SOBER: Highly Parallel Bayesian Optimization and Bayesian Quadrature over Discrete and Mixed Spaces

Masaki Adachi
Machine Learning Research Group, University of Oxford
Toyota Motor Corporation
masaki@robots.ox.ac.uk

Satoshi Hayakawa, Harald Oberhauser
Mathematical Institute, University of Oxford
{hayakawa,oberhauser}@maths.ox.ac.uk

Saad Hamid, Martin Jørgensen, Michael A. Osborne
Machine Learning Reserach Group, University of Oxford,
{saad, martinj, mosb}@robots.ox.ac.uk

Abstract

Batch Bayesian optimisation and Bayesian quadrature have been shown to be sample-efficient methods of performing optimisation and quadrature where expensive-to-evaluate objective functions can be queried in parallel. However, current methods do not scale to large batch sizes — a frequent desideratum in practice (e.g. drug discovery or simulation-based inference). We present a novel algorithm, SOBER, which permits scalable and diversified batch global optimisation and quadrature with arbitrary acquisition functions and kernels over discrete and mixed spaces. The key to our approach is to reformulate batch selection for global optimisation as a quadrature problem, which relaxes acquisition function maximisation (non-convex) to kernel recombination (convex). Bridging global optimisation and quadrature can efficiently solve both tasks by balancing the merits of exploitative Bayesian optimisation and explorative Bayesian quadrature. We show that SOBER outperforms 11 competitive baselines on 12 synthetic and diverse real-world tasks.

1 Introduction

Bayesian optimisation (BO) is a sample-efficient model-based global optimiser. BO typically use a Gaussian process (GP), whose predictive mean and variance guide the optimiser where to evaluate next by maximising the acquisition function (AF). Flexibility and superb sample-efficiency enable a range of expensive-to-evaluate applications, e.g. drug discovery [18, 22], materials [1], and hyperparameter optimisation [16, 70]. Batch BO offers faster convergence by querying multiple locations at once (batch acquisition). Bayesian quadrature (BQ), a model-based blackbox integration method akin to BO, widely applied in simulation-based inference. The task is to estimate both the posterior *and* marginal likelihood (also called evidence) of blackbox functions (simulators), applied to science (e.g. astrophysics [48, 47], batteries [3]).

Challenges in Batch Bayesian Optimisation Tasks. Despite its many successes, batch BO has several challenges. Firstly, *batch size scalability*: The extensive overhead of many batch BO methods limits the batch size to be around 10. Scaling to a larger batch size is preferable for a variety of real-world problems. For instance, high-throughput drug discovery might evaluate 384 compounds

ICML 2023 Workshop: Sampling and Optimization in Discrete Space (SODS 2023).

Table 1: Comparing batch BO algorithms. The scalability to large batch sizes is defined by the computational complexity smaller or equivalent complexity to Thompson sampling (TS) (See Supp. B). Comb. explos. refers to combinatorial explosion when applied to discrete inputs with large batch size. The sparsity means more diversified samples than random Monte Carlo. x, f, and h spaces refer to the input, function, and hyperparameter spaces. The black-box evidence represents the ability to estimate the integral of the black-box function over input space x, equivalent to the task of BQ.

Batch BOs	large batch	mixed space	No comb. explos.	arbitrary AF	x-space sparsity	f-space sparsity	h-space sparsity	blackbox evidence
Hallucination [4]	×	✓	×	✓	✓	×	×	×
LP [19]	✓	×	×	✓	✓	×	×	×
TS [30]	✓	✓	✓	×	×	×	×	×
Decoupled TS [69]	✓	✓	✓	×	×	✓	×	×
DPP [37]	×	✓	×	×	✓	×	×	×
DPP-TS [45]	✓	✓	✓	×	✓	×	×	×
MC-SAA [5]	✓	✓	×	×	✓	×	×	×
GIBBON [43]	×	✓	×	×	✓	×	×	×
TurBO [15]	✓	×	×	✓	✓	×	×	×
SOBER (Ours)	✓	✓	✓	✓	✓	✓	✓	✓

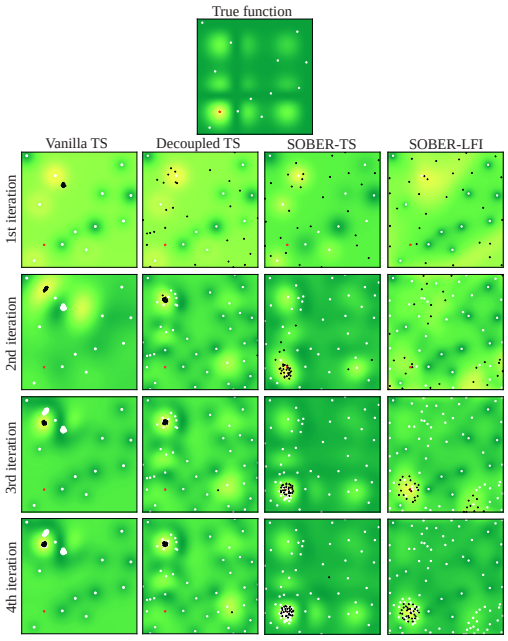

Figure 1: Thompson sampling is under-explorative. The white, black, and red points are the observed, querying, and true maximum locations.

in a batch experiment [8]. Further, *in-Silico* materials discovery can query thousands of simulations in parallel via computer clusters. As a second challenge, many batch BO methods are targeted at *continuous inputs*, yet the examples mentioned above of drug discovery, namely molecules, are inherently discrete. Selecting batch acquisition samples in a discrete space leads to combinatorial explosion with increasing batch size. Lastly, *batch diversity*: scalable methods, such as Thompson sampling (TS) [30, 36], are too exploitative. The two leftmost columns in Figure 1 exemplifies typical behaviour – getting stuck in a local minimum and wasting batch samples in the majority of TS methods. This tendency amplifies in noisy and multimodal cases. The larger the batch size we query, the larger the regret becomes, as the batch samples could have been used for exploring other regions.

Figure 2 shows drug discovery is a case where the input space is noisy and multimodal. This needle-in-the-haystack situation is challenging for finding the bias-variance trade-off via optimising GP hyperparameters. It leads to having two modes: low-noise and high-noise in the hyperposterior space. A low-noise mode regards every tiny change as a peak, whereas high-noise sees every peak as noise. While the low-noise mode tends to get stuck in local minima, the high-noise mode can not find the best drug. Such problems are usually solved by adopting fully Bayesian GP (FBGP), which uses an ensemble of GP models with hyperparameters sampled from hyperposteriors [53]. Yet, FBGP causes significant overhead that challenges for expensive batch BOs (e.g. determinantal point process (DPP) [37]).

Challenges in Batch Bayesian Quadrature Tasks. Unlike BO, BQ endures 'over-exploration' when applied for Bayesian inference. The exploration space in BQ is defined as the prior distribution, whereas the target distribution to approximate, posterior, is much sharper than prior and shows negligibly small values almost everywhere. Guiding BQ to explore the posterior mode is key to sample-efficiency.

Contribution. We propose the hallucination-free scalable batching: solving optimisation as Bayesian estimation via recombination (SOBER). SOBER bridges BO and BQ, combining both merits to solve the above issues of each method — While exploitative BO alleviates over-exploration for BQ, explorative BQ mitigates the stuck-in-local-minima issue via diversified sampling for BO (see two rightmost columns of Figure 1). Table 1 summarises a comparison with the batch BO baselines. With the given features, we particularly focus on drug discovery (large-scale discrete batch BO with graph/string input, batch BO task), and simulation-based inference (simultaneous estimation of both posterior and evidence, batch BQ task). Empirically, SOBER shows better sample-efficiency as well as faster wall-clock time computation than 11 baselines in both batch BO and BQ over 12 tasks. We emphasise that SOBER is an extension of both BO and BQ and is compatible with existing methods.

2 Related Work and Challenges

Batch Bayesian Optimisation. Batch BOs are summarised in Table 1 (see Supp. A.1 for primer). While BO is an inherently sequential algorithm, assuming queries to an oracle are performed one after another, batch BO aims to query multiple locations in one go. However, most AFs are designed only for querying one point. Classic methods, like hallucination [4] and local penalisation (LP) [19], tackled this by simulating sequential processes, both of which are successful in small batch size n, but not scalable due to large computational overhead. In recent work, MC-SAA [5] proposed quasi-Monte Carlo (qMC)-based AF approximation, and GIBBON [43] proposed diversified sampling methods using a specific AF. However, discrete and mixed spaces also present challenges for the above algorithms. The simplest way to maximise AFs over a discrete space is to take the argmax of all possible candidates. Yet, the higher the dimension and larger the number of categorical classes, the more infeasibly large the combination becomes (combinatorial explosion). This is particularly challenging for MC-SAA and GIBBON, as both optimise the batch querying points as optimisation variables. Namely, querying large batch sizes requires enumerating all possible permutations of *both batch samples and discrete variables*, leading to a combinatorial explosion. DPP [37] also proposed diversified sampling with rejection sampling, however, it produces prohibitive overhead and is not scalable. Alternatively, there exist BO works for discrete and mixed spaces that propose bespoke AF [6, 54, 10] or special kernel [67, 46, 11, 64]. All consider sequential setting — we do not compare against sequential BO. Rather, these special AF/kernels are compatible with our method, SOBER.

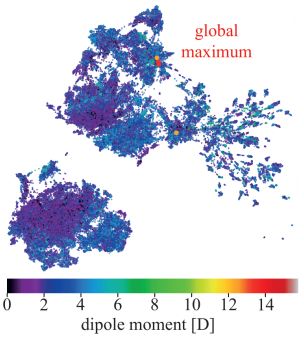

Figure 2: Drug discovery is finding the needle in the hey stack. 2D UMAP [42] shows only 5 out of 133,055 molecules on the QM9 dataset met desideratum (> 10D).

As such, the existing scalable discrete batch BO methods are TS-based (TS [30], decoupled TS [69], DPP-TS [45], TurBO [15]). TS is approximated by the sequential argmax of random samples over input space, which is completely gradient-free, allowing for scalable batching over a discrete space. Yet, batch TS is not so diversified. Decoupled TS diversifies TS batch samples by decoupled sampling that sparsifies the GP function space. Although decoupled sampling yields scalable batch sampling, diversification is not enough as shown in Figure 1. DPP-TS tried to take the best of both worlds of DPP and TS, and is faster than DPP, but still slower than others. TurBO [15] introduced multiple local BOs bounded with trust regions, and allocates batching budgets based on TS. This succeeded in scalable batching via maintaining local BOs that are compact, via shrinking trust regions, based on heuristics with many hyperparameters. Selecting hyperpameters is non-trivial and TurBO cannot apply to discrete and non-Euclidean space, for which kernels do not have lengthscale hyperparameters for the trust region update heuristic (e.g. Tanimoto kernel for drug discovery [51]). (See Supp. B).

Batch Bayesian Quadrature. A primer on BQ can be found in [28] and Supp. A.2. BQ shares analogous challenges on batching with BO. While BO needs accurate approximation around the global optimum, integration approximation error in BQ is affected by whole function space. Thus, BQ adopts uncertainty sampling AF [47, 23]. BatchWSABI [66] was the first to extend BQ to a batch setting using LP [19]. Later, Bayesian alternately subsampled quadrature (BASQ) achieved efficient batching with kernel recombination and is the only method for scalable batching [2]. However, BASQ is over-explorative; logBASQ [3] mitigates this by using the log-warped GP.

2.1 Bayesian Alternately Subsampled Quadrature (BASQ)

The difficulty of scalable batching is the sequential AF maximisation strategy, of which the large overhead and the continuity assumption hinder scalable discrete space optimisation. The TS strategy is too exploitative and less diversified. BASQ [2] achieved scalable batch BQ with a different approach from existing works. Instead of AF maximisation, BASQ reframed batch uncertainty sampling (batch BQ) as kernel quadrature (KQ). KQ is the problem of sample-efficient integration approximation as a weighted sum $\int f(x)\mathrm{d}\pi(x) \approx \mathbf{w}_{\text{batch}}^\top f(\mathbf{X}_{\text{batch}})$, of which the integrand function f belongs to reproducing kernel Hilbert space (RKHS), typically applied to the positive semi-definite kernel. Let the kernel K be the posterior predictive covariance of the GP. Surprisingly, minimising worst-case integration error via selecting n weighted samples $(\mathbf{w}_{\text{batch}}, \mathbf{X}_{\text{batch}})$ is *equivalent* to minimising the total predictive uncertainty (proof in [32]). Thus, solving KQ can obtain n batch samples which are not only diversified, but also of provably small total GP predictive variance (uncertainty sampling). This duality enables us to import the advanced methods from KQ community. Particularly, subsample-based kernel quadrature (SKQ) offers multiple advantages over existing batch BQ/BO methods. First, *scalability in batch size*, of which computational complexity is smaller than baselines (see Supp. B). Second, a *gradient-free solver*, which is ideal for non-Euclidean discrete spaces. Third, *no limitation in prior and kernel modelling*, unlike baselines [66]. SKQ reformulates the n point selection problem as extracting the subset of discrete probability measure $\pi_{\text{batch}} = (\mathbf{w}_{\text{batch}}, \mathbf{X}_{\text{batch}})$ from large samples approximating the prior (we refer to empirical measure) $\pi_{\text{emp}} = (\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}})$, where $\mathbf{X}_{\text{batch}} \in \mathbb{R}^{n\times d}$, $\mathbf{X}_{\text{rec}} \in \mathbb{R}^{N\times d}$, $\mathbf{X}_{\text{batch}} \subset \mathbf{X}_{\text{rec}}$, $N \gg n$. Empirical measure is constructed by sampling from π, $\mathbf{X}_{\text{rec}} \sim \pi$, and its expectation approximates the mean of π well, $\int f\mathrm{d}\pi \approx \mathbf{w}_{\text{rec}}^\top f(\mathbf{X}_{\text{rec}}) \approx \mathbf{w}_{\text{batch}}^\top f(\mathbf{X}_{\text{batch}})$ for f in the RKHS. This formulation can be interpreted as minimising maximum mean discrepancy (MMD) [21] between π_{emp} and π_{batch} [32]. The MMD distance is widely recognised for measuring the distance between distributions. As such, SKQ is a sparse discretisation problem of an uncertain region subject to approximate the original π with given n samples with weights.

Amongst SKQs, random convex hull quadrature (RCHQ) [26] achieved provably state-of-the-art convergence rate of integral approximation as well as computationally-tractable complexity. This rate and complexity come from two approaches: Nyström method [12] and kernel recombination [62]. The Nyström method approximates the kernel using randomised singular value decomposition (SVD) [25] of the Gram matrix. SVD eigendecomposes the Gram matrix $K(\mathbf{X}_{\text{rec}}, \mathbf{X}_{\text{rec}}) = \mathbf{U}\mathrm{diag}(\mathbf{\Lambda})\mathbf{U}^\top$, then the test functions, defined as $\varphi_j(x) := u_j^\top K(\mathbf{X}_{\text{nys}}, x)$, where $u_j \in \mathbf{U}$ is an eigenvector, $\lambda_j \in \mathrm{diag}(\mathbf{\Lambda})$ is an eigenvalue, and $\mathbf{X}_{\text{nys}} \in \mathbb{R}^{M\times d}$ drawn from π, can approximate the kernel $K(x, x') \approx \sum_{j=1}^{n-1} \lambda_j^{-1}\varphi_j(x)\varphi_j(x')$. RCHQ constructs sparse KQ rules via $n-1$ test functions, permitting incorporation of the spectral decay in the kernel for faster convergence. *Kernel recombination* is the algorithm to solve such SKQ problems with best known complexity. This algorithm is completely gradient-free. In summary, the advantages of BASQ derive from: (1) Scalability, (2) gradient-free solver for discrete and non-Euclidean spaces, and (3) no limitation in prior and kernel selection, as comes from reformulation as SKQ. Constructing π_{emp} is just sampling, so any kind of distribution can be used. The Nyström approximation only requires Gram matrix, which any kernel can compute.

3 Proposed Method: SOBER

3.1 Global Optimisation as Bayesian Quadrature: Duality in Probability Measure

We reframe batch BO as a batch BQ problem. Consider the following dual formulation [56]:

$$x^*_{\text{true}} = \arg\max_x f_{\text{true}}(x) \quad \overset{\text{dual}}{\iff} \quad \delta_{x^*_{\text{true}}} \in \arg\max_\pi \int f_{\text{true}}(x)\mathrm{d}\pi(x), \tag{1}$$

where δ_x is the delta distribution at x, π is a probability distributions over the x^*_{true}, and x^*_{true} is the location of the global maximum of f_{true}. Figure 3 illustrates the algorithm flow.

Why necessary? This formulation is defined as black-box integration. We can then solve the batch BO problem as batch BQ using BASQ to harvest the benefits: (1) scalable diversified batching, (2) applicability to discrete and non-Euclidean space, and (3) flexibility of GP models with arbitrary kernel and priors over input. These benefits are unachievable with existing methods as previously mentioned.

How is it different from BASQ? Eq. (1) updates π over each iteration, whereas BASQ keeps π unchanged. The more accurately the GP surrogate approximates f_{true} through batch-sequential

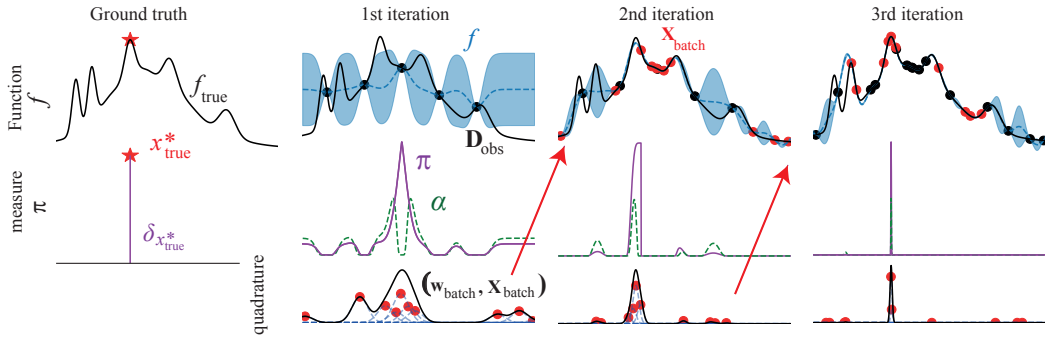

Figure 3: SOBER algorithm. Finding the location of global maximum x^*_{true} is equivalent to finding the delta distribution $\delta_{x^*_{\text{true}}}$. Based on the surrogate f, we approximate the probability of global maximum $P(x^*_{\text{true}} \mid \mathbf{D}_{\text{obs}})$ as π defined in §3.3. We can also set the user-defined acquisition function α to adjust batch samples (max-value entropy search in this case). Kernel quadrature gives a weighted point set $(\mathbf{w}_{\text{batch}}, \mathbf{X}_{\text{batch}})$ that makes a discrete probability measure approximating π in a way that is adaptive to the current GP [32]. Here, we have used a weighted kernel density estimation based on $(\mathbf{w}_{\text{batch}}, \mathbf{X}_{\text{batch}})$ to approximately visualise the kernel quadrature. Over iterations, π shrinks toward global maximum, which ideally becomes the delta function in a single global maximum case.

updates, the "narrower" π gets. Ideally, π eventually reaches the global maximum $\delta_{x^*_{\text{true}}}$ in a single global maximum case. As such, solving Eq. (1) by updating π is equivalent to finding the maximum x^*_{true}. While BASQ is a pure exploration algorithm, SOBER introduces exploitation by shrinking π.

3.2 Batch Selection as Kernel Quadrature

The batch selection in SOBER is performed by RCHQ. We propose the *objective*-RCHQ, solving:

$$\text{Find an } n\text{-point subset} \quad \mathbf{X}_{\text{batch}} \subset \mathbf{X}_{\text{rec}}, \tag{2a}$$

$$\text{s.t.} \quad \max \quad \mathbf{w}_{\text{batch}}^\top \alpha(\mathbf{X}_{\text{batch}}), \tag{2b}$$

$$\mathbf{w}_{\text{batch}}^\top \mathbf{1} = \mathbf{w}_{\text{rec}}^\top \mathbf{1} = 1, \quad \mathbf{w}_{\text{batch}} \geq \mathbf{0}, \quad \mathbf{w}_{\text{batch}}^\top \varphi_i(\mathbf{X}_{\text{batch}}) = \mathbf{w}_{\text{rec}}^\top \varphi_i(\mathbf{X}_{\text{rec}}), \quad 1 \leq i < n, \tag{2c}$$

where α is an arbitrary AF. As with the known constraints of $n-1$ KQ rules with Nyström approximation in Eq. (2c), we added the AF maximisation term Eq. (2b). There exist theoretical guarantees for specific choices of AFs depending on the spectral decay of an integral operator determined by the pair (K, π) [2, 27]. Empirically, the convergence as n increase is fast even without the AF term. We can utilise this degree of freedom for incorporating AF tailored for specific domain.

3.3 Sequentially Updating π

π is vital in defining the feasible region by exploiting the information from the surrogate model f. We wish to design π to shrink towards the global maximum, which ideally becomes the delta function in a single global maximum case. We discuss two variants; TS and likelihood-free inference (LFI).

TS-based. SOBER-TS adopts the maximum of the current GP surrogate x^*, $P(x^*|\mathbf{D})$ as π. This definition is well known as TS. We prepare N candidates with parallel decoupled TS [69], which alleviates the sampling overhead and fosters batch diversity via sparsifying the sampled functions from GP posterior (f-space sparsity). As such, SOBER-TS can be understood as re-selecting the TS samples that can maximise the sum of the AF and minimise the predictive variance. However, TS cannot provide the closed-form $P(x^*|\mathbf{D})$ distribution for faster sampling.

LFI-based. Considering π definition under uncertain global maximiser x^*, x^*_{true} might be any location with values possibly larger than the current maximum $\eta := \max \mathbb{E}[f(x^*)]$. Given this definition, we can define the "tentative" likelihood over input space $P(f(x) \geq \eta | x, \mathbf{D})$ [24], given by:

$$L(x|\theta, \sigma_n^2, \eta) := P\big(f(x|\mathbf{D}) \geq \eta\big) \propto \Phi\left[\frac{m(x|\theta) - \eta}{\sqrt{C(x,x|\theta)}}\right], \tag{3}$$

where the Φ is cumulative density function (CDF) of the standard normal distribution. This likelihood definition is the same as the probability of improvement (PI) AF [38]. Ideally, π can eventually reach the global maximum $\delta_{x^*_{\text{true}}}$ in a single global maximum case, where $\eta = f_{\text{true}}(x^*_{\text{true}}) = \mathbb{E}[f(x^*_{\text{true}})]$. Now we can estimate posterior belief of the maximum location π via the following Bayes' rule, $\pi(x) \propto L(x)\pi'(x)$. For each iteration, $L(\cdot|\theta, \sigma_n^2, \eta)$ is updated and π' is π of the previous iteration. In the beginning, π' is the prior belief. In the typical BO setting, we use the bounded uniform prior, but we can incorporate the experts' knowledge via a stronger prior. We do not examine the sensitivity to prior π' in this paper, but the following papers show strong empirical performance of a stronger prior [41, 34]. Closed-form expression of SOBER-LFI can offer faster sampling.

How to sample from π. SOBER handles discrete, continuous, or mixed inputs. The difference is the sampler for empirical measure π_{emp}. The simplest scenario is if all discrete candidates are available *a priori* and enumerable. As RCHQ accepts weighted samples $\pi_{\text{emp}} = (\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}})$ as importance sampling, sampling is just computing the weights $\mathbf{w}_{\text{rec}} = \pi(\mathbf{X}_{\text{rec}}) / \left[\pi(\mathbf{X}_{\text{rec}})^\top \mathbf{1}\right]$. If all combinations are innumerable or unavailable, we sample $\mathbf{X}_{\text{rec}}$ from the discrete prior π', which is user-defined. Once sampled, the procedure is the same: compute $\mathbf{w}_{\text{rec}}$, then pass the empirical measure π_{emp} to RCHQ. We update the hyperparameters of the prior π' via maximum likelihood estimation (MLE) from the weighted sample $(\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}})$. A continuous space can be regarded as an innumerable discrete space, so it can be handled similarly. The only difference is the prior update. We use weighted kernel density estimation (KDE) for the update, for speed and flexibility. Mixed space is the combination of discrete and continuous space, so the prior is the combination of both. Importantly, the prior does not need to precisely approximate π as the importance weights $\mathbf{w}_{\text{rec}}$ will correct the difference. See Supp. D for details.

3.4 Auxiliary Algorithms

The fundamental idea of SOBER is simple: reframe batch BO as a batch BQ problem. While objective-RCHQ plays a role of exploration, shrinking π contributes to exploitation. The balance between exploration and exploitation can be adjustable via AF in objective-RCHQ, π definitions, or kernel selection, similarly to standard BO. While the above functionalities suffice minimal components of SOBER, the following auxiliary ones help bringing out the full potential of SOBER on each case.

Noisy Functions. As seen in Figure 2, noisy functions need FBGP modelling [53] but is expensive. We propose **quadrature distillation** to mitigate this. As FBGP is the Monte Carlo (MC) integration for marginal predictive posterior over hyperposterior (blackbox function), we can reframe this as another BQ problem. This reformulation extracts a small subset of weighted hypersamples from random ones. Such a distilled subset can offer sample-efficient integral approximattion over the hyperposterior, which can accelerate various expensive integrations, such as fast FBGP, fast information-theoretic AF, and fully Bayesian LFI formulation (see Supp. C).

Non-Smooth Functions. RCHQ exploits spectral decay of kernel via Nyström method for faster convergence. However, non-smooth or sharp functions have a long-tailed decay in eigenvalues, worsening the convergence rate of quadrature algorithms. We propose **automatic KQ selection** algorithm to cope with this. SKQ is not only RCHQ — kernel thinning [14]is also subsample-based and independent on the spectral decay. Which performs better depends on the level of smoothness. Importantly, the quadrature performance is quantifiable as worst-case integration error, thus we can automate the selection of two SKQ algorithms by monitoring the error (see Supp. D).

Simulation-based Inference. As LFI is originally introduced to solve simulation-based inference, now our SOBER-LFI is also capable of solving simulation-based inference. Sampling from π efficiently squeezes the region to be explored toward only the vicinity of posterior mode. As such, BO reformulation can introduce exploitation to BQ via π update. When compared to the original BO-based LFI (BOLFI) [24], SOBER has two benefits; evidence estimation and exact posterior estimation. While BOLFI is designed to approximate only the posterior distribution using the approximated likelihood definition, SOBER can estimate both the posterior and model evidence in one go, using the exact likelihood definition based on BASQ. We propose **dual GPs**; one for sampling with BOLFI, one for BQ modelling with BASQ. (see Supp. E)

3.5 Summary of Contribution

We reformulated the batch BO task as the dual problem defined by Eq. (1). Now, estimating the global maximum is equivalent to updating π. We introduced two variants: TS and LFI. Both offer

Table 2: Experimental Setup. BO/BQ refers to which task is to be solved, syn./real refers to the problem whether synthetic or real-world ones. The dimensions represent the number of dimensions over input space categorised into continuous (cont.), categorical (cat.), and binary (bin.), total refers to the summation of each dimension. Batch is the batch size n, and prior is π'. For prior, Bernoul. and Categor. refer to Bernoulli and categorical distributions with equal weights. (see Suppl. F)

				dimensions d						prior π'	
experiments	BO /BQ	syn. /real	objective	cont.	cat.	bin.	total	batch n	kernel K	cont.	disc.
Ackley	BO	syn	$\log_{10}\min(f)$	3	-	20	23	200	RBF	$\mathcal{U}(-1,1)$	Bernoul.
Rosenbrock			$\log_{10}\min(f)$	1	6	-	7	100	RBF	$\mathcal{U}(-5,10)$	Categor.
Hartmann			$\log_{10}\max(f)$	6	-	-	6	100	RBF	$\mathcal{U}(0,1)$	-
Snekel			$\log_{10}\max(f)$	4	-	-	4	100	RBF	$\mathcal{U}(0,10)$	-
Pest		real	$\min(f)$	-	15	-	15	200	RBF	-	Categor.
MaxSat			$\min(f)$	-	-	28	28	200	RBF	-	Bernoul.
Ising			$\log_{10}\min(f)$	-	-	24	24	100	RBF	-	Bernoul.
SVM			$\min(f)$	3	-	20	23	200	RBF	$\mathcal{U}(0,1)$	Bernoul.
Malaria			$\log_{10}\min(f)$	-	-	2048	2048	100	Tanimoto	-	Categor.
Solvent			$-\max\log_{10} f$	-	-	2048	2048	200	Tanimoto	-	Categor.
2 RC	BQ	real	$\int f(x)\mathrm{d}\pi'(x)$	6	-	-	6	100	RBF	Gaussian	-
5 RC			$\int f(x)\mathrm{d}\pi'(x)$	12	-	-	12	100	RBF	Gaussian	-

approximation of π using the information of the current surrogate model f. In either case, once the empirical measure $\pi_{\text{emp}} = (\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}})$ is constructed by sampling from π, the 'objective-RCHQ' chooses batch samples that minimise the GP posterior variance and maximise the user-defined AF. Additionally, quadrature distillation provides fast FBGP for noisy functions, automatic KQ selection permits the best of both worlds over RCHQ and kernel thinning for non-smooth or sharp functions, and dual GPs offer simulation-based inference. Note that the auxiliary algorithms are additional augmentations of general GP-based BO against the shared challenges. These design choices are simple, whether or not the objective function is noisy, non-smooth, or sharp. If the additional overhead allows, we can use all in default, but SOBER works even without these (see Supp. D).

4 Link between Global Optimisation and Quadrature

4.1 Theoretical Background

Similar approaches to ours can be seen in existing batch BOs, such as TurBO [15] (shrinking trust regions), batch TS [36], and DPP-BO [37]. For instance, the motivation of DPP is to diversify the batch samples [45], but the underlying idea can be linked with quadrature. As explained in Sec. 2.1, quadrature can be seen as sparsely discretising an uncertain region, which again can be regarded as pure exploration with diversified batch samples for BO via Eq. (1). For a more detailed theoretical background, we can start with classical batch TS. First of all, [36] show that the batch TS (synTS in their paper) satisfies the reasonable order of Bayesian regret (BR) (their Theorem 2). Their proof is essentially given by estimating $\frac{1}{t}\sum_{i=1}^{t}\frac{1}{N}\sum_{j=1}^{N}\mathbb{E}[f_{\text{true}}(x^*_{\text{true}}) - f_{\text{true}}(x_{i,j})]$, where $(x_{i,j}) \in \mathbf{X}_{\text{batch},i}$ is the batch sample at the i-th iteration. We can view the value $f_{\text{true}}(x^*_{\text{true}}) - \frac{1}{N}\sum_{j=1}^{N} f_{\text{true}}(x_{i,j})$ as the difference between $f_{\text{true}}(x^*_{\text{true}})$ and the MC estimate of the integral of f_{true} over π_i. In SOBER, we approximate this integral by an n-point weighted subset of $\mathbf{X}_{\text{batch},i}$ at each iteration using kernel recombination — on which we can prove the integral error (see Theorem 1 in [2])[1]. Namely, we can quantify how close our integral estimate $\mathbf{w}_{\text{batch}}^\top f(\mathbf{X}_{\text{batch}})$ is to the true integral $\int f_{\text{true}}(x)\mathrm{d}\pi(x)$ or the empirical mean $\mathbf{w}_{\text{rec}}^\top f(\mathbf{X}_{\text{rec}})$. Similar discussion can be done with DPP-BO [37, 45] for general AFs.

Table 3: Experimental BO results on convergence of objectives atfter 15 iterations. Objective values are average $\pm$ 1 standard error over 10 repeated runs. See Table 2 for the objectives of each task. Mean rank is the average rank over 10 datasets for each method.

baselines	Ackley	Rosen-brock	Hart-mann	Shekel	Pest	MaxSat	Ising	SVM	Mal-aria	Sol-vent	Mean rank
Random	0.232 ± 0.01	3.509 ± 0.02	0.448 ± 0.01	-1.280 ± 0.06	8.514 ± 0.06	-25.50 ± 0.30	-2.008 ± 0.12	0.363 ± 0.01	-2.593 ± 0.03	-0.972 ± 0.01	7.1
TS	0.093 ± 0.01	1.382 ± 0.31	0.505 ± 0.01	-0.195 ± 0.01	9.397 ± 0.12	-30.37 ± 0.54	0.027 ± 0.07	0.349 ± 0.01	-2.703 ± 0.03	-0.990 ± 0.01	6.0
decoupled TS	0.054 ± 0.01	2.490 ± 0.14	0.502 ± 0.01	0.072 ± 0.05	7.911 ± 0.01	-32.53 ± 0.25	-0.157 ± 0.01	0.348 ± 0.01	incompatible RFF kernel		5.0
DPP-TS	0.071 ± 0.01	0.976 ± 0.26	0.495 ± 0.01	-0.821 ± 0.08	7.963 ± 0.05	-31.18 ± 0.45	-0.320 ± 0.01	0.357 ± 0.01	prohibitively slow (>7 days)		6.0
TurBO	0.200 ± 0.01	**-0.017 ± 0.01**	0.499 ± 0.01	0.458 ± 0.01	9.309 ± 0.03	-18.25 ± 0.47	0.169 ± 0.02	0.362 ± 0.01	no lengthscale in kernel		7.0
GIBBON	0.007 ± 0.02	2.800 ± 0.20	0.368 ± 0.02	0.375 ± 0.03	8.890 ± 0.05	-33.58 ± 1.08	-0.228 ± 0.01	0.365 ± 0.01	combinatorial explosion		6.4
Halluci-nation	0.142 ± 0.01	0.704 ± 0.28	0.504 ± 0.01	-0.013 ± 0.02	8.176 ± 0.04	-31.05 ± 0.46	-2.345 ± 0.03	0.327 ± 0.01	prohibitively slow (>7 days)		4.6
LP	-0.063 ± 0.01	3.727 ± 0.19	0.511 ± 0.01	-0.189 ± 0.08	9.111 ± 0.11	-20.98 ± 1.39	-1.527 ± 0.28	0.343 ± 0.01	Non-Euclidean space		5.9
SOBER -TS	-0.008 ± 0.02	1.934 ± 0.250	0.513 ± 0.01	0.242 ± 0.02	8.377 ± 0.01	-32.29 ± 0.27	-2.749 ± 0.01	0.349 ± 0.01	incompatible RFF kernel		3.9
SOBER -LFI	**-2.180 ± 0.01**	0.653 ± 0.17	**0.514 ± 0.01**	**0.713 ± 0.01**	**7.070 ± 0.03**	**-34.84 ± 0.12**	**-2.796 ± 0.01**	**0.320 ± 0.01**	**-2.765 ± 0.03**	**-1.064 ± 0.01**	**1.1**

4.2 Empirical Analysis

We empirically investigate the relationship between regret and π-shrinkage. We consider the following two metrics for π-shrinkage; mean variance (MV) $\mathbb{V}\text{ar}[\pi(x)]$ and mean distance (MD) $|x^*_{\text{true}} - \mathbb{E}[\pi(x)]|$. MV corresponds to the π-shrinkage, of which smaller value indicates shrinking. MD represents the Euclidean distance between the mean of π and the true global maximum x^*_{true}. We compared these two metrics against BR, BR $:= |y^*_{\text{true}} - \mathbf{w}^\top_{\text{batch}} f_{\text{true}}(\mathbf{X}_{\text{batch}})|$. BR is the batch estimation regret (referred as BR for Theorem 2 in [36]). Experimental results (see Supp. F.) show the strong linear correlations between all of these 3 metrics (> 0.95), clearly explaining the π-shrinkage is a good measure of BR as the dual objective in Eq. (1). In other words, π (MC estimate of x^*) shrinks toward true global maximum x^*_{true} with being smaller variance (more confident), and both linearly correlated to minimising BR, can be visually confirmed in Figure 3.

5 Experiments

We evaluate the sample efficiency and overhead of SOBER on **12 experiments against 11 baselines** (8 baselines for BO; random, TS, decoupled TS, DPP-TS, TurBO, GIBBON, hallucination, LP, and 3 baselines for BQ; batchWSABI, BASQ, and logBASQ). 12 experimental conditions are summarised in Table 2 (10 experiments for BO, 4 synthetic and 6 real-world datasets, and 2 real-world experiments for BQ). See Supp. F for full experimental details. Our code is built upon PyTorch-based libraries [49, 17, 5, 22]. All experiments were averaged over 10 repeats, computed in parallel with multicore CPUs[2] for a fair comparison, although GPUs can accelerate SOBER. As explained in Sec. 2, TurBO, GIBBON, Hallucination, and LP suffer from a combinatorial explosion in discrete space. To enable the comparison, we adopt thresholding[3] only for their discrete or mixed experiments. Most algorithms cannot be applied for various reasons to drug discovery tasks (see Table 3).

Results. Tables 3 and 4 illustrate the convergence and wall-clock time of sampling overhead at 15th iteration, respectively. As mean rank shows, SOBER-LFI outperforms all 8 baselines on 9 out of 10 experiments as well as maintaining the smallest average overhead. Diversified but squeezed batching

[1]Although, more investigation is needed to obtain end-to-end convergence guarantees.

[2]Performed on MacBook Pro 2019, 2.4 GHz 8-Core Intel Core i9, 64 GB 2667 MHz DDR4

[3]We optimise discrete variables as continuous ones, then the solutions are classified via nearest neighbours.

Table 4: Cumulative wall-clock time for 15 iterations and averaged over 10 repeated runs (log10 second).

baselines	Ackley	Rosen-brock	Hart-mann	Shekel	Pest	MaxSat	Ising	SVM	Mal-aria	Sol-vent	Mean rank
Random	-1.92	-1.96	-1.26	-1.17	-1.92	-1.89	-1.64	0.82	1.40	1.49	-
TS	2.71	3.10	2.79	2.86	3.00	3.70	3.22	3.36	2.71	2.85	3.1
decoupled TS	**2.20**	**2.04**	**2.01**	**2.04**	3.17	3.22	3.65	3.90	-	-	2.6
DPP-TS	4.85	4.56	4.35	4.62	5.67	4.49	4.73	4.73	-	-	7.4
TurBO	3.42	3.06	2.12	3.07	**2.91**	2.97	3.45	3.58	-	-	3.3
GIBBON	4.92	4.18	3.71	3.52	3.72	4.71	4.25	4.41	-	-	6.8
Hallucination	4.52	4.09	4.42	3.68	4.68	4.75	4.14	5.05	-	-	7.4
LP	5.50	5.48	5.23	4.78	3.84	5.48	5.10	4.53	-	-	8.5
SOBER-TS	3.10	3.43	3.16	3.17	3.30	4.01	3.20	3.21	-	-	4.1
SOBER-LFI	2.58	2.19	2.08	2.65	2.99	**2.96**	**2.28**	**2.31**	**2.43**	**2.35**	**1.5**

with SOBER-LFI is demonstrated to work well over various levels of multimodal and noisy functions over continuous, discrete, and mixed spaces. SOBER-LFI was only outperformed on the unimodal noiseless Rosenbrock function, which is in favour of hill-climbing algorithms like TS and TurBO. Still, SOBER-LFI takes second place, showing that updating π can efficiently squeeze the sampling region around the global maximum. In drug discovery tasks, SOBER-LFI showed fast convergence in both optimisation and computation, where most algorithms do not apply. Particularly, the solvent dataset clearly exemplifies the stuck behaviours (see Supp. F for learning curves). TS converges fast in the early stage, but it can not get out of the local maxima, resulting in a final regret being identical to random search, whereas SOBER-LFI does not suffer.

Figure 4 illustrates that SOBER also outperformed the BQ baselines. While the original BASQ over-explores the prior distribution and shows plateaus, logBASQ alleviates this behaviour via log-warp modelling. Nonetheless, SOBER showed significantly faster convergence than all competitors in both posterior and evidence inference of all tasks by squeezing π toward the posterior mode.

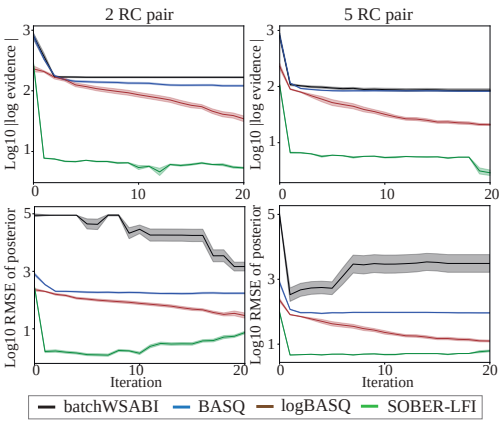

Figure 4: Batch BQ baseline comparisons across 2 real-world simulation-based inference tasks. *Top*: Log of log evidence. *Bottom*: Log RMSE of posterior as function of iterations. Lines and shaded area denote mean $\pm$ 1 std. error. SOBER-LFI consistently outperforms all three baselines.

We used type-II MLE GP throughout the experiments (no auxiliary algorithms from Sec. 3.4 except simulation-based inference). Supp. F further illustrates the effect of AF, batch size n, and hyperparameters (N, M). Amongst the AFs, information-theoretic AFs (MES and GIBBON) can boost the convergence rate than the LFI AF with a negligible overhead increase. The hyperparameters N, M are quite intuitive: the discretisation accuracy of the input x and function f spaces, respectively. Unsurprisingly, the larger these values become, the faster the convergence becomes but the larger the overhead is. Our default values $(N = 20000, M = 500)$ are competitive throughout the experiments. These values can be adjusted to the cost of queries [2] [4]. Moreover, the larger the batch size n becomes, the faster the convergence[5]. The ablation study shows that each component (temporary likelihood π, the iterative π update, and the objective-RCHQ) contributes to faster convergence. Additionally, FBGP with quadrature distillation can accelerate convergence, especially in noisy functions, while maintaining the overhead is competitive enough to the baselines.

[4] see guidelines in [2]

[5] We emphasise the convergence acceleration in the large batch is not typically achievable with other baseline methods, such as GIBBON (See Figure 6 in [43]).

References

[1] Masaki Adachi. High-dimensional discrete Bayesian optimization with self-supervised representation learning for data-efficient materials exploration. In *NeurIPS 2021 AI for Science Workshop*, 2021.

[2] Masaki Adachi, Satoshi Hayakawa, Martin Jørgensen, Harald Oberhauser, and Michael A Osborne. Fast Bayesian inference with batch Bayesian quadrature via kernel recombination. *Advances in Neural Information Processing Systems*, 35, 2022.

[3] Masaki Adachi, Yannick Kuhn, Birger Horstmann, Michael A Osborne, and David A Howey. Bayesian model selection of lithium-ion battery models via Bayesian quadrature. *arXiv preprint arXiv:2210.17299*, 2022.

[4] Javad Azimi, Alan Fern, and Xiaoli Fern. Batch Bayesian optimization via simulation matching. *Advances in Neural Information Processing Systems*, 23, 2010.

[5] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. BoTorch: a framework for efficient Monte-Carlo Bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

[6] Ricardo Baptista and Matthias Poloczek. Bayesian optimization of combinatorial structures. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 462–471. PMLR, 10–15 Jul 2018. URL: `https://proceedings.mlr.press/v80/baptista18a.html`.

[7] François-Xavier Briol, Chris J Oates, Mark Girolami, Michael A Osborne, and Dino Sejdinovic. Probabilistic integration. *Statistical Science*, 34(1):1–22, 2019.

[8] Arnaud Carpentier, Ila Nimgaonkar, Virginia Chu, Yuchen Xia, Zongyi Hu, and T Jake Liang. Hepatic differentiation of human pluripotent stem cells in miniaturized format suitable for high-throughput screen. *Stem Cell Research*, 16(3):640–650, 2016.

[9] Henry R Chai and Roman Garnett. Improving quadrature for constrained integrands. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2751–2759. PMLR, 2019.

[10] Samuel Daulton, Xingchen Wan, David Eriksson, Maximilian Balandat, Michael A Osborne, and Eytan Bakshy. Bayesian optimization over discrete and mixed spaces via probabilistic reparameterization. *Advances in neural information processing systems*, 35, 2022.

[11] Aryan Deshwal, Syrine Belakaria, and Janardhan Rao Doppa. Mercer features for efficient combinatorial bayesian optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):7210–7218, 2021. URL: `https://ojs.aaai.org/index.php/AAAI/article/view/16886`, `doi:10.1609/aaai.v35i8.16886`.

[12] Petros Drineas and Michael W. Mahoney. On the nystrom method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6(72):2153–2175, 2005. URL: `http://jmlr.org/papers/v6/drineas05a.html`.

[13] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL: `http://archive.ics.uci.edu/ml`.

[14] Raaz Dwivedi and Lester Mackey. Kernel thinning. *arXiv preprint arXiv:2105.05842*, 2021.

[15] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

[16] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. *Advances in neural information processing systems*, 28, 2015.

[17] Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. *Advances in neural information processing systems*, 31, 2018.

[18] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[19] Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch Bayesian optimization via local penalization. In *Artificial intelligence and statistics*, pages 648–657. PMLR, 2016.

[20] Franz Graf, Hans-Peter Kriegel, Matthias Schubert, Sebastian Pölsterl, and Alexander Cavallaro. 2d image registration in ct images using radial image descriptors. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2011: 14th International Conference, Toronto, Canada, September 18-22, 2011, Proceedings, Part II 14*, pages 607–614. Springer, 2011.

[21] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[22] Ryan-Rhys Griffiths, Leo Klarner, Henry Moss, Aditya Ravuri, Sang T Truong, Bojana Rankovic, Yuanqi Du, Arian Rokkum Jamasb, Julius Schwartz, Austin Tripp, et al. GAUCHE: A library for Gaussian processes in chemistry. In *ICML 2022 2nd AI for Science Workshop*, 2022.

[23] Tom Gunter, Michael A Osborne, Roman Garnett, Philipp Hennig, and Stephen J Roberts. Sampling for inference in probabilistic models with fast Bayesian quadrature. *Advances in neural information processing systems*, 27, 2014.

[24] Michael U Gutmann and Jukka Corander. Bayesian optimization for likelihood-free inference of simulator-based statistical models. *Journal of Machine Learning Research*, 2016.

[25] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[26] Satoshi Hayakawa, Harald Oberhauser, and Terry Lyons. Positively weighted kernel quadrature via subsampling. *Advances in Neural Information Processing Systems*, 35:6886–6900, 2022.

[27] Satoshi Hayakawa, Harald Oberhauser, and Terry Lyons. Sampling-based Nyström approximation and kernel quadrature. *arXiv preprint arXiv:2301.09517*, 2023.

[28] Philipp Hennig, Michael A Osborne, and Hans P Kersting. *Probabilistic Numerics: Computation as Machine Learning*. Cambridge University Press, 2022.

[29] Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(6), 2012.

[30] José Miguel Hernández-Lobato, James Requeima, Edward O Pyzer-Knapp, and Alán Aspuru-Guzik. Parallel and distributed Thompson sampling for large-scale accelerated exploration of chemical space. In *International conference on machine learning*, pages 1470–1479. PMLR, 2017.

[31] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

[32] Ferenc Huszár and David Duvenaud. Optimally-weighted herding is Bayesian quadrature. *arXiv preprint arXiv:1204.1664*, 2012.

[33] Carl Hvarfner, Frank Hutter, and Luigi Nardi. Joint entropy search for maximally-informed Bayesian optimization. *arXiv preprint arXiv:2206.04771*, 2022.

[34] Carl Hvarfner, Danny Stoll, Artur Souza, Marius Lindauer, Frank Hutter, and Luigi Nardi. πBO: Augmenting acquisition functions with user beliefs for Bayesian optimization. *arXiv preprint arXiv:2204.11051*, 2022.

[35] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

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

[37] Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. Batched Gaussian process bandit optimization via determinantal point processes. *Advances in Neural Information Processing Systems*, 29, 2016.

[38] Harold J Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of basic engineering*, 86:97 – 106, 1964.

[39] Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[40] Gustavo Malkomes, Charles Schaff, and Roman Garnett. Bayesian optimization for automated model selection. *Advances in Neural Information Processing Systems*, 29, 2016.

[41] Matthias Mayr, Carl Hvarfner, Konstantinos Chatzilygeroudis, Luigi Nardi, and Volker Krueger. Learning skill-based industrial robot tasks with user priors. In *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, pages 1485–1492. IEEE, 2022.

[42] Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[43] Henry B Moss, David S Leslie, Javier Gonzalez, and Paul Rayson. GIBBON: General-purpose information-based Bayesian optimisation. *arXiv preprint arXiv:2102.03324*, 2021.

[44] Iain Murray, Ryan Adams, and David MacKay. Elliptical slice sampling. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 541–548. JMLR Workshop and Conference Proceedings, 2010.

[45] Elvis Nava, Mojmir Mutny, and Andreas Krause. Diversified sampling for batched Bayesian optimization with determinantal point processes. In *International Conference on Artificial Intelligence and Statistics*, pages 7031–7054. PMLR, 2022.

[46] Changyong Oh, Jakub Tomczak, Efstratios Gavves, and Max Welling. Combinatorial Bayesian optimization using the graph Cartesian product. *Advances in Neural Information Processing Systems*, 32, 2019.

[47] Michael Osborne, Roman Garnett, Zoubin Ghahramani, David K Duvenaud, Stephen J Roberts, and Carl Rasmussen. Active learning of model evidence using Bayesian quadrature. *Advances in neural information processing systems*, 25, 2012.

[48] Michael Osborne, Roman Garnett, Stephen Roberts, Christopher Hart, Suzanne Aigrain, and Neale Gibson. Bayesian quadrature for ratios. In *Artificial Intelligence and Statistics*, pages 832–840. PMLR, 2012.

[49] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[50] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[51] Liva Ralaivola, Sanjay J Swamidass, Hiroto Saigo, and Pierre Baldi. Graph kernels for chemical informatics. *Neural networks*, 18(8):1093–1110, 2005.

[52] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

[53] Christoffer Riis, Francisco N Antunes, Frederik Boe Hüttel, Carlos Lima Azevedo, and Francisco Camara Pereira. Bayesian active learning with fully Bayesian Gaussian processes. *arXiv preprint arXiv:2205.10186*, 2022.

[54] Binxin Ru, Ahsan Alvi, Vu Nguyen, Michael A Osborne, and Stephen Roberts. Bayesian optimisation over multiple continuous and categorical inputs. In *International Conference on Machine Learning*, pages 8276–8285. PMLR, 2020.

[55] Binxin Ru, Michael A Osborne, Mark McLeod, and Diego Granziol. Fast information-theoretic Bayesian optimisation. In *International Conference on Machine Learning*, pages 4384–4392. PMLR, 2018.

[56] Alessandro Rudi, Ulysse Marteau-Ferey, and Francis Bach. Finding global minima via kernel approximations. *arXiv preprint arXiv:2012.11978*, 2020.

[57] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, 1992.

[58] Edward Snelson, Zoubin Ghahramani, and Carl Rasmussen. Warped Gaussian processes. *Advances in neural information processing systems*, 16, 2003.

[59] Thomas Spangenberg, Jeremy N Burrows, Paul Kowalczyk, Simon McDonald, Timothy NC Wells, and Paul Willis. The open access malaria box: a drug discovery catalyst for neglected diseases. *PloS one*, 8(6):e62906, 2013.

[60] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[61] S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets. Retrieved May 18, 2023, from `http://www.sfu.ca/~ssurjano`.

[62] M. Tchernychova. *Carathéodory cubature measures*. PhD thesis, University of Oxford, 2016.

[63] George R Terrell and David W Scott. Variable kernel density estimation. *The Annals of Statistics*, pages 1236–1265, 1992.

[64] Alexander Thebelt, Calvin Tsay, Robert Lee, Nathan Sudermann-Merx, David Walz, Behrang Shafei, and Ruth Misener. Tree ensemble kernels for bayesian optimization with known constraints over mixed-feature spaces. *Advances in Neural Information Processing Systems*, 35:37401–37415, 2022.

[65] Julien Villemonteix, Emmanuel Vazquez, and Eric Walter. An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, 44(4):509–534, 2009.

[66] E. Wagstaff, S. Hamid, and M. A. Osborne. Batch selection for parallelisation of Bayesian quadrature. arXiv preprint, 2018.

[67] Xingchen Wan, Vu Nguyen, Huong Ha, Binxin Ru, Cong Lu, and Michael A Osborne. Think global and act local: Bayesian optimisation over high-dimensional categorical and mixed search spaces. *arXiv preprint arXiv:2102.07188*, 2021.

[68] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient Bayesian optimization. In *International Conference on Machine Learning*, pages 3627–3635. PMLR, 2017.

[69] James Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Deisenroth. Efficiently sampling functions from Gaussian process posteriors. In *International Conference on Machine Learning*, pages 10292–10302. PMLR, 2020.

[70] Jian Wu, Saul Toscano-Palmerin, Peter I Frazier, and Andrew Gordon Wilson. Practical multi-fidelity Bayesian optimization for hyperparameter tuning. In *Uncertainty in Artificial Intelligence*, pages 788–798. PMLR, 2020.

A Background

A.1 Bayesian Optimisation

BO is an algorithm for solving the black-box global optimisation problems defined by:

$$x^*_{\text{true}} = \arg\max_x f_{\text{true}}(x), \tag{4}$$

where x^*_{true} is the true global maximum of the true black-box function f_{true}. The underlying assumptions here are:

1. f_{true} is a black-box function, and we do not know the function information except for querying the values at certain locations (oracles).
2. We can query multiple locations at the same time without additional overhead, and a larger batch size is desirable for faster convergence in wall-clock time.

With the given assumption, the desiderata are:

1. The queries are expensive, and we wish to minimise the number of queries for fast convergence.
2. The total overhead of the batch acquisition algorithm should be smaller for fast computation.

For sample-efficient global optimisation, BO typically utilities a surrogate GP model f of f_{true}. We set zero mean GP as a prior over function space, and set Gaussian likelihood. As both are Gaussian and conjugate, the predictive posterior with noisy observations has the closed-form as Gaussian, denoted by:

$$P(f \mid \mathbf{D}_{\text{obs}}, x) = \mathcal{GP}\Big(f; m(x), C(x, x')\Big), \tag{5a}$$

$$m(x) = K(x, \mathbf{X}_{\text{obs}})\Big[K(\mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{obs}}) + \sigma^2\mathbf{I}\Big]^{-1}\mathbf{y}_{\text{obs}}, \tag{5b}$$

$$C(x, x') = K(x, x') - K(x, \mathbf{X}_{\text{obs}})\Big[K(\mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{obs}}) + \sigma^2\mathbf{I}\Big]^{-1}K(\mathbf{X}_{\text{obs}}, x'), \tag{5c}$$

where $\mathbf{D}_{\text{obs}} = (\mathbf{X}_{\text{obs}}, \mathbf{y}_{\text{obs}})$ is the observed dataset, $\mathbf{y}_{\text{obs}} := f_{\text{true}}(\mathbf{X}_{\text{obs}})$ are the oracles queried in parallel, $m(\cdot)$ and $C(\cdot, \cdot)$ are the mean and covariance of the predictive posterior, $K(\cdot, \cdot \mid \theta)$ is the kernel, θ is the kernel hyperparameters, $\mathbf{I}$ is the identity matrix, σ_n^2 is the variance of Gaussian noise, and x and x' are the locations where we wish to predict, $x, x' \notin \mathbf{X}_{\text{obs}}$.

m and C guide our beliefs toward the region where the true global maximum x^*_{true} possibly locates. Such a guiding mechanism is obtained through maximising an AF, which selects the next query via its maximisation. There are several types of AFs, such as expected improvement (EI) [35], upper confidence bound (UCB) [60], information-theoretic AFs [65, 29, 68, 33], FBGP-based AFs [31, 57, 53]. More sample-efficient AF tends to be more computationally expensive.

Queried observations $\mathbf{D}_{\text{obs}}$ serially update the GP surrogate model f so it can predict the output of f_{true} more accurately. When updating GP with given $\mathbf{D}_{\text{obs}}$, GP hyperparameters θ are also updated. There are two ways of updating hyperparameters; type-II MLE and FBGP. While type-II MLE is the point estimation of optimal hyperparamter in terms of the marginal likelihood of the GP, FBGP estimates the hyperposterior, typically performed by Markov chain Monte Carlo (MCMC), then represent the predictive posterior as the ensemble of GP with the hyperparameters randomly sampled from hyperposterior.

A.2 Bayesian Quadrature

BQ is an algorithm for evaluating integrals given by:

$$Z_{\text{true}} = \int f_{\text{true}}(x)\mathrm{d}\pi(x), \tag{6}$$

where f_{true} is the black-box function we wish to integrate against a known probability measure π. The difference from BO is the objective being integration, not global optimisation. The integration problem is widely recognised in statistical learning: expectations, variances, marginalisation, ensembles, Bayesian model selection, and Bayesian model averaging. BQ is, like BO, solved by GP-surrogate-model-based active learning. The batch acquisition methods are also shared with batch BO. The methodological differences are:

1. BQ typically assumes a specific kernel to make the integration analytical (e.g. RBF kernel).
2. While BO requires to approximate the black-box function only in the vicinity of the global optimum, BQ needs to approximate the whole region of interest defined by the probability measure π.

Thus, BQ is a purely explorative algorithm, and the uncertainty sampling AF is often applied.

The classic method to estimate the integral exploits Gaussianity. Let π be multivariate normal distribution $\pi(x) = \mathcal{N}(x; \mu_\pi, \mathbf{\Sigma}_\pi)$, and the kernel K be RBF kernel, which can be represented as Gaussian $K(\mathbf{X}_{\text{ob}}, x) = v\sqrt{|2\pi\mathbf{W}|}\mathcal{N}(\mathbf{X}_{\text{ob}}; x, \mathbf{W})$, where v is kernel variance and $\mathbf{W}$ is the diagonal covariance matrix whose diagonal elements are the lengthscales of each dimension. As the product of two Gaussians is a Gaussian, the integrand becomes a Gaussian and its integral has the closed form, as such:

$$\int m(x)\pi(x)dx = v\left[\int \mathcal{N}(x; \mathbf{X}_{\text{obs}}, \mathbf{W})\mathcal{N}(x; \mu_\pi, \mathbf{\Sigma}_\pi)dx\right]^\top K(\mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{obs}})^{-1}\mathbf{y}_{\text{obs}}, \tag{7}$$

$$= v\left[\int \mathcal{N}(x; \mathbf{X}_{\text{obs}}, \mathbf{W})\mathcal{N}(x; \mu_\pi, \mathbf{\Sigma}_\pi)dx\right]^\top \boldsymbol{\omega}, \tag{8}$$

$$= v\mathcal{N}(\mathbf{X}_{\text{obs}}; \mu_\pi, \mathbf{W} + \mathbf{\Sigma}_\pi)^\top\boldsymbol{\omega}, \tag{9}$$

where $\boldsymbol{\omega} := K(\mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{obs}})^{-1}\mathbf{y}_{\text{obs}}$. As such, the integration of GP over the measure π is analytical. The more accurately the GP can approximate the true function f_{true}, the more accurately the above integration estimation approximates.

As such, classical BQs have additional limitations on prior and kernel selections. To make the integration closed-form, the prior needs to be uniform or Gaussian, and the kernel also needs to be limited selection (e.g. RBF kernel, see Table 1 in [7]).

B Details on Related Works

B.1 Compatibility

The compatibility of each batch BO method is summarised in Table 1.

Hallucination Hallucination [4] tackled batch BO by simulating sequential process by putting "fantasy" oracles estimated by GP, translating batch selection into sequential problem. Hallucination is successful in low batch size n, but not scalable. Even a single iteration of AF maximisation is not trivial due to non-convexity, but they repeat this over n times and produce prohibitive overhead. For discrete and mixed space, maximizing AF requires enumerating all possible candidates. However, the higher the dimension and larger the number of categorical classes, the more infeasibly large the combination becomes (combinatorial explosion).

Local Penalisation (LP) LP [19], simulates only AF shape change, without fantasy oracles, by penalising AF assuming Lipschitz continuity. This succeeds in speeding up the hallucination algorithm. However, the principled limitations are inherited (combinatorial explosion). Large batch sizes are also not applicable because maximising AF still produces large overhead. This is because maximising AF is typically computed by multi-start optimiser, but the number of random seeds need to increase dependent on the number of dimensions and multimodality of true function. This optimiser also does not guarantee to be globally maximised, which contradicts the assumption of AF (only optimal if it is globally maximised.). Furthermore, Lipschitz continuity assumption limits its applicable range to be only for continuous space.

Thompson Sampling (TS) TS [30] is a random sampling method of $P(x^* \mid \mathbf{D}_{\text{obs}})$ by maximising the function samples drawing from predictive posterior in Eq. (5a). Due to its random sampling nature, exactly maximising the function samples are not strict relative to hallucination or LP. Thus, in practice, TS is typically done with taking argmax of function samples amongst the candidates of random samples over input space. This two-step sampling nature (random samples over input space $\rightarrow$ subsamples with argmax of random function samples) allows us for domain-agnostic BO. However, this scheme itself is a type of AF, so other AF is not naïvely supported. Moreover, due

Table 5: Comparion with batch BO algorithms. The scalability to large batch size is defined by the computational complexity that is smaller or equivalent complexity than TS.

Batch BOs	large batch	mixed space	No combinatorial explosion	sampling complexity
Hallucination [4]	×	✓	×	$n(NC_{\text{acq}} + C_{\text{update}} + N \log N)$
Local penalisation [19]	✓	×	×	$n(NC_{\text{acq}} + N \log N)$
TS [30]	✓	✓	✓	$n(NC_{\text{func}} + N \log N)$
Decoupled TS [69]	✓	✓	✓	$n(NMC_{\varphi} + N \log N)$
DPP [37]	×	✓	×	$n^3 NC_{\text{reject}}$
DPP-TS [45]	✓	✓	✓	$I_{\text{MCMC}}(NC_{\text{func}} + n^3)$
MC-SAA [5]	×	✓	×	$P_n^N C_{\text{acq}} + P_n^N \log P_n^N$
GIBBON [43]	×	✓	×	$P_n^N C_{\text{acq}} + P_n^N \log P_n^N$
SOBER (Ours)	✓	✓	✓	$NMC_{\varphi} + NC_{\text{acq}} + n^3 \log(N/n)$

to the random sampling nature, the selected batch samples are not sparsified to efficiently explore uncertain regions.

Decoupled TS Decoupled TS [69] achieved faster but more diversified sampling than TS by separating out the prior from the data. Only this method and SOBER sparsifies the samples via f-space (function space), which allows for general-purpose sparsifications. However, this does not guarantee to be f-space sparsification leads to better sparcification in x-space. Figure 1 visualises that decoupled TS can provide more diversified samples than TS, however, it still suffered from being stuck in local minima. Also, AFs are not custamisable.

Determinantal Point Process (DPP) DPP-BO [37] proposed diversified sampling using DPP via maximising the determinant of Gram matrix of the selected batch samples. However, this maximisation problem in general is NP-hard and the best known sampling method is a rejection sampling, which produces significant overhead. Thus, large batch sizes are prohibitive. This method also cannot support discrete and mixed spaces due to combinatorial explosion.

DPP-TS DPP-TS tried to take the best of both worlds of DPP and TS. Thanks to the randomised sampling of TS, this method can apply to discrete and mixed spaces, avoid combinatorial explosion, and is faster than original DPP. Still, the computation is much slower than non-DPP-based alternatives.

MC-SAA MC-SAA [5] adopts qMC-based AF approximation, which achieves faster computations with a deterministic function sampler. Still, underlying strategy of AF maximisation is shared with classic methods (hallucination, LP), hence the limitations are also shared. Thus, AF maximisation requires enumerating all combinations of possible candidates, easily becomes combinatorial explosion. Moreover, this method is only applicable to those MC approximation can be applied, so information-theoretic AF cannot be applied.

GIBBON GIBBON [43] proposed diversified sampling methods using a specific AF (lower bound approximation of max-value entropy search [68]). Querying large batch sizes requires enumerating all possible permutations of *both batch samples and discrete variables*. This requirement introduces a combinatorial explosion. Moreover, even in continuous variables, GIBBON becomes less diversified for large batch size (See Figure 6 in [43]).

TurBO TurBO [15] introduced multiple local BOs bounded with trust regions, and allocates batching budgets based on TS. This succeeded in scalable batching via maintaining local BOs that are compact, via shrinking trust regions, based on heuristics with many hyperparameters. Selecting hyperpameters is non-trivial and TurBO cannot apply to discrete and non-Euclidean space, for which kernels do not have lengthscale hyperparameters for the trust region update heuristic (e.g. Tanimoto kernel for drug discovery [51]).

B.2 Sampling Complexity

Table 5 summarises the computational complexity of each baseline method for sampling. The sampling complexity shows the upper bounds of N discrete candidates. Notation remarks: C_{reject} is the cost of rejection sampling from DPP, C_{acq} is the cost of evaluating the AF at a point, C_{update} is the cost of updating the GP, including the Gram matrix inversion and hyperparameter optimisation, C_{func} is the cost of evaluating a sampled function at a point, I_{MCMC} is the number of MCMC iterations until convergence, MC_φ is the cost of evaluating M components of the kernels approximated by Nyström method or random Fourier feature, $P_n^N := n!/(N-n)! \gg nN$ is the total number of all permutations, N is the number of candidate points, n is the batch size. The computational costs are in order of $C_{\text{reject}} \gg C_{\text{update}} \gg C_{\text{func}} \geq C_{\text{acq}} > MC_\varphi$, and the number of samples are in order of $N > M > n$, $I_{\text{MCMC}} \gg n$. The empirical comparison can be seen in Table 4. Our method was the fastest.

C Quadrature Distillation

C.1 Fast Fully Bayesian Gaussian Process

C.1.1 Existing Method: MCMC

While BO works well with a single set of optimised hyperparameters (type-II MLE) on most functions, some noisy cases such as drug discovery shown in Figure 2 requires FBGP modelling. GP marginalisation is typically done with MC integration via sampling from hyperposterior Π. The hyperposterior Π is given by:

$$\Pi(\theta) := p(\theta \mid \mathbf{D}_{\text{obs}}), \tag{10}$$

$$= \frac{\int p(\mathbf{y}_{\text{obs}} \mid f) p(f \mid \theta, \mathbf{X}_{\text{obs}}) p(\theta) \mathrm{d}f}{\iint p(\mathbf{y}_{\text{obs}} \mid f) p(f \mid \theta, \mathbf{X}_{\text{obs}}) \mathrm{d}f \mathrm{d}p(\theta)}, \tag{11}$$

$$= \frac{L(\theta)\Pi'(\theta)}{\int L(\theta) \mathrm{d}\Pi'(\theta)}, \tag{12}$$

$$= \frac{L(\theta)\Pi'(\theta)}{\mathcal{L}_{\text{hyper}}}, \tag{13}$$

where

$$\Pi'(\theta) := p(\theta), \tag{14}$$

$$L(\theta) := \int p(\mathbf{y}_{\text{obs}} \mid f) p(f \mid \theta, \mathbf{X}_{\text{obs}}) \mathrm{d}f, \tag{15}$$

$$= \mathcal{N}\Big(\mathbf{y}_{\text{obs}}; m(\mathbf{X}_{\text{obs}} \mid \theta), C(\mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{obs}} \mid \theta)\Big), \tag{16}$$

$$\mathcal{L}_{\text{hyper}} = \int L(\theta) \mathrm{d}\Pi'(\theta), \tag{17}$$

$\Pi'(\theta)$ is the hyperprior, L is the marginal likelihood, $\mathcal{L}_{\text{hyper}}$ is the marginal hyperlikelihood. And the predictive posterior for the test inputs x can be approximated with MCMC with M samples as

$$p(y \mid \mathbf{D}_{\text{obs}}, x, x') := \iint p(y \mid f, x, x') p(f \mid \theta, \mathbf{D}_{\text{obs}}) \mathrm{d}f \mathrm{d}p(\theta \mid \mathbf{D}_{\text{obs}}), \tag{18}$$

$$= \int p(y \mid \theta, \mathbf{D}_{\text{obs}}, x, x') \mathrm{d}p(\theta \mid \mathbf{D}_{\text{obs}}), \tag{19}$$

$$= \int L(x \mid \theta) \mathrm{d}\Pi(\theta), \tag{20}$$

$$\approx \frac{1}{M} \sum_{m=1}^{M} L(x \mid \theta_m) \tag{21}$$

where

$$L(x \mid \theta) := \mathcal{N}\Big(y; m(x \mid \theta), C(x, x' \mid \theta)\Big) \tag{22}$$

$$\theta_m \sim \Pi \tag{23}$$

However, its convergence rate $\mathcal{O}(1/\sqrt{n})$ is poor, requiring thousands of ensemble GP models for marginalisation. This significantly slows down batch BO computation as BASQ requires such an ensemble kernel. Hence, we introduce quadrature distillation (QD) trick for faster marginalisation by using the smaller set of weighted hypersamples selected by the quadrature, distilling random hypersamples from hyperprior.

C.1.2 MCMC-based Quadrature Distillation

The easiest to implement is to apply existing MCMC scheme to sample from hyperposterior Π. Let $\boldsymbol{\theta}_{\text{cand}}$ be the candidates sampled from Π via MCMC, $\boldsymbol{\theta}_{\text{cand}} \sim \Pi$. Then, simply applying RCHQ with an arbitrary kernel yields the small subset of $(\mathbf{w}_{\text{QD}}, \boldsymbol{\theta}_{\text{QD}})$. We refer this weighted subset of hypersamples to QD samples, which are the approximation of Π. Intuitively, this can be understood as the weighted samples can more sample-efficiently approximate the distribution than random samples that approximate the distributions with frequencies. With the QD samples, we can approximate the marginal predictive posterior of FBGP, given by:

$$p(y \mid \mathbf{D}_{\text{obs}}, x, x') \approx \mathbf{w}_{\text{QD}}^{\top} L(x \mid \boldsymbol{\theta}_{\text{QD}}), \tag{24}$$

$$\begin{aligned} m_{\text{QD}}(x) &= \int m(x \mid \Theta) \mathrm{d}\Pi(\theta), \\ &\approx \mathbf{w}_{\text{QD}}^{\top} m(x \mid \boldsymbol{\theta}_{\text{QD}}), \end{aligned} \tag{25}$$

$$\begin{aligned} V_{\text{QD}}(x) &= \int C(x, x \mid \theta) \mathrm{d}\Pi(\theta), \\ &\approx \mathbf{w}_{\text{QD}}^{\top} \left[C(x, x \mid \boldsymbol{\theta}_{\text{QD}}) + m^2(x \mid \boldsymbol{\theta}_{\text{QD}}) \right] - m_{\text{QD}}^2(x). \end{aligned} \tag{26}$$

$$\begin{aligned} C_{\text{QD}}(x, x') &= \int C(x, x' \mid \theta) \mathrm{d}\Pi(\theta), \\ &\approx \sum_{i}^{H} w_{\text{i, QD}} \left[\left(m(x \mid \theta_{\text{i, QD}}) - m_{\text{QD}}(x) \right)^T \left(m(x' \mid \theta_{\text{i, QD}}) - m_{\text{QD}}(x') \right) \right], \end{aligned} \tag{27}$$

where $\mathbf{w}_{\text{QD}} \in \mathbb{R}^H$ is the QD weights, $\boldsymbol{\theta}_{\text{QD}} \in \mathbb{R}^{D \times H}$ is the QD samples, D is the number of types of hyperparameters (e.g. lengthscale, variance), and H is the number of QD samples, which is much smaller than the number of MCMC samples M, $H \ll M$. Thus, we can estimate the marginal predictive posterior of FBGP as the small set of weighted hypersamples via QD. As all the above marginalisation shares the same signed measure Π, so the quadrature with QD is good approximation.

C.1.3 BQ-based Quadrature Distillation

MCMC-based QD suffices for most cases, however, MCMC produces significant overhead, so we wish to avoid this sampling scheme for quick estimation. Thus we adopt BQ-based quadrature distrillation. First, we apply BQ by placing GP on hyperlikelihood $L(\theta)$,

$$\begin{aligned} \Pi(\theta) &= \frac{L(\theta)\Pi'(\theta)}{\int L(\theta) \mathrm{d}\Pi'(\theta)}, \\ &\approx \frac{\mathbb{E}\Big[p(\ell \mid \theta, \mathbf{L}_{\text{obs}}, \boldsymbol{\theta}_{\text{obs}})\Big]\Pi'(\theta)}{\int \mathbb{E}\Big[p(\ell \mid \theta, \mathbf{L}_{\text{obs}}, \boldsymbol{\theta}_{\text{obs}})\Big] \mathrm{d}\Pi'(\theta)}, \\ &\approx \frac{m_{\text{hyper}}(\theta)\Pi'(\theta)}{\int m_{\text{hyper}}(\theta) \mathrm{d}\Pi'(\theta)}, \\ &= \frac{m_{\text{hyper}}(\theta)\Pi'(\theta)}{\left[\int K_{\text{hyper}}(\theta, \boldsymbol{\theta}_{\text{obs}}) K_{\text{hyper}}(\boldsymbol{\theta}_{\text{obs}}, \boldsymbol{\theta}_{\text{obs}})^{-1} \mathrm{d}\Pi'(\theta)\right] \mathbf{L}_{\text{obs}}}, \\ &= \frac{m_{\text{hyper}}(\theta)\Pi'(\theta)}{\mathbf{w}_{\text{BQ}}'^{\top} \mathbf{L}_{\text{obs}}}, \end{aligned} \tag{28}$$

where

$$P(\ell \mid \theta, \mathbf{L}_{\text{obs}}, \boldsymbol{\theta}_{\text{obs}}) \sim \mathcal{GP}(\ell; m_{\text{hyper}}(\theta), C_{\text{hyper}}(\theta, \theta')), \tag{29a}$$

$$\mathbf{w}_{\text{BQ}}' := \int K_{\text{hyper}}(\theta, \boldsymbol{\theta}_{\text{obs}}) K_{\text{hyper}}(\boldsymbol{\theta}_{\text{obs}}, \boldsymbol{\theta}_{\text{obs}})^{-1} \mathrm{d}\Pi'(\theta), \tag{29b}$$

$$\mathbf{L}_{\text{obs}} := L(\boldsymbol{\theta}_{\text{obs}}). \tag{29c}$$

Here, we place hyper-GP on the marginal likelihood L defined by Eq. (16). Importantly, this measure is the hyperprior Π', not the hyperposterior Π like the MCMC-based cases. We draw hypersamples $\boldsymbol{\theta}_{\text{obs}} \in \mathbb{R}^{D\times M}$, from the hyperprior $\Pi'(\cdot) := P(\theta)$. Then, we evaluate the marginal likelihood $\mathbf{L}_{\text{obs}} = \mathcal{L}(\boldsymbol{\theta}_{\text{obs}})$ in parallel. We select multivariate normal distribution for hyperprior $\Pi'(\theta) := \mathcal{N}(\theta; \mu_{\text{hyper}}, \boldsymbol{\Sigma}_{\text{hyper}})$ based on [40], and Gaussian kernel for hyper-GP. Then, weights $\mathbf{w}'_{\text{BQ}}$ in Eq. (29b) become analytical:

$$\begin{aligned}
\mathbf{w}'_{\text{BQ}} &:= \int K_{\text{hyper}}(\theta, \boldsymbol{\theta}_{\text{obs}}) K_{\text{hyper}}(\boldsymbol{\theta}_{\text{obs}}, \boldsymbol{\theta}_{\text{obs}})^{-1} \mathrm{d}\Pi'(\theta), \\
&= v\sqrt{|2\pi\mathbf{W}|}\left[\int \mathcal{N}(\theta; \boldsymbol{\theta}_{\text{obs}}, \mathbf{W}) \mathcal{N}(\theta; \mu_{\text{hyper}}, \boldsymbol{\Sigma}_{\text{hyper}}) \mathrm{d}\theta\right] K_{\text{hyper}}(\boldsymbol{\theta}_{\text{obs}}, \boldsymbol{\theta}_{\text{obs}})^{-1}, \qquad (30) \\
&= v\sqrt{|2\pi\mathbf{W}|}\mathcal{N}(\boldsymbol{\theta}_{\text{obs}}; \mu_{\text{hyper}}, \mathbf{W} + \boldsymbol{\Sigma}_{\text{obs}}) K_{\text{hyper}}(\boldsymbol{\theta}_{\text{obs}}, \boldsymbol{\theta}_{\text{obs}})^{-1}, \qquad (31)
\end{aligned}$$

With the given weighted samples $(\mathbf{w}'_{\text{BQ}}, \boldsymbol{\theta}_{\text{obs}})$, we can approximate the marginalisation. For instance, the marginal predictive posterior can be approximated as follows:

$$\begin{aligned}
p(y \mid \mathbf{D}_{\text{obs}}, x, x') &\approx \int L(x \mid \theta) \frac{m_{\text{hyper}}(\theta)}{\mathbf{w}'^{\top}_{\text{BQ}} \mathbf{L}_{\text{obs}}} \mathrm{d}\Pi'(\theta), \\
&\approx \mathbf{w}^{\top}_{\text{BQ}} L(x \mid \boldsymbol{\theta}_{\text{obs}}),
\end{aligned} \qquad (32)$$

where $\mathbf{w}_{\text{BQ}} := \mathbf{w}'_{\text{BQ}} \odot \mathbf{L}_{\text{obs}} / (\mathbf{w}'^{\top}_{\text{BQ}} \mathbf{L}_{\text{obs}})$. Obviously, Eq. (32) is the same approximation with the MCMC-based QD approximation in Eq. (24). The difference is the number of hypersamples; the number of hyperprior samples M is larger than that of quadrature distillation. Therefore, we can further distill the BQ samples, $(\mathbf{w}_{\text{QD}}, \boldsymbol{\theta}_{\text{QD}}) \subset (\mathbf{w}_{\text{BQ}}, \boldsymbol{\theta}_{\text{obs}})$ via RCHQ. The kernel for RCHQ is the kernel of hyper-GP, K_{hyper}. The kernel hyperparameters are optimised by type-II MLE. These procedure does not require MCMC. Thus, this can offer faster computation.

C.2 Fast Fully Bayesian Acquisition Functions

Many AFs, including the LFI, are dependent on not only kernel hyperparameters θ, but also the current maximum η. To achieve fast marginalisation of AFs, we need to incorporate η as one of arguments in estimation.

C.2.1 Parabolic Transform for Max-value Estimation

Estimating the current maximum location conditioned on θ is computationally challenging. We inherit the parabolic transform of GP surrogate model from [55]:

$$\begin{aligned}
f(x \mid \theta) &= \eta - \frac{1}{2} g(x)^2, && (33a) \\
&:= \mathcal{GP}\big(f(x); m(x \mid \theta, \eta), C(x, x \mid \theta)\big), && (33b) \\
g(x \mid \theta) &:= \mathcal{GP}\big(g(\cdot); m_g(\cdot \mid \theta), C_g(\cdot, \cdot \mid \theta)\big), && (33c) \\
m(x \mid \theta, \eta) &:= \eta - \frac{1}{2}\left[m_g(x)^2 + C_g(x, x)\right], && (33d) \\
C(x, x' \mid \theta) &:= \frac{1}{2} C_g(x, x')^2 + m_g(x)^{\top} C_g(x, x') m_g(x'), && (33e)
\end{aligned}$$

where $f(\cdot)$ is the surrogate model that approximates $f_{\text{true}}(\cdot)$, $g(\cdot)$ is the square-root warped GP [58] of $f(\cdot)$. The predictive mean $m_g(\cdot)$ and covariance $C_g(\cdot, \cdot)$ of the warped GP $g(\cdot)$ are expressed with normal GPs in Eqs. (5b) - (5c). The predictive mean $m(\cdot)$ and covariance $C(\cdot, \cdot)$ are approximated via moment-matching [23]. $\mathbf{D}_g = (\mathbf{X}_{\text{obs}}, \mathbf{y}_{g,\text{obs}})$ is the observed data for the warped GP, and $\mathbf{y}_{g,\text{obs}} := \sqrt{2(\eta - \mathbf{y}_{\text{obs}})}$. Now, η becomes a GP hyperparameter via $\mathbf{y}_{g,\text{obs}}$. All we have to do for QD is just replacing the definition of GP to this warped GP.

C.2.2 Marginal Expected Improvement Acquisition Function

The marginal EI AF [35] can be calculated with parabolic-transformed FBGP formulation:

$$\alpha_{\text{EI}}(x) := \mathbf{w}_{\text{QD}}^{\top}\left[(m(x \mid \boldsymbol{\theta}_{\text{QD}}) - \boldsymbol{\eta}) \odot \Phi(Z \mid \boldsymbol{\theta}_{\text{QD}})\right] + \mathbf{w}_{\text{QD}}^{\top}\left[\sqrt{C(x, x \mid \boldsymbol{\theta}_{\text{QD}})} \odot \phi(Z \mid \boldsymbol{\theta}_{\text{QD}})\right] \quad (34)$$

$$Z := \frac{m(x \mid \boldsymbol{\theta}_{\text{QD}}) - \boldsymbol{\eta}}{\sqrt{C(x, x \mid \boldsymbol{\theta}_{\text{QD}})}} \quad (35)$$

where $\Phi(x), \phi(x)$ are CDF and probability density function (PDF) of the normal distribution, $\boldsymbol{\eta} \in \boldsymbol{\theta}_{\text{QD}}$ is the distilled max value η.

C.2.3 Marginal Upper Confidence Bound Acquisition Function

The marginal UCB AF [60] can be calculated using FBGP formulation without parabolic-transformation:

$$\alpha_{\text{UCB}}(x) := \mathbf{w}_{\text{QD}}^{\top} m(x \mid \boldsymbol{\theta}_{\text{QD}}) + \sqrt{\beta}\mathbf{w}_{\text{QD}}^{\top}\sqrt{C(x, x \mid \boldsymbol{\theta}_{\text{QD}})} \quad (36)$$

where β is the BO hyperparameter, usually 0.2 is selected.

C.2.4 Max-value Entropy Search Acquisition Function

The max-value entropy search (MES) AF [68] can be calculated via parabolic-transformed FBGP formulation via `FITBO` formulation [55]:

$$\alpha_{\text{FITBO}}(x) := H[p(y \mid \mathbf{D}_{\text{obs}}, x)] - \mathbb{E}_{p(\eta \mid \mathbf{D}_{\text{obs}})}\Big[H[p(y \mid \mathbf{D}_{\text{obs}}, x, \eta)]\Big], \quad (37)$$

$$p(y \mid \mathbf{D}_{\text{obs}}, x) = \int p(y \mid \mathbf{D}_{\text{obs}}, x, \eta)\mathrm{d}p(\eta \mid \mathbf{D}_{\text{obs}}), \quad (38)$$

$$H[p(y \mid \mathbf{D}_{\text{obs}}, x)] = \int \ln p(y \mid \mathbf{D}_{\text{obs}}, x)\mathrm{d}p(y \mid \mathbf{D}_{\text{obs}}, x), \quad (39)$$

$$\mathbb{E}_{p(\eta \mid \mathbf{D}_{\text{obs}})}\Big[H[p(y \mid \mathbf{D}_{\text{obs}}, x, \eta)]\Big] = \int H[p(y \mid \mathbf{D}_{\text{obs}}, x, \eta)]\mathrm{d}p(\eta \mid \mathbf{D}_{\text{obs}}). \quad (40)$$

FITBO AF can be discretised via MC integration:

$$\alpha_{\text{FITBO}}(x \mid \mathbf{D}_{\text{obs}}) := H\left[\frac{1}{M}\sum_{i}^{M} p(y \mid \mathbf{D}_{\text{obs}}, x, \theta_i, \eta_i)\right] - \frac{1}{2M}\sum_{i}^{M} \log[2\pi e(C(x, x \mid \mathbf{D}_{\text{obs}}, \theta_i, \eta_i) + \sigma_{n,i})]. \quad (41)$$

Quadrature distillation can approximate the above AF as:

$$\alpha_{\text{FITBO}}(x \mid \mathbf{D}_{\text{obs}}) \approx H\left[\mathbf{w}_{\text{QD}}^{\top} m(x \mid \boldsymbol{\theta}_{\text{QD}})\right] - \frac{1}{2}\mathbf{w}_{\text{QD}}^{\top} \log[2\pi e(C(x, x \mid \mathbf{D}_{\text{obs}}, \boldsymbol{\theta}_{\text{QD}}) + \boldsymbol{\sigma}_{n,\text{QD}}^2)]. \quad (42)$$

For faster computation, moment-matching approximation yields the first term as:

$$H\left[\frac{1}{M}\sum_{i}^{M} p(y \mid \mathbf{D}_{\text{obs}}, x, \theta_i, \eta_i)\right] \approx \frac{1}{2}\log[2\pi e(\mathbb{V}\text{ar}[y] + \sigma_{n,i}^2)], \quad (43)$$

$$\mathbb{V}\text{ar}[y] = \frac{1}{M}\sum_{i}^{M}\Big(C(x, x \mid \theta_i) + m^2(x \mid \theta_i)\Big) - \mathbb{E}[y\theta_i]^2, \quad (44)$$

$$\mathbb{E}[y] = \frac{1}{M}\sum_{i}^{M} m(x \mid \theta_i). \quad (45)$$

Hence,

$$\begin{aligned}\alpha_{\text{FITBO}}(x \mid \mathbf{D}_{\text{obs}}) \approx{}& \frac{1}{2}\log[2\pi e(\mathbb{V}\text{ar}[y] + \mathbf{w}_{\text{QD}}^{\top}\boldsymbol{\sigma}_{n,\text{QD}}^2)] \\ &- \frac{1}{2}\mathbf{w}_{\text{QD}}^{\top}\log[2\pi e(C(x, x \mid \mathbf{D}_{\text{obs}}, \boldsymbol{\theta}_{\text{QD}}) + \boldsymbol{\sigma}_{n,\text{QD}}^2)],\end{aligned} \quad (46)$$

$$\mathbb{V}\text{ar}[y] = \mathbf{w}_{\text{QD}}^{\top}\left(C(x, x \mid \boldsymbol{\theta}_{\text{QD}}) + m^2(x \mid \boldsymbol{\theta}_{\text{QD}})\right) - \left[\mathbf{w}_{\text{QD}}^{\top} m(x \mid \boldsymbol{\theta}_{\text{QD}})\right]^2. \quad (47)$$

Table 6: Quadrature distillation algorithm

Algorithm 2: Quadrature distillation

if MCMC-based?
  $\Pi_{\text{emp}} := (\mathbf{w}_{\text{hyper}}, \boldsymbol{\theta}_{\text{obs}}) = (\mathbf{1}/M, \boldsymbol{\theta}_{\text{obs}}), \quad \boldsymbol{\theta}_{\text{obs}} \sim \Pi(\cdot)$ # MCMC sampling from hyperposterior
elif BQ-based?
  $\boldsymbol{\theta}_{\text{obs}} \sim \Pi'(\cdot)$ # random sampling from hyperprior
  $\ell(\theta) \leftarrow \text{TrainGP}(\boldsymbol{\theta}_{\text{obs}}, L(\boldsymbol{\theta}_{\text{obs}}))$ # train hyper-GP
  $\Pi_{\text{emp}} := (\mathbf{w}_{\text{BQ}}, \boldsymbol{\theta}_{\text{BQ}}) \leftarrow \text{BayesQuad}(\ell(\cdot), \boldsymbol{\theta}_{\text{obs}}, L(\boldsymbol{\theta}_{\text{obs}}))$ # Bayesian quadrature
$(\mathbf{w}_{\text{QD}}, \boldsymbol{\theta}_{\text{QD}}) = \text{RCHQ}(\Pi_{\text{emp}}, K_{\text{hyper}})$ # quadrature distillation via RCHQ
return $\mathbf{w}_{\text{QD}}, \boldsymbol{\theta}_{\text{QD}}$

C.2.5 Bayesian Query-by-Committee Acquisition Function

The Bayesian query-by-committee (B-QBC) AF is defined by [53]:

$$\alpha_{\text{BQBC}}(x) := \mathbb{V}\text{ar}_{p(\theta|\mathbf{D}_{\text{obs}})}\Big[m(x \mid \theta)\Big], \tag{48}$$

$$= \mathbb{E}_{p(\theta|\mathbf{D}_{\text{obs}})}\Big[(m(x \mid \theta) - \hat{m}(x))^2\Big]. \tag{49}$$

The quadrature distillation approximates this without parabolic-transformation, as follows:

$$\alpha_{\text{BQBC}}(x) \approx \mathbf{w}_{\text{QD}}^\top \Big[(m(x \mid \boldsymbol{\theta}_{\text{QD}}) - \mathbf{w}_{\text{QD}}^\top m(x \mid \boldsymbol{\theta}_{\text{QD}}))^2\Big]. \tag{50}$$

A variant AF, query by mixture of Gaussian process (QB-MGP), can also be approximated by the quadrature distillation without parabolic-transformation:

$$\alpha_{\text{QB-MGP}}(x) := \mathbb{E}_{p(\theta|\mathbf{D}_{\text{obs}})}\Big[C(x, x \mid \theta)\Big] + \mathbb{E}_{p(\theta|\mathbf{D}_{\text{obs}})}\Big[(m(x \mid \theta) - \hat{m}(x))^2\Big]. \tag{51}$$

$$\approx \mathbf{w}_{\text{QD}}^\top C(x, x \mid \boldsymbol{\theta}_{\text{QD}}) + \mathbf{w}_{\text{QD}}^\top \Big[(m(x \mid \boldsymbol{\theta}_{\text{QD}}) - \mathbf{w}_{\text{QD}}^\top m(x \mid \boldsymbol{\theta}_{\text{QD}}))^2\Big]. \tag{52}$$

C.3 Quadrature Distillation Algorithm

The algorithm flow of the quadrature distillation is shown in Table 6. Each procedure will be explained step by step.

D Algorithm

The whole algorithm flow of SOBER is shown in Table 7. QuadDistil is short for quadrature distillation explained in Section C. Sampling procedure is deliniated in Section D.3. AutoKQ is short for automatic kernel quadrature selection, explained in Section D.2.

D.1 Sampling from π

SOBER is a sample-based gradient-free approach, and so can handle discrete, continuous or mixed inputs. The only difference is the sampler for $\mathbf{X}_{\text{rec}}$. The simplest scenario is if all discrete candidates are available *a priori* and enumerable. As RCHQ accepts weighted samples $\pi_{\text{emp}} = (\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}})$ for importance sampling, all we have to do is to calculate the weights $\mathbf{w}_{\text{rec}}$. This is simply the normalised posterior $\pi(\mathbf{X}_{\text{rec}})/\big[\pi(\mathbf{X}_{\text{rec}}) \cdot \mathbf{1}\big]$. If all combinations are innumerable or unavailable, we sample $\mathbf{X}_{\text{rec}}$ from the discrete prior π', which the user can define the arbitrarily. Once sampled, the procedure is the same: we compute $\mathbf{w}_{\text{rec}}$, then pass the empirical measure π_{emp} to RCHQ. We update the hyperparameters of the prior π' via MLE from the weighted sample $(\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}})$. Continuous space can be regarded as innumerable discrete space, so it can be handled similarly. The only difference is the prior update. We use weighted KDE for the update, for speed and flexibility. Mixed space is the

Table 7: SOBER algorithm.

Algorithm 1: SOBER

Input: prior π', hyperprior $\Pi'(\theta)$,
observed dataset $\mathbf{D}_{\text{obs}} = (\mathbf{X}_{\text{ob}}, \mathbf{y}_{\text{ob}})$
Output: maximum $\arg\max[\mathbf{y}_{\text{ob}}]$, evidence $\mathbb{E}[m(x)]$

$f \leftarrow$ InitialiseGP($\mathbf{D}_{\text{obs}}$)
while convergence:
  if FBGP:
    $\mathbf{w}_{\text{QD}}, \boldsymbol{\theta}_{\text{QD}} \leftarrow$ QuadDistil($f, \Pi'(\theta)$)
    $\pi, \alpha, K(\cdot,\cdot) \leftarrow$ FBGP($f, \pi', \mathbf{w}_{\text{QD}}, \boldsymbol{\theta}_{\text{QD}}$)
  else:
    $\pi, \alpha, K(\cdot,\cdot) \leftarrow$ Type-II MLE(f)
  $\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}}, \mathbf{X}_{\text{nys}} \sim$ Sampling(π, π')
  $\mathbf{X}_{\text{batch}}, \mathbf{w}_{\text{batch}} \leftarrow$ AutoKQ($\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}}, \mathbf{X}_{\text{nys}}, \alpha, K(\cdot,\cdot)$)
  $\mathbf{y}_{\text{batch}} =$ ParallelQuery($f_{\text{true}}(\mathbf{X}_{\text{batch}})$)
  $\mathbf{D}_{\text{obs}} \leftarrow \mathbf{D}_{\text{obs}} \cup \mathbf{D}_{\text{batch}}$
  $f \leftarrow$ UpdateGP($f, \mathbf{D}_{\text{obs}}$)
  $\pi' \leftarrow \pi$
  $\mathbb{E}[m(x)], \mathbb{V}\text{ar}[m(x)] \leftarrow$ KQ($f, \mathbf{X}_{\text{batch}}, \mathbf{w}_{\text{batch}}$)
return $\arg\max[\mathbf{y}_{\text{ob}}], \mathbb{E}[m(x)]$

combination of discrete and continuous space, which also can be regarded as innumerable discrete space. The prior update is the combination of the above two by assuming the discrete and continuous parameters are independent. Importantly, the prior does not need to precisely approximate π as the importance weights $\mathbf{w}_{\text{rec}}$ will correct the difference.

D.2 Automatic Kernel Quadrature Selection

Table 8: AutoKQ selection algorithm

Algorithm 4: AutoKQ selection

$\mathbf{X}_{\text{rchq}}, \mathbf{w}_{\text{rchq}}, \mathbb{V}\text{ar}[m(x)]_{\text{rchq}} \leftarrow$ RunRCHQ($\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}}, \boldsymbol{\varphi}(\cdot), \alpha(\cdot), \mathbf{X}_{\text{nys}}, f(\cdot)$)
$\mathbf{X}_{\text{kt}}, \mathbf{w}_{\text{kt}}, \mathbb{V}\text{ar}[m(x)]_{\text{kt}} \leftarrow$ RunKernelThinning($\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}}, \boldsymbol{\varphi}(\cdot), \alpha(\cdot), \mathbf{X}_{\text{nys}}, f(\cdot)$
if $\mathbb{V}\text{ar}[m(x)]_{\text{rchq}} < \mathbb{V}\text{ar}[m(x)]_{\text{kt}}$:
  return $\mathbf{X}_{\text{rchq}}, \mathbf{w}_{\text{rchq}}$
else:
  return $\mathbf{X}_{\text{kt}}, \mathbf{w}_{\text{kt}}$

function RunRCHQ($\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}}, \boldsymbol{\varphi}(\cdot), \alpha(\cdot), \mathbf{X}_{\text{nys}}, f(\cdot)$):
  $\boldsymbol{\varphi}(\cdot) \leftarrow$ Nyström($\mathbf{X}_{\text{nys}}, f(\cdot)$)
  $\mathbf{X}_{\text{rchq}}, \mathbf{w}_{\text{rchq}} \leftarrow$ RCHQ($\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}}, \boldsymbol{\varphi}(\cdot), \alpha(\cdot)$)
  $\mathbb{V}\text{ar}[m(x)]_{\text{rchq}} \leftarrow$ KQ($f(\cdot), \mathbf{X}_{\text{batch}}, \mathbf{w}_{\text{batch}}$)
  return $\mathbf{X}_{\text{rchq}}, \mathbf{w}_{\text{rchq}}, \mathbb{V}\text{ar}[m(x)]_{\text{rchq}}$

function RunKernelThinning($\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}}, \boldsymbol{\varphi}(\cdot), \alpha(\cdot), \mathbf{X}_{\text{nys}}, f(\cdot)$):
  $\mathbf{X}_{\text{kt}}, \mathbf{w}_{\text{kt}} \leftarrow$ KernelThinning($\mathbf{X}_{\text{rec}}, f(\cdot), \alpha(\cdot)$)
  $\mathbb{V}\text{ar}[m(x)]_{\text{kt}} \leftarrow$ KQ($f(\cdot), \mathbf{X}_{\text{batch}}, \mathbf{w}_{\text{batch}}$)
  return $\mathbf{X}_{\text{kt}}, \mathbf{w}_{\text{kt}}, \mathbb{V}\text{ar}[m(x)]_{\text{kt}}$

Table 8 illustrates the algorithm flow of automatic kernel quadrature selection algorithm. We compare the worst-case integration error of each algorithm, then pick the batch queries of which integration error is smaller. The choice between these two KQ methods can be made automatically by comparing

the worst-case error $\mathrm{wce}(Q_{\mathbf{w}_{\text{batch}},\mathbf{X}_{\text{batch}}})$:

$$\begin{aligned}
&\mathrm{wce}(Q_{\mathbf{w}_{\text{batch}},\mathbf{X}_{\text{batch}}}) \\
&:= \sup_{\|f\|_{\mathcal{H}}\leq 1} |\mathbf{w}_{\text{batch}} f(\mathbf{X}_{\text{batch}}) - \int f(x)\pi(x)\mathrm{d}x|, \qquad (53)\\
&\approx \mathbf{w}_{\text{batch}}^\top K(\mathbf{X}_{\text{batch}},\mathbf{X}_{\text{batch}})\mathbf{w}_{\text{batch}} - 2\mathbf{w}_{\text{batch}}^\top K(\mathbf{X}_{\text{batch}},\mathbf{X}_{\text{rec}})\mathbf{w}_{\text{rec}} + \mathbf{w}_{\text{rec}}^\top K(\mathbf{X}_{\text{rec}},\mathbf{X}_{\text{rec}})\mathbf{w}_{\text{rec}},
\end{aligned}$$

where $\mathcal{H}$ is the reproducing kernel Hilbert space. The third term in Eq. (53) is not dependent on the KQ methods, so we can avoid expensive $N \times N$ computations. RCHQ is selected in the early stage because the smooth kernel makes the eigenvalue decay short-tailed. In the late stage, the kernel thinning is chosen when the region is narrowed.

D.3 Sampling algorithm

The algorithm flow of the sampling is shown in the Table 9. The details will be explained step by step.

Table 9: Sampling algorithm

Algorithm 3: Subsampling

$\mathbf{X}_{\text{rec}} \sim \pi'(\cdot)$ # sampling from prior
$\mathbf{w}_{\text{rec}} = \frac{L(\mathbf{X}_{\text{rec}})}{\pi'(\mathbf{X}_{\text{rec}})} \cdot \frac{\pi'(\mathbf{X}_{\text{rec}})^\top \mathbf{1}}{L(\mathbf{X}_{\text{rec}})^\top \mathbf{1}}$ # compute the weights
if $\mathrm{len}(\mathbf{w}_{\text{rec}} > 0) < n$:
    $\pi'(\cdot) \leftarrow \pi'_{\text{initial}}(\cdot)$ # return to the initial prior when overexploitive
if continuous:
    $\pi(\cdot) = \mathrm{WKDE}(\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}})$ # weighted kernel density estimation
    $\mathbf{X}_{\text{rec}} \sim \pi(\cdot)$ # resample from WKDE
    $\mathbf{w}_{\text{rec}} = \frac{L(\mathbf{X}_{\text{rec}})}{\pi(\mathbf{X}_{\text{rec}})} \cdot \frac{\pi(\mathbf{X}_{\text{rec}})^\top \mathbf{1}}{L(\mathbf{X}_{\text{rec}})^\top \mathbf{1}}$ # recompute the weights
else if discrete and enumerable:
    $\mathbf{X}_{\text{rec}} = \pi'(\cdot)$ # all discrete candidates
    $\mathbf{w}_{\text{rec}} = \frac{L(\mathbf{X}_{\text{rec}})}{L(\mathbf{X}_{\text{rec}})^\top \mathbf{1}}$ # normalised weights
else if innumerable discrete:
    $\pi(\cdot) \leftarrow \mathrm{OptHypersMLE}(\pi'(\cdot), \mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}})$ # MLE hyperparameter optimisation
    $\mathbf{X}_{\text{rec}} \sim \pi(\cdot)$ # resample from WKDE
    $\mathbf{w}_{\text{rec}} = \frac{L(\mathbf{X}_{\text{rec}})}{\pi(\mathbf{X}_{\text{rec}})} \cdot \frac{\pi(\mathbf{X}_{\text{rec}})^\top \mathbf{1}}{L(\mathbf{X}_{\text{rec}})^\top \mathbf{1}}$ # recompute the weights
else mixed:
    $\pi(\cdot) \leftarrow \mathrm{CombineBothPrior}(\pi'(\cdot), \mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}})$ # Combine continuous and discrete prior
    $\mathbf{w}_{\text{rec}} = \frac{L(\mathbf{X}_{\text{rec}})}{\pi(\mathbf{X}_{\text{rec}})} \cdot \frac{\pi(\mathbf{X}_{\text{rec}})^\top \mathbf{1}}{L(\mathbf{X}_{\text{rec}})^\top \mathbf{1}}$ # recompute the weights
$\mathbf{X}_{\text{nys}} \sim \mathrm{Deweighted}(\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}})$ # deweighted random subset extraction
return $\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}}, \mathbf{X}_{\text{nys}}$

D.3.1 Weighted Kernel Density Estimation

The mean and covariance of the weighted kernel density estimation (WKDE) is estimated with the unbiased data covariance matrix given by:

$$\mu_{\text{wkde}} := \mathbf{w}_{\text{rec}}^\top \mathbf{X}_{\text{rec}}, \qquad (54)$$

$$\begin{aligned}
\mathbf{\Sigma}_{\text{wkde}} &:= \frac{\mathbf{w}_{\text{rec}}^\top \mathbf{1}}{(\mathbf{w}_{\text{rec}}^\top \mathbf{1})^2 - (\mathbf{w}_{\text{rec}}^2)^\top \mathbf{1}} \sum_i^N w_{\text{i, rec}} (X_{\text{i, rec}} - \mu_{\text{wkde}})^T (X_{\text{i, rec}} - \mu_{\text{wkde}}), \qquad (55)\\
&:= \frac{1}{1 - (\mathbf{w}_{\text{rec}}^2)^\top \mathbf{1}} \sum_i^N w_{\text{i, rec}} (X_{\text{i, rec}} - \mu_{\text{wkde}})^T (X_{\text{i, rec}} - \mu_{\text{wkde}}), \qquad (56)
\end{aligned}$$

where $X_{\text{i, rec}} \in \mathbf{X}_{\text{rec}}$ and $w_{\text{i, rec}} \in \mathbf{w}_{\text{rec}}$ is the i-th element of $\mathbf{X}_{\text{rec}}$ and $\mathbf{w}_{\text{rec}}$, respectively. The bandwidth of the kernel is estimated by the Scott's method [63].

D.3.2 Maximum Likelihood Estimation of Discrete Prior

The optimisation of hyperparamters of the discrete prior distributions was done via MLE from the weighted samples $(\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}})$. We denote the PDF of Bernoulli distribution (binary) and the categorical distribution as $\text{Bernoul}(x; \mathbf{w}_{\text{Ber}}), \text{Categor}(x; \mathbf{w}_{\text{Cat}})$, where $\mathbf{w}_{\text{Ber}} \in \mathbb{R}^d$ and $\mathbf{w}_{\text{Cat}} \in \mathbb{R}^{d \times C}$ are the weights hyperparaeters, C is the number of categories in the input parameters. The weighted log-PDF can be expressed as follows:

$$\text{LL} := \mathbf{w}_{\text{rec}}^\top \log \text{Bernoul}(\mathbf{X}_{\text{rec}}; \mathbf{w}_{\text{Ber}}) \tag{57}$$

$$\text{LL} := \mathbf{w}_{\text{rec}}^\top \log \text{Categor}(\mathbf{X}_{\text{rec}}; \mathbf{w}_{\text{Cat}}) \tag{58}$$

We optimise each weight hyperparameters via maximising the log-likelihood (LL) via L-BFGS-B [39]. PyTorch [49] auto-differentiation gives the gradient for L-BFGS-B. To make weights bounded $[0, 1]$, we transformed original LL space via the sigmoid function during optimisation. We start the optimisation of weights to be equal chance (discrete uniform distribution).

D.3.3 Deweighted sampling

Samples for the Nyström method are better to be spatially sparse to well represent the whole kernel shape. We adopt the deweighted sampling to constract the small subset of uniformly distributed samples $\mathbf{X}_{\text{nys}}$ from the weighted samples $(\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}})$. We resample from the categorical distribution with the inverse weights $(1/\mathbf{w}_{\text{rec}})$, then the resampled samples are uniformly distributed.

E Simulation-based inference

E.1 Simulation-based inference

The simulator emulates typically time-evolving signals from the physical device modelled by simultaneous differential equations. The solution of the differential equation is basically not analytical, requiring numerical approximation such as the finite element method. Each equation has parameters, such as coefficients of differential terms, which determine the signal shape. Estimating the parameters that can reproduce the observed signal is a typically tricky task because simulation is not differentiable with regard to each parameter. Although auto-differentiation can mitigate this problem, the parameter posterior is typically multimodal, so local optimisation algorithms based on differentiation struggles to find the global optimum. More importantly, this inverse problem often has no unique solution mathematically. Hence, rather than estimating one deterministic parameter set, inferring the parameter posterior is more practically important. Moreover, having dozens of plausible simulators with differing levels of assumption is a common situation where we need to select the parsimonious model that best describes the given dataset. Bayesian model evidence can provide a selection criterion. Therefore, estimating both Bayesian model evidence *and* parameter posterior is a frequent desideratum in practice. Furthermore, running simulators is expensive to evaluate, so parallelising the computation via computer clusters is of practical importance.

Let $\mathbf{y}_{\text{obs}}$ be the observed signal from the physical device, and we wish to estimate the simulator parameters Θ. This can be formulated as Bayesian inference, given by:

$$p(\Theta) := \pi'(\Theta) := \mathcal{N}(\Theta; \mu_\pi, \mathbf{\Sigma}_\pi) \tag{59}$$

$$p(\mathbf{D}_{\text{obs}}|\Theta, M) := \ell_{\text{true}}(\Theta) := \prod_j^m \mathcal{N}(\text{err}_j(\theta); \mathbf{0}, \sigma^2_{\text{noise}}), \tag{60}$$

$$p(\mathbf{D}_{\text{obs}}|M) := \mathcal{N}\left(\mathop{\mathbb{E}}_{x \in \pi}[\ell_{\text{true}}(\Theta)], \mathop{\mathbb{V}\text{ar}}_{x \in \pi}[\ell_{\text{true}}(\Theta)] \right), \tag{61}$$

$$p(\Theta|\mathbf{D}_{\text{obs}}, M) = \frac{p(\mathbf{D}_{\text{obs}}|\Theta, M)p(\Theta)}{p(\mathbf{D}_{\text{obs}}|M)} = \frac{\ell_{\text{true}}(\Theta)\pi(\Theta)}{\mathbb{E}_{x \in \pi}[\ell_{\text{true}}(\Theta)]}, \tag{62}$$

where

$$\mathbf{D}_{\text{obs}} := \{\mathbf{x}_{\text{obs}}, \mathbf{y}_{\text{obs}}\} \in \mathbb{R}^{m\times 1}, \tag{63}$$

$$\theta := \{\theta_i\} \in \mathbb{R}^{d-1}, \tag{64}$$

$$\Theta := \{\theta, \sigma^2_{\text{noise}}\} \in \mathbb{R}^{d}, \tag{65}$$

$$y_{\text{sim},j}(\theta) := M(\theta, \mathbf{x}_{\text{obs}}), \tag{66}$$

$$\text{err}_j(\theta) := \left[y_{\text{obs},j} - y_{\text{sim},j}(\theta)\right]^2. \tag{67}$$

$M(\theta, \mathbf{x}_{\text{obs}})$ is the simulation model, which returns the prediction $y_{\text{sim},j}(\theta)$ at given simulation parameter θ at the j-th time step. We wish to estimate the model evidence $\mathbb{E}_{x\in\pi}[\ell_{\text{true}}(\Theta)]$ and the parameter posterior $p(\Theta|\mathbf{D}_{\text{obs}}, M)$.

E.2 Bayesian Quadrature Formulation

In naive BQ, we place GP on the likelihood as such:

$$\ell(\Theta) \sim \mathcal{GP}\big(\ell(\Theta); \mu_\ell(\Theta), \sigma_\ell(\Theta, \Theta')\big). \tag{68}$$

The evidence can be estimated with BASQ formulation, given by:

$$\mathop{\mathbb{E}}_{x\in\pi}[m(x)] \approx \mathbf{w}^\top_{\text{batch}} m(\mathbf{X}_{\text{batch}}), \tag{69a}$$

$$\mathop{\mathbb{V}\text{ar}}_{x\in\pi}[m(x)] \approx \mathbf{w}^\top_{\text{batch}} C(\mathbf{X}_{\text{batch}}, \mathbf{X}_{\text{batch}})\mathbf{w}_{\text{batch}} - 2\mathbf{w}^\top_{\text{batch}} C(\mathbf{X}_{\text{batch}}, \mathbf{X}_{\text{rec}})\mathbf{w}_{\text{rec}} + \mathbf{w}^\top_{\text{rec}} C(\mathbf{X}_{\text{rec}}, \mathbf{X}_{\text{rec}})\mathbf{w}_{\text{rec}}. \tag{69b}$$

The posterior can be estimated with the surrogate model and the estimated evidence via Eq. (62).

However, the likelihood is typically transformed into the logarithmic space because its dynamic range is wider than the numerical over-/underflow limits. Thus, log-warped GP [47, 9, 3] is often applied. Particularly, we consider moment-matched log-transformed (MMLT) [9] GP, modelled as such:

$$f(x) = \exp[g(x)] - 1, \tag{70a}$$

$$:= \mathcal{GP}\big(f(x); m(x), C(x,x)\big), \tag{70b}$$

$$g(x) := \mathcal{GP}\big(g(\cdot); m_g(\cdot), C_g(\cdot,\cdot)\big), \tag{70c}$$

$$m(x) := \exp\left[m_g(x) + \frac{1}{2}C_g(x,x)\right], \tag{70d}$$

$$C(x,x') := m_g(x)m_g(x')\left[C_g(x,x) - 1\right]. \tag{70e}$$

The warped GP stores log-transformed values $\mathbf{y}_g = \log(\mathbf{y}+1)$, so we can avoid the over-/underflows. Adachi et al. [3] further extended MMLT GP so as to accommodate with BASQ modelling. They adopted the four-layered GP combining MMLT and parabolic transformation. The reason why they add the parabolic transformation is to copy the exponetiated function information to not only the surrogate function but also prior update. However, this deep warped structure causes additional predictive errors due to the cumulative approximation errors from each layer's moment-matching method.

E.3 Likelihood-free Inference Formulation

Alternately, BO-based LFI [24] models GP differently. They placed GP on the discrepancy, rather than the likelihood, defined as :

$$\Delta_{\text{true}}(\theta) := \log||\mathbf{y}_{\text{obs}} - \mathbf{y}_{\text{sim}}(\theta)|| \tag{71}$$

$$\Delta(\Theta) \sim \mathcal{GP}\big(\Delta(\Theta); \mu_\Delta(\theta), \sigma_\Delta(\theta, \theta')\big) \tag{72}$$

LFI adopts the tentative likelihood defined by Eq. 3 as the likelihood at each iteration. The true likelihood can be estimated *a posteriori*. The benefits of this modelling are as follows:

1. Avoiding extreme dynamic range of likelihood; $\Delta_{\text{true}}(\theta)$ has much more moderate range.

2. We can reformulate BQ as BO. BO is more suitable for solving simulation-based inference as only the vicinity of the MAP location has meaningful value. As almost everywhere has zero likelihood, so BQ formulation is over-exploring if the prior is misspecified.
3. We can obtain the "temporary" likelihood $L(\theta)$ that approaches the true likelihood $\ell_{\text{true}}(\Theta)$ asymptotically over iterations. This likelihood can be regarded as "updated prior". This can also mitigate the prior misspecification.

They reformulate the posterior inference as the BO to find the global minimum of the discrepancy $\Delta_{\text{true}}(\theta)$. The resulting GP surrogate model is used to approximate the posterior. They do not go beyond the posterior inference, so evidence estimation cannot be done with BOLFI.

E.4 SOBER-LFI Formulation

We wish to take the best of both world; LFI GP modelling suitable for sampling and exact evidence estimation via BQ modelling. Thus, we adopt the dual GPs; one for sampling, and the other for BQ modelling. While the sampling GP is modelled with the inverse discrepancy ($-\Delta_{\text{true}}(\theta)$ so as to be the maximisation objective), the BQ GP is modelled with log-likelihood with MMLT GP. Importantly, we can query both $\{\Delta_{\text{true}}(\theta), \ell_{\text{true}}(\Theta)\}$ with negligible overhead as the time-consuming part is $\mathbf{y}_{\text{sim}}(\theta)$. Once we get $\mathbf{y}_{\text{sim}}(\theta)$, calculating both $\{\Delta_{\text{true}}(\theta), \ell_{\text{true}}(\Theta)\}$ are very cheap.

The sampling GP is used for setting up the sampling function π, in the same manner explained in Section D.3. One difference is that the π becomes extremely sharper than the BO task. WKDE-based sampling can fail to sampling from π. Hence, we adopted elliptical slice sampling (ESS) [44]. Importance sampling permits using all of the samples from ESS without the burn-in period. The weights can be calculated via the π defined with the sampling GP. Note that ESS is more expensive than WKDE, so the additional overhead had to be produced instead. As such, the sampling GP constructs the empirical measure $\pi_{\text{emp}} = (\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}})$.

On the other hand, BQ GP constructs the surrogate model for likelihood. The posterior and evidence inference can be made in the same manner explained in Section E.2.

Batch acquisition via objective RCHQ becomes a mix of both GPs. The kernel is defined by the BQ GP in Eq. 70e. The objective with AF is defined by the sampling GP.

F Experiments

F.1 Batch Bayesian Optimisation

We examined our method, SOBER, with the following 10 datasets. All experiments are averaged over 16 iterations with varied random seeds. **random** is the random samples drawn from the prior distribution for each task. As these random samples are aware of categorical or mixed variables, it often performs better than baseline methods that cannot handle them (e.g. TurBO, LP). **TS** and **decoupled TS** was implemented with BoTorch library [5]. The candidates are sampled from the prior distribution, and each TS algorithm took the argmax of function samples over candidates. **DPP-TS** is provided by the author of the paper [45]. **TurBO** and **GIBBON** were implemented based on the official tutorials of BoTorch. Both **Hallucination** and **LP** were implemented using GPyTorch [17], and we select the standard EI AF. Both SOBER-TS and SOBER-LFI experiments were conducted without FBGP, and the hyperparameters (N, M) are fixed with $(20, 000, 500)$ throughout the experiments. The SOBER software is coded with GPyTorch, BoTorch, and BASQ code [2]. All codes are provided in the supplementary and will be open-sourced.

Synthetic: Ackley function Ackley funciton is defined as:

$$f(x) := -a \exp\left[-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right] - \exp\left[\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right] + a + \exp(1) \tag{73}$$

where $a = 20, c = 2\pi, d = 23$. We take the negative Ackley function as the objective of BO to make this optimisation problem maximisation. We modified the original Ackley function into a 23-dimensional function with the mixed spaces of 3 continuous and 20 binary inputs from $[0, 1]^{20}$, following [10]. The batch size n is 200. The continuous prior is the uniform distribution ranging

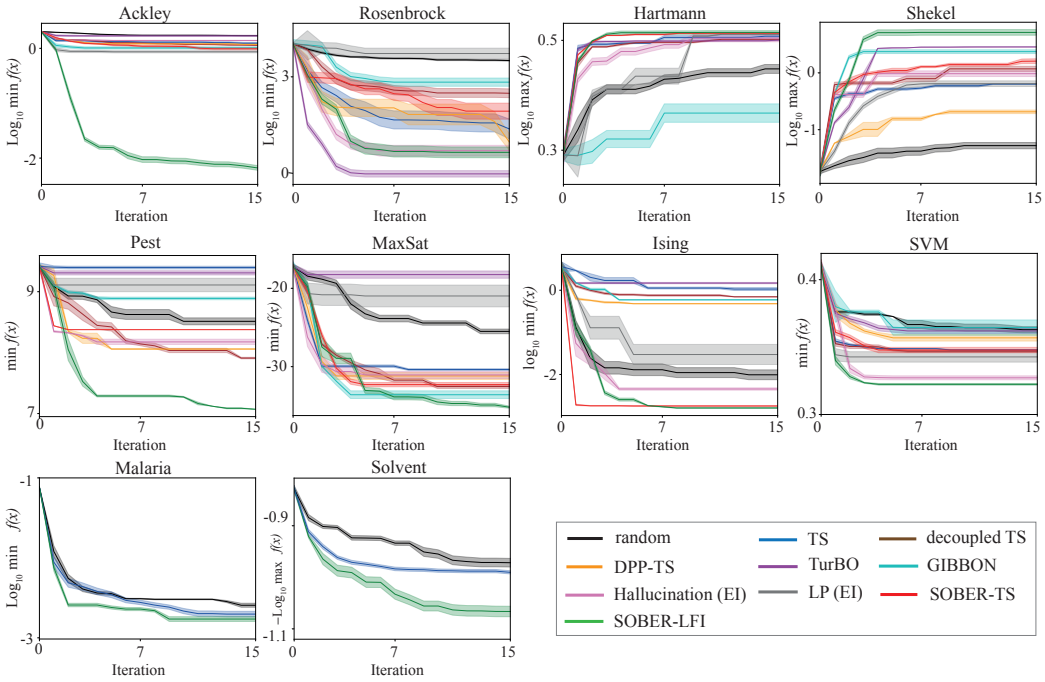

Figure 5: We evaluate SOBER across 4 synthetic functions and 6 real-world drug discovery datasets. Top: Log regret or log best observations, Bottom: Log overhead in seconds as function of iterations. Lines and shaded area denote mean $\pm$ 1 standard error. The batch size is 100 or 200 (see Table 2).

from [-1, 1]. The binary prior is the Bernoulli distribution with unbiased weights 0.5. We assume each of continuous and binary priors at each dimension are independent.

Synthetic: Rosenbrock function Rosenbrock function is defined as:

$$f(x) := \left[\sum_{i=1}^{d-1} \left\{100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right\}\right] \tag{74}$$

where $d = 7$. We take the negative Rosenbrock function as the objective of BO to make this optimisation problem maximisation. We modified the original Rosenbrock function into a 7-dimensional function with the mixed spaces of 1 continuous and 6 discrete variables, following [10]. The first 1 dimension is discretised to be categorical variables, with 4 possible values $x_1 \in \{-5, 0, 5, 10\}$. The other 6 dimensions are continuous with bounds $[-5, 10]^6$. The batch size n is 100. The continuous prior is the uniform distribution ranging from [-5, 10]. The discrete prior is the categorical distribution with unbiased weights 0.5. We assume each of continuous and discrete priors at each dimension are independent.

Synthetic: Hartmann function Hartmann 6-dimensional function [61] is defined as:

$$f(x) := -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{6} A_{ij}(x_j - P_{ij})^2\right), \tag{75}$$

$$\alpha = (1.0, 1.2, 3.0, 3.2)^\top, \tag{76}$$

$$\mathbf{A} = \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}, \tag{77}$$

$$\mathbf{P} = \begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{pmatrix} \tag{78}$$

We take the negative Hartmann function as the objective of BO to make this optimisation problem maximisation. All input variables are continuous with bounds $[0, 1]^6$. The batch size n is 100. The continuous prior is the uniform distribution ranging from [0, 1].

Synthetic: Shekel function Shekel function [61] is defined as:

$$f(x) := -\sum_{i=1}^{10} \left(\sum_{j=1}^{4} (x_j - C_{ji})^2 + \beta_i\right), \tag{79}$$

$$\beta = \frac{1}{10}(1, 2, 2, 4, 4, 6, 3, 7, 5, 5)^\top, \tag{80}$$

$$\mathbf{C} = \begin{pmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \end{pmatrix}, \tag{81}$$

We take the negative Shekel function as the objective of BO to make this optimisation problem maximisation. All input variables are continuous with bounds $[0, 10]^6$. The batch size n is 100. The continuous prior is the uniform distribution ranging from [0, 10].

Real-world: Pest Control Pest Control (Pest in the main) is proposed in [46], which is a multi-categorical optimisation problem (15 dimensions, 5 categories for each dimension). We wish to optimise the effectiveness of pesticide by choosing the 5 actions (selection of pesticides from 4 different firms, or not using any of it), but penalised by their prices. This choice is a sequential decision of 15 stages, and the objective function is expressed as the cumulative loss function with the total of both cost and the portion having pest. The batch size n is 200. We set the categorical prior with equal weights for each choice (discrete uniform distribution). Code is used in https://github.com/xingchenwan/Casmopolitan [67].

Real-world: Maximum Satisfiability Maximum satisfiability (MaxSat in the main) is proposed in [46], which is 28 dimensional binary optimisation problem. The objective is to find boolean values that maximise the combined weighted satisfied clauses for the dataset provided by Maximum Satisfiability competition 2018. The batch size n is 200. We set the Bernoulli distribution prior with equal weights (discrete uniform distribution). Both code and dataset are used in https://github.com/xingchenwan/Casmopolitan [67].

Real-world: Ising Model Sparsification Ising Model Sparsification (Ising in the main) is proposed in [46], which is 24 dimensional binary optimisation problem. The objective is to sparsify an Ising model using the regularised Kullback-Leibler divergence between a zero-field Ising model and the partition function, considering 4×4 grid of spins with regularisation coefficient $\lambda = 10^{-4}$. The batch size n is 100. We set the Bernoulli distribution prior with equal weights (discrete uniform distribution). Code is used in https://github.com/QUVA-Lab/COMBO [46].

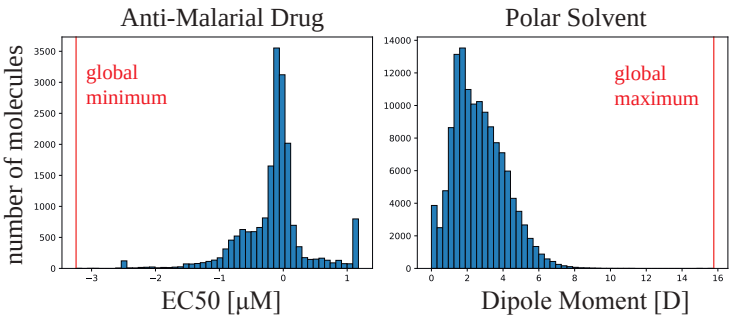

Figure 6: The histograms of the target values in the real-world drug discovery datasets

Real-world: Support Vector Machine Feature Selection Support vector machine feature selection (SVM in the main) is proposed in [10], which is 23 dimensional mixed-type input optimisation problem (20 dimensional binary and 3 dimensional continuous variables). The objective is jointly performing feature selection (20 features) and hyperparameter optimisation (3 hyperparameters) for a support vector machine (SVM) trained in the CTSlice UCI dataset [20, 13]. The batch size n is 100. We set the Bernoulli distribution prior with equal weights (discrete uniform distribution) for 20 binary inputs, and uniform prior with bounds $[0, 1]^6$. Code is used in https://github.com/facebookresearch/bo_pr [10].

Real-world: Anti-Malarial drug discovery The dataset with 20,746 small molecules represented as 2048-dimensional binary features were taken from the P. falciparum whole cell screening derived by the Novatis-GNF Malaria Box [59]. The target variable is the EC50 value, which is defined as the concentration of the drug which gives half the maximal response. The lower the concentration, the more effective (better) the drug. We take the nagative EC50 to make this optimisation problem maximisation. The batch size n is 100. We set the categorical prior with equal weights for each molecule (discrete uniform distribution). The dataset is downloaded from https://www.mmv.org/mmv-open/malaria-box/malaria-box-supporting-information. The raw molecule inputs as SMILES are converted into 2048-dimensional binary features with Gauche https://github.com/leojklarner/gauche [22]. The Tanimoto kernel is also coded with Gauche. This drug discovery dataset is challenging for most baseline methods: For decoupled TS in BoTorch, which is based on random Fourier features [50], is not compatible with the Tanimoto kernel in GAUCHE [22]. SOBER-TS is also based on decoupled TS, so it is not applicable neither. Tanimoto kernel is more computationally demanding than RBF, so DPP-TS and hallucination could not be finished the computation within one week. This is practically too slow, so we judged these two are not applicable for active learning purposes. TurBO is also not applicable because the Tanimoto kernel has no lengthscale hyperparameter, so TurBO cannot update the trust region. GIBBON suffers from combinatorial explosion, as enumerating all combination of both batch and drug candidates becomes infeasible (${}^{20746}\mathrm{P}_{100} \approx 3.9 \times 10^{431}$. For LP, which explicitly assumes continuity over Euclidean space, so we cannot apply it to non-Euclidean and discrete space.

Real-world: Polar solvent for batteries The dataset with 133,055 small molecules represented as 2048-dimensional features were optimised and predicted by the quantum-chemical computations using density functional theory, known as QM9 dataset [52]. The target variable is the dipole moment, which is basically correlated with the solvation capability in electrolytes in lithium-ion batteries, increasing the ratio of electro-mobile Li-ions. The higher the dipole moment becomes, the larger (better) the ionic conductivity does. The batch size n is 200. We set the categorical prior with equal weights for each molecule (discrete uniform distribution). The dataset is downloaded from http://quantum-machine.org/datasets/ [52]. The coding was done with Gauche [22]. Most baseline methods cannot be applied with the same reasons explained above.

Figure 6 shows the distribution of target values in two real-world drug discovery datasets. The optimal molecules are outliers from the dataset distribution, so it clearly shows these tasks are needle-in-the-haystack situations.

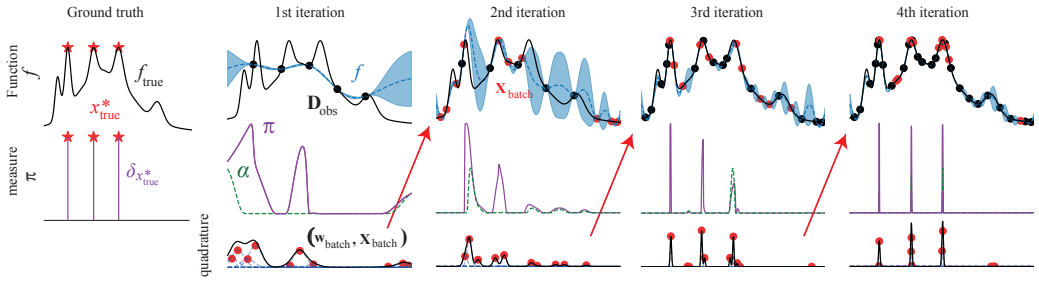

Figure 7: Visualising the multiple true global maxima case

F.2 Batch Bayesian Quadrature

We tested our algorithm, SOBER, with the simulation-based inference tasks as the batch BQ method. All experiments are averaged over 16 iterations with varied random seeds. The number of candidate samples drawn from the prior distribution is fixed to be 20,000 ($N = 20,000$) for a fair comparison. As the ground truth of posterior and evidence cannot be obtained for the simulation-based inference, we use the empirical metric to evaluate the quality of each inference. For posterior evaluation, we drew the 10,000 test samples from the normal distribution centered at the ground truth parameters and the covariance with the identity matrix of which each element is 5×10^{-6}. Then, we computed the root-mean-squared error (RMSE) between the estimated log-likelihood and true log-likelihood. For evidence, we simply adopted the negative of estimated evidence.

Real-world: 2 RC Pairs ECM 2 RC Pair equivalent circuit model (ECM) is the simplest lithium-ion battery simulator with 6-dimensional continuous variables [3]. We generated synthetic signal using the model with 100 frequency steps equispaced over log-angular frequency regime, then added the Gaussian noise with the amplitude of $\exp(1)$ to the $R_{\text{total}} = \exp(2)$ signal from the canonical ECM.

Real-world: 5 RC Pairs ECM 5 RC Pair ECM is more complex lithium-ion battery simulator with 12-dimensional continuous variables [3]. We generated synthetic signal using the model with 100 frequency steps equispaced over log-angular frequency regime, then added the Gaussian noise with the amplitude of $\exp(1)$ to the $R_{\text{total}} = \exp(2)$ signal from the canonical ECM.

F.3 Additional Experiments

F.3.1 Visualising Multiple Global Maxima Case

We visualised the case of multiple true global maxima. As shown in Figure 7, π converged to multiple delta distributions.

F.3.2 Empirical Convergence Analysis

We empirically investigated the relationship between regret and π shrinkage. Recall that empirical measure $\pi_{\text{emp}} = (\mathbf{w}_{\text{rec}}, \mathbf{X}_{\text{rec}})$ is sampled from π, and batch measure $\pi_{\text{batch}} = (\mathbf{w}_{\text{batch}}, \mathbf{X}_{\text{batch}})$ is the subset further extracted from π_{emp}. As such, all measures approximate the same distribution π, only the level of discretisation differs.

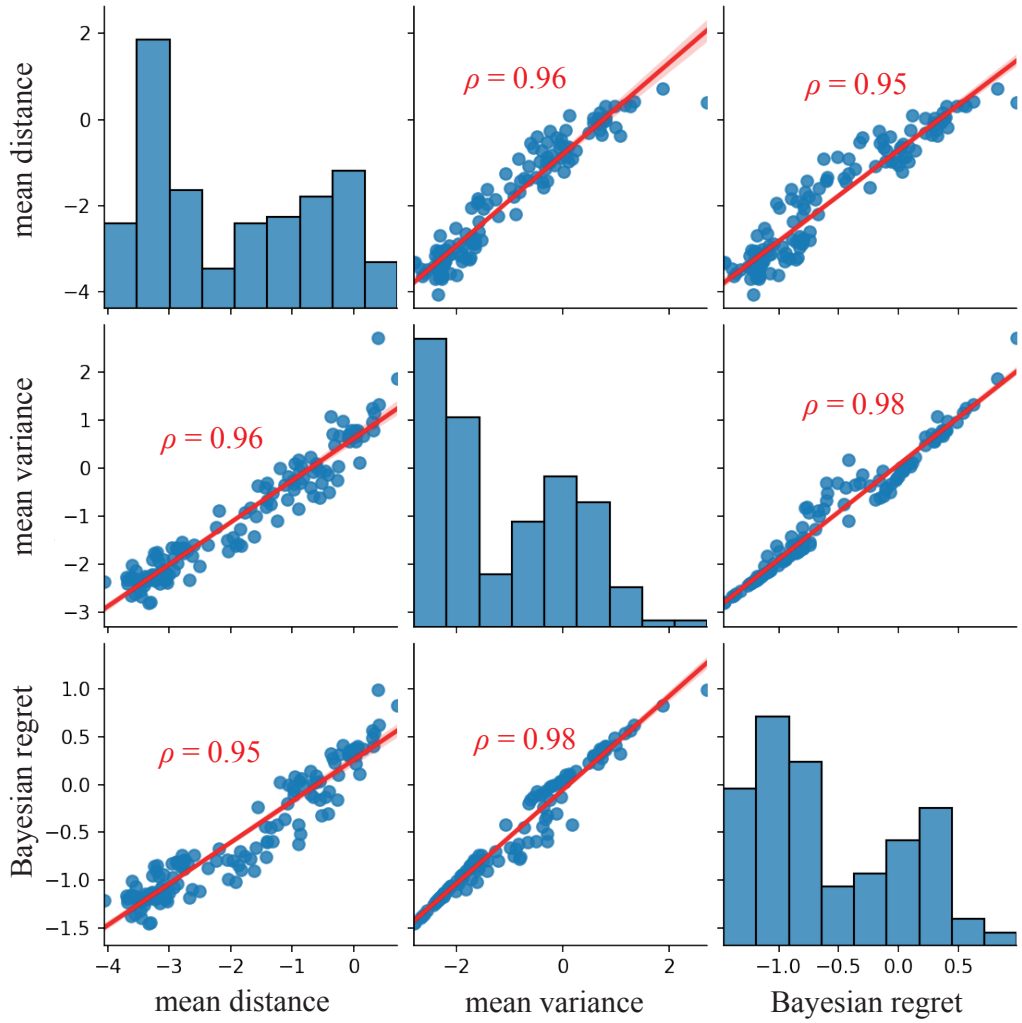

Figure 8: The correlations between regret and measure optimisation

Given two measures π_{emp} and π_{batch}, we consider the following two metrics for π shrinkage; MV $\mathbb{V}\text{ar}[\pi(x)]$ and MD $|x^*_{\text{true}} - \mathbb{E}[\pi(x)]|$. The empirical measure π_{emp} can approximate these as such:

$$\mathbb{E}[\pi(x)] := \int x \mathrm{d}\pi(x) \approx \mathbf{w}_{\text{rec}}^\top \mathbf{X}_{\text{rec}}, \tag{82a}$$

$$\mathbb{V}\text{ar}[\pi(x)] := \int |x - \mathbb{E}[\pi(x)]|^2 \mathrm{d}\pi(x) \approx \mathbf{w}_{\text{rec}}^\top \text{diag}\Big[(\mathbf{X}_{\text{rec}} - \mathbb{E}[\pi(x)])^\top (\mathbf{X}_{\text{rec}} - \mathbb{E}[\pi(x)])\Big], \tag{82b}$$

$$|x^*_{\text{true}} - \mathbb{E}[\pi(x)]| \approx \sqrt{\sum_{k=1}^{d} (x^*_{\text{true},k} - (\mathbf{w}_{\text{rec}}^\top \mathbf{X}_{\text{rec}})_k)^2}, \tag{82c}$$

$$\text{BR} := |y^*_{\text{true}} - \mathbf{w}_{\text{batch}}^\top f_{\text{true}}(\mathbf{X}_{\text{batch}})|. \tag{82d}$$

MV, and MD corresponds to the π shrinkage, of which smaller value indicates shrinking. MD represents the Euclidean distance between the mean of π and the true global maximum x^*_{true}. We compared these two metrics against BR. BR is the batch estimation regret (referred as BR for Theorem 2 in [36]). Experiments were done using SOBER-LFI on the Ackley function (see Table 2) over 10 iterations with 16 repeats (160 data points). Figure 8 shows the linear correlation matrix of these 3 metrics. Both MD and MV are highly correlated with BR, clearly explaining the π shrinkage as the

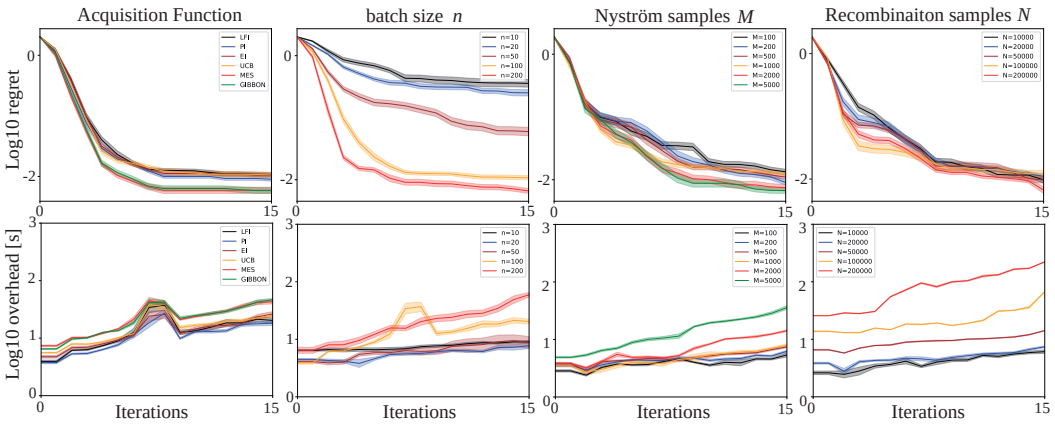

Figure 9: Hyperparameter sensitivity analysis using the Ackley function. Lines and shade area denote mean $\pm$ 1 standard error.

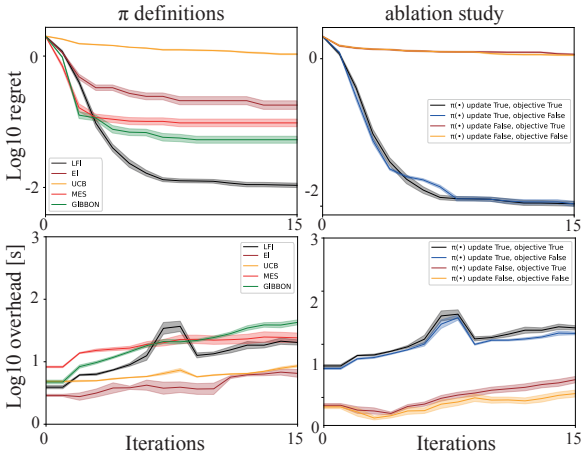

Figure 10: Ablation study using the Ackley function. Lines and shade area denote mean $\pm$ 1 standard error.

dual objective in Eq. (1) is the good measure of BR. In other words, π (MC estimate of x^*) shrinks toward true global maximum x^*_{true} with being smaller variance (more confident), and both linearly correlated to minimising the Bayesian regret, BR.

F.3.3 Hyperparameter sensitivity

We tested the hyperparameter sensitivity of SOBER-LFI using the Ackley function. We examined the effect of AFs α, batch size n, the number of Nyström samples M, and the number of recombination samples N. We averaged the results from 16 experiments with varied random seeds, and terminated at the 15th batch acquisition. The baseline conditions are $n = 100$, $\alpha =$ LFI, $M = 500$, and $N = 20,000$. For AF, the information-theoretic AFs can boost the convergence rate, whereas the others do not change significantly. For the batch size n, the convergence rate can be improved in accordance with the batch size. For quadrature hyperparameters M and N, the larger the number of samples becomes, the faster the convergence does. However, increasing the number of samples leads to additional overhead increase. Our default conditions are competitive throughout the experiments.

F.3.4 Ablation Study

We performed the ablation study to analyse each algorithm's effects on convergence rate. Firstly, we compared the various π definitions defined by the AFs. As shown in Figure 10, LFI and PI definitions are the clear performants. This is because the other AFs are designed to guide the sequential sampling,

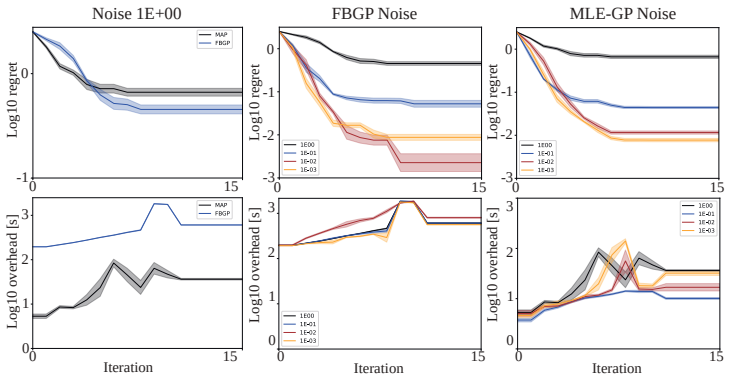

Figure 11: Efficacy of Fully Bayesian Gaussian process modelling using the noisy Ackley function. Lines and shade area denote mean $\pm$ 1 standard error.

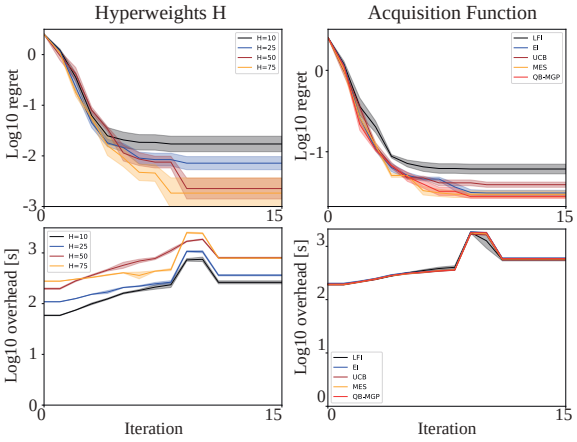

Figure 12: SOBER-LFI consistently outperforms with small overhead. Lines and shade area denote mean $\pm$ 1 standard error.

of which global maxima sensitively changed over iterations. LFI and PI show the possibility of global maxima, which gradually squeezes the region toward the true global maxima. Thus, in SOBER-LFI formulation, LFI AF is well-suited as the definition of π. As another ablation study, we compared whether or not updating π and using AF in the objective RCHQ. As a result, unsurprisingly, updating π is the most influential on the convergence rate. The objective RCHQ does not significantly influence the convergence when we select LFI as the objective. However, information-theoretic AFs can boost the convergence, as shown in Figure 9.

F.3.5 Fully Bayesian Gaussian Process

We further tested the effect of FBGP modelling on the convergence rate. To examine the efficacy, we adopted the noisy Ackley function. We added the Gaussian noise to the queried values from the Ackley function. The amplitude of the noise is varied from 10^{-3} to 1 in a logarithmic order. The baseline conditions are $n = 100$, $\alpha =$ LFI, $M = 500$, $N = 20,000$, and $H = 50$. Figure 11 illustrates that FBGP modelling with quadrature distillation can boost the convergence rate while maintaining the overhead feasibly small (the overhead of FBGP is smaller than DPP-TS with type-II MLE kernel.)

Furthermore, we examined the effects of the number of hyperweights H and the AFs. While H are influential on both convergence rate and overhead, the default value $H = 50$ are reasonably competitive. With regard to the effect of AFs, QB-MGP AF was the performant.