),$$

then, under the assumptions of Theorem 1, using $C = \gamma_{B-1}^{\mathrm{init}}$, the regret of the two-stage algorithm is bounded by $R_T = O(T^{\mathrm{init}} + \sqrt{T\gamma_T^{\mathrm{init}}\alpha_T \exp 2C})$. In the following, we show that it is indeed possible to construct an initialization set D^{init} such that the size T^{init} is dominated by $\sqrt{T\gamma_T^{\mathrm{init}}\alpha_T \exp(2C)}$, and – crucially – that $C = \gamma_{B-1}^{\mathrm{init}}$ can be bounded *independently* of the batch size B.

The initialization set D^{init} is constructed via uncertainty sampling: start with $D_0^{\mathrm{init}} = \emptyset$, and for each $t = 1, \ldots, T^{\mathrm{init}}$, greedily determine the most uncertain decision

$$\boldsymbol{x}_t^{\mathrm{init}} = \operatorname*{argmax}_{\boldsymbol{x} \in D} \sigma_{t-1}^2(\boldsymbol{x})$$

and set $D_t^{\mathrm{init}} = D_{t-1}^{\mathrm{init}} \cap \boldsymbol{x}_t^{\mathrm{init}}$. Note that uncertainty sampling is a special case of the GP-BUCB algorithm with a constant prior mean of 0 and the requirement that for all $1 \leq t \leq T^{\mathrm{init}}$, $\mathrm{fb}[t] = 0$, i.e., no feedback is taken into account for the first T^{init} iterations.

Under the above procedure, we have the following key result about the maximum residual information gain γ^{init}:

Lemma 3. *Suppose uncertainty sampling is used to generate an initialization set D^{init} of size T^{init}. Then*

$$\gamma_{B-1}^{init} \leq \frac{B-1}{T^{init}} \gamma_{T^{init}}. \tag{3.16}$$

Proof. The proof of this lemma is presented in Appendix A.2. □

Whenever γ_T is sublinear in T (i.e., $\gamma_T = o(T)$), then the bound on $\gamma_{B-1}^{\text{init}}$ given by Inequality (3.16) converges to zero for sufficiently large T^{init}; thus for *any* constant $C > 0$, we can choose T^{init} as a function of B such that $\gamma_{B-1}^{\text{init}} < C$. Using this choice of C in Theorem 1 bounds the post-initialization regret. In order to derive bounds on T^{init}, we in turn need a bound on γ_T which is analytical and sublinear. Fortunately, Srinivas et al. (2010) prove suitable bounds on how the information gain γ_T grows for some of the most commonly used kernels. We summarize our analysis below in Theorem 4. For sake of notation, define R_T^{seq} to be the regret bound of Srinivas et al. (2010) associated with the sequential GP-UCB algorithm (i.e., Corollary 2 with $B = 1$).

Theorem 4. *Suppose one of the conditions of Theorem 1 is satisfied. Further, suppose the kernel and T^{init} are as listed in Table 3.1, and $B \geq 2$. Fix $\delta > 0$. Let R_T be the regret of the two-stage initialized GP-BUCB algorithm, which ignores feedback for the first T^{init} rounds. Then there exists a constant C' independent of B such that for any $T \geq 0$, it holds with probability at least $1 - \delta$ that*

$$R_T \leq C' R_T^{seq} + 2||f||_\infty T^{init}, \tag{3.17}$$

where C' takes the value shown in Table 3.1.

The results in Table 3.1 are derived in Appendix A.2. Note that the particular values of C' used in Table 3.1 are not the only ones possible; they are chosen simply because they yield relatively clean algebraic forms for T^{init}. Relative to Theorem 1, which depends on B and T through the product $\exp(2C)T\alpha_T\gamma_T$, Theorem 4 replaces this product with a sum of two terms, one in each of B and T; the term $C'R_T^{\text{seq}}$ in Inequality (3.17) is the cost paid for running the algorithm post-initialization (independent of B, dependent on T), whereas the second term is the cost of performing the initialization (dependent on B, independent of T). Notice that whenever $B = O(\text{polylog}(T))$, $T^{\text{init}} = O(\text{polylog}(T))$, and further, note $R_T^{\text{seq}} = \Omega(\sqrt{T})$. Thus, as long as the batch size does not grow too quickly, the term $O(T^{\text{init}})$ is dominated by $C'R_T^{\text{seq}}$ and the regret bounds of GP-BUCB are only a constant factor, *independent of* B, worse than those of GP-UCB.

3.4 Adaptive Parallelism: GP-AUCB

While the analysis of the GP-BUCB algorithm in Sections 3.3.4 and 3.3.5 used feedback mappings fb[t] specified by the problem instance, it may be useful to let the algorithm control when to request

Kernel Type	**Size T^{init} of Initialization Set D^{init}**	**Regret Multiplier C'**
LINEAR: $\gamma_t \leq \eta d \log(t+1)$	$\max \left[\begin{array}{c} \log(B) \\ \frac{\log\eta + \log d + 2\log(B)}{2\log(B)-1} e\eta d(B-1)\log(B) \end{array} \right]$	$\exp(2/e)$
MATÉRN: $\gamma_t \leq \nu t^{\epsilon}$	$(\nu(B-1))^{1/(1-\epsilon)}$	e
RBF: $\gamma_t \leq \eta(\log(t+1))^{d+1}$	$\max \left[\begin{array}{c} (\log(B))^{d+1} \\ \left(\frac{e}{d+1} \frac{\log\eta + (d+2)\log(B)}{2\log(B)-1}\right)^{d+1} \eta(B-1)(\log(B))^{d+1} \end{array} \right]$	$\exp((\frac{2d+2}{e})^{d+1})$

Table 3.1: Initialization set sizes for Theorem 4.

feedback, and to allow this feedback period to vary in some range not easily described by any constant B. For example, allowing the algorithm to control parallelism is desirable in situations where the cost of executing the algorithm's queries to the oracle depends on both the number of batches and the number of individual actions or experiments in those batches. Consider a chemical experiment, in which cost is a weighted sum of the time to complete the batch of reactions and the cost of the reagents needed for each individual experiment. In such a case, confronting an initial state of relative ignorance about the reward function, it may be desirable to avoid using a wasteful level of parallelism. Motivated by this, we develop an extension to our requirement that $t - \text{fb}[t] \leq B$; we will instead simply require that the mapping $\text{fb}[t]$ and the sequence of actions selected by the algorithm be such that there exists some bound C, holding for all $t \geq 1$ and $\boldsymbol{x} \in D$, on $I(f(\boldsymbol{x}); \boldsymbol{y}_{\text{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\text{fb}[t]})$, the hallucinated information as of round t with respect to any value $f(\boldsymbol{x})$. This requirement on $\text{fb}[t]$ in terms of C may appear stringent, but in actual fact it can be easily satisfied by on-line, data-driven construction of the mapping $\text{fb}[t]$ after having pre-selected a value for C. The GP-AUCB algorithm controls feedback adaptively through precisely such a mechanism.

Section 3.4.1 introduces GP-AUCB and states a corollary regret bound for this algorithm. A few comments on local versus global stopping criteria for adaptivity of algorithms follow in Section 3.4.2.

3.4.1 GP-AUCB Algorithm

The key innovation of the GP-AUCB algorithm is in choosing $\text{fb}[t]$ online, using a limit on the amount of information hallucinated within the batch. Such adaptive batch length control is possible because we can actually measure online the amount of hallucinated information using Equation (3.4), even in the absence of the observations themselves. When this value exceeds a pre-defined constant C, the algorithm terminates the batch, setting fb for the next batch to the current t (i.e., $\text{fb}[t+1] = t$), and waits for the oracle to return values for the pending queries. The resulting algorithm, GP-AUCB, is shown in Algorithm 3. The GP-AUCB algorithm can also be employed in the delay setting, but rather than using the hallucinated information to decide whether or not to terminate the current batch, the algorithm chooses whether or not to submit an action in this round; the algorithm submits an action if the hallucinated information is $\leq C$ and refuses to submit an action ("balks") if the hallucinated information is $> C$.

In comparison to GP-BUCB, by concerning itself with only the batches *actually* chosen, rather than worst-case batches, the GP-AUCB algorithm eliminates the requirement that C be greater than the information which could be gained in *any* batch, and thus makes the information gain bounding argument less conservative; for such a C,

$$I(f(\boldsymbol{x}); \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) \leq I(f; \boldsymbol{y}_{\mathrm{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\mathrm{fb}[t]}) \leq C, \ \forall \boldsymbol{x} \in D, \ \forall t \geq 1,$$

that is, via the monotonicity of conditional mutual information, the information gain locally under GP-AUCB is bounded by the information gain with respect to f as a whole, which is constrained to be $\leq C$ by the stopping condition. Using such an adaptive stopping condition and the corresponding value of β_t, Equation (3.14) can be used to maintain a guarantee of confidence interval correctness for batches of variable length. In particular, the batch length may possibly become quite large as the shape of f is better and better understood and the variance of $f(\boldsymbol{x}_t)$ tends to decrease. Further, if exploratory actions are chosen, the high information gain of these actions contributes to a relatively early arrival at the information gain threshold C and thus relatively short batch length, even late in the algorithm's run. In this way, the batch length is chosen in response to the algorithm's need to explore or exploit as dictated by the decision rule, Equation (3.7), not simply following a monotonically increasing schedule.

This approach meets the conditions of Theorem 1, allowing the regret of GP-AUCB to be bounded for both the batch and delay settings with the following corollary:

Corollary 5. *If the GP-AUCB algorithm is employed with a specified constant value C, for which $I(f; \boldsymbol{y}_{fb[t]+1:t-1} \mid \boldsymbol{y}_{1:fb[t]}) \leq C, \forall t \in \{1, \ldots, T\}$, δ is a specified constant in the interval $(0, 1)$, one of the conditions of Theorem 1 is met, and under the resulting feedback mapping $fb[t]$, $\beta_\tau = \exp(2C)\alpha_{fb[\tau]}, \forall \tau \in \{1, \ldots, t\}$, then*

$$\Pr\left\{R_T \leq \sqrt{C_1 T \exp(2C)\alpha_T \gamma_T} + 2, \quad \forall T \geq 1\right\} \geq 1 - \delta$$

where $C_1 = 8/\log(1 + \sigma_n^{-2})$.

Note that it is also possible to combine the results of Section 3.3.5 with Corollary 5 to produce a two-stage adaptive algorithm which can deliver high starting parallelism, very high parallelism as the run proceeds, and a low information gain bound C, yielding a low regret bound.

Despite the advantages of this approach, the value of C is rather abstract and is certainly less natural for an experimentalist to specify than a maximum batch size or delay length B. However, C can be selected to deliver batches with a specified minimum size B_{min}. To ensure this occurs, C can be set such that $C > \gamma_{(B_{min}-1)}$, i.e., no set of queries of size less than B_{min} could possibly gain enough information to end the batch. Further, if we choose C such that $C < \gamma_{(B_{min})}$, it is possible

Algorithm 3 GP-AUCB

Input: Decision set D, GP prior μ_0, σ_0, kernel $k(\cdot,\cdot)$, information gain threshold C.
Set $\text{fb}[t'] = 0,\ \forall t' \geq 1$, $G = 0$.
for $t = 1, 2, \ldots, T$ **do**
  if $G > C$ **then**
    Obtain $y_{t'} = f(\boldsymbol{x}_{t'}) + \varepsilon_{t'}$ for $t' \in \{\text{fb}[t-1], \ldots, t-1\}$
    Perform Bayesian inference to obtain $\mu_{t-1}(\cdot)$
    Set $G = 0$
    Set $\text{fb}[t'] = t-1,\ \forall t' \geq t$
  end if
  Choose $\boldsymbol{x}_t = \text{argmax}_{\boldsymbol{x} \in D}[\mu_{\text{fb}[t]}(\boldsymbol{x}) + \beta_t^{1/2}\sigma_{t-1}(\boldsymbol{x})]$
  Set $G = G + \frac{1}{2}\log\left(1 + \sigma_n^{-2}\sigma_{t-1}^2(x_t)\right)$
  Compute $\sigma_t(\cdot)$
end for

to select a batch of size B_{min} which does attain the required amount of information to terminate the batch, and thus B_{min} truly can be thought of as the minimum batch size which could be produced by the GP-AUCB algorithm. Often, however, γ_t is not available directly, and cannot be obtained except for combinatorial optimization; in such a case, if B_{min} is very small, this combinatorial optimization may be tractable, and if B_{min} is too large, greedy maximum entropy sampling can be used to bound $\gamma_{(B_{min}-1)}$ from above, allowing the selection of values for C which satisfy the specification on minimum batch size, if with a large degree of conservatism. While relating C to some B_{min} is not the only way to choose the constant C intelligently, doing so gives a clear way to specify C in a more intuitive and relatable way.

It is also possible to choose a very small value for the constant C and produce nearly sequential actions early, while retaining late-run parallelism and producing a very low regret bound. This can be seen if B_{min} is set to 1; following the analysis above, such a C must satisfy the inequalities $\gamma_0 = 0 < C < \gamma_1$, i.e., C can be a very small positive number. Following rearrangement, the regret of GP-AUCB is bounded by Corollary 5 as $R_T \leq exp(C) R_T^{\text{seq}}$, where R_T^{seq} is the bound of Corollary 2 with $B = 1$, the regret of the GP-UCB algorithm. Since for very small C, $\exp(C)$ is nearly 1, the regret bound of GP-AUCB is only a very little more than for GP-UCB. With regard to action selection, choosing C to be a small positive value should result in GP-AUCB beginning its run by acting sequentially, since most actions gain information greater than C. However, the algorithm has the potential to construct batches of increasing length as $T \to \infty$; even assuming the worst case, that all observations are independent and each gains the same amount of information, the batch length allowed with a given posterior is lower-bounded by $B_{max} \geq C/log(1 + \sigma_n^{-2}\tilde{\sigma}_{\text{fb}[t]}^2)$ where $\tilde{\sigma}^2$ is the largest variance among the actions selected in the batch. If the algorithm converges toward the optimal subset $\boldsymbol{X}^* \subseteq D$, as the regret bound suggests it will, and $\boldsymbol{X}^*$ is of finite size, then the variances of the actions selected (and thus the denominator in the expression above) can be expected to generally become very small, producing batches of very long length, even for very small

C. Choosing C as a small positive value thus produces the potential for naturally occurring late-run parallelism for very little additional regret relative to GP-UCB.

3.4.2 Local Stopping Conditions Versus Global Stopping Conditions

Both GP-BUCB and GP-AUCB rely upon bounds on the information gain of the hallucinated observations with respect to f as a whole, but Theorem 1 is stated in terms of a bound on the information gain with respect to $f(\boldsymbol{x})$, which must hold $\forall t \geq 1, \forall \boldsymbol{x} \in D$. During the proof of the regret bound, the information gain threshold is a vehicle for ensuring that the confidence intervals do not shrink to too small a ratio, as in Equation (3.14); this enables a choice of β_t which ensures that $C_t^{\text{batch}}(\boldsymbol{x}) \supseteq C_{\text{fb}[t]+1}^{\text{seq}}(\boldsymbol{x})$ for all $t \geq 1$ and $\boldsymbol{x} \in D$. However, as has been stated above, the standard deviation can be calculated on-line, even without the actual observations, thus enabling exhaustive checking of the ratio $\sigma_{\text{fb}[t]}(\boldsymbol{x})/\sigma_{t-1}(\boldsymbol{x})$ for every $\boldsymbol{x} \in D$; assuming this calculation is practical, this strongly suggests that it would be possible to create an algorithm for which the standard deviation ratio is directly checked, rather than being bounded from above through the information gain. Doing so would also imply the existence of some information gain bound $C \geq (f(\boldsymbol{x}); \boldsymbol{y}_{\text{fb}[t]+1:t-1} | \boldsymbol{y}_{\text{fb}[t]})$, $\forall t \geq 1$, $\boldsymbol{x} \in D$, without requiring recourse to the multiple upper bounding arguments used for GP-BUCB and GP-AUCB. This formulation appears attractive because it might enable the algorithm to avoid waiting for early observations when such waiting may be unnecessary, e.g., when the subsequent actions will be additional initialization and not close to previous actions in the batch. This allows the choice of a very small C, e.g., $C = (1+\epsilon)\gamma_1$, such that if the algorithm makes a single observation at a point, and perhaps some other distant points, it will not stop the batch. Due to the small value of C, this translates to a quite small regret bound under Theorem 1.

If we assume a flat prior, such a procedure would tend to create a first batch which thoroughly initializes over the whole set, since most points would be scattered widely, and therefore the local information gain stopping condition would not be met until actions were proposed close together. For practical purposes, it is therefore necessary to introduce a limit on batch size B_{max} such that the first batch is not of a size approaching that of D. By doing so, the tendency is to produce an algorithm which actually tends to *just* produce batches of the maximal size, i.e., $B = B_{max}$ in the simple parallel case, albeit with a tighter regret bound. If nothing else, this tends to offer justification of the common practice with GP-BUCB of setting β_t much smaller than the theory would suggest setting it, such that the algorithm tends to exploit more heavily.

We implement this algorithm, denoting it GP-AUCB Local, and show it in some of the experiments and figures in Section 3.6, along with the Hybrid Batch Bayesian Optimization algorithm (HBBO) of Azimi et al. (2012b). HBBO implements a similar local check on a hallucinated posterior, though this check is expressed in terms of expected prediction error versus the true posterior if all information

had been acquired, rather than information gain, and the local stopping condition is only checked at $\boldsymbol{x}_t$, rather than all $\boldsymbol{x}$ in D.

3.5 Lazy Variance Calculations

In this section, we introduce the notion of lazy variance calculations, which may be used to greatly accelerate the computation of many UCB-based algorithms, including GP-UCB, GP-BUCB, and GP-AUCB, without any loss of performance.

Though GP-UCB, GP-BUCB, and GP-AUCB may be implemented as linear algebraic operations and are thus amenable to computational implementation without sampling, the execution time of the algorithms may still be lengthy, particularly as the number of observations becomes larger. The major computational bottleneck is calculating the posterior mean $\mu_{\mathrm{fb}[t]}(\boldsymbol{x})$ and variance $\sigma^2_{t-1}(\boldsymbol{x})$ for the candidate decisions, as required to calculate the decision rule and choose an action $\boldsymbol{x}_t$. The mean is updated only whenever feedback is obtained, and – upon computation of the Cholesky factorization of $K(\boldsymbol{X}_{\mathrm{fb}[t]}, \boldsymbol{X}_{\mathrm{fb}[t]}) + \sigma_n^2 I$ (which only needs to be done once whenever new feedback arrives) – the calculation of the posterior mean $\mu_{\mathrm{fb}[t]}(\boldsymbol{x})$ takes $O(t)$ additions and multiplications. On the other hand, σ^2_{t-1} must be recomputed for every $\boldsymbol{x} \in D$ after every round, and requires solving backsubstitution, which requires $O(t^2)$ computations. For large decision sets D, the variance computation thus dominates the computational cost of GP-BUCB.

Fortunately, for any fixed decision $\boldsymbol{x}$, $\sigma^2_t(\boldsymbol{x})$ is non-increasing in t. This fact can be exploited to dramatically improve the running time of GP-BUCB, at least for decision sets D which are finitely discretized or are themselves finite. The key idea is that instead of recomputing $\sigma_{t-1}(\boldsymbol{x})$ for all decisions $\boldsymbol{x}$ in every round t, we can maintain an upper bound $\widehat{\sigma}_{t-1}(\boldsymbol{x})$, initialized to $\widehat{\sigma}_0(\boldsymbol{x}) = \infty$. In every round, we lazily apply the GP-BUCB rule with this upper bound to identify

$$\boldsymbol{x}_t = \operatorname*{argmax}_{\boldsymbol{x} \in D} \left[\mu_{\mathrm{fb}[t]}(\boldsymbol{x}) + \beta_t^{1/2} \widehat{\sigma}_{t-1}(\boldsymbol{x}) \right]. \tag{3.18}$$

We then recompute $\widehat{\sigma}_{t-1}(\boldsymbol{x}_t) \leftarrow \sigma_{t-1}(\boldsymbol{x}_t)$. If $\boldsymbol{x}_t$ still lies in the argmax of Equation (3.18), we have identified the next decision to make, and set $\widehat{\sigma}_t(\boldsymbol{x}) = \widehat{\sigma}_{t-1}(\boldsymbol{x})$ for all remaining decisions $\boldsymbol{x}$. Minoux (1978) proposed a similar technique, concerning calculating the greedy action for submodular maximization, which the above procedure generalizes to the bandit setting. A similar idea was also employed by Krause et al. (2008) in the Gaussian process setting for experimental design. The lazy variance calculation method leads to dramatically improved empirical computational speed, discussed in Section 3.6.

Locally stopped algorithms (Section 3.4.2) may have stopping conditions which are dependent on the uncertainty at every $\boldsymbol{x} \in D$, but they also benefit from lazy variance calculations. Since the global conditional information gain bounds the local information gain for all $\boldsymbol{x} \in D$, as in Inequality

(3.11), we obtain the implication

$$I(f; \boldsymbol{y}_{\text{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\text{fb}[t]}) \leq C \implies \nexists \boldsymbol{x} \in D : I(f(\boldsymbol{x}); \boldsymbol{y}_{\text{fb}[t]+1:t-1} \mid \boldsymbol{y}_{1:\text{fb}[t]}) > C$$

that is, that until the stopping condition for GP-AUCB is met, the stopping condition for GP-AUCB Local is also not met, and thus no local calculations need be made. In implementing GP-AUCB Local, we may run what is effectively lazy GP-AUCB until the global stopping condition is met, at which time we transition to GP-AUCB Local. For a fixed maximum batch size B_{max}, it is often the case that local variance calculations become only very rarely necessary after the first few batches.

3.6 Computational Experiments

We compare GP-BUCB with several alternatives: (1) The strictly sequential GP-UCB algorithm ($B = 1$) receiving feedback from each action without batching or delay; (2) Two versions of a state of the art algorithm for Batch Bayesian optimization proposed by Azimi et al. (2010), which can use either a UCB or Maximum Expected Improvement (MEI) decision rule, herein SM-UCB and SM-MEI respectively. Similarly, we compare GP-AUCB against two other adaptive algorithms: (1) HBBO, proposed by Azimi et al. (2012b), which checks an expected prediction error stopping condition and makes decisions using either an MEI or a UCB decision rule; and (2) GP-AUCB Local, a local information gain-checking adaptive algorithm described briefly in Section 3.4.2. We also present some experimental comparisons across these two sets of algorithms.

In Section 3.6.1, we describe the computational experiments which were performed in more detail. Each of the described computational experiments was performed for each data set. These data sets and the corresponding experimental results are presented in Section 3.6.2. Results of the computational time comparisons are reserved to Section 3.6.3. We also briefly highlight the tradeoffs inherent in adaptive parallelism in Section 3.6.4.

3.6.1 Experimental Comparisons

We performed a number of different experiments using this set of algorithms; (1) A simple experiment in the batch case, in which the non-adaptive batch length algorithms are compared against one another, using a single batch length of $B = 5$ (Figure 3.2); (2) A corresponding experiment in the delay case, comparing GP-UCB, GP-BUCB, GP-AUCB, and GP-AUCB Local against one another, using a delay of $B = 5$ (Figure 3.3); (3) An experiment examining how changes in the batch length over the range $B = 5, 10$, and 20 affect performance of the non-adaptive algorithms (Figure 3.4), and a similar experiment where the maximum batch lengths for the adaptive algorithms vary over the same values (Figure 3.5); (4) A corresponding experiment in the delay setting, examining how

varying the delay length over the set $5, 10$, and 20 affects algorithm performance (Figure 3.6); and (5) an experiment which examines execution time for various algorithms in the batch case, comparing basic and lazy versions (see Section 3.5) of the algorithms presented (Figure 3.7). All experiments were performed in MATLAB using custom code, which we make publicly available for use. [1]

Comparisons of reward and regret among the algorithms discussed above are presented in terms of their cumulative regret, as well as their simple regret (how close did the points considered ever get to the maximum function value). Execution time comparisons are performed using wall-clock time elapsed since the beginning of the experiment, recorded at ends of algorithmic timesteps. All experiments were repeated for 200 trials, with independent tie-breaking and observation noise for each trial. Additionally, in those experimental cases where the reward function was a draw from a GP (the SE and Matérn problems), each trial used an independent draw from the same GP.

In the theoretical analysis in Section 3.3, the crucial elements in proving the regret bounds of GP-BUCB and GP-AUCB are C, the bound on the information which can be hallucinated within a batch and β_t, the exploration-exploitation tradeoff parameter, which is set with reference to C to ensure confidence interval containment of the reward function. For practical purposes, it is often necessary to define β_t and the corresponding parameter of GP-UCB, α_t, in a fashion which makes the algorithm considerably more aggressive than the regret bound requires. This removes the high-probability guarantees in the regret bound, but often produces excellent empirical performance. On the other hand, leaving the values for α_t and β_t as would be indicated by the theory results in heavily exploratory behavior and very little exploitation. In this chapter, in all algorithms which use the UCB or BUCB decision rules, the value of α_t has been set such that it has a small premultiplier (0.05 or 0.1, see Table 3.2), yielding substantially smaller values for α_t. Further, despite the rigors of analysis explored above in Section 3.3, we choose to set $\beta_t = \alpha_{\mathrm{fb}[t]}$ for the batch and delay algorithms, without reference to the value of C or the batch length B. Taking either of these measures removes the guarantees of correctness as carefully crafted in Section 3.3. However, as is verified by the experiments comparing batch sizes, this is not a substantial detriment to performance, even for large batch sizes, and indeed, the batch algorithms remain generally quite competitive with the sequential GP-UCB algorithm. One experimental advantage of this approach is that (with some limitations necessitated by the adaptive algorithms) the various algorithms using a UCB decision rule are using the same exploration-exploitation tradeoff parameter at the same iteration, including GP-UCB, GP-BUCB, GP-AUCB, and even SM-UCB and HBBO when using the UCB decision rule. This choice enables us to remove a confounding factor in comparing how well the algorithms overcome the disadvantages inherent in the batch and delay settings.

[1] See Appendix E.

3.6.2 Data Sets

We empirically evaluate GP-BUCB and GP-AUCB on several synthetic benchmark problems as well as two real applications. For each of the experimental data sets used in this chapter, the kernel functions and experimental constants are listed in Table 3.2. Where applicable, the covariance function from the GPML toolbox (Rasmussen and Nickisch, 2010) used is also listed by name. For all experiments, $\delta = 0.1$ (see Theorem 1) for UCB-based algorithms and tolerance $\epsilon = 0.02$ for HBBO. Each of the experiments discussed above was performed for each of the data sets described below and their results are presented, organized by experimental comparison (e.g., delay, adaptive batch size, etc.), in the accompanying Figures.

Problem Setting	Kernel Function	Hyperparameters	Noise Variance σ_n^2	Premultiplier (on α_t, β_t)
Matérn	covMaterniso	$l = 0.1$, $\sigma^2 = 0.5$	0.0250	0.1
SE	covSEiso	$l = 0.2$, $\sigma^2 = 0.5$	0.0250	0.1
Rosenbrock	RBF	$l^2 = 0.1$, $\sigma^2 = 1$	0.01	0.1
Cosines	RBF	$l^2 = 0.03$, $\sigma^2 = 1$	0.01	0.1
Vaccine	covLINone	$t_2 = 0.8974$	1.1534	0.05
SCI	covSEard	$l = [0.988, 1.5337, 1.0051, 1.5868]$, $\sigma^2 = 1.0384$	0.0463	0.1

Table 3.2: Experimental kernel functions and parameters.

3.6.2.1 Synthetic Benchmark Problems

We first test GP-BUCB and GP-AUCB in conditions where the true prior is known. A set of 100 example functions was drawn from a zero-mean GP with Matérn kernel over the interval $[0, 1]$. The kernel, its parameters, and the noise variance were known to each algorithm. The decision set D was the discretization of $[0, 1]$ into 1000 evenly spaced points. These experiments were also repeated with a Squared-Exponential kernel. Broadly speaking, these two problems were quite easy; the functions were fairly smooth, and for all algorithms considered, the optimum was found nearly every time, even for long batch sizes or delay lengths. Even for long batch lengths, as in Figures 3.4(a) and 3.4(b), which show substantial disadvantages in the average regret plots, the first batch has essentially all of the information needed to find the optimum, such that minimum regret after receiving the observations in the first batch is essentially zero. Similar sorts of results are present in the delay length experiments, where the adaptive algorithms which balk at spending queries early are able to do very well after receiving only a very few observations.

The Rosenbrock and Cosines test functions used by Azimi et al. (2010) were also considered, using the same Squared Exponential kernel as employed in their experiments, though with somewhat different lengthscales. The Rosenbrock test function shows a very strong skew toward actions near the upper end of the reward range, such that the minimum regret is often nearly zero before the first feedback is obtained. Since under conditions of the same batch length, most algorithms perform very comparably in terms of both average and minimum regret, the most interesting results are in the

case concerning delay length changes, Figure 3.6(c). In this figure, it is possible to see that GP-AUCB balks too often in this setting, leading to substantial losses in performance relative to GP-AUCB Local and GP-BUCB. The Cosines test function also shows broadly similar results across specific problem instances, where there is not a tremendous spread amongst the algorithms tested. Because the Cosines function is multi-modal, the average regret seems to show two-phase convergence behavior, in which all algorithms may be converging to a local optimum and then subsequently finding the global optimum. The overly frequent balking by GP-AUCB present in the Rosenbrock test function is also present for longer delays in the Cosines function, as can be seen in 3.6(g).

In both the Rosenbrock and Cosines delay experiments, one reason this behavior may occur has to do with the kernel chosen and how this interacts with the stopping condition, which requires that the information gain (either with respect to the reward function f as a whole or with respect to $f(\boldsymbol{x})$, $\forall \boldsymbol{x} \in D$) be less than a chosen constant C. With a flat prior, both GP-AUCB and GP-AUCB Local initially behave like Greedy Maximum Entropy Sampling (GMES). Since GMES gains a great deal of information globally, GP-AUCB tends to balk; on the other hand, since GMES scatters queries widely, the information gained with respect to any individual reward $f(\boldsymbol{x})$ is small, and so GP-AUCB Local tends not to balk much or at all. Since the information gain is calculated using the kernel used by the algorithm to model the function, misspecification of this kernel's longer-ranged properties may be particularly problematic for GP-AUCB, as opposed to GP-AUCB Local, which is (initially) more dependent on the local properties of the kernel and the assumed noise.

3.6.2.2 Automated Vaccine Design

We also tested GP-BUCB and GP-AUCB on a database of Widmer et al. (2010), as considered for experimental design by Krause and Ong (2011). This database describes the binding affinity of various peptides with a Major Histocompatibility Complex (MHC) Class I molecule, of importance when designing vaccines to exploit peptide binding properties. Each of the peptides which bound with the MHC molecule is described by a set of chemical features in $\mathbb{R}^{45}$, where each dimension corresponds to a chemical feature of the peptide. The binding affinity of each peptide, which is treated as the reward or payoff, is described as an offset IC_{50} value. The experiments used an isotropic linear kernel fitted on a different MHC molecule from the same data set. Since the data describe a phenomenon which has a measurable limit, many members of the data set are optimal; out of 3089 elements of D, 124, or about 4%, are in the maximizing set. In the simple batch experiments, Figures 3.2(h) and 3.2(k), GP-BUCB performs competitively with SM-MEI and SM-UCB, both in terms of average and minimum regret, and converges to the performance of GP-UCB. In the simple delay setting, Figures 3.3(h) and 3.3(k), both GP-BUCB and GP-AUCB produce superior minimum regret curves to that of GP-UCB, while performing comparably in terms of long-run average regret; this indicates that the more thorough initialization of GP-AUCB and GP-BUCB versus GP-UCB

may enable them to avoid early commitment to the wrong local optimum, finding a member of the maximizing set more consistently. This is consistent with the results of the non-adaptive batch size comparison experiment, Figures 3.4(h) and 3.4(k), which shows that as the batch size B grows, the algorithm must pay more "up front" due to its more enduring ignorance, but also tends to avoid missing the optimal set entirely. This same sort of tradeoff of average regret against minimum regret is clearly visible for the GP-AUCB Local variants in the experiments sweeping maximal batch size for adaptive algorithms, Figures 3.5(h) and 3.5(k).

3.6.2.3 Spinal Cord Injury (SCI) Therapy

Lastly, we compare the algorithms on a pre-recorded data set of leg muscle activity triggered by therapeutic spinal electrostimulation in spinal cord injured rats. This setting is intended to mimic the on-line experiments conducted in Chapter 4. Much greater detail is given regarding the experimental design in that chapter, but, in brief, the procedure is as follows. From the 3-by-9 grid of electrodes on the array, a pair of electrodes is chosen to activate, with the first element of the pair used as the cathode and the second used as the anode. Electrode configurations were represented in $\mathbb{R}^4$ by the cathode and anode locations on the array. These active array electrodes create an electric field which may influence both incoming sensory information in dorsal root processes and the function of interneurons within the spinal cord, but the precise mechanism of action is poorly understood. Since the goal of this therapy is to improve the motor control functions of the lower spinal cord, the designated experimental objective is to choose the stimulus electrodes which maximize the resulting activity in lower limb muscles, as measured by electromyography (EMG). We used data with a stimulus amplitude of 5V and sought to maximize the peak-to-peak amplitude of the recorded EMG waveforms from the right medial gastrocnemius muscle in a time window corresponding to a single interneuronal delay. Note that in Chapter 4, the muscle chosen is the left tibialis anterior, but the procedure is fundamentally the same. This objective function attempts to measure the degree to which the selected stimulus activates interneurons controlling reflex activity in the spinal gray matter. This response signal is non-negative and for physical reasons does not generally rise above 3mV. A squared-exponential ARD kernel was fitted using experimental data from 12 days post-injury. Algorithm testing was done using an oracle composed of data from 116 electrode pairs tested on the 14th day post-injury.

Like the Vaccine data set, the SCI data set displayed a number of behaviors which indicate that the problem instance was difficult; in particular, the same tendency that algorithms which initialized more thoroughly would eventually do better in both minimum and average regret was observed. This tendency is visible in the simple batch setting (Figures 3.2(i) and 3.2(l)), where GP-UCB was not clearly superior to either GP-BUCB or GP-AUCB; this is surprising because being required to work in batches, rather than one query at a time, might be expected to give the algorithm less information

at any given round, and should thus be a disadvantage; this under-exploration in GP-UCB may be a result of the value of the exploration-exploitation tradeoff parameter α being chosen to promote greater aggressiveness across all algorithms. Interestingly, this data set also displays both a small gap between the best and second-best values of the reward function (approximately 0.9% of the range) and a large gap between the best and third-best (approximately 7% of the range). When examining how many out of the individual experimental runs simulated selected $\boldsymbol{x}^* = \text{argmax}_{\boldsymbol{x} \in D} f(\boldsymbol{x})$ on the 200th query in the simple batch case, only 20% of GP-UCB runs choose $\boldsymbol{x}^*$; the numbers are considerably better for GP-BUCB, SM-UCB, and SM-MEI, at 35%, 30.5%, and 36%, but are still not particularly good. If the first sub-optimal $\boldsymbol{x}$ is also included, these numbers improve substantially, to 63.5% for GP-UCB and 84%, 91%, and 96.5% for GP-BUCB, SM-UCB, and SM-MEI. These results indicate that the second-most optimal $\boldsymbol{x}$ is actually easier to find than the most optimal, to a fairly substantial degree. It is also important to place these results in the context of the experimental setting; even assuming that there truly is a difference between these pairs of SCI electrodes, the rewards produced are so close to one another as to likely produce no therapeutic difference between the most optimal and second-most optimal actions. Since all of GP-BUCB, SM-UCB, and SM-MEI more consistently found one of the two best actions in the decision set than GP-UCB, all of them showed strong performance in comparison to GP-UCB.

3.6.3 Computational Performance

Another test of interest across the set of experiments discussed above was the degree to which lazy variance calculations, as described in Section 3.5, reduced the computational overhead of each of the algorithms discussed. These results are presented in Table B.6 and Figure 3.7. Note that for algorithms which appear as both lazy and non-lazy versions, the only functional difference between the two is the procedure by which the action is selected, not the action selection itself; all computational gains are without sacrificing accuracy and without requiring any algorithmic approximations. All computational time experiments were performed on a desktop computer (quad-core Intel i7, 2.8 GHz, 8 GB RAM, Ubuntu 10.04) running a single MATLAB R2012a process.

For all data sets, the algorithms lie in three broad classes: Class 1, comprised of the lazy GP-UCB family of algorithms; Class 2, the non-lazy versions of the GP-UCB family of algorithms, as well as the HBBO UCB and MEI variants; Class 3, consisting of the SM-MEI and SM-UCB algorithms, in both lazy and non-lazy versions. Class 1 algorithms run to completion about one order of magnitude faster than those in Class 2, which in turn are about one order of magnitude faster than those in Class 3. The various versions of the simulation matching algorithm of Azimi et al. (2010), require multiple samples from the posterior over f to aggregate together into a batch, the composition of which is intended to match or cover the performance of the corresponding sequential UCB or MEI algorithm. The time difference between Class 2 and Class 3, approximately one order of magnitude,

reflects the choice to run 10 such samples. Within Class 3, our implementation of the lazy version of SM-MEI is slower than the non-lazy version, largely due to the increased overhead of sorting the decision rule and computing single values of the variance; a more efficient implementation of either or both of these elements could perhaps improve on this tradeoff. The lazy algorithms also tend to expend a large amount of computational time early, computing upper bounds on later uncertainties, but tend to make up for this early investment later; this is even visible with regard to the lazy version of SM-UCB, which is initially slower than the non-lazy version, but scales better and, in all six data sets examined, ends up costing substantially less computational time by the 200th query.

3.6.4 Parallelism: Costs and Tradeoffs

Parallelism is motivated by the setting in which each round or opportunity to submit a query is expensive, but the additional marginal cost of taking an action at that round is not very large. It is interesting to consider more precisely what we mean by "expensive" or "not very large." For a given relationship between the individual costs, one can examine which algorithms most effectively trade these costs off against one another. One way to do this is to experimentally measure the costs incurred by several algorithms solving the same problem. In the following discussion, we examine the delay case, in which the algorithm is faced with a choice of which action (if any) to take at each round. A similar examination of cost tradeoffs can be made in the batch case.

Given N sample runs, a successful algorithm should have a (nearly) monotonically decreasing sample average regret curve, defined as $\bar{r}(T) = \sum_{n=1}^{N} R_{T,n}/T$, where $R_{T,n}$ is the cumulative regret of the nth run at round T. This curve can be inverted to find the first round $\tau(\bar{r})$ in which the average regret is at or below a particular value $\bar{r}$. The sample mean cost of running the algorithm until round $\tau(\bar{r})$ can then be computed. The cost of the nth run is the sum of two contributions, the first for running $\tau(\bar{r})$ rounds of the algorithm, and the second for the actual execution of $a_n(\tau(\bar{r}))$ actions. Parameterizing the relative costs of each round and each action using w, the average cost $C(\bar{r}, w) = (1 - w)\tau(\bar{r}) + w \cdot \bar{a}(\tau(\bar{r}))$ corresponding to a particular average regret value $\bar{r}$ can be obtained, where $\bar{a}(\tau(\bar{r})) = \sum_{n=1}^{N} a_n(\tau(\bar{r}))$. Note that $w \in [0, 1]$ translates to any constant, non-negative ratio of the cost of a single action to that of a single round. As a technical point, note that this average cost is not exactly equivalent to specifying $\bar{r}$, continuing each individual run until $R_{T,n}/T \leq \bar{r}$, and averaging the individual costs incurred in so doing; such a method, while more intuitive, must deal with the problem that a run may fail to ever attain a specified value of $\bar{r}$, e.g., a run could fail to converge to the optimum. This failure to converge happens with a non-zero probability, and is theoretically treated in Theorem 1 through δ. The post-hoc calculation of $C(\bar{r}, w)$ as proposed here is robust to this case, giving an estimate of the expected cost to run the algorithm until a round in which the expected cumulative average regret is below $\bar{r}$.

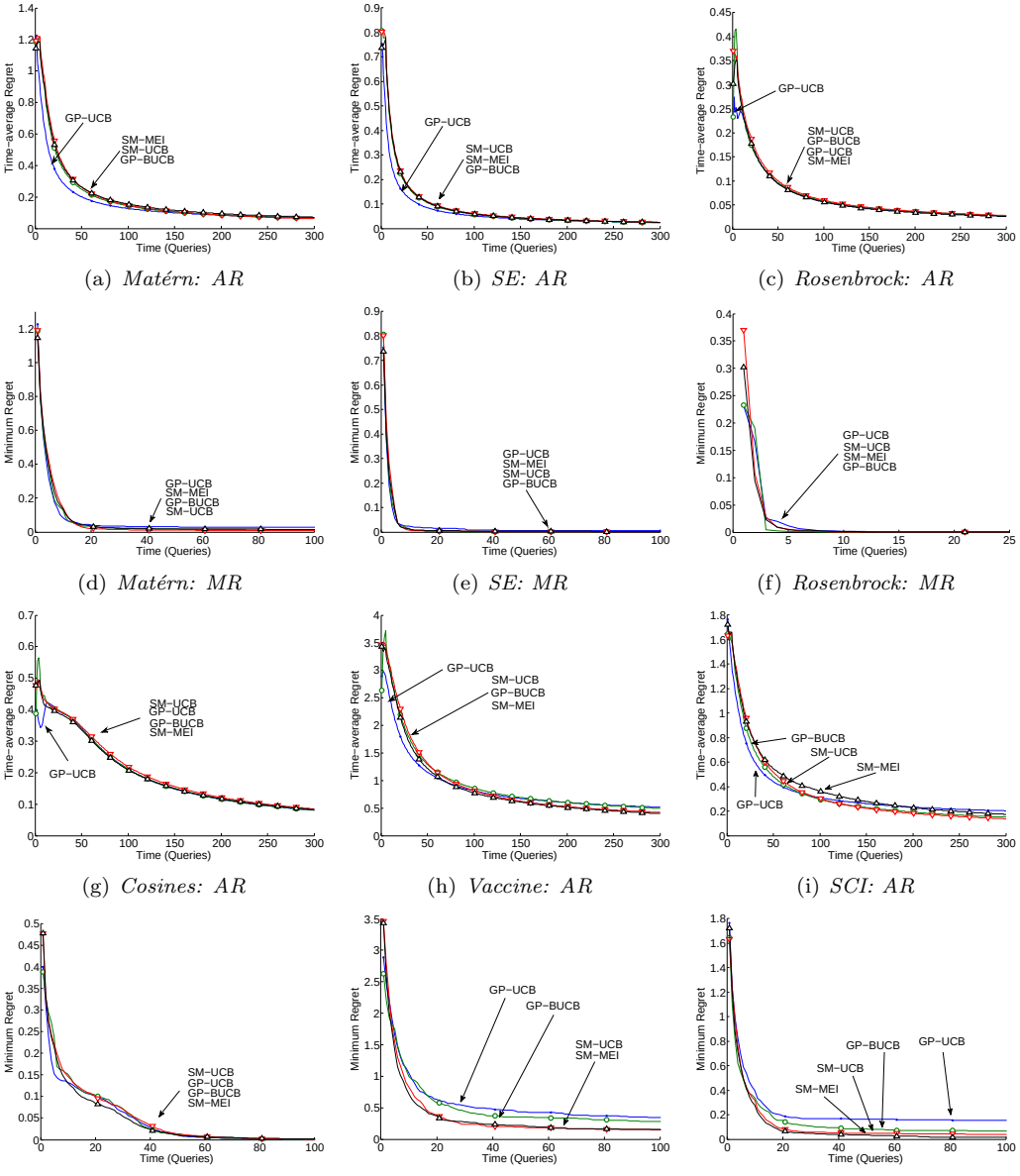
Among a set of algorithms, and given a test problem, one can find which among them has the

lowest value of $C(\bar{r}, w)$ at any particular point in the $\bar{r}, w$ space. Similarly, for any fixed value of w, it is possible to once more invert the function and plot $\bar{r}_w(C)$; this plot resembles conventional average regret plots, and corresponds to intersection of the set of $C(\bar{r}, w)$ surfaces with the plane at a fixed w. A comparison between three algorithms on the SCI data set, with simple delay $B = 5$, is shown in Figure 3.8. In this scenario, GP-AUCB costs the least through most of the parameter space, due to its tendency to pass in early rounds, when the potential for exploitation is lowest. The advantage changes to the fully sequential algorithm when w is large (i.e., parallelism is expensive), and to GP-BUCB when w is small. Many real-world situations lie somewhere between these extremes, suggesting that GP-AUCB may be useful in a variety of scenarios.

3.7 Conclusions

We develop the GP-BUCB and GP-AUCB algorithms for parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization. The analytical framework used to bound the regret of GP-BUCB and GP-AUCB is generalized to include all GP-UCB-type algorithms. We prove Theorem 1, which provides high-probability bounds on the cumulative regret of algorithms in this class, which hold for both the batch and delay setting. These bounds consequently provide guarantees on the convergence of such algorithms. Further, we prove Theorem 4, which establishes a high-probability regret bound for initialized GP-BUCB. This bound scales independently of the batch size or delay length B, if B is constant or polylogarithmic in T. Finally, we introduce lazy variance calculations, which dramatically accelerate computation of GP-based active learning decision rules.

Across the experimental settings examined, GP-BUCB and GP-AUCB performed comparably with state of the art parallel and adaptive parallel Bayesian optimization algorithms, which are not equipped with theoretical bounds on regret. GP-BUCB and GP-AUCB also perform comparably to the sequential GP-UCB algorithm, indicating that GP-BUCB and GP-AUCB successfully overcome the disadvantages of only receiving delayed or batched feedback. With respect to cost to achieve a given level of regret, GP-AUCB appears to offer substantial advantages over the fully parallel or fully sequential approaches. We believe that our results provide an important step towards solving complex, large-scale exploration-exploitation tradeoffs.



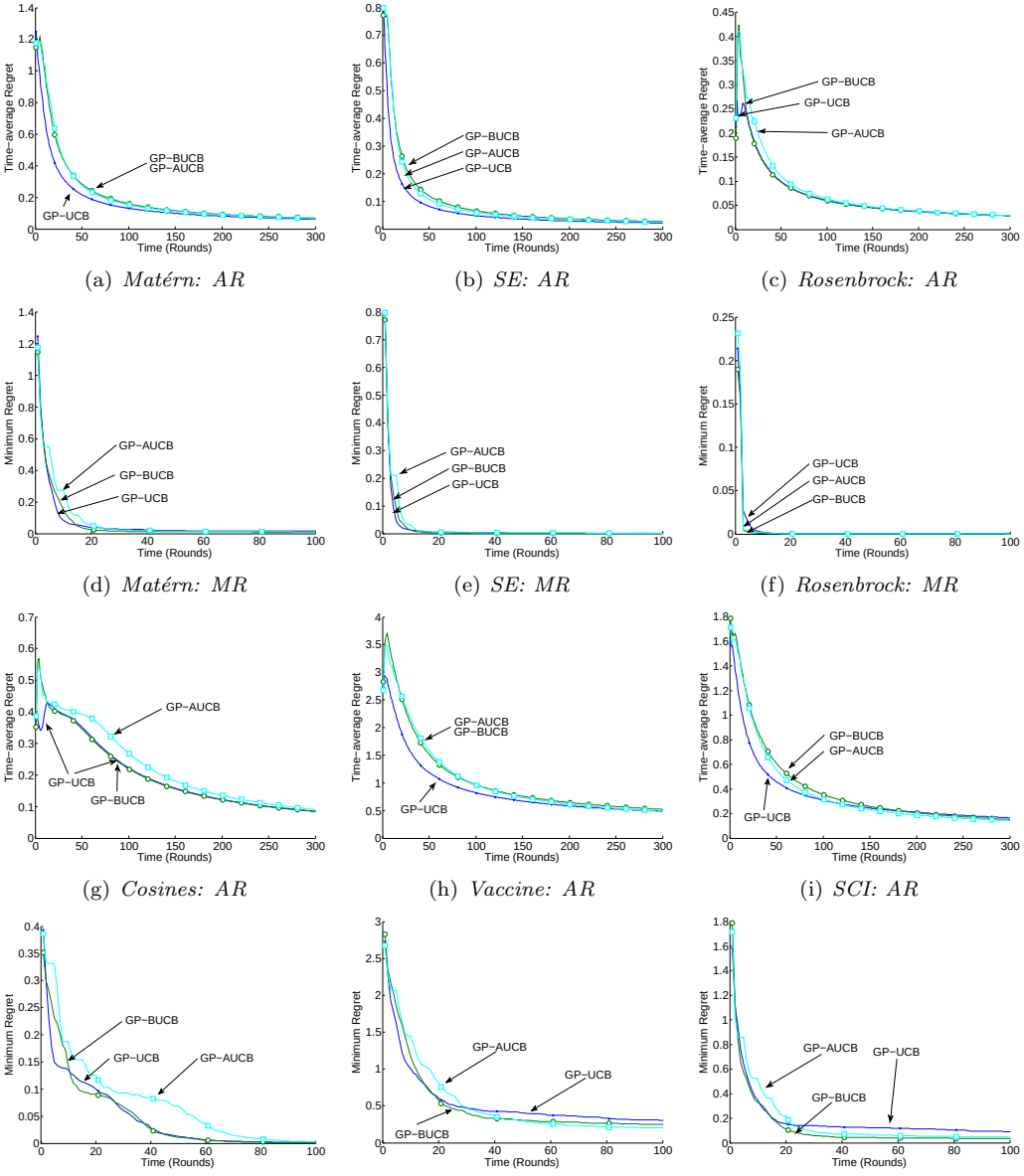
(a) *Matérn: AR* (b) *SE: AR* (c) *Rosenbrock: AR*

(d) *Matérn: MR* (e) *SE: MR* (f) *Rosenbrock: MR*

(g) *Cosines: AR* (h) *Vaccine: AR* (i) *SCI: AR*

(j) *Cosines: MR* (k) *Vaccine: MR* (l) *SCI: MR*

Figure 3.2: Time-average (AR) and minimum (MR) regret plots, batch setting, for a batch size of 5.



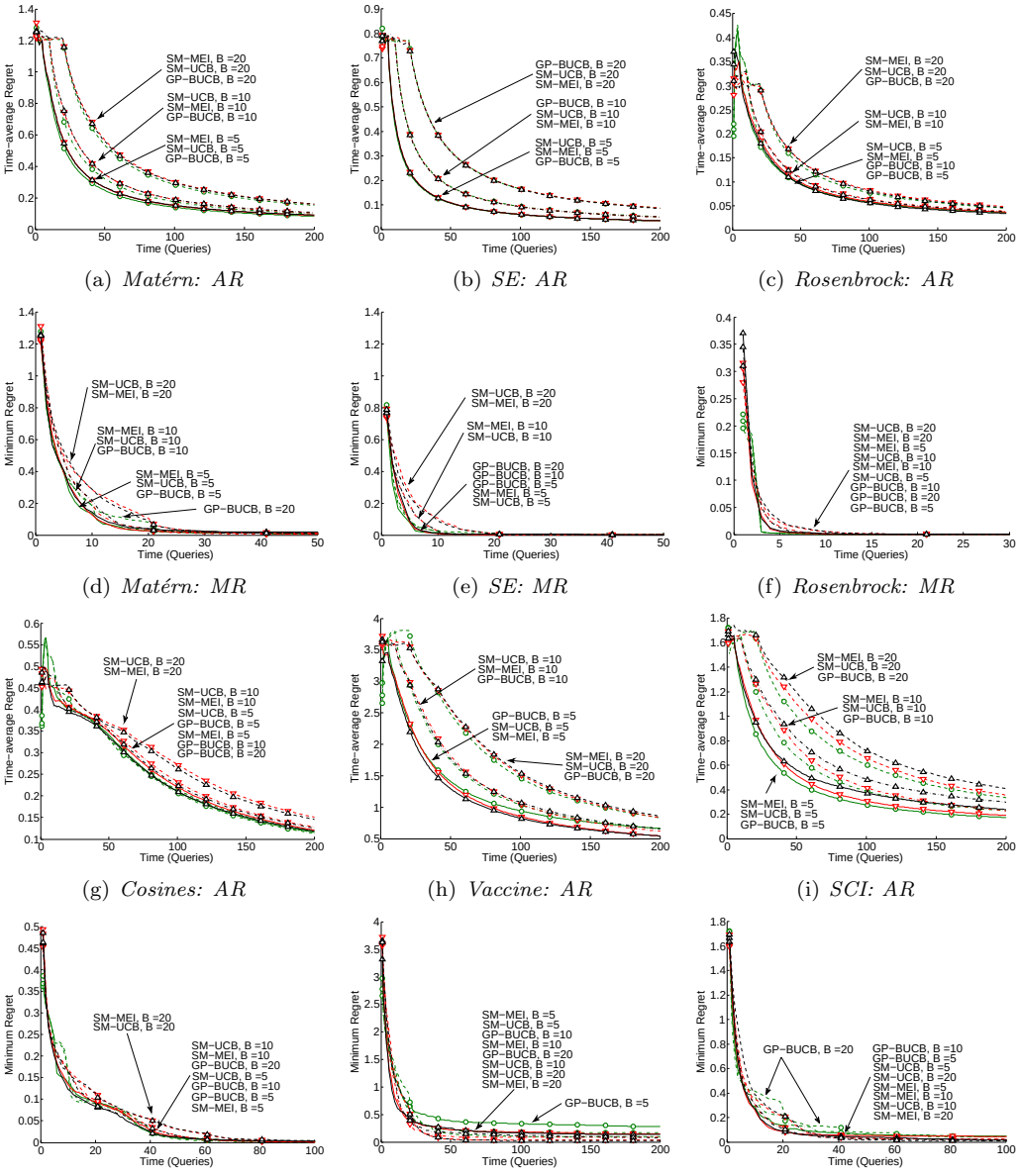
(a) *Matérn: AR* (b) *SE: AR* (c) *Rosenbrock: AR*

(d) *Matérn: MR* (e) *SE: MR* (f) *Rosenbrock: MR*

(g) *Cosines: AR* (h) *Vaccine: AR* (i) *SCI: AR*

(j) *Cosines: MR* (k) *Vaccine: MR* (l) *SCI: MR*

Figure 3.3: Time-average (AR) and minimum (MR) regret plots, delay setting, with a delay length of 5 rounds between action and observation.



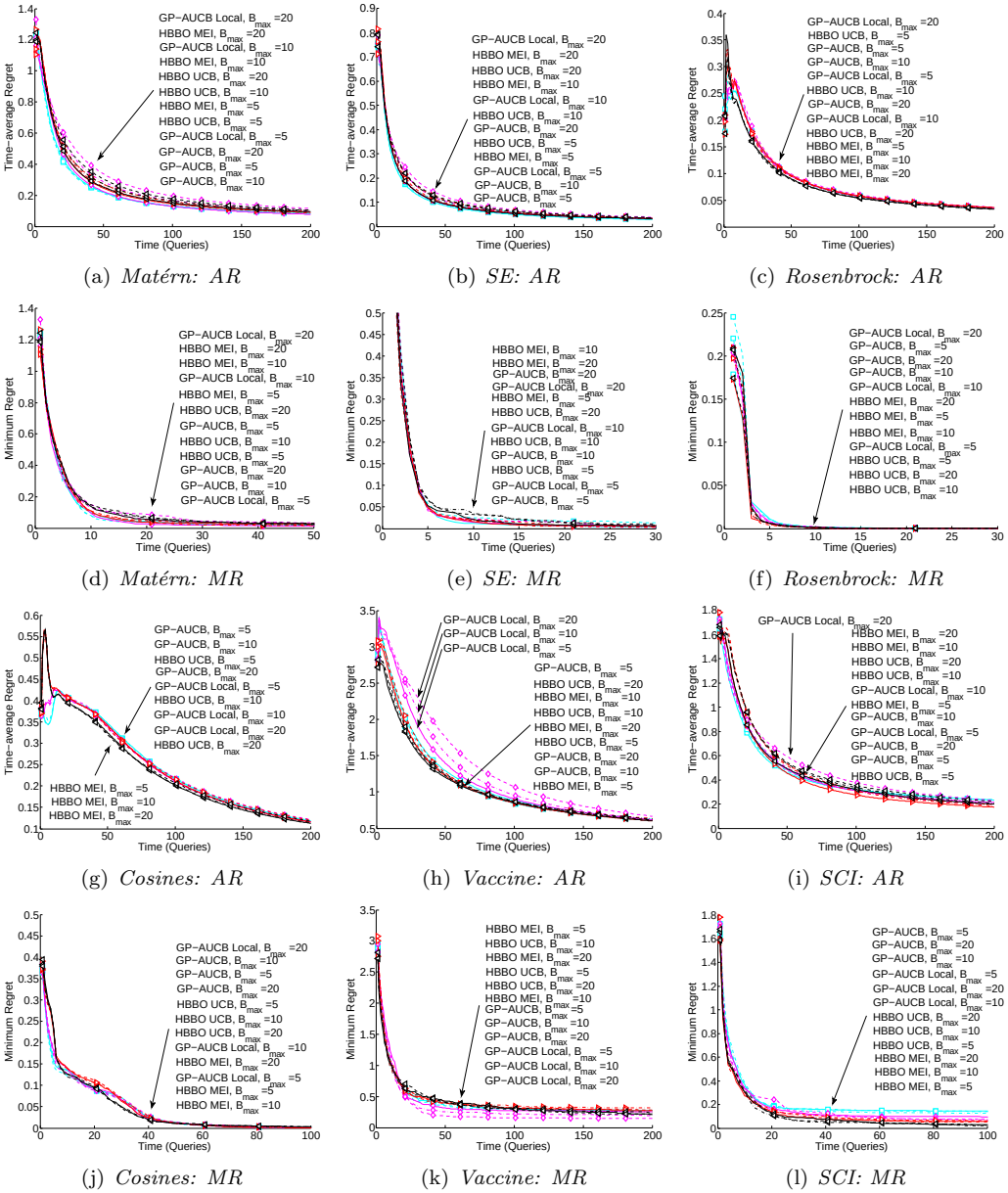
(a) *Matérn: AR* (b) *SE: AR* (c) *Rosenbrock: AR*

(d) *Matérn: MR* (e) *SE: MR* (f) *Rosenbrock: MR*

(g) *Cosines: AR* (h) *Vaccine: AR* (i) *SCI: AR*

(j) *Cosines: MR* (k) *Vaccine: MR* (l) *SCI: MR*

Figure 3.4: Time-average (AR) and minimum (MR) regret plots, non-adaptive batch algorithms, batch sizes 5, 10, and 20.



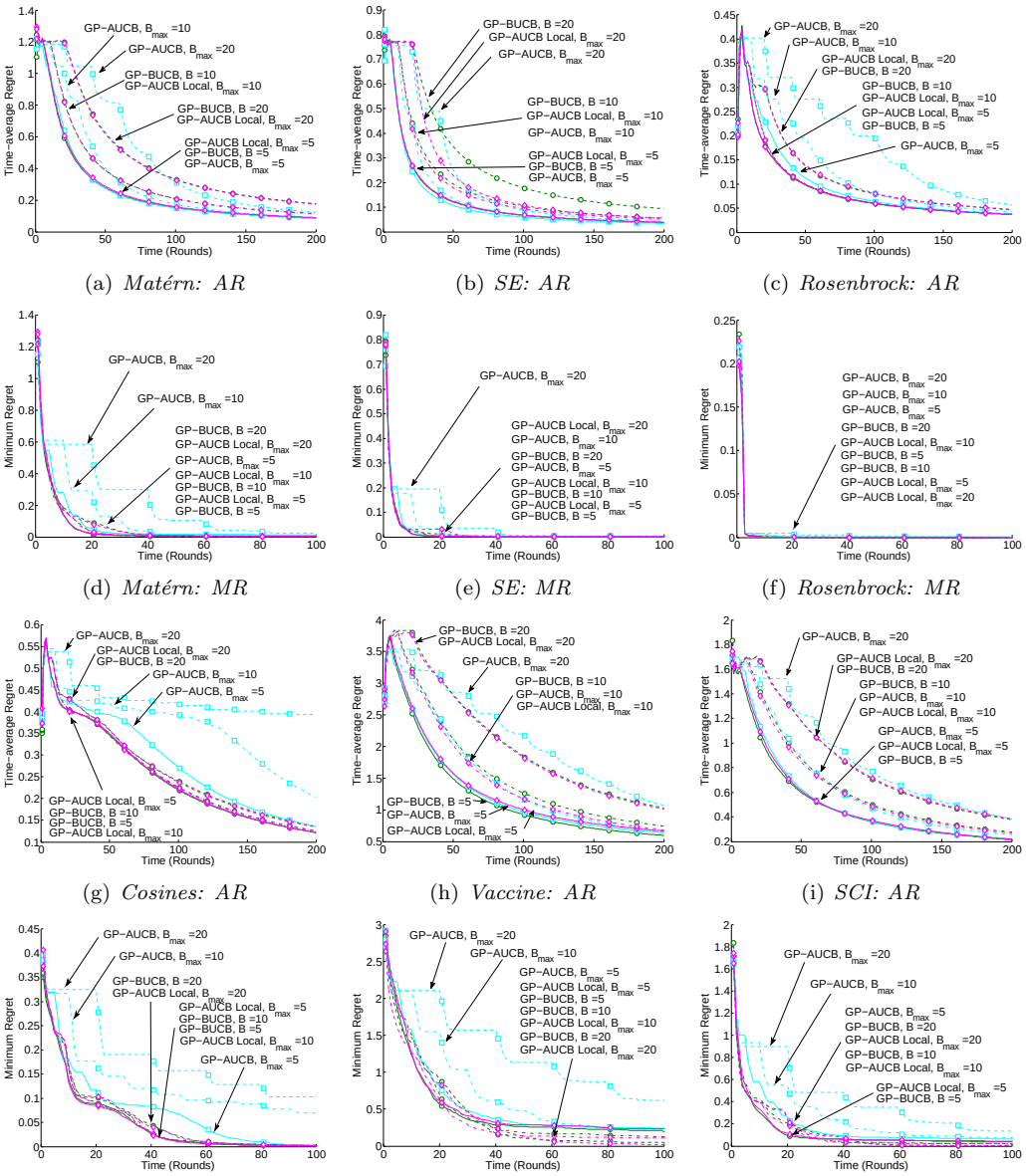
(a) *Matérn: AR* (b) *SE: AR* (c) *Rosenbrock: AR*

(d) *Matérn: MR* (e) *SE: MR* (f) *Rosenbrock: MR*

(g) *Cosines: AR* (h) *Vaccine: AR* (i) *SCI: AR*

(j) *Cosines: MR* (k) *Vaccine: MR* (l) *SCI: MR*

Figure 3.5: Time-average (AR) and minimum (MR) regret plots, adaptive batch algorithms, maximum batch sizes 5, 10, and 20. For the adaptive algorithms, minimum batch size B_{min} was set to 1, as in HBBO. The algorithms tended to run fully sequentially at the beginning, but quite rapidly switched to maximal parallelism.



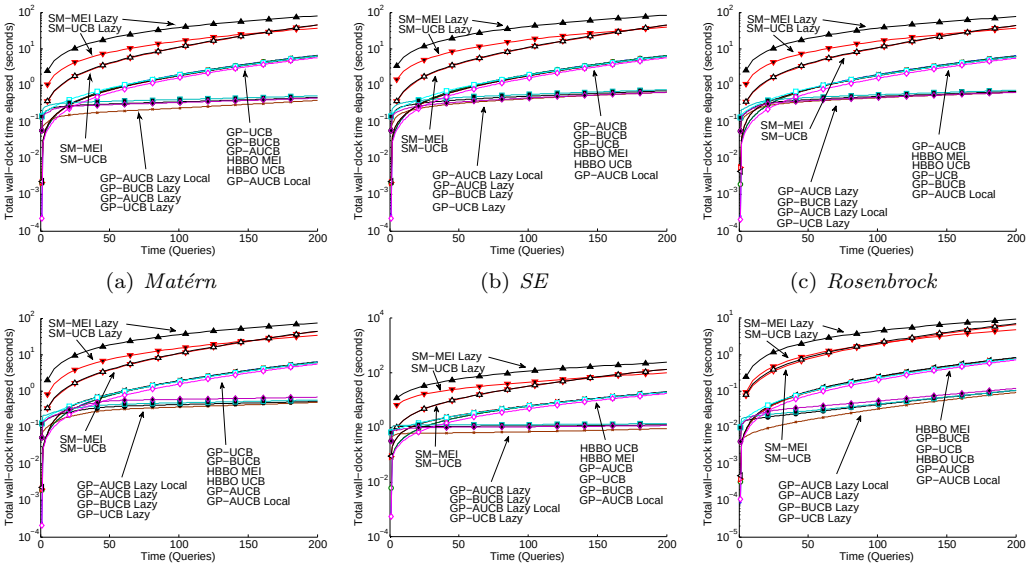
(a) *Matérn: AR* (b) *SE: AR* (c) *Rosenbrock: AR*

(d) *Matérn: MR* (e) *SE: MR* (f) *Rosenbrock: MR*

(g) *Cosines: AR* (h) *Vaccine: AR* (i) *SCI: AR*

(j) *Cosines: MR* (k) *Vaccine: MR* (l) *SCI: MR*

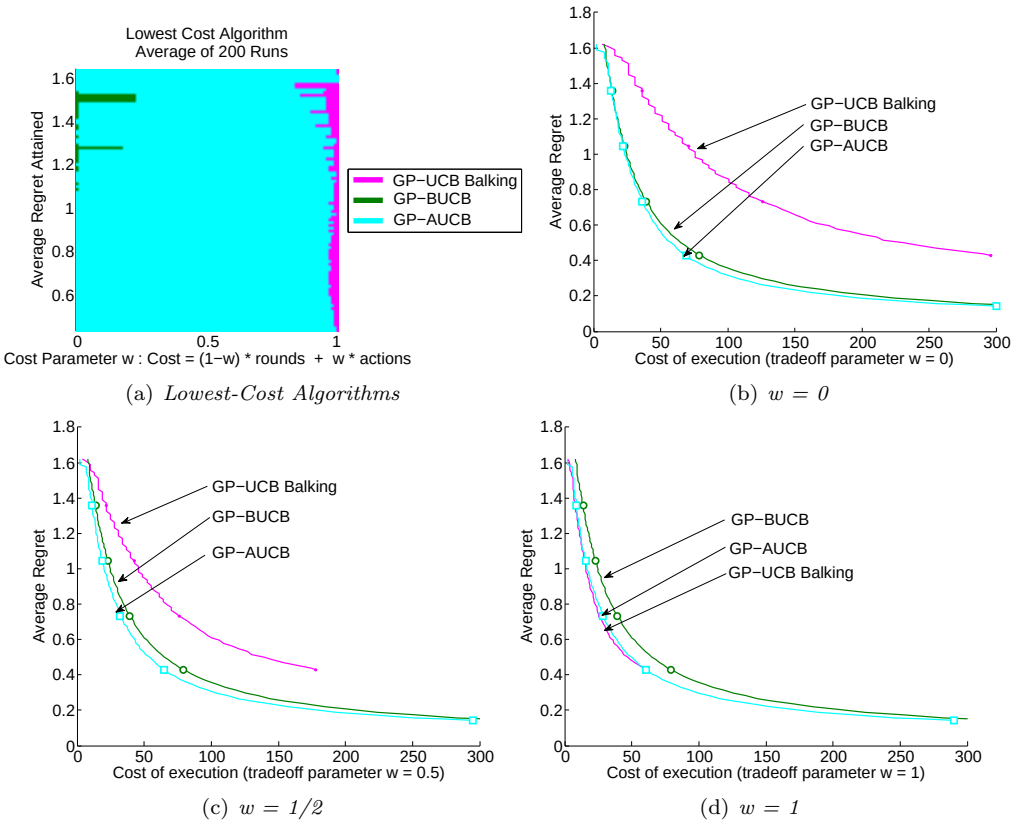
Figure 3.6: Time-average (AR) and minimum (MR) regret plots, delay setting, with delay lengths of 5, 10, and 20 rounds between action and observation. Note that the adaptive algorithms, GP-AUCB and GP-AUCB Local, may balk at some rounds. The time-average regret is calculated over the number of actions actually executed as of that round; this means that the number of queries submitted as of any particular round is hidden with respect to the plots shown, and may vary across runs of the same algorithm.



(a) *Matérn* (b) *SE* (c) *Rosenbrock*

(d) *Cosines* (e) *Vaccine* (f) *SCI*

Figure 3.7: Elapsed computational time in batch experiments, B = 5. Note the logarithmic vertical scaling in all plots. Note also the substantial separation between the three groups of algorithms, discussed in Section 3.6.3.



(a) *Lowest-Cost Algorithms* (b) $w = 0$

(c) $w = 1/2$ (d) $w = 1$

Figure 3.8: Parameterized cost comparison on the SCI data set, simple delay case, $B = 5$. The same experiment, with a different set of algorithms shown, is presented in Figure 3.3(i). Figure 3.8(a): the space of cost tradeoff parameter w and attained average regrets $\bar{r}$ is colored according to which algorithm has the lowest mean cost at the round in which the mean, time-average regret is first $\leq \bar{r}$. The algorithm denoted GP-UCB Balking refuses to submit another query while one is pending, i.e., it is the GP-UCB algorithm obeying the delay constraint of the problem setting. Figures 3.8(b), 3.8(c), and 3.8(d) show $\bar{r}$ as a function of C and correspond to vertical slices through Figure 3.8(a) at the left, center, and right. Since GP-AUCB and GP-UCB Balking pass on some rounds, the terminal cost of GP-AUCB and GP-UCB Balking is possibly < 300.