Journal of Machine Learning Research 22 (2021) 1-78 Submitted 12/20; Published 7/21

Are we Forgetting about Compositional Optimisers in Bayesian Optimisation?

Antoine Grosnit * ANTOINE.GROSNIT@HUAWEI.COM
Alexander I. Cowen-Rivers * ALEXANDER.COWEN.RIVERS@HUAWEI.COM
Rasul Tutunov * RASUL.TUTUNOV@HUAWEI.COM
Huawei R&D UK

Ryan-Rhys Griffiths RRG27@CAM.AC.UK
Huawei R&D UK
University of Cambridge

Jun Wang W.J@HUAWEI.COM
Haitham Bou-Ammar † HAITHAM.AMMAR@HUAWEI.COM
Huawei R&D UK
University College London

Editor: Ryan Adams

Abstract

Bayesian optimisation presents a sample-efficient methodology for global optimisation. Within this framework, a crucial performance-determining subroutine is the maximisation of the acquisition function, a task complicated by the fact that acquisition functions tend to be non-convex and thus nontrivial to optimise. In this paper, we undertake a comprehensive empirical study of approaches to maximise the acquisition function. Additionally, by deriving novel, yet mathematically equivalent, compositional forms for popular acquisition functions, we recast the maximisation task as a compositional optimisation problem, allowing us to benefit from the extensive literature in this field. We highlight the empirical advantages of the compositional approach to acquisition function maximisation across 3958 individual experiments comprising synthetic optimisation tasks as well as tasks from Bayesmark. Given the generality of the acquisition function maximisation subroutine, we posit that the adoption of compositional optimisers has the potential to yield performance improvements across all domains in which Bayesian optimisation is currently being applied. An open-source implementation is made available at `https://github.com/huawei-noah/noah-research/tree/CompBO/BO/HEBO/CompBO`.

Keywords: Black Box Optimisation, Bayesian Optimisation, Compositional Optimisation, Acquisition Functions, Empirical Analysis

∗. Equal contribution
†. Honorary position at UCL

1. Introduction

Bayesian optimisation is a method for optimising black-box objective functions (Kushner, 1964; Močkus, 1975; Jones et al., 1998). The black-box optimisation (BBO) problem describes the search for the global maximiser $\mathbf{x}^*$ of an unknown objective function $f(\mathbf{x})$. The objective function is unknown in the sense that an analytical form is unavailable. However, the objective may still be evaluated pointwise at arbitrary query locations within the bounds of the design space. A further characteristic of the BBO problem is that each query is expensive in terms of time, and as such, it is desirable to query as few points as possible in the search for the global maximiser.

Real world examples of BBO problems are ubiquitous. Illustrative examples include hyperparameter tuning in machine learning (Falkner et al., 2018; Kandasamy et al., 2018; White et al., 2019; Gabillon et al., 2020; Cowen-Rivers et al., 2020a; Turner et al., 2021), where the black-box objective is the mapping between a set of model hyperparameters $\mathbf{x}$ and the validation set performance $f(\mathbf{x})$, as well as automatic chemical design (Gómez-Bombarelli et al., 2018; Korovina et al., 2020; Moss and Griffiths, 2020; Griffiths and Hernández-Lobato, 2020), where the black-box objective is the mapping between a molecule $\mathbf{x}$ and its suitability as a drug candidate $f(\mathbf{x})$. Further examples of BBO problems appear as subroutines of optimisation algorithms such as immune optimisation (Zhang et al., 2015; Mahapatra et al., 2015), ant colony optimisation (Yoo and Han, 2014; Speranskii, 2015) and genetic algorithms (Peng and Li, 2015), in reinforcement learning when accounting for safety (Cowen-Rivers et al., 2020b; Abdullah et al., 2019), in multi-agent systems to compute Nash equilibria (Yang et al., 2020; Aprem and Roberts, 2018), in speech recognition (Moss et al., 2020b) and more broadly across domains spanning architecture (Costa et al., 2015), supply chain networks (Aziz et al., 2021), human motion prediction (Bourached et al., 2020), the fine arts (Stork et al., 2021), astrophysics (Griffiths et al., 2021), chemical engineering (Ploskas et al., 2018), materials science (Cheng et al., 2020) and biology (Shah and Sahinidis, 2012; Moss et al., 2020a).

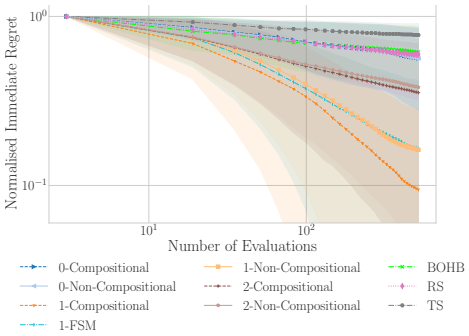

Figure 1: Summary plot for 3100 synthetic BBO experiments showing first-order compositional optimisers outperform others. Lower regret indicates better performance.

Various strategies exist for optimising black-box objective functions including zero-order methods (Valko et al., 2013; Grill et al., 2015; Gabillon et al., 2020), resource allocation methods (Li et al., 2017; Falkner et al., 2018) and surrogate model-based methods (Snoek et al., 2012; Shahriari et al., 2016; Frazier, 2018). In this paper, we focus on Bayesian optimisation, a sequential, data-efficient, surrogate model-based approach that is particularly effective when function evaluations are costly. The two core components of the Bayesian optimisation algorithm are a probabilistic surrogate model and an acquisition function. The probabilistic surrogate model fa-

cilitates data efficiency by making use of the full optimisation history to represent the black-box function and additionally leverages uncertainty estimates to guide exploration. Given that the true sequential risk describing the optimality of a sequence of queries is computationally intractable, an acquisition function is a myopic heuristic which acts as a proxy to the true sequential risk. The acquisition function measures the utility of a query point $\mathbf{x}$ by its mean value under the surrogate model (exploitation) as well as its uncertainty under the surrogate model (exploration). At each round of the Bayesian optimisation algorithm, the acquisition function is maximised to select the next query point.

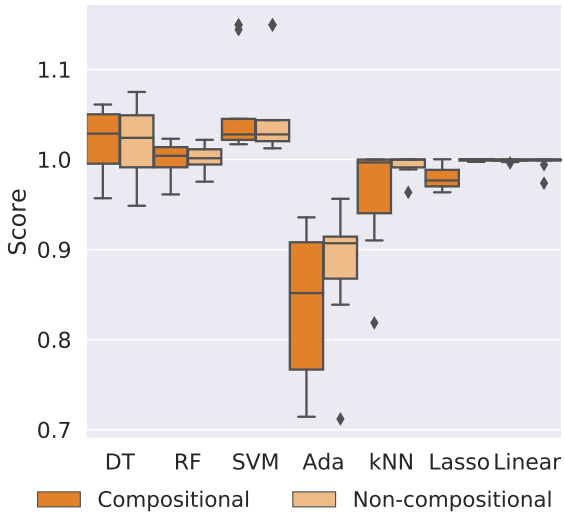

Figure 2: Bayesmark regression summary amalgamating the results from 54 Bayesmark regression tasks when we compare compositional and non-compositional optimisers. Higher score is better. Boxplots show median, lower and upper quartiles of the scores. Results show that compositional optimisers outperform non-compositional optimisers on half of the tasks.

It has been argued that maximisation of the acquisition function is an important, yet neglected determinant of the performance of Bayesian optimisation schemes (Wilson et al., 2018b). The vast majority of acquisition functions however, constitute a serious challenge from the standpoint of optimisation; a characteristic exacerbated in the batch setting, where acquisition functions are routinely non-convex, high-dimensional and intractable (Wilson et al., 2018b). Many strategies exist for optimising acquisition functions including gradient-based methods (Duchi et al., 2011; Hinton et al., 2012; Kingma and Ba, 2015), evolutionary methods (Igel et al., 2006; Jastrebski and Arnold, 2006; Hansen, 2016) as well as variations of random search (Schumer and Steiglitz, 1968; Schrack and Choit, 1976; Bergstra and Bengio, 2012). In this work, we choose to focus on gradient-based methods which were recently shown to be highly effective for optimising a wide class of Monte Carlo acquisition functions (Wilson et al., 2018b).

The most commonly-used acquisition functions in practical applications (Snoek et al., 2012) are Monte Carlo acquisition functions in the sense that they are formulated as integrals with respect to the current probabilistic belief over the unknown function f (Shahriari et al., 2016; Wilson et al., 2018b); these integrals are typically intractable and as such are approximated by the corresponding Monte Carlo (MC) estimate. In order to admit gradient-based optimisation, a reparametrisation trick (Kingma and Welling, 2014; Rezende et al., 2014), introduced first as infinitesimal perturbation analysis (Cao, 1985; Glasserman, 1988), is applied to facilitate differentiation through the MC estimates with respect to the parameters of the surrogate model. It was shown in (Wilson et al., 2018b) that acquisition functions estimated via MC integration are consistently amenable to gradient-based optimisation via standard first and second-order

methods including SGA (Bottou and Bousquet, 2007), Adam (Kingma and Ba, 2015), RMSprop (Hinton et al., 2012), AdaGrad (Duchi et al., 2011) and L-BFGS-B (Zhu et al., 1997).

In this work, we exploit the observation that most common acquisition functions exhibit compositional structure and hence can be equivalently reformulated in a compositional form (Wang et al., 2017a). Such a reformulation allows a broader class of optimisation techniques to be applied for acquisition function optimisation (Tutunov et al., 2020; Ghadimi et al., 2020; Wang et al., 2017b) and in practice can more often enable better numerical performance to be achieved in comparison with standard first and second-order methods. The compositional form is achieved for the expected improvement (EI), simple regret (SR), upper confidence bound (UCB) and probability of improvement (PI) acquisition functions by first exposing the finite-sum form of the reparameterised acquisition functions derived by (Wilson et al., 2018b) and second introducing a deterministic outer function when considering the problem from a matrix-vector perspective. It should be noted that reformulating the acquisition function in a compositional form is distinct from the setting where the black-box function has a compositional form (Astudillo and Frazier, 2019).

In order to both improve and analyse the optimisation performance on the compositional form of the acquisition function, we introduce several algorithmic adaptations. Firstly, we present (C)L-BFGS; a modification to the L-BFGS algorithm to enable the handling of nested compositional forms. Secondly, we develop AdamOS, a variant of the Adam optimiser (Kingma and Ba, 2015) which borrows the hyperparameter settings of CAdam (Tutunov et al., 2020) and facilitates performance comparison between compositional and non-compositional optimisers. Lastly, we formulate a generalised iterative update rule for first-order compositional optimisers and show how the updates of a number of first-order optimisers may be expressed in this manner.

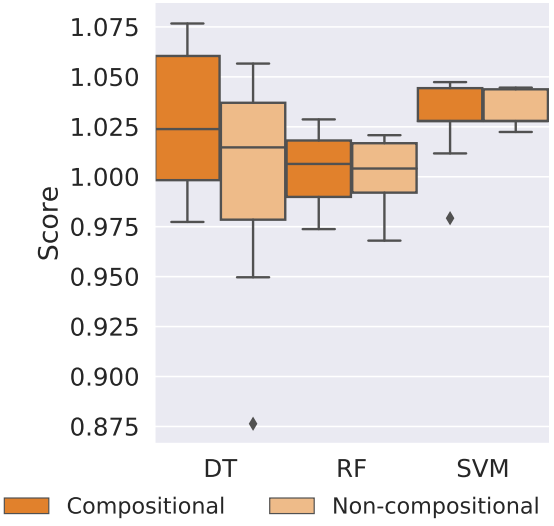

Figure 3: Bayesmark classification summary amalgamating the results from 54 Bayesmark classification tasks, we compare compositional and non-compositional optimisers. Higher score is better. Boxplots show lower, median and upper quartiles of the data. Results show compositional optimisers outperforming non-compositional optimisers across all tasks.

In our empirical study, we seek to identify the most effective means of optimising the acquisition function under a range of experimental conditions including input dimensionality, presence or absence of observation noise and choice of acquisition function. We investigate twenty-eight optimisation schemes, spanning zeroth, first and second-order optimisers as well as both compositional and non-compositional methods. Additionally, we seek to answer the following questions: Are there benefits to the finite-sum formulation of the reparameterised acquisition functions compared to the more frequently-encountered empirical risk minimisation formulation? Are

compositional or non-compositional approaches to optimisation more effective and if so, under what conditions are they more effective? What are the performance-related trade-offs in memory-efficient implementations of compositional acquisition functions? How does the wall-clock time of compositional optimisation methods compare to non-compositional optimisation methods, and how does this vary with the dimensionality of the input space? How do compositional optimisers fare when faced with noisy observations?

In order to answer these questions, we first perform a set of experiments across five noiseless synthetic function tasks. Using this set of noiseless experiments as a filter for the most effective optimisers, we then perform a second set of experiments on the Bayesmark data sets which are noisy and bear a closer resemblance to real-world problems than the synthetic tasks. Our results for the synthetic experiments are summarised in Figure 1 whilst our results for the Bayesmark data sets are summarised in Figure 2 and Figure 3 for the regression and classification challenges respectively. In sum total, our empirical study comprises 3958 individual experiments.

The paper is organised as follows: First, we introduce the necessary background on the Bayesian optimisation framework. Second, we hone in on the acquisition function maximisation subroutine of Bayesian optimisation with the intent to understand the efficacy of compositional optimisation schemes. We provide a general overview of compositional optimisation and derive compositional forms for the four most popular myopic acquisition functions. Third, we discuss state-of-the-art compositional solvers, namely CAdam, NASA, SCGA and ASCGA. Fourth, we detail our experimental setup and present the empirical results. Fifth, we analyse the experimental results, draw conclusions and indicate avenues for future work as well as descriptions of open problems in acquisition function maximisation.

2. Bayesian Optimisation

We consider a sequential decision-making approach to the global optimisation of smooth functions $f : \mathcal{X} \to \mathbb{R}$ over a bounded input domain $\mathcal{X} \subseteq \mathbb{R}^d$. At each decision round, i, we select an input $\mathbf{x}_i \in \mathcal{X}$ and observe the value of the *black-box* function $f(\mathbf{x}_i)$. We allow the returned value to be either deterministic i.e., $y_i = f(\mathbf{x}_i)$ or stochastic with $y_i = f(\mathbf{x}_i) + \epsilon_i$, where ϵ_i denotes a bounded-variance random variable. Our goal is to rapidly approach the maximum $\mathbf{x}^\star = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$ in terms of cumulative regret $R_T = \sum_{t=1}^{T} r_t$ where $r_t = f(\mathbf{x}^\star) - f(\mathbf{x}_t^{(\text{new})})$ is the distance between maximum function value $f(\mathbf{x}^\star)$ and function value at the algorithm's best recommendation at round t denoted as $\mathbf{x}_t^{(\text{new})}$. Since both $f(\cdot)$ and $\mathbf{x}^\star$ are unknown, solvers need to trade off exploitation and exploration during the search process.

To reason about the unknown function, typical Bayesian optimisation algorithms assume smoothness and adopt Bayesian modelling as a principle to carry out inference about the properties of $f(\cdot)$ in light of the observations. Here, one introduces a prior to encode beliefs over the smoothness properties and an observation model to describe collected data, $\mathcal{D}_i = \{\mathbf{x}_l, y_l\}_{l=1}^{n_i}$, up to the i^{th} round with n_i denoting the total acquired data so far. Using these two components in addition to Bayes rule, we can then compute a posterior $p(f(\cdot)|\mathcal{D}_i)$ to encode all knowledge of $f(\cdot)$ allowing us to account for the location of the maximum.

2.1 Bayesian Optimisation with Gaussian Processes

A Gaussian process (GP) offers a flexible and sample-efficient procedure for placing priors over unknown functions (Rasmussen and Williams, 2006). These models are fully specified by a mean function $m(\mathbf{x})$ and a covariance function, or kernel, $k(\mathbf{x}, \mathbf{x}')$ that encodes the smoothness assumptions on $f(\cdot)$. Given any finite collection of inputs $\mathbf{x}_{1:n_i}$, the outputs are jointly Gaussian given by:

$$f(\mathbf{x}_{1:n_i})|\boldsymbol{\theta} \sim \mathcal{N}\left(m(\mathbf{x}_{1:n_i}), \mathbf{K}_{\boldsymbol{\theta}}(\mathbf{x}_{1:n_i}, \mathbf{x}_{1:n_i})\right),$$

where $[m(\mathbf{x}_{1:n_i})]_k = m(\mathbf{x}_k)$ denotes the mean vector, and $\mathbf{K}_{\boldsymbol{\theta}}(\mathbf{x}_{1:n_i}, \mathbf{x}_{1:i}) \in \mathbb{R}^{n_i \times n_i}$ the covariance matrix with its $(k,l)^{th}$ entry computed as $[\mathbf{K}_{\boldsymbol{\theta}}(\mathbf{x}_{1:n_i}, \mathbf{x}_{1:n_i})]_{k,l} = k_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{x}_l)$. Here, $k_{\boldsymbol{\theta}}(\cdot,\cdot)$ represents a parameterised kernel with unknown hyperparameters $\boldsymbol{\theta}$ corresponding to lengthscales or signal amplitudes for example. For ease of presentation following (Rasmussen and Williams, 2006), we use a zero-mean prior in our notation here. In terms of the choice of Gaussian process kernel, there are a wide array of options which encode prior modelling assumptions about the latent function. Two of the most commonly-encountered kernels in the Bayesian optimisation literature are the squared exponential (SE) and Matérn(5/2) kernels

$$[\mathbf{K}_{\boldsymbol{\theta}}^{\text{SE}}(\mathbf{x}_{1:n_i}, \mathbf{x}_{1:n_i})]_{k,l} = k_{\boldsymbol{\theta}}^{\text{SE}}(\mathbf{x}_k, \mathbf{x}_l) = \exp\left(-\frac{1}{2}r^2\right)$$
$$[\mathbf{K}_{\boldsymbol{\theta}}^{\text{Matérn(5/2)}}(\mathbf{x}_{1:n_i}, \mathbf{x}_{1:n_i})]_{k,l} = k_{\boldsymbol{\theta}}^{\text{Matérn(5/2)}}(\mathbf{x}_k, \mathbf{x}_l) = \exp\left(-\sqrt{5}r\right)\left(1 + \sqrt{5}r + \frac{5}{3}r^2\right),$$

where $r = \sqrt{(\mathbf{x}_k - \mathbf{x}_l)^\mathsf{T} \operatorname{diag}\left(\boldsymbol{\theta}^2\right)^{-1} (\mathbf{x}_k - \mathbf{x}_l)}$ and $\boldsymbol{\theta} \in \mathbb{R}^d$ denotes the d-dimensional hyperparameters with $\boldsymbol{\theta}^2$ executed element-wise. As noted in (Rasmussen and Williams, 2006), both these kernels are suited for situations where little is known about the latent function in question. The Matérn kernel, however, is arguably suitable for a broader class of real-world Bayesian optimisation problems as it imposes less restrictive smoothness assumptions on $f(\cdot)$ (Stein, 2012). Following initial experimentation with linear, cosine, squared exponential and various Matérn kernels, we chose the Matérn(5/2) kernel to perform all experiments with.

Given the data $\mathcal{D}_i$, and assuming Gaussian-corrupted observations $y_i = f(\mathbf{x}_i) + \epsilon_i$ with $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, we can write the joint distribution over the data and an arbitrary evaluation input $\mathbf{x}$ as:

$$\left[\begin{array}{c} \mathbf{y}_{1:n_i} \\ f(\mathbf{x}) \end{array}\right] \Bigg| \, \boldsymbol{\theta} \sim \mathcal{N}\left(\mathbf{0}, \left[\begin{array}{cc} \mathbf{K}_{\boldsymbol{\theta}}^{(i)} + \sigma^2\mathbf{I} & \mathbf{k}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}) \\ \mathbf{k}_{\boldsymbol{\theta}}^{(i),\mathsf{T}}(\mathbf{x}) & k_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{x}) \end{array}\right]\right),$$

where $\mathbf{K}_{\boldsymbol{\theta}}^{(i)} = \mathbf{K}_{\boldsymbol{\theta}}(\mathbf{x}_{1:n_i}, \mathbf{x}_{1:n_i})$ and $\mathbf{k}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}) = \mathbf{k}_{\boldsymbol{\theta}}(\mathbf{x}_{1:n_i}, \mathbf{x})$. With the above joint distribution derived, we can now easily compute the predictive posterior through marginalisation

(Rasmussen and Williams, 2006) leading us to $f(\mathbf{x})|\mathcal{D}_i, \boldsymbol{\theta} \sim \mathcal{N}\left(\boldsymbol{\mu}_i(\mathbf{x};\boldsymbol{\theta}), \sigma_i(\mathbf{x};\boldsymbol{\theta})^2\right)$ with:

$$\mu_i(\mathbf{x};\boldsymbol{\theta}) = \mathbf{k}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x})^\mathsf{T}(\mathbf{K}_{\boldsymbol{\theta}}^{(i)} + \sigma^2\mathbf{I})^{-1}\mathbf{y}_{1:n_i}$$
$$\sigma_i(\mathbf{x};\boldsymbol{\theta})^2 = k_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{x}) - \mathbf{k}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x})^\mathsf{T}(\mathbf{K}_{\boldsymbol{\theta}}^{(i)} + \sigma^2\mathbf{I})^{-1}\mathbf{k}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}).$$

Of course, the above can be generalised to the case when a predictive posterior over q arbitrary evaluation points, $\mathbf{x}_{1:q}^\star$, needs to be computed as is the case in batched adaptations of Bayesian optimisation. In such a setting $\mathbf{f}(\mathbf{x}_{1:q}^\star)|\mathcal{D}_i, \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_i(\mathbf{x}_{1:q}^\star;\boldsymbol{\theta}), \boldsymbol{\Sigma}_i(\mathbf{x}_{1:q}^\star;\boldsymbol{\theta}))$ with:

$$\boldsymbol{\mu}_i(\mathbf{x}_{1:q}^\star;\boldsymbol{\theta}) = \mathbf{K}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}_{1:q}^\star, \mathbf{x}_{1:n_i})(\mathbf{K}_{\boldsymbol{\theta}}^{(i)} + \sigma^2\mathbf{I})^{-1}\mathbf{y}_{1:n_i}$$
$$\boldsymbol{\Sigma}_i(\mathbf{x}_{1:q}^\star;\boldsymbol{\theta}) = \mathbf{K}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}_{1:q}^\star, \mathbf{x}_{1:q}^\star) - \mathbf{K}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}_{1:q}^\star, \mathbf{x}_{1:n_i})(\mathbf{K}_{\boldsymbol{\theta}}^{(i)} + \sigma^2\mathbf{I})^{-1}\mathbf{K}_{\boldsymbol{\theta}}^{(i),\mathsf{T}}(\mathbf{x}_{1:q}^\star, \mathbf{x}_{1:n_i}).$$

The remaining ingredient needed in a GP pipeline is a process to determine the unknown hyperparameters $\boldsymbol{\theta}$ given a set of observation $\mathcal{D}_i$. In standard GPs (Rasmussen and Williams, 2006), $\boldsymbol{\theta}$ are fit by minimising the negative log marginal likelihood (NLML) leading us to the following optimisation problem:

$$\min_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \frac{1}{2}\det\left(\mathbf{C}_{\boldsymbol{\theta}}^{(i)}\right) + \frac{1}{2}\mathbf{y}_{1:n_i}^\mathsf{T}\mathbf{C}_{\boldsymbol{\theta}}^{(i),-1}\mathbf{y}_{1:n_i} + \frac{n_i}{2}\log 2\pi, \text{ with } \mathbf{C}_{\boldsymbol{\theta}}^{(i)} = \mathbf{K}_{\boldsymbol{\theta}}^{(i)} + \sigma^2\mathbf{I}. \quad (1)$$

The objective in Equation 1 represents a non-convex optimisation problem making GPs susceptible to local minima. Various off-the-shelf optimisation solvers ranging from first-order (Kingma and Ba, 2015; Bottou and Bousquet, 2007) to second-order (Zhu et al., 1997; Amari, 1998) methods have been rigorously studied in the literature. In our experiments, we made use of a set of implementations provided in GPyTorch (Gardner et al., 2018) that relied on a scipy (Virtanen et al., 2020) implementation of L-BFGS-B (Zhu et al., 1997) for determining $\boldsymbol{\theta}$. It is also worth noting that gradients of the loss in Equation 1 require inverting an $n_i \times n_i$ covariance matrix leading to an order of $\mathcal{O}(n_i^3)$ complexity in each optimisation step. In large data regimes, variational GPs have proved to be a scalable methodology through the usage of $m << n_i$ inducing points (Titsias, 2009; Hensman et al., 2013).

In Bayesian optimisation however, data is typically sparse due to the expense of evaluating even one query of the black-box function, which makes the application of sparse GPs less attractive in these scenarios. While other scalable surrogate models such as Bayesian neural networks (BNNs) and Random Forest have featured in the literature (Snoek et al., 2015b; Hutter et al., 2011b), each come with disadvantages. Many BNN-based approaches rely on approximate inference, and hence uncertainty estimates may deteriorate in quality relative to exact GPs while the Random-Forest-based SMAC algorithm is not amenable to gradient-based optimisation due to a discontinuous response surface (Hutter et al., 2011a; Shahriari et al., 2016). As such, we restrict our focus to exact GPs and direct the reader to external sources for discussion on alternative surrogate models such as sparse GPs (McIntire et al., 2016), BNNs (Snoek et al., 2015a; Springenberg et al., 2016; Hernández-Lobato et al., 2017), neural processes (Kim et al., 2018) as well as heteroscedastic GPs (Calandra, 2017; Griffiths et al., 2019).

2.2 Acquisition Functions

Having introduced a distribution over latent black-box functions and specified mechanisms for updating hyperparameters, we now discuss the process by which novel query points are suggested for collection in order to improve the surrogate model's best guess for the global optimiser $\mathbf{x}^\star$. In Bayesian optimisation, proposing novel query points is performed through maximising an acquisition function $\alpha(\cdot|\mathcal{D}_i)$ that trades off exploration and exploitation by utilising statistics from $p(f(\cdot)|\mathcal{D}_i)$, i.e., $\mathbf{x}_{i+1} = \arg\max_{\mathbf{x}} \alpha(\mathbf{x}|\mathcal{D}_i)$. Acquisition functions can be taxonomised into myopic and non-myopic forms (González et al., 2016). The former class involves integrals defined in terms of beliefs over unknown outcomes from the black-box function, while the latter class constitutes more complicated nested integrals. In this paper, we focus on representative examples of standard myopic acquisitions whilst considering entropy search as a widely-used non-myopic acquisition. We detail these acquisitions next.

Expected Improvement: One of the most popular acquisition functions is expected improvement (Močkus, 1975; Jones et al., 1998), which determines new query points by maximising expected gain relative to the function values observed so far. Formally, denote by $\mathbf{x}_i^+$ an input point in $\mathcal{D}_i$ for which $f(\cdot)$ is maximised, i.e., $\mathbf{x}_i^+ = \arg\max_{\mathbf{x}\in\mathbf{x}_{1:n_i}} f(\mathbf{x})$. Given $\mathbf{x}_i^+$, we define an expected improvement acquisition to compute the expected positive gain in function value compared to the best incumbent point in $\mathcal{D}_i$ as:

$$\alpha_{\text{EI}}(\mathbf{x}|\mathcal{D}_i) = \mathbb{E}_{f(\mathbf{x})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\max\{(f(\mathbf{x}) - f(\mathbf{x}_i^+)), 0\}\right] = \mathbb{E}_{f(\mathbf{x})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\text{ReLU}(f(\mathbf{x}) - f(\mathbf{x}_i^+))\right],$$

where ReLU represents a rectified linear unit with $\text{ReLU}(a) = \max\{0, a\}$. The above can be generalised to support a batch form generating $\mathbf{x}_{1:q}$ query points as introduced in (Ginsbourger et al., 2008). Here, we first compute the multi-dimensional predictive posterior $f(\mathbf{x}_{1:q})|\mathcal{D}_i, \boldsymbol{\theta}$ as described in Section 2.1 and then define the maximal gain across all q-batches as:

$$\alpha_{\text{q-EI}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{\mathbf{f}(\mathbf{x}_{1:q})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\max_{j\in 1:q}\{\text{ReLU}(\mathbf{f}(\mathbf{x}_{1:q}) - f(\mathbf{x}_i^+)\mathbf{1}_q)\}\right], \tag{2}$$

where $\mathbf{1}_q$ denotes a q-dimensional vector of ones and as such, the $\text{ReLU}(\cdot)$ is to be executed element-wise. In words, Equation 2 simply computes the expected maximal improvement across all q-dimensional predictions compared to the best incumbent point in $\mathcal{D}_i$. This form of acquisition is termed joint parallel acquisition function maximisation in (Wilson et al., 2018b) (other forms being greedy and incremental) and is chosen for the experiments in this paper due to its usage in the BoTorch library (Balandat et al., 2020). In joint parallel acquisition function maximisation, each query point is treated as a dimension of the acquisition surface and the set of batch points is optimised on this surface cf. figure 2 of (Wilson et al., 2018b) for an illustration.

Probability of Improvement: Another commonly-used acquisition function in Bayesian optimisation is the probability of improvement criterion which measures the probability of acquiring gains in the function value compared to $f(\mathbf{x}_i^+)$ (Kushner, 1964). Such a probability is measured through an expected Heaviside step function as follows:

$$\alpha_{\text{PI}}(\mathbf{x}|\mathcal{D}_i) = \mathbb{E}_{f(\mathbf{x})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\mathbb{1}\{f(\mathbf{x}) - f(\mathbf{x}_i^+)\}\right],$$

with $\mathbb{1}\{f(\mathbf{x}) - f(\mathbf{x}_i^+)\} = 1$ if $f(\mathbf{x}) \geq f(\mathbf{x}_i^+)$ and zero otherwise. Analogous to expected improvement, we can extend $\alpha_{\text{PI}}(\mathbf{x}|\mathcal{D}_i)$ to a batch form by generalising the step function to support-vectored random variables in addition to adopting maximal gain across all batches as an improvement metric:

$$\alpha_{\text{q-PI}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{\mathbf{f}(\mathbf{x}_{1:q})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\max_{j\in 1:q}\left\{\mathbb{1}\{\mathbf{f}(\mathbf{x}_{1:q}) - f(\mathbf{x}_i^+)\mathbf{1}_q\}\right\}\right], \tag{3}$$

where $\mathbb{1}\{\mathbf{f}(\mathbf{x}_{1:q}) - f(\mathbf{x}_i^+)\mathbf{1}_q\}$ returns a q-dimensional binary vector with $[\mathbb{1}\{\mathbf{f}(\mathbf{x}_{1:q}) - f(\mathbf{x}_i^+)\}]_j = 1$ if $[\mathbf{f}(\mathbf{x}_{1:q})]_j \geq [f(\mathbf{x}_i^+)\mathbf{1}_q]_j$ and zero otherwise for all $j \in \{1, \ldots, q\}$.

Simple Regret: In simple regret, new query points are determined by maximising expected outcomes, i.e., $\alpha_{\text{SR}}(\mathbf{x}|\mathcal{D}_i) = \mathbb{E}_{f(\mathbf{x})|\mathcal{D}_i,\boldsymbol{\theta}}[f(\mathbf{x})]$. This, in turn, can also be generalised to a batch mode by considering the maximal improvement across all q batches leading to:

$$\alpha_{\text{q-SR}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{\mathbf{f}(\mathbf{x}_{1:q})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\max_{j\in 1:q}\{\mathbf{f}(\mathbf{x}_{1:q})\}\right].$$

Upper Confidence Bound: In this type of acquisition, the learner trades off the mean and variance of the predictive distribution to gather new query points for function evaluation (Srinivas et al., 2010). In the standard form, an upper-confidence bound acquisition can simply be written as: $\alpha_{\text{UCB}}(\mathbf{x}|\mathcal{D}_i) = \mu_i(\mathbf{x};\boldsymbol{\theta}) + \sqrt{\beta}\sigma_i(\mathbf{x};\boldsymbol{\theta})$ with $\beta \in \mathbb{R}$ being a free tuneable hyperparameter. Although widely used, such a form of the upper-confidence bound is not directly amendable to parallelism. To circumvent this problem, the authors in (Wilson et al., 2018b) have shown an equivalent form for the expectation by exploiting reparameterisation leading to:

$$\alpha_{\text{UCB}}(\mathbf{x}|\mathcal{D}_i) = \mu_i(\mathbf{x};\boldsymbol{\theta}) + \sqrt{\beta}\sigma_i(\mathbf{x};\boldsymbol{\theta}) = \mathbb{E}_{f(\mathbf{x})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\mu_i(\mathbf{x};\boldsymbol{\theta}) + \sqrt{\beta\pi/2}|\gamma_i(\mathbf{x};\boldsymbol{\theta})|\right],$$

with $\gamma_i(\mathbf{x};\boldsymbol{\theta}) = f(\mathbf{x}) - \mu_i(\mathbf{x};\boldsymbol{\theta})$. Given such a formulation, we can now follow similar reasoning to previous generalisations of acquisition functions and consider a batched version by taking the maximum over all q query points:

$$\alpha_{\text{q-UCB}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{\mathbf{f}(\mathbf{x}_{1:q})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\max_{j\in 1:q}\left\{\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \sqrt{\beta\pi/2}|\boldsymbol{\gamma}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})|\right\}\right],$$

where $\boldsymbol{\gamma}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) = \mathbf{f}(\mathbf{x}_{1:q}) - \boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})$.

Entropy Search: In (Hennig and Schuler, 2012), an information-theoretic approach is introduced to select novel query points based on an approximation of the posterior entropy for the global optimiser $\mathbf{x}^\star$. The next point $\mathbf{x}_{i+1}$ is chosen to minimise the posterior entropy $\mathbb{E}_{f(\mathbf{x}|\mathcal{D}_i,),\boldsymbol{\theta}}\left[\mathbb{H}[p(\mathbf{x}^*|\mathcal{D}_i \cup \{\mathbf{x}, f(\mathbf{x})\})]\right]$ and hence minimises the uncertainty over the location of $\mathbf{x}^+$. In (Wilson et al., 2018b), a parallel implementation is introduced via a $q-$batch form for the entropy search acquisition function

$$\alpha_{\text{q-ES}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = -\mathbb{E}_{\mathbf{f}(\mathbf{x}_{1:q})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\text{H}\left[\mathbb{E}_{f\left(\mathbf{x}_{1:u}^{(\text{g})}\right)|\mathcal{D}_i\cup\{\mathbf{x}_{1:q};\mathbf{f}(\mathbf{x}_{1:q})\},\boldsymbol{\theta}}\left[\mathbb{1}\{\mathbf{f}(\mathbf{x}_{1:u}^{(\text{g})}) - \max_{j\in 1:u} f(\mathbf{x}_j^{(\text{g})})\mathbf{1}_u\}\right]\right]\right],$$

where $\mathbf{x}_{1:u}^{(\mathrm{g})}$ is a grid of u discrete locations sampled from the input domain $\mathcal{X}$ according to a discretisation measure $\mathcal{U}(\cdot|\mathcal{D}_i)$, $\mathrm{H}[\cdot]$ is the Shannon entropy and $\mathbb{1}\{\mathbf{f}(\mathbf{x}_{1:u}^{(\mathrm{g})}) - \max_{j\in 1:u} f(\mathbf{x}_j^{(\mathrm{g})})\mathbf{1}_u\}$ returns a u-dimensional binary vector with $[\mathbb{1}\{\mathbf{f}(\mathbf{x}_{1:u}^{(\mathrm{g})}) - \max_{j\in 1:u} f(\mathbf{x}_j^{(\mathrm{g})})\mathbf{1}_u\}]_\ell = 1$ if $f(\mathbf{x}_\ell^{(\mathrm{g})}) = \max_{j\in 1:u} f(\mathbf{x}_j^{(\mathrm{g})})$ and zero otherwise for all $\ell \in \{1,\dots,u\}$

Following the introduction of GP surrogate models and acquisition functions, we are now ready to present a canonical template for the Bayesian optimisation algorithm. The main steps are summarised in the pseudocode of Algorithm 1.

Algorithm 1 Batched Bayesian Optimisation with GPs

1: **Inputs:** Total number of outer iterations N, initial randomly-initialised data set $\mathcal{D}_0 = \{\mathbf{x}_l, y_l \equiv f(\mathbf{x}_l)\}_{l=1}^{n_0}$, batch size q, acquisition function type
2: **for** $i = 0 : N-1$:
3: Fit the GP model to the current data set $\mathcal{D}_i$ by $\min_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$ from Equation 1
4: Find q points by solving $\mathbf{x}_{1:q}^{(\mathrm{new})} = \arg\max_{\mathbf{x}_{1:q}} \alpha_{\text{q-type}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$)
5: Evaluate new inputs by querying the black-box to acquire $\mathbf{y}_{1:q}^{(\mathrm{new})} = f(\mathbf{x}_{1:q}^{(\mathrm{new})})$
6: Update the data set creating $\mathcal{D}_{i+1} = \mathcal{D}_i \cup \{\mathbf{x}_l^{(\mathrm{new})}, y_l^{(\mathrm{new})}\}_{l=1}^{q}$
7: **end for**
8: **Output:** Return the best-performing query point from the data $\mathbf{x}^\star = \arg\max_{\mathbf{x}\in\mathcal{D}_N} f(\mathbf{x})$

First, a GP model is fit to the available data (see line 3 of Algorithm 1) enabling the computation of the predictive distribution needed to maximise the acquisition function (line 4). Having acquired new query points, the learner then updates the data set $\mathcal{D}_i$ after which the above process repeats until a total number of iterations N is reached. At the end of the main loop, Algorithm 1 outputs $\mathbf{x}^\star$, the best performing input from all acquired data $\mathcal{D}_N$.

Clearly, maximising acquisition functions plays a crucial role in Bayesian optimisation as this step constitutes the process by which the learner yields concrete exploratory actions to improve the guess for the global optimum $\mathbf{x}^\star$. The majority of acquisition functions, however, are often intractable, posing formidable challenges during the optimisation step in line 4 of Algorithm 1. In order to tackle these challenges, researchers have proposed a plethora of methods that can generally be categorised into three main groups. Approximation techniques, the first group, replace the quantity of interest with a more readily-computable one e.g. (Cunningham et al., 2011) apply expectation propagation (Minka, 2001a,b; Opper et al., 2001) as an approximate integration method while (Wang and Jegelka, 2017) apply a mean field approximation to enable a Gumbel sampling approximation to their max-value entropy search acquisition function. As noted in (Wilson et al., 2018b), these methods tend to work well in practice but may not converge to the true value of the optimiser. On the other hand, solutions provided in the second group (Chevalier and Ginsbourger, 2013) derive near-analytic expressions in the sense that they contain terms such as low-dimensional multivariate normal cumulative density functions that cannot be computed exactly but for which high-quality estimators exist (Genz, 1992, 2004). As noted again by (Wilson et al., 2018b), these methods rarely scale to high dimensions. Finally, the third group comprises Monte Carlo (MC) methods (Osborne et al., 2009; Hennig and Schuler, 2012; Snoek et al.,

2012) which provide unbiased estimators to $\alpha(\cdot|\mathcal{D}_i)$. MC methods have been successfully used in the context of acquisition function maximisation to the extent that they form the backbone of modern Bayesian optimisation libraries such as BoTorch (Balandat et al., 2020).

As such, given their prevalence in present-day implementations, we restrict our attention to MC techniques and note three classes of widely-used optimisers. Zeroth-order procedures (Hazan, 2016; Gabillon et al., 2020), such as evolutionary algorithms (van Rijn et al., 2016; Blank and Deb, 2020), only use function value information for determining the maximum of the acquisition. First-order methods (Kingma and Ba, 2015; Bottou and Bousquet, 2007), on the other hand, utilise gradient information during the ascent step, while second-order methods exploit (approximations to) Hessians (Byrd et al., 1995; Zhu et al., 1997; Boyd and Vandenberghe, 2004; Tutunov et al., 2015, 2019) in their update. During the implementation of first and second-order optimisers, one realises the need for differentiating through an MC estimator with respect to the parameters of the generative distribution $\mathcal{P}(\cdot)$. As described in (Wilson et al., 2018b), this can be achieved through reparameterisation in two steps: 1) reparameterising samples from $\mathcal{P}(\cdot)$ as draws from a simpler distribution $\hat{\mathcal{P}}(\cdot)$, and 2) interchanging integration and differentiation by exploiting sample-path derivatives. After reparameterisation, the designer faces two implementation choices which we refer to as ERM-BO and FSM-BO akin to the distinction between empirical risk minimisation (Gonen and Shalev-Shwartz, 2017) and finite sum (Schmidt et al., 2017) optimisation forms[1].

In an ERM-BO construction, samples from $\hat{\mathcal{P}}(\cdot)$ are acquired at every iteration of the optimisation algorithm as needed. In contrast, in an FSM-BO setting, all samples from $\hat{\mathcal{P}}(\cdot)$ are obtained upfront and mini-batched during gradient computations. Due to memory consideration, especially in high-dimensional scenarios, the ERM-BO version has been mostly preferred and studied in the literature (Knudde et al., 2017; Balandat et al., 2020).

In this paper however, we are interested in both views and desire to shed light on best practices when optimising acquisition functions. To accomplish such a goal, we carefully probe both settings and realise that an FSM-BO implementation enables a novel connection to a compositional (nested expectation) formulation that sanctions new compositional solvers not previously attempted. Next, we derive such a connection, present memory-efficient optimisation algorithms for FSM-BO, and demonstrate empirical gains in large-scale experiments.

3. Acquisition Function Maximisation

The first step in investigating different implementations of BO is to derive relevant reparameterised forms of the acquisition functions in Section 2.2. When reparameterising one reinterprets samples $\mathbf{y}_k \sim \mathcal{P}(\mathbf{y};\boldsymbol{\theta})$ as a deterministic map $\boldsymbol{\lambda}_{\boldsymbol{\theta}}(\cdot)$ of a simpler random variable $\mathbf{z}_k \sim \hat{\mathcal{P}}(\mathbf{z})$, that is $\mathbf{y} = \lambda_{\boldsymbol{\theta}}(\mathbf{z})$. Under these conditions, the expectation of some loss $\mathcal{L}(\cdot)$ under $\mathbf{y}$ can be rewritten in terms of $\hat{\mathcal{P}}(\mathbf{z})$ as $\mathbb{E}_{\mathbf{y}\sim\mathcal{P}(y;\boldsymbol{\theta})}[\mathcal{L}(\mathbf{y})] = \mathbb{E}_{\mathbf{z}\sim\hat{\mathcal{P}}(\mathbf{z})}[\mathcal{L}(\lambda_{\boldsymbol{\theta}}(\mathbf{z}))]$ allow-

1. Of course, an empirical risk and a finite sum formulation become equivalent as samples grow large. In reality, infinite samples cannot possibly be acquired hence our two-class categorisation.

ing us, under further technical conditions (Wilson et al., 2018b), to push gradients inside expectations when needed.

Before diving into ascent direction computation, we first present reparameterised acquisition formulations as derived in (Wilson et al., 2018b). First, we realise that all batched acquisition functions in Section 2.2 involve an expectation over the GP's predictive posterior $\mathbf{f}(\mathbf{x}_{1:q})|\mathcal{D}_i, \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}), \boldsymbol{\Sigma}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}))$. Second, we recall that if a random variable is Gaussian distributed, one can reparameterise by choosing $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and then applying $\boldsymbol{\lambda}_{\boldsymbol{\theta}}(\mathbf{z}) = \boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}$ with $\mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{L}_i^{\mathsf{T}}(\mathbf{x}_{1:q};\boldsymbol{\theta}) = \boldsymbol{\Sigma}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})$. Using such a deterministic transformation $\boldsymbol{\lambda}_{\boldsymbol{\theta}}(\mathbf{z})$, the original random variable's distribution remains unchanged indicating a mean $\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})$ and covariance $\boldsymbol{\Sigma}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})$. Now, we can easily replace $\boldsymbol{\lambda}_{\boldsymbol{\theta}}(\mathbf{z})$ in each of the expected improvement, simple regret, upper confidence bound, and entropy search acquisitions leading us to the following batch-reparameterised formulations:

$$\alpha_{\text{rq-EI}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\max_{j\in 1:q}\left\{\text{ReLU}\left(\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z} - f(\mathbf{x}_i^+)\mathbf{1}_q\right)\right\}\right], \quad (4)$$

$$\alpha_{\text{rq-SR}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\max_{j\in 1:q}\left\{\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}\right\}\right], \quad (5)$$

$$\alpha_{\text{rq-UCB}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\max_{j\in 1:q}\left\{\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \sqrt{\beta\pi/2}|\mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}|\right\}\right]. \quad (6)$$

When it comes to probability of improvement, the direct insertion of $\lambda_{\boldsymbol{\theta}}(\mathbf{z})$ into Equation 3 is difficult due to the discrete nature of the utility measure that violates differentiablity assumptions in reparameterisation (Jang et al., 2017). To overcome this issue, we follow (Wilson et al., 2018b) and adopt the concrete (continuous to discrete) approximation to replace the discontinuous mapping (Maddison et al., 2017) such that transformed and original variables are close in distribution. Sticking to the formulation presented (Wilson et al., 2018b), we loosen the indicator part of $\alpha_{\text{q-PI}}(\cdot)$ from Equation 3 and write:

$$\max_{j\in 1:q}\left\{\mathbb{1}\{\mathbf{f}(\mathbf{x}_{1:q}) - f(\mathbf{x}_i^+)\mathbf{1}_q\}\right\} \approx \max_{j\in 1:q}\left\{\text{Sig}\left(\frac{\mathbf{f}(\mathbf{x}_{1:q}) - f(\mathbf{x}_i^+)\mathbf{1}_q}{\tau}\right)\right\},$$

where $\text{Sig}(\cdot)$ is executed component-wise and denotes the sigmoid function with $\tau \in \mathbb{R}_+$ representing its temperature parameter that yields an exact approximation as $\tau \to 0$. Given the approximation above and using a multivariate standard normal (instead of a uniform, see (Maddison et al., 2017)) as $\hat{\mathcal{P}}(\mathbf{z})$, we derive the following reparameterised form for the probability of improvement acquisition:

$$\alpha_{\text{rq-PI}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\max_{j\in 1:q}\left\{\text{Sig}\left(\frac{\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z} - f(\mathbf{x}_i^+)\mathbf{1}_q}{\tau}\right)\right\}\right]. \quad (7)$$

Finally, for the entropy search acquisition function, the above reparametrisation trick should be applied twice: for the outer posterior distribution $\mathbf{f}(\mathbf{x}_{1:q})|\mathcal{D}_i, \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}), \boldsymbol{\Sigma}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}))$ and for the inner posterior distribution $\mathbf{f}\left(\mathbf{x}_{1:u}^{(\text{g})}\right)|\mathcal{D}_i \cup$

$\{\mathbf{x}_{1:q}; \mathbf{f}(\mathbf{x}_{1:q})\}, \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_i^{(\mathrm{g})}(\mathbf{x}_{1:u}^{(\mathrm{g})}; \mathbf{f}(\mathbf{x}_{1:q}), \boldsymbol{\theta}), \boldsymbol{\Sigma}_i^{(\mathrm{g})}(\mathbf{x}_{1:u}^{(\mathrm{g})}; \boldsymbol{\theta}))$ with:

$$\boldsymbol{\mu}_i^{(\mathrm{g})}(\mathbf{x}_{1:u}^{(\mathrm{g})}; \mathbf{f}(\mathbf{x}_{1:q}), \boldsymbol{\theta}) = \underbrace{\mathbf{K}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}_{1:u}^{(\mathrm{g})}, \mathcal{D}_i \cup \mathbf{x}_{1:q})}_{\mathbf{K}_{\boldsymbol{\theta}}^{(\mathrm{g}),(i)}} \begin{bmatrix} \mathbf{K}_{\boldsymbol{\theta}}^{(i)} + \sigma^2\mathbf{I} & \mathbf{K}_{\boldsymbol{\theta}}^{(i)}(\mathcal{D}_i, \mathbf{x}_{1:q}) \\ \mathbf{K}_{\boldsymbol{\theta}}^{(i),\mathsf{T}}(\mathcal{D}_i, \mathbf{x}_{1:q}) & \mathbf{K}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}_{1:q}, \mathbf{x}_{1:q}) \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{y}_{1:n_i} \\ \mathbf{f}(\mathbf{x}_{1:q}) \end{bmatrix},$$

$$\boldsymbol{\Sigma}_i^{(\mathrm{g})}(\mathbf{x}_{1:u}^{(\mathrm{g})}; \boldsymbol{\theta}) = \mathbf{K}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}_{1:u}^{(\mathrm{g})}, \mathbf{x}_{1:u}^{(\mathrm{g})}) - \mathbf{K}_{\boldsymbol{\theta}}^{(\mathrm{g}),(i)} \begin{bmatrix} \mathbf{K}_{\boldsymbol{\theta}}^{(i)} + \sigma^2\mathbf{I} & \mathbf{K}_{\boldsymbol{\theta}}^{(i)}(\mathcal{D}_i, \mathbf{x}_{1:q}) \\ \mathbf{K}_{\boldsymbol{\theta}}^{(i),\mathsf{T}}(\mathcal{D}_i, \mathbf{x}_{1:q}) & \mathbf{K}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}_{1:q}, \mathbf{x}_{1:q}) \end{bmatrix}^{-1} \mathbf{K}_{\boldsymbol{\theta}}^{(\mathrm{g}),(i),\mathsf{T}}.$$

Due to the nested expectation structure of the entropy search acquisition function $\alpha_{\text{q-ES}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$, in order to rewrite it in the reparametrised form we consider two deterministic transformations $\boldsymbol{\lambda}_{\boldsymbol{\theta}}^{(i)}(\mathbf{z}) = \boldsymbol{\mu}_i(\mathbf{x}_{1:q}; \boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q}; \boldsymbol{\theta})\mathbf{z}$ with Cholesky decomposition $\mathbf{L}_i(\mathbf{x}_{1:q}; \boldsymbol{\theta})\mathbf{L}_i^{\mathsf{T}}(\mathbf{x}_{1:q}; \boldsymbol{\theta}) = \boldsymbol{\Sigma}_i(\mathbf{x}_{1:q}; \boldsymbol{\theta})$ and $\boldsymbol{\varrho}_{\boldsymbol{\theta}}^{(i)}(\boldsymbol{\omega}) = \boldsymbol{\mu}_i^{(\mathrm{g})}(\mathbf{x}_{1:u}^{(\mathrm{g})}; \mathbf{f}(\mathbf{x}_{1:q}), \boldsymbol{\theta}) + \mathbf{L}_i^{(\mathrm{g})}(\mathbf{x}_{1:u}^{(\mathrm{g})}; \boldsymbol{\theta})\boldsymbol{\omega}$ with Cholesky decomposition $\mathbf{L}_i^{(\mathrm{g})}(\mathbf{x}_{1:u}; \boldsymbol{\theta})\mathbf{L}_i^{(\mathrm{g}),\mathsf{T}}(\mathbf{x}_{1:u}; \boldsymbol{\theta}) = \boldsymbol{\Sigma}_i^{(\mathrm{g})}(\mathbf{x}_{1:u}^{(\mathrm{g})}; \boldsymbol{\theta})$. Choosing random vectors $\mathbf{z} \sim \mathcal{N}(\mathbf{0}_q, \mathbf{I}_{q\times q})$ and $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}_u, \mathbf{I}_{u\times u})$[2] in the above transformations $\boldsymbol{\lambda}_{\boldsymbol{\theta}}^{(i)}(\cdot)$, $\boldsymbol{\varrho}_{\boldsymbol{\theta}}^{(i)}(\cdot)$ respectively, and applying the following smooth approximation for the step function:

$$\mathbb{1}\{\mathbf{f}(\mathbf{x}_{1:u}^{(\mathrm{g})}) - \max_{j\in 1:u} f(\mathbf{x}_j^{(\mathrm{g})})\mathbf{1}_u\} \approx \mathrm{SM}\left(\frac{\boldsymbol{\mu}_i^{(\mathrm{g})}(\mathbf{x}_{1:u}^{(\mathrm{g})}; \boldsymbol{\lambda}_{\boldsymbol{\theta}}^{(i)}(\mathbf{z}), \boldsymbol{\theta}) + \mathbf{L}_i^{(\mathrm{g})}(\mathbf{x}_{1:u}^{(\mathrm{g})}; \boldsymbol{\theta})\boldsymbol{\omega}}{\tau}\right)$$

with $\mathrm{SM}(\cdot)$ being a softmax function and $\tau \in \mathbb{R}_+$ a temperature parameter controlling the approximation accuracy, we arrive to the batch-reparametrised form for the entropy search acquisition function:

$$\alpha_{\text{rq-ES}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = -\mathbb{E}_{\mathbf{z}}\left[\mathrm{H}\left[\mathbb{E}_{\boldsymbol{\omega}}\left[\mathrm{SM}\left(\frac{\boldsymbol{\mu}_i^{(\mathrm{g})}(\mathbf{x}_{1:u}^{(\mathrm{g})}; \boldsymbol{\lambda}_{\boldsymbol{\theta}}^{(i)}(\mathbf{z}), \boldsymbol{\theta}) + \mathbf{L}_i^{(\mathrm{g})}(\mathbf{x}_{1:u}^{(\mathrm{g})}; \boldsymbol{\theta})\boldsymbol{\omega}}{\tau}\right)\right]\right]\right] \tag{8}$$

Given reparameterised acquisitions, we now turn our attention to ERM- and FSM-BO depicting both implementations and presenting novel compositional procedures that are sample and memory efficient.

3.1 ERM-BO using Stochastic Optimisation

Mainstream implementations of BO cast the inner optimisation problem (line 4 in Algorithm 1) in an empirical risk form $\max_{\mathbf{x}_{1:q}} \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[\mathcal{L}(\mathbf{x}_{1:q}; \mathbf{z})]$ with $\mathcal{L}(\mathbf{x}_{1:q}; \mathbf{z})$ dependent on the acquisition's type, e.g., $\max_{j\in 1:q}\{\boldsymbol{\mu}_i(\mathbf{x}_{1:q}; \boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q}; \boldsymbol{\theta})\mathbf{z}\}$ in the simple regret case. Such a connection enables tractable optimisation through the usage of numerous zero, first, and second-order optimisers developed in the literature (van Rijn et al., 2016; Bottou et al., 2018; Sun et al., 2019). Since such an implementation is fairly common in practice (Knudde et al., 2017; Balandat et al., 2020) and not to burden the reader with unnecessary notation, we defer the exact details of the optimisers used in our experiments to appendices B, C and D. Here, we briefly mention that we surveyed three zero-order optimisers, eight first-order algorithms and one well-known approximate second-order method.

2. Here $\mathbf{0}_a$ and $\mathbf{I}_{a\times a}$ denote $a-$dimensional vector of zeros and a by a identity matrix respectively

Zeroth-Order Optimisers in ERM-BO: Zeroth-order methods optimise objectives based on function value information and have emerged from many different fields. In the online learning literature, for example, development of zeroth-order methods is mostly theoretical aiming at efficient and optimal regret guarantees (Hazan, 2016; Lattimore and Szepesvári, 2020; Gabillon et al., 2020) – a challenging topic in itself. Empirical successes of such procedures have been achieved in isolated instances (Shalev-Shwartz and Singer, 2007; Viappiani and Boutilier, 2009; Contal et al., 2013; Chen et al., 2013; Bresler et al., 2016; Ariu et al., 2020; Hallak et al., 2020). Mainstream implementation of zeroth-order optimisers for BO, however, are of the evolutionary type updating generations of $\mathbf{x}$ through a process of adaptation and mutation (Bentley, 1999).

In our experiments, we used three such strategies, varying from simple to advanced. The most simple among the three was random search (RS) which acts as a low-memory, low-compute baseline. The second, corresponds to a covariance matrix evolutionary strategy (CMA-ES) that generates updates of the mean and covariance of a multivariate normal based on average sample ranks gathered from function value information (Hansen and Ostermeier, 1996; van Rijn et al., 2016). The third and final algorithm was differential evolution (DE) which is widely considered a go-to in evolutionary optimisation (Price, 1996; Baioletti et al., 2020), e.g., NSGA I and II (Deb et al., 2002) as implemented in (Blank and Deb, 2020). DE continuously updates a population of candidate solutions via component-wise mutation performing selection according to a mutation probability p_{mutation}. More details are available in Appendix B.

First-Order Optimisers in ERM-BO: First-order optimisation techniques rely on gradient information to compute updates of $\mathbf{x}$. They are iterative in nature running for a total of T iterations and executing a variant of the following rule at each step[3]:

$$\mathbf{x}_{1:q,t+1} = \delta_t \mathbf{x}_{1:q,t} + \eta_t \underbrace{\frac{\boldsymbol{\phi}_t^{(1)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}, \left\{\beta_k^{(1)}\right\}_{k=0}^{t}\right)}{\boldsymbol{\phi}_t^{(2)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}^2, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}^2, \left\{\beta_k^{(2)}\right\}_{k=0}^{t}, \epsilon\right)}}_{\text{(General update)},} \tag{9}$$

where δ_t is a weighting that depends on the type of algorithm used, η_t is a typically decaying learning rate, $\boldsymbol{\phi}_t^{(1)}(\cdot)$ and $\boldsymbol{\phi}_t^{(2)}(\cdot)$ are history-dependent mappings that vary between algorithms with the ratio executed element-wise, $\left\{\beta_k^{(1)}\right\}_{k=0}^{t}$ and $\left\{\beta_k^{(2)}\right\}_{k=0}^{t}$ are history-weighting parameters, and ϵ a small positive constant used to avoid division by zero. Additionally, $\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}$ represent sub-sampled gradient estimators that are acquired using Monte-Carlo samples of $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. It is also worth noting that differentiating through the max operator that appears in all acquisitions can be performed either using sub-gradients or by propagating through the max value of the corresponding vector.

To elaborate our generalised form, we realise that one can easily recover Adam's (Kingma and Ba, 2015) update equation by setting $\delta_1 = \cdots = \delta_T = 1$, $\beta_1^{(1)} = \cdots = \beta_T^{(1)} = \beta_1$,

3. For simplicity in the notation for acquisition functions $\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)$ we drop the subscript with the type.

$\beta_1^{(2)} = \cdots = \beta_T^{(2)} = \beta_2$, and $\boldsymbol{\phi}_t^{(1)}$ and $\boldsymbol{\phi}_t^{(2)}$ to:

$$\boldsymbol{\phi}_t^{(1)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}, \beta_1\right) = \frac{1-\beta_1}{1-\beta_1^t}\sum_{k=0}^{t}\beta_1^k\overline{\nabla\alpha(\mathbf{x}_{1:q,t-k}|\mathcal{D}_i)},$$

$$\boldsymbol{\phi}_t^{(2)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}^2, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}^2, \beta_2, \epsilon\right) = \sqrt{\frac{1-\beta_2}{1-\beta_2^t}\sum_{k=0}^{t}\beta_2^k\overline{\nabla\alpha(\mathbf{x}_{1:q,t-k}|\mathcal{D}_i)}^2} + \epsilon.$$

Of course, Adam is yet another special case of Equation 9. For notational convenience, we defer the detailed derivations of other optimisers including SGA (Robbins and Monro, 1951), RProp (Riedmiller and Braun, 1993), RMSprop (Hinton et al., 2012), AdamW (Loshchilov and Hutter, 2019), AdamOS (an Adam adaptation with new hyperparameters that we propose in this paper), AdaGrad (Duchi et al., 2011), and AdaDelta (Zeiler, 2012) to Appendix C.

Second-Order Optimisers in ERM-BO: Along with gradient information, second-order optimisers utilise Hessian (sometimes the Fisher matrix instead (Amari, 1997; Pascanu and Bengio, 2014)) information for maximising objective functions. The general iterative update equation for a second-order method is given by:

$$\mathbf{x}_{1:q,t+1} = \mathbf{x}_{1:q,t} - \eta_t\left[\overline{\nabla^2\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}\right]^{-1}\overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)} \qquad \text{(General update)},$$

where $\overline{\nabla^2\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}$ is an approximation to the true Hessian $\nabla^2\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)$ as evaluated on the current iterate $\mathbf{x}_{1:q,t}$, and $\overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}$ denotes a gradient estimate that is acquired through Monte Carlo samples as described above. It is worth emphasising the need for the approximation $\overline{\nabla^2\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}$ to $\nabla^2\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)$ due to the large size of the true Hessian matrix ($\mathbb{R}^{dq\times dq}$ in our case), as well as the necessity to compute an inverse at every iteration of the update. Numerous approximation techniques with varying degrees of accuracy have been proposed in the literature (Shanno, 1970; Mokhtari and Ribeiro, 2014, 2015; Byrd et al., 2016). In this paper, however, we make use of L-BFGS (Zhu et al., 1997) due to its widespread adoption in both GPs and BO (Rasmussen and Williams, 2006; Balandat et al., 2020). Exact details and pseudocode for L-BFGS are comprehensively presented in Appendix D.

3.2 FSM-BO & Connections to Compositional Optimisation

Rather than considering the problem of acquisition function maximisation as an instance of empirical risk minimisation, we can follow an alternative route and focus on finite sum approximations. To do so, imagine we acquire M independent and identically-distributed samples from $\mathcal{N}(\mathbf{0}, \mathbf{I})$, $\{\mathbf{z}_m\}_{m=1}^M$, upfront before the beginning of any acquisition function optimisation step. Assuming fixed samples for now, we can write finite-sum forms of the reparameterised acquisition functions (those from Section 3) using a simple Monte Carlo

estimator as follows:

$$\alpha^{(\text{FSM})}_{\text{rq-EI}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \frac{1}{M}\sum_{m=1}^{M}\max_{j\in 1:q}\left\{\text{ReLU}\left(\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m - f(\mathbf{x}_i^+)\mathbf{1}_q\right)\right\}, \quad (10)$$

$$\alpha^{(\text{FSM})}_{\text{rq-SR}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \frac{1}{M}\sum_{m=1}^{M}\max_{j\in 1:q}\left\{\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m\right\}, \quad (11)$$

$$\alpha^{(\text{FSM})}_{\text{rq-UCB}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \frac{1}{M}\sum_{m=1}^{M}\max_{j\in 1:q}\left\{\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \sqrt{\beta\pi/2}|\mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m|\right\}, \quad (12)$$

$$\alpha^{(\text{FSM})}_{\text{rq-PI}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \frac{1}{M}\sum_{m=1}^{M}\max_{j\in 1:q}\left\{\text{Sig}\left(\frac{\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m - f(\mathbf{x}_i^+)\mathbf{1}_q}{\tau}\right)\right\}. \quad (13)$$

As for the entropy search acquisition function given in Equation 8, due to its nested expectation form simply replacing both expectations with their corresponding MC estimates leads to a biased estimate of $\alpha_{\text{rq-ES}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$. Instead, we use a collection of independent random vectors $\{\mathbf{z}_m\}_{m=1}^M$ sampled from $\mathcal{N}(\mathbf{0}_q, \mathbf{I}_{q\times q})$ to construct a Monte Carlo estimate for the outer expectation and write:

$$\alpha^{(\text{FSM})}_{\text{rq-ES}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = -\frac{1}{M}\sum_{m=1}^{M}\text{H}\left[\mathbb{E}_{\boldsymbol{\omega}}\left[\text{SM}\left(\frac{\boldsymbol{\mu}_i^{(\text{g})}(\mathbf{x}_{1:u}^{(\text{g})};\boldsymbol{\lambda}_{\boldsymbol{\theta}}^{(i)}(\mathbf{z}_m),\boldsymbol{\theta}) + \mathbf{L}_i^{(\text{g})}(\mathbf{x}_{1:u}^{(\text{g})};\boldsymbol{\theta})\boldsymbol{\omega}}{\tau}\right)\right]\right]. \quad (14)$$

At this stage, we can execute any off-the-shelf optimiser to maximise the finite sum version of the acquisitions, i.e., Equations 9 to 12. Contrary to ERM-BO which samples new $\mathbf{z}$ vectors at each iteration, the FSM formulation fixes $\{\mathbf{z}_m\}_{m=1}^M$ and mini-batches from this fixed pool to compute necessary gradients and Hessian estimates for first and second-order methods respectively. At first sight, one might believe that ERM and FSM are the only plausible approximation forms of acquisition functions in BO. Upon further investigation, however, we realise that finite sum myopic acquisitions adhere to yet another configuration that is still to be (well-) explored in the literature. Not only does this new form allow for novel solvers not yet attempted in acquisition function maximisation, but also seems to significantly outperform both ERM-and FSM-BO in practice, cf. Section 4.

3.2.1 Comp-BO: A Compositional Form for Myopic Acquisition Functions

Recently, the optimisation community has displayed an increased interest in developing specialised algorithms for compositional (or nested) objectives due to their prevalence in subfields of machine learning, e.g., in model-agnostic-meta-learning (Tutunov et al., 2020), semi-implicit variational inference (Yin and Zhou, 2018), dynamic programming and reinforcement learning (Wang et al., 2017b). In each of these examples, compositional solvers have demonstrated efficiency advantages when compared to other algorithms which begs the question as to whether these improvements can be ported to Bayesian optimisation.

From a definition perspective, compositional problems involve maximising an objective that consists of a non-linear nesting of expectations of random variables:

$$\max_{\mathbf{x}_{1:q}} \mathbb{E}_\nu[f_\nu(\mathbb{E}_\omega[\mathbf{g}_\omega(\mathbf{x}_{1:q})])], \tag{15}$$

where ν and ω are (not necessarily iid) random variables sampled from $\mathcal{P}_\nu(\cdot)$ and $\mathcal{P}_\omega(\cdot)$ respectively (Wang and Liu, 2016), $f_\nu(\cdot)$ a stochastic function, and $\mathbf{g}_\omega(\cdot)$ is a stochastic map. Hence to benefit from such techniques, our first step consists of transforming the finite-sum versions of the acquisition functions above into a composed (or nested) form that abides by the structure in Equation 15. Interestingly, this can easily be achieved if we look at the problem from a matrix-vector perspective. To illustrate, consider $\alpha^{(\text{FSM})}_{\text{rq-EI}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$ and define $\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q})$ to be a $q \times M$ matrix such that the ω^{th} column is set to $\mathbf{v}^{(\text{EI})}_\omega = \text{ReLU}\left(\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_\omega - f(\mathbf{x}_i^+)\mathbf{1}_q\right) \in \mathbb{R}^q$ with ω uniformly distributed in $[1:M]$, and set the other columns to $\mathbf{0}_q$:

$$\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q}) = [\mathbf{0}_q, \ldots, \mathbf{v}^{(\text{EI})}_\omega, \ldots, \mathbf{0}_q].$$

Clearly, if we consider the expectation with respect to $\omega \sim \text{Uniform}([1:M])$, we arrive at the following matrix that sums all information across $\{\mathbf{z}_m\}_{m=1}^M$:

$$\mathbb{E}_\omega[\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q})] = \frac{1}{M}[\mathbf{v}^{(\text{EI})}_1, \ldots, \mathbf{v}^{(\text{EI})}_m, \ldots, \mathbf{v}^{(\text{EI})}_M],$$

with $\mathbf{v}^{(\text{EI})}_m = \text{ReLU}\left(\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m - f(\mathbf{x}_i^+)\mathbf{1}_q\right)$ being a q-dimensional vector. To attain the original form of $\alpha^{(\text{FSM})}_{\text{rq-EI}}(\cdot)$, we further introduce a deterministic outer function $f^{(\text{EI})} : \mathbb{R}^{q\times M} \to \mathbb{R}$ as follows:

$$\alpha^{(\text{Comp})}_{\text{rq-EI}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = f^{(\text{EI})}(\mathbb{E}_\omega[\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q})]) = \frac{1}{M}\sum_{m=1}^{M}\max_{j\in 1:q}\mathbf{v}^{(\text{EI})}_m = \alpha^{(\text{FSM})}_{\text{rq-EI}}.$$

Importantly, the above shows that a finite-sum expected improvement acquisition can be written in a compositional (nested) form with $\alpha^{(\text{FSM})}_{\text{rq-EI}} = f(\mathbb{E}_\omega[\mathbf{g}_\omega(\mathbf{x})])$. In our derivations, we have considered a deterministic outer function $f(\cdot)$ leading us to a special case of Equation 15 where $P_\nu(\cdot)$ is Dirac. Such a consideration is mostly due to the fact that q is typically in the order of tens or hundreds in BO allowing for exact outer summations. In the case of large batch sizes, our formulation can easily be generalised to a stochastic setting exactly matching a compositional form as shown in Appendix A.

Following the same strategy above, we can now reformulate all other acquisition functions as instances of compositional optimisation. Next, we list these results and refer the reader to Appendix A for a detailed exposition. First, we choose $\omega \sim \text{Uniform}([1:M])$ and then consider the following inner matrix mappings:

$$\begin{aligned}
\mathbf{g}^{(\text{PI})}_\omega(\mathbf{x}_{1:q}) &= [\mathbf{0}_q, \ldots, \mathbf{v}^{(\text{PI})}_\omega, \ldots, \mathbf{0}_q] \in \mathbb{R}^{q\times M},\\
\mathbf{g}^{(\text{SR})}_\omega(\mathbf{x}_{1:q}) &= [\mathbf{0}_q, \ldots, \mathbf{v}^{(\text{SR})}_\omega, \ldots, \mathbf{0}_q] \in \mathbb{R}^{q\times M}\\
\mathbf{g}^{(\text{UCB})}_\omega(\mathbf{x}_{1:q}) &= [\mathbf{0}_q, \ldots, \mathbf{v}^{(\text{UCB})}_\omega, \ldots, \mathbf{0}_q] \in \mathbb{R}^{q\times M}
\end{aligned}$$

and for the entropy search acquisition $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}_u, \mathbf{I}_{u\times u})$:

$$\mathbf{g}^{(\mathrm{ES})}_{\boldsymbol{\omega}}(\mathbf{x}_{1:q}) = [\mathbf{v}^{(\mathrm{ES})}_{1,\boldsymbol{\omega}}, \mathbf{v}^{(\mathrm{ES})}_{2,\boldsymbol{\omega}}, \ldots, \mathbf{v}^{(\mathrm{ES})}_{M,\boldsymbol{\omega}}] \in \mathbb{R}^{u\times M}.$$

where the $q-$dimensional vectors $\mathbf{v}^{(\mathrm{PI})}_m, \mathbf{v}^{(\mathrm{SR})}_m$, and $\mathbf{v}^{(\mathrm{UCB})}_m$ are defined as (for $m \in [1:M]$):

$$\begin{aligned}
\mathbf{v}^{(\mathrm{PI})}_m &= \frac{1}{\tau}\left[\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m - f(\mathbf{x}^+_i)\mathbf{1}_q\right], \\
\mathbf{v}^{(\mathrm{SR})}_m &= \boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m, \\
\mathbf{v}^{(\mathrm{UCB})}_m &= \boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \sqrt{\beta\pi/2}\,\left|\mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m\right|.
\end{aligned}$$

and $u-$dimensional vector $\mathbf{v}^{(\mathrm{ES})}_{m,\boldsymbol{\omega}}$ is defined as (for $m \in [1:M]$ and $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}_u, \mathbf{I}_{u\times u})$):

$$\mathbf{v}^{(\mathrm{ES})}_{m,\boldsymbol{\omega}} = \mathrm{SM}\left(\frac{\boldsymbol{\mu}^{(\mathrm{g})}_i(\mathbf{x}^{(\mathrm{g})}_{1:u}; \boldsymbol{\lambda}^{(i)}_{\boldsymbol{\theta}}(\mathbf{z}_m), \boldsymbol{\theta}) + \mathbf{L}^{(\mathrm{g})}_i(\mathbf{x}^{(\mathrm{g})}_{1:u};\boldsymbol{\theta})\boldsymbol{\omega}}{\tau}\right).$$

Now, properly selecting the outer functions $f^{(\mathrm{PI})}(\cdot), f^{(\mathrm{SR})}(\cdot)$, and $f^{(\mathrm{UCB})}(\cdot)$ gives us:

$$\alpha^{(\mathrm{Comp})}_{\text{rq-PI}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = f^{(\mathrm{PI})}(\mathbb{E}_\omega[\mathbf{g}^{(\mathrm{PI})}_\omega(\mathbf{x}_{1:q})]) = \frac{1}{M}\sum_{m=1}^{M}\max_{j\in 1:q}\left\{\mathrm{Sig}\left(\mathbf{v}^{(\mathrm{PI})}_m\right)\right\} = \alpha^{(\mathrm{FSM})}_{\text{rq-PI}}(\mathbf{x}_{1:q}|\mathcal{D}_i),$$

$$\alpha^{(\mathrm{Comp})}_{\text{rq-SR}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = f^{(\mathrm{SR})}(\mathbb{E}_\omega[\mathbf{g}^{(\mathrm{SR})}_\omega(\mathbf{x}_{1:q})]) = \frac{1}{M}\sum_{m=1}^{M}\max_{j\in 1:q}\left\{\mathbf{v}^{(\mathrm{SR})}_m\right\} = \alpha^{(\mathrm{FSM})}_{\text{rq-SR}}(\mathbf{x}_{1:q}|\mathcal{D}_i),$$

$$\alpha^{(\mathrm{Comp})}_{\text{rq-UCB}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = f^{(\mathrm{UCB})}(\mathbb{E}_\omega[\mathbf{g}^{(\mathrm{UCB})}_\omega(\mathbf{x}_{1:q})]) = \frac{1}{M}\sum_{m=1}^{M}\max_{j\in 1:q}\left\{\mathbf{v}^{(\mathrm{UCB})}_m\right\} = \alpha^{(\mathrm{FSM})}_{\text{rq-UCB}}(\mathbf{x}_{1:q}|\mathcal{D}_i).$$

Finally, properly selecting the stochastic outer function $f^{(\mathrm{ES})}_\nu(\cdot)$ with $\nu \sim \mathrm{Uniform}([1:M])$ gives us:

$$\begin{aligned}
\alpha^{(\mathrm{Comp})}_{\text{rq-ES}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_\nu\left[f^{(\mathrm{ES})}_\nu\left(\mathbb{E}_{\boldsymbol{\omega}}\left[\mathbf{g}^{(\mathrm{ES})}_{\boldsymbol{\omega}}(\mathbf{x}_{1:q})\right]\right)\right] = \\
- \frac{1}{M}\sum_{m=1}^{M}\mathrm{H}\left[\mathbb{E}_{\boldsymbol{\omega}}\left[\mathbf{v}^{(\mathrm{ES})}_{m,\boldsymbol{\omega}}\right]\right] = \alpha^{(\mathrm{FSM})}_{\text{rq-ES}}(\mathbf{x}_{1:q}|\mathcal{D}_i).
\end{aligned}$$

Clearly, the results above recover the formulations of the acquisition functions given in Equations 11 - 13 while making them amenable to compositional solvers, a new class of optimisers not yet well-studied in the Bayesian optimisation literature. We detail such compositional optimisers next.

Zeroth-Order Compositional Solvers for BO: Of course, the compositional forms presented above are still suitable for zeroth-order methods (Section 3.1). The distinguishing factor from non-compositional forms is the evaluation process of nested objectives which

requires careful consideration. In the case of $\alpha^{(\text{Comp})}_{\text{rq-EI}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$, for example, the inner expectation $\mathbb{E}_\omega[\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x})]$ in Equation 15 can be evaluated using a Monte Carlo approximation:

$$\mathbb{E}_\omega[\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q})] \approx \frac{1}{K}\sum_{m=1}^{K}\mathbf{g}^{(\text{EI})}_{\omega_m}(\mathbf{x}_{1:q}), \quad \text{with } K < M \text{ being a mini-batch of } \{\mathbf{z}_m\}_{m=1}^{M}.$$

Furthermore, the outer function is estimated by $f^{(\text{EI})}(\mathbb{E}_\omega[\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q})]) \approx f^{(\text{EI})}\left(\frac{1}{K}\sum_{m=1}^{K}\mathbf{g}^{(\text{EI})}_{\omega_m}(\mathbf{x}_{1:q})\right)$, where such an estimate asymptotically ($K \to \infty$) converges to the true expectation due to the continuity of $f^{(\text{EI})}(\cdot)$:

$$\lim_{K\to\infty} f^{(\text{EI})}\left(\frac{1}{K}\sum_{m=1}^{K}\mathbf{g}^{(\text{EI})}_{\omega_m}(\mathbf{x}_{1:q})\right) = f^{(\text{EI})}(\mathbb{E}_\omega[\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q})]).$$

Clearly, this observation allows us to straightforwardly apply any of the three considered zero-order methods (CMA-ES, DE, and RS) for determining updates of $\mathbf{x}_{1:q}$. Certainly, such Monte Carlo approximations are not distinctive for $\alpha^{(\text{Comp})}_{\text{rq-EI}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$, allowing us to follow the same scheme for $\alpha^{(\text{Comp})}_{\text{rq-PI}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$, $\alpha^{(\text{Comp})}_{\text{rq-SR}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$, and $\alpha^{(\text{Comp})}_{\text{rq-UCB}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$.

First-Order Compositional Solvers for BO: In contrast to zeroth-order compositional methods, where the only difference between them and their non-compositional counterparts is in the evaluation of the objective function, first-order compositional optimisers require more sophisticated techniques due to the difficulty associated in acquiring unbiased gradients of nested objectives. To elaborate, let us carry on with our running example and consider the gradient of $\alpha^{(\text{Comp})}_{\text{rq-EI}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = f^{(\text{EI})}(\mathbb{E}_\omega[\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q})])$. Using the chain rule, we can easily see that such a gradient involves a product of the Jacobian of $\mathbf{g}_\omega(\mathbf{x}_{1:q})$ with the gradient of $f^{(\text{EI})}(\cdot)$ that is to be evaluated around the inner mapping[4]:

$$\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}_{\text{rq-EI}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_\omega[\nabla_{\text{vec}(\mathbf{x}_{1:q})}\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q})]^\intercal \nabla_{\boldsymbol{\zeta}} f^{(\text{EI})}(\boldsymbol{\zeta})\mid_{\boldsymbol{\zeta}=\mathbb{E}_\omega[\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q})]},$$

where we use $\text{vec}(\mathbf{x}_{1:q}) \in \mathbb{R}^{dq}$ to denote an unrolled vector across all dimensions d and batch sizes q. When attempting to acquire an unbiased estimate of $\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}_{\text{rq-EI}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$, we realise that the first term can be approximated by simple Monte Carlo:

$$\mathbb{E}_\omega[\nabla_{\text{vec}(\mathbf{x}_{1:q})}\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q})] \approx \frac{1}{K_1}\sum_{m=1}^{K_1}\nabla_{\text{vec}(\mathbf{x}_{1:q})}\mathbf{g}^{(\text{EI})}_{\omega_m}(\mathbf{x}_{1:q}),$$

with $K_1 < M$ being a batch size. The second part, however, is tougher to estimate as it involves a gradient of a non-linear nesting of an expected value, i.e., $\nabla_{\boldsymbol{\zeta}} f^{(\text{EI})}(\boldsymbol{\zeta})\mid_{\boldsymbol{\zeta}=\mathbb{E}_\omega[\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q})]}$. To resolve this problem, in the compositional optimisation literature (Wang et al., 2017a; Tutunov et al., 2020), typically an auxiliary variable $\mathbf{u}$ is

4. Of course, a simple solution corresponds to a Nested Monte Carlo approach that approximates both inner and outer mappings with samples from ω and ν and then executes standard off-the-shelf algorithms. In our experiments, we make use of such a technique which we refer to as Adam-Nested (see Section 4) but realise that dedicated first-order compositional solvers tend to outperform such a scheme.

introduced and an exponentially-weighted average of $\boldsymbol{\zeta}$ is used, resulting in asymptotically-vanishing biases. To acquire such behaviour, not only do we need to update $\mathbf{x}_{1:q}$ but we also need to modify $\mathbf{u}$ and our estimation of $\boldsymbol{\zeta}$. As such, most compositional solvers execute three subroutines (main $\mathbf{x}_{1:q}$, auxiliary $\mathbf{u}$ and $\boldsymbol{\zeta}$) between iterations t and $t+1$ – the first to generate $\mathbf{x}_{1:q,t+1}$, the second for $\mathbf{u}_{t+1}$ and the third for $\boldsymbol{\zeta}_{t+1}$. Rather than presenting every subroutine for all utilised algorithms across all acquisition functions, here we keep the exposition general and provide a set of unifying update rules, deferring exact details to Appendix E. To that end, we introduce four history-dependent mappings $\boldsymbol{\phi}_t^{(1)}(\cdot)$, $\boldsymbol{\phi}_t^{(2)}(\cdot)$, $\boldsymbol{\phi}_t^{(3)}(\cdot)$ and $\boldsymbol{\phi}_t^{(4)}(\cdot)$. $\boldsymbol{\phi}_t^{(1)}(\cdot)$ and $\boldsymbol{\phi}_t^{(2)}(\cdot)$ act on sub-sampled gradient histories, and their corresponding squares, for updating $\mathbf{x}_{1:q,t}$ as follows:

Main variable update: (16)

$$\mathbf{x}_{1:q,t+1} = \mathbf{x}_{1:q,t} + \eta_t \frac{\boldsymbol{\phi}_t^{(1)}\left(\left\{\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}\right\}_{k=0}^{t}, \left\{\gamma_k^{(1)}\right\}_{k=0}^{t}\right)}{\boldsymbol{\phi}_t^{(2)}\left(\left\{\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}^2\right\}_{k=0}^{t}, \left\{\gamma_k^{(2)}\right\}_{k=0}^{t}, \epsilon\right)},$$

where η_t is a learning rate, $\{\gamma_k^{(1)}\}_{k=0}^t$ and $\{\gamma_k^{(2)}\}_{k=0}^t$ are history-dependent weightings that vary across algorithms. In Equation 16, we also use $\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}$ to define a compositional gradient estimate that can be written as:

$$\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)} = \left[\frac{1}{K_1}\sum_{m=1}^{K_1}\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\mathbf{g}_{\omega_m}^{(\mathrm{type})}(\mathbf{x}_{1:q,k})\right]^{\mathsf{T}}\nabla_{\boldsymbol{\zeta}}f^{(\mathrm{type})}(\boldsymbol{\zeta}_k), \quad (17)$$

with $\mathbf{g}_{\omega_m}^{(\mathrm{type})}$ and $f^{(\mathrm{type})}$ denoting the inner and outer mapping of a compositional formulation where type $\in \{\mathrm{EI}, \mathrm{PI}, \mathrm{SR}, \mathrm{UCB}\}$. With $\mathbf{x}_{1:q,t+1}$ computed, the next step is to update $\mathbf{u}_t$ and $\boldsymbol{\zeta}_t$ which can be achieved through $\boldsymbol{\phi}_t^{(3)}(\cdot)$ and $\boldsymbol{\phi}_t^{(4)}(\cdot)$ in the following manner:

$$\mathbf{u}_{t+1} = \boldsymbol{\phi}_{t+1}^{(3)}\left(\mathbf{x}_{1:q,0},\ldots,\mathbf{x}_{1:q,t+1},\{\beta_k\}_{k=0}^t\right), \quad (18)$$

$$\boldsymbol{\zeta}_{t+1} = \boldsymbol{\phi}_{t+1}^{(4)}\left(\overline{\mathbf{g}^{(\mathrm{type})}(\mathbf{u}_1)},\ldots,\overline{\mathbf{g}^{(\mathrm{type})}(\mathbf{u}_{t+1})},\{\beta_k\}_{k=0}^t,\boldsymbol{\zeta}_0,\mathbf{u}_0\right), \quad (19)$$

where $\{\beta_k\}_{k=0}^t$ is a set of free parameters[5], $\boldsymbol{u}_0$ and $\boldsymbol{\zeta}_0$ are initialisations that in turn depend on $\mathbf{x}_{1:q,0}$. Furthermore, in Equation 19 we used $\overline{\mathbf{g}^{(\mathrm{type})}(\cdot)}$ to represent a Monte Carlo estimate of the inner mapping, i.e.,

$$\overline{\mathbf{g}^{(\mathrm{type})}(\cdot)} = \frac{1}{K_2}\sum_{m=1}^{K_2}\mathbf{g}_{\omega_m}^{(\mathrm{type})}(\cdot),$$

where $K_2 < M$ is a batch size and type $\in \{\mathrm{EI}, \mathrm{PI}, \mathrm{SR}, \mathrm{UCB}\}$. As an illustrative example, we note that one can recover CAdam (Tutunov et al., 2020) by instantiating the above as

5. It is worth noting that in Appendix H we provide a complete set of all hyperparameters used across all 28 optimisers.

follows:

$$\boldsymbol{\phi}_t^{(1)}\left(\left\{\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}\right\}_{k=0}^{t},\left\{\gamma_k^{(1)}\right\}_{k=0}^{t}\right)=$$
$$\sum_{k=0}^{t}(1-\gamma_k^{[1]})\prod_{j=k+1}^{t}\gamma_j^{[1]}\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)},$$

$$\boldsymbol{\phi}_t^{(2)}\left(\left\{\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}^2\right\}_{k=0}^{t},\left\{\gamma_k^{(2)}\right\}_{k=0}^{t},\epsilon\right)=$$
$$\sqrt{\sum_{k=0}^{t}(1-\gamma_k^{[2]})\prod_{j=k+1}^{t}\gamma_j^{[2]}\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}^2}+\epsilon,$$

$$\boldsymbol{\phi}_t^{(3)}\left(\mathbf{x}_{1:q,0},\ldots,\mathbf{x}_{1:q,t},\{\beta_k\}_{k=0}^{t-1}\right)=(1-\beta_{t-1}^{-1})\mathbf{x}_{1:q,t-1}+\beta_{t-1}^{-1}\mathbf{x}_{1:q,t},$$

$$\boldsymbol{\phi}_t^{(4)}\left(\overline{\mathbf{g}^{(\text{type})}(\mathbf{u}_1)},\ldots,\overline{\mathbf{g}^{(\text{type})}(\mathbf{u}_t)},\{\beta_k\}_{k=0}^{t-1},\boldsymbol{\zeta}_0,\mathbf{u}_0\right)=\sum_{k=1}^{t}\beta_{k-1}\prod_{j=k}^{t-1}(1-\beta_j)\overline{\mathbf{g}^{(\text{type})}(\mathbf{u}_k)}.$$

Of course, CAdam is just an instance of the generic update rules presented in Equations 16-19. Other first-order compositional methods, such as NASA (Ghadimi et al., 2020), ASCGA (Wang et al., 2017a), SCGA (Wang et al., 2017a) and Adam applied to a nested Monte Carlo objective can all be derived from our general form as demonstrated in Appendix E.

Second-Order Compositional Solvers for BO: For a holistic comparison against ERM-BO, we prefer to use the three same optimisation categories of zero-, first-, and second-order methods in Comp-BO. Although significant progress towards first-order compositional optimisers has been achieved in the literature, second-order techniques tackling the objective in Equation 15 are yet to be developed. In this paper, we take a first step towards developing second-order compositional methods and propose an adaption of the standard L-BFGS algorithm to handle nested compositional forms. To start, we note that any second-order technique considers function curvature in its update through the usage of Hessian information:

$$\mathbf{x}_{1:q,t+1}=\mathbf{x}_{1:q,t}+\eta_t\left[\overline{\nabla^2_{\text{vec}(\mathbf{x}_{1:q})\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}\right]^{-1}\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,t}|\mathcal{D}_i)},$$

where $\overline{\nabla^2_{\text{vec}(\mathbf{x}_{1:q})\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}$ and $\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}$ are stochastic approximations of the Hessian and the gradient of $\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,t}|\mathcal{D}_i)$ and η_t is a learning rate. A compositional structure however, imposes practical limitations for the applicability of any arbitrary second-order method due to two essential difficulties. The first relates to the computation of the Hessian, while the second relates to calculating its inverse. When evaluating $\nabla^2_{\text{vec}(\mathbf{x}_{1:q})\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q}|\mathcal{D}_i)$, we encounter an expensive 3-tensor-vector product – $\mathcal{O}(d^2q^3M)$ with d, q and M denoting the dimensionality, batch size of input queries and $\mathbf{z}$ respectively – of the following form:

$$\nabla^2_{\text{vec}(\mathbf{x}_{1:q})\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q}|\mathcal{D}_i)=$$
$$\mathbf{J}(\mathbf{x}_{1:q})^{\mathsf{T}}\nabla^2_{\boldsymbol{\zeta}\boldsymbol{\zeta}}f(\mathbb{E}_\omega[\mathbf{g}_\omega(\mathbf{x}_{1:q})])\mathbf{J}(\mathbf{x}_{1:q})+\nabla_{\text{vec}(\mathbf{x}_{1:q})}\mathbf{J}(\mathbf{x}_{1:q})\times_1\nabla_{\boldsymbol{\zeta}}f(\mathbb{E}_\omega[\mathbf{g}_\omega(\mathbf{x}_{1:q})]),$$

where $\mathbf{J}(\mathbf{x}_{1:q}) = \mathbb{E}[\nabla_{\text{vec}(\mathbf{x}_{1:q})}\mathbf{g}_\omega(\mathbf{x}_{1:q})]$ is the Jacobian of the inner mapping $\mathbb{E}_\omega[\mathbf{g}_\omega(\mathbf{x}_{1:q})]$, the 3-tensor $\nabla_{\text{vec}(\mathbf{x}_{1:q})}\mathbf{J}(\mathbf{x}_{1:q})$ is the Hessian of $\mathbb{E}_\omega[\mathbf{g}_\omega(\mathbf{x}_{1:q})]$, and $\times_1$ is a mode-1 product between a 3-tensor and a vector. Apart from needing such expensive products – a total of $\mathcal{O}(dq^3M(d+M))$ – the update rule introduced above further escalates the computational burden by requiring an inverse that is generally cubic in the number of dimensions, i.e., $\mathcal{O}(d^3q^3)$ in our case. Hence, a feasible approximation for computing $[\nabla^2_{\text{vec}(\mathbf{x}_{1:q})\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q}|\mathcal{D}_i)]^{-1}$ plays a crucial role in the success of any second-order method for compositional objectives. As introduced earlier, BFGS-type methods ameliorate the expense of the calculations by utilising the recursive Sherman-Morison formulae that we also follow here (Riedel, 1992). For such an application, we require two curvature pairs $\mathbf{s}_t$ and $\mathbf{h}_t$ for recursively approximating the inverse of the Hessian. Namely if $\mathbf{s}_t = \mathbf{x}_{1:q,t} - \mathbf{x}_{1:q,t-1}$ and $\mathbf{h}_t = \nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,t}|\mathcal{D}_i) - \nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,t-1}|\mathcal{D}_i)$, one can show that

$$\mathbf{A}_t = \left[\mathbf{I} - \frac{\mathbf{s}_t\mathbf{h}_t^\top}{\mathbf{h}_t^\top\mathbf{s}_t}\right]\mathbf{A}_{t-1}\left[\mathbf{I} - \frac{\mathbf{h}_t\boldsymbol{s}_t^\top}{\mathbf{h}_t^\top\mathbf{s}_t}\right] + \frac{\mathbf{s}_t\mathbf{s}_t^\top}{\mathbf{h}_t^\top\mathbf{s}_t},$$

provides a valid approximation to the t^{th} iteration Hessian inverse when initialising $\mathbf{A}_0 = \mathbf{I}$. That is $\mathbf{A}_t \approx \left[\nabla^2_{\text{vec}(\mathbf{x}_{1:q})\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,t}|\mathcal{D}_i)\right]^{-1}$ and memory cost is reduced to $\mathcal{O}(Tdq)$, with T being total number of update iterations. Hence, a BFGS-type update can now be written as:

$$\mathbf{x}_{1:q,t+1} = \mathbf{x}_{1:q,t+1} + \eta_t\mathbf{A}_t\underbrace{\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}}_{\text{Gradient Monte-Carlo estimate}}.$$

3.2.2 Memory-Efficient Implementations for Comp-BO

Although the ERM-BO and FSM-BO strategies discussed in Sections 3.1 and 3.2 share commonalities such as the sampling of the reparametrisation variable $\mathbf{z} \in \mathbb{R}^q$ and the use of Monte Carlo estimates, one important difference between the approaches is memory complexity - the total amount of space in storage (be that disk or cloud) needed for the complete execution of an optimisation method. It is worthwhile mentioning that the key difference between memory and time resources is that the former can be erased and reused multiple times while the latter cannot, and this distinction plays an important role in the analysis of applied optimisation algorithms.

For ERM-BO methods, the total amount of required memory is defined by the size of the largest mini-batch sampled during the execution and the memory needed for the iterative update. Since in all ERM-BO algorithms we use mini-batches of a constant size $K = 128$, and at each iteration t we store only the current iterative value $\mathbf{x}_{1:q,t} \in \mathbb{R}^{dq}$ the overall memory complexity is therefore bounded by $\mathcal{O}(Kq + dq)$.

Similarly to empirically-founded techniques, in FSM-BO methods we also store at each step t the current value of the iterate $\mathbf{x}_{1:q,t} \in \mathbb{R}^{dq}$ and utilise a mini-batch of samplings of size $K \ll M$. However in contrast to the ERM-BO case, the upfront sampling of M reparameterisation random variables $\mathbf{z}$ used in the FSM-BO scenario leads to an $\mathcal{O}(Mq+dq)$

bound for the overall memory capacity. On one hand, large values of M are preferable as they provide a better approximation to the true acquisition functions given in Equations 4 - 7, yet on the other hand, such values of M make finite-sum methods memory stringent.

To remedy this problem, we propose memory-efficient adaptations of compositional methods: CAdam-ME, NASA-ME and Nested-MC-ME. In a nutshell, all these methods exploit the observation that at any given iteration, stochastic compositional optimisers only require uniform sub-sampling from the fixed collection of M reparametrisation variables $\mathbf{z}$. Hence instead of storing M samples upfront, one can draw K of them from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ at each iteration resulting in an overall memory complexity given by $\mathcal{O}(Kq + dq)$. For a detailed description of the memory-efficient methods CAdam-ME, NASA-ME and Nested-MC-ME, we refer the reader to Appendix F.

4. Experiments & Results

Having presented a comprehensive set of optimisation techniques suitable for maximising acquisition functions, we now wish to systematically evaluate their empirical performance. Specifically, we design our experimental setup with the intention of answering the following questions:

1. Do Finite-Sum Minimisation acquisition functions provide any benefits compared to the more frequently-used Empirical Risk Minimisation versions?
2. Do compositional optimisers provide any advantages over non-compositional optimisers?
3. What are the practical savings for using memory-efficient implementations of compositional acquisition functions?
4. Are compositional methods more computationally expensive than non-compositional optimisation methods and how does runtime scale as a function of the input dimensionality?
5. How do compositional optimisers perform when optimising real-world black-box functions with noisy evaluations?

In order to answer Questions 1-4, we run twenty-eight optimiser variants on five synthetic, noiseless BBO problems for which the true maxima are known. Knowing the true maxima allows for exact computation of the normalised immediate regret

$$r_t = \frac{|f(\tilde{\mathbf{x}}_\mathbf{t}) - f(\mathbf{x}^*)|}{|f(\tilde{\mathbf{x}}_\mathbf{0}) - f(\mathbf{x}^*)|}, \tag{20}$$

where $f(\mathbf{x}^*)$ is the function value at the global optimiser $\mathbf{x}^*$, $\tilde{\mathbf{x}}_\mathbf{t}$ is the algorithm's recommendation at round t and $f(\tilde{\mathbf{x}}_\mathbf{0})$ is the regret upon initialisation at round 0. The use of analytic functions also facilitates the treatment of input dimensionality as an experiment

variable. In order to answer question 5, we focus on the tasks from Bayesmark. These tasks possess noise in the evaluations and are more representative of real-world BBO problems. For these latter experiments we take forward the best-performing optimisers observed in the synthetic function experiments. A pictorial summary of the experimental setup is provided in Figure 4.

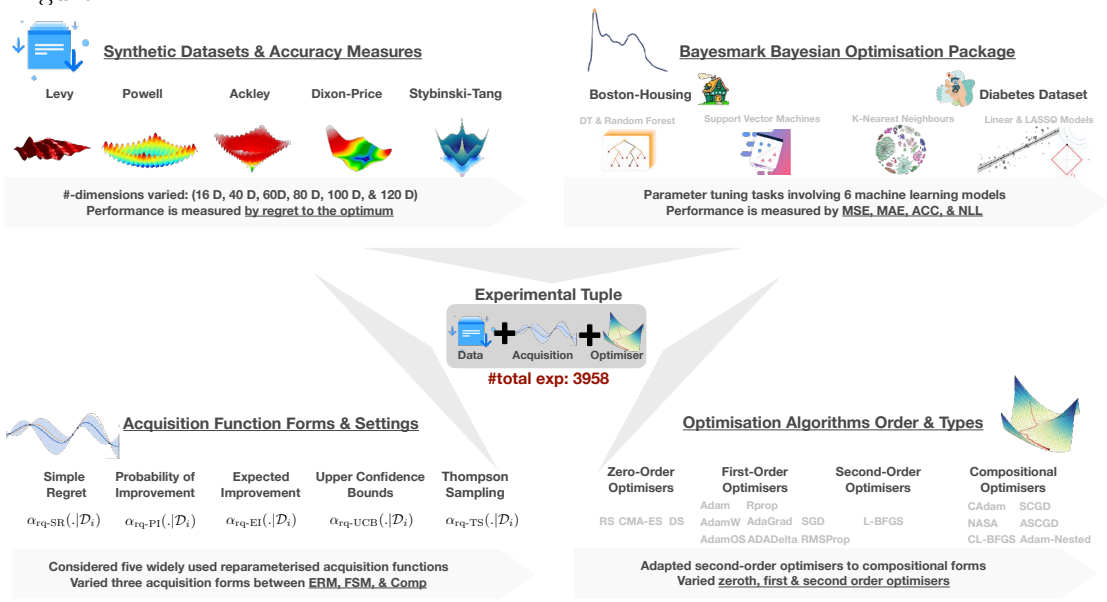

Figure 4: Experiment Overview: **Top Left**: Synthetic functions (noiseless). **Top Right**: Bayesmark data (noisy). **Bottom Left**: Five classes of acquisition function in ERM, Finite-Sum, and Compositional forms. **Bottom Right**: Four classes of optimiser. Each experiment tuple comprises a data set, an acquisition function and an optimiser. The study comprises 3958 experiments in total.

Surrogate Model: For all tasks, we use a GP with constant mean function set to the empirical mean of the data, and a Matérn(5/2) kernel with lengthscale parameter $\boldsymbol{\theta}$. At each acquisition step k, the hyperparameters of the GP kernel are estimated based on the current observed input-output pairs $\mathcal{D}_k$ by optimising the negative log marginal likelihood with a *Gamma* prior over $\boldsymbol{\theta}$. To facilitate the fitting procedure of the surrogate model, we standardise the outputs and apply an affine transformation to the inputs so that the search domain lies in $[0, 1]^d$. At the beginning of each experiment, three points are drawn uniformly at random within the search domain to initialise the surrogate model.

Additionally, in order to provide some indication as to how the GP-based surrogate model schemes, endowed with compositional optimisation of the acquisition function, perform against other surrogates, we also compare against the BOHB algorithm (Falkner et al., 2018), a hybrid approach based on Bayesian optimisation and the Hyperband algorithm (Li et al., 2017). BOHB has recently been demonstrated to outperform Bayesian optimisation across a range of problems in the multi-fidelity setting, that is where multiple objective functions exist possessing varying degrees of accuracy and cost associated with querying them

(Song et al., 2019). In order to enable comparison in the single-fidelity contexts considered in our experiments, we simply ignore the budget handling from Hyperband.

Acquisition Functions: We consider the batched versions of each acquisition function presented in Section 2.2, namely EI, PI, SR and UCB under ERM, FSM and compositional forms. Additionally, we employ Thompson sampling (Thompson, 1933) as a baseline in order to provide an indication as to how the compositionally-optimised acquisition functions perform against another popular batch acquisition function.

Optimisers: Acquisition function maximisation is carried out using the zero-order optimisers RS, CMA-ES and DE from the pymoo library (Blank and Deb, 2020), the non-compositional first-order optimisers Adadelta, Adagrad, Adam, AdamW, RMSprop, Rprop and SGA taken from PyTorch (Paszke et al., 2019), the second-order optimiser L-BFGS-B from the SciPy library (Virtanen et al., 2020), as well as the compositional optimisers ASCGA, CAdam, Nested-MC, NASA and SCGA that we implemented on top of the BoTorch library (Balandat et al., 2020). Except when using non-memory-efficient compositional methods, we used quasi-MC normal Sobol sequences (Owen, 2003) instead of *i.i.d.* normal samples in order to obtain lower variance estimates of the value and gradient of the acquisition function as recommended by (Balandat et al., 2020). For the L-BFGS-B optimiser, the minibatch of samples was fixed in all cases. To ensure fairness in performance comparison, the same number of optimisation steps T (set to 64) and minibatch size m (set to 128), is used for each method at each acquisition step. As acquisition function maximisation is a non-convex problem, it is sensitive to the initialisation set. As such, we use multiple restart points (Wang et al., 2020) that we first obtain by drawing 1024 batches uniformly at random in the modified search space $[0, 1]^{q \times d}$, and second using the default heuristic from (Balandat et al., 2020) to select only 32 promising initialisation batches. Consequently, at each inner optimisation step of BO, the Random Search optimisation strategy is granted $32 \times T \times m$ evaluations of the acquisition function at random batches. Similarly, CMA-ES and DE are run for 64 evolution and mutation steps, and the aforementioned initialisation strategy is used to generate the 32 members of the initial population.

It is known that first-order stochastic optimisers can be very sensitive to the choice of hyperparameter settings (Balandat et al., 2020; Schmidt et al., 2020). Therefore, to limit the effect of choice of hyperparameter settings for the different optimisers, we conducted each experiment in two phases. An experiment in this instance is characterised by the 3-tuple consisting of a black-box function, an acquisition function and an optimiser.

In the first phase, we ran BO hyperparameter tuning to identify the best optimiser hyperparameters, in the sense that these hyperparameters provide the lowest final regret for the given task. This first phase allows us to compare optimisers in their most favourable settings, and therefore we hope that under-performance cannot be the result of a poor choice of hyperparameters but would reflect a real weakness of the considered method in tackling BO's inner optimisation problem.

In the second phase, we ran the black-box maximisation task using the acquisition function and optimiser with hyperparameters fixed to be the best ones identified during

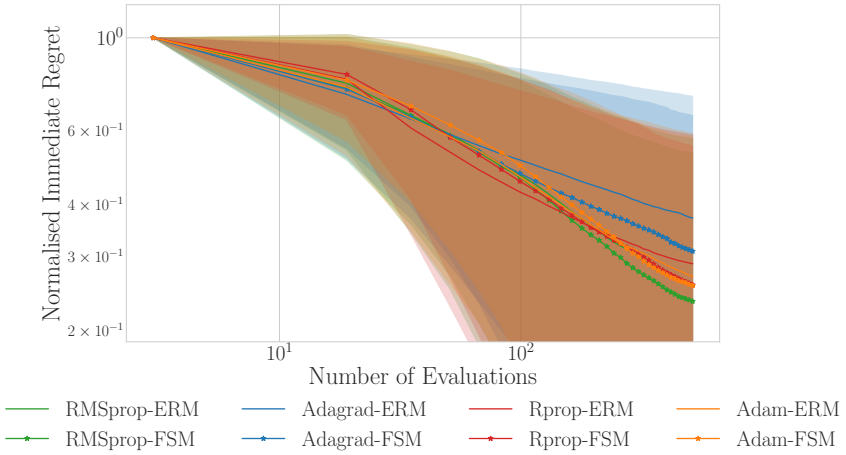

Figure 5: Summary plot comparing the evolution of the normalised immediate regret averaged over all tasks when using first-order methods with either the ERM or FSM formulation of the acquisition function. The results of 960 experiments are summarised. We observe a small advantage of the FSM formulation over the ERM formulation across every optimiser. Statistical significance is discussed in Appendix G.2.

the first phase. The set and range of the considered hyperparameters are summarised in Table 4 for non-compositional optimisers, and in Table 5 for compositional optimisers.

4.1 FSM vs. ERM

In the following experiment, we consider five non-separable, non-convex, synthetic black-box functions chosen to have a variety of optimisation landscapes and that are commonly-used benchmarks for optimisation algorithms (Jamil and Yang, 2013; Laguna and Marti, 2005). We include the unimodal functions *Dixon-Price* and *Powell* as well as the multimodal *Levy*, *Ackley* and *Styblinski-Tang* functions. We run experiments for (negative) versions of these functions with search domain specified as in (Jamil and Yang, 2013; Laguna and Marti, 2005). We consider optimisation problems across dimensionalities in the set (16D, 40D, 60D, 80D, 100D and 120D) in order to observe the impact of the input space dimension on the optimisers' performance. At each acquisition step, a batch of $q = 16$ points is acquired as a result of batch acquisition function maximisation. We run each BO algorithm with 32 acquisition steps and observe the normalised immediate regret from Equation 20 as the performance metric.

Results Summary Figure 5 aggregates by optimiser category, (zero-order non-compositional, first-order compositional,...), the results of 960 experiments involving each combination of optimisation task, acquisition function and optimiser. The best performances obtained inside each category are accounted for. Specifically, given a category and an acquisition step, the lowest normalised immediate regrets obtained at this step by an optimiser belonging to this category are included and the average and standard deviation obtained over all optimisation tasks and all acquisition functions, are reported.

In light of these results we will now answer Question 1:

> Question 1
>
> Do Finite-Sum Minimisation acquisition functions provide any benefits compared to the more frequently-used Empirical Risk Minimisation versions?

When looking at the top four first-order non-compositional optimisers, Figure 5 shows in all cases that the FSM version outperforms the ERM version when averaging the normalised immediate regret scores over all optimisation tasks and acquisition functions. This can be seen in an un-aggregated breakdown in both Figure 18 and Figure 6. This is an interesting discovery, and to the best of our knowledge, we are the first to observe this. We now proceed to our second question.

4.2 Compositional vs. Non-Compositional Optimisation

To synthesise the results obtained over all combinations of synthetic function (Levy, Ackley, Powell, Dixon-Price, Styblinski-Tang), input dimensionality (16D, 40D, 60D, 80D, 100D and 120D), and acquisition function (EI, PI, SR, UCB), we show in Figure 1 the evolution of the normalised immediate regret for each category of optimiser. We confirm the observation of (Wilson et al., 2018b) that gradient-based approaches outperform zero-order methods. Evolutionary strategies perform comparably to Random Search (which we exclude from its category as a global baseline). The poor performance of zero-order methods can be explained by the dimensionality of the acquisition function domain, ranging from 16×16 to 16×120 and the strict limitation on the number of optimisation steps. Results obtained with BOHB are also similar to Random Search, although it is worth mentioning that the experimental setting is single-fidelity and not multi-fidelity where BOHB has been observed to perform well. The performance of Thompson sampling (TS) coincides with the observation in the literature that TS has difficulty scaling beyond 8-10 dimensions (Wilson et al., 2020). We run GPflow (De G. Matthews et al., 2017) implementations of function-space, weight-space and decoupled TS with the default hyperparameters from (Wilson et al., 2020). We report these results in our summary plots and note that scaling such information-based acquisition functions constitutes an important direction for future work, see Section 5.

On examining gradient-based methods, we observe that quasi-Newton (C)L-BFGS-B is consistently outperformed by first-order methods, which was not observed in (Balandat et al., 2020) where only a small-dimensional experiment with no batch acquisition (i.e. $q = 1$) was presented. From this global summary, our results favour first-order optimisers, with a relative advantage being given to compositional methods associated with the FSM approximation. On the other hand, non-compositional optimisers do not seem to be amenable to ERM or FSM formulation.

To show a breakdown of all experiments, we present in Figure 6 the best performances yielded by each category of optimiser for each input dimensionality and acquisition function considered. From this figure, we can first observe that the dimensionality of the BO problem does not seem to have a significant impact on the relative performances between the different types of methods, that is, for any dimension, the best first-order gradient method

			Dim.	16		40		60		80		100		120		**Tot.**	
	Order	Optimiser	Ref.	#Best (%)	NFR	#Best (%)	NFR	#Best (%)	NFR	#Best (%)	NFR	#Best (%)	NFR	#Best (%)	NFR	#Best (%)	NFR
Non-Comp	0	RS	App. B.1	0	.33	0	.51	0	.60	0	.64	0	.68	0	.75	0	.59
		CMA-ES	App. B.2	0	.30	0	.49	0	.76	0	.80	0	.81	0	.85	0	.67
		DE	App. B.3	0	.29	0	.45	0	.61	0	.66	0	.66	0	.70	0	.56
		Subtot.		0	.31	0	.48	0	.66	0	.70	0	.72	0	.77	0	.61
	1	SGA	App. C.1	0	.18	0	.28	0	.33	0	.42	0	.35	0	.48	0	.34
		Adagrad	App. C.2	5	.36	5	.55	5	.66	5	.75	5	.87	10	.89	6	.68
		RMSprop	App. C.3	10	.29	5	.45	15	.47	0	.58	0	.53	**15**	.64	8	.49
		Adam	App. C.4	5	.35	15	.46	5	.51	5	.53	**20**	.61	10	.70	10	.52
		Adadelta	App. C.5	0	.20	0	.44	5	.32	0	.46	0	.45	0	.48	1	.39
		Rprop	App. C.6	0	.36	0	.49	10	.57	5	.61	0	.59	10	.66	4	.55
		AdamW	App. C.7	0	.18	0	.24	5	.22	5	.22	5	.25	5	.23	3	.22
		Adamos	App. C.8	0	.17	0	.26	0	.26	0	.28	5	.30	5	.34	2	.27
		Subtot.		20	.26	25	.40	45	.42	20	.48	35	.49	**55**	.55	33	.43
	2	L-BFGS-B	App. D	0	.19	0	.29	0	.39	0	.45	0	.45	0	.51	0	.38
		Subtot.		0	.19	0	.29	0	.39	0	.45	0	.45	0	.51	0	.38
	Tot.			20	.27	25	.41	45	.48	20	.53	35	.55	55	.60	33	.47
Comp	0	CMA-ES	App. B.2	0	.30	0	.49	0	.76	0	.82	0	.83	0	.87	0	.68
		DE	App. B.3	0	.30	0	.46	0	.61	0	.64	0	.67	0	.71	0	.57
		Subtot.		0	.30	0	.47	0	.69	0	.73	0	.75	0	.79	0	.62
	1	SCGA	App. E.1	10	.12	0	.18	0	.33	0	.44	0	.52	0	.62	2	.37
		ASCGA	App. E.2	5	.11	5	.17	0	.34	0	.48	0	.53	0	.60	2	.37
		CAdam	App. E.3	20	.09	25	.12	**35**	.19	**25**	.14	**20**	.14	10	.22	**22**	.15
		NASA	App. E.4	**45**	.08	**35**	.21	15	.31	20	.39	10	.40	5	.55	**22**	.32
		Nested-MC	App. E.5	0	.17	10	.22	5	.23	5	.26	5	.29	0	.38	4	.26
		CAdam-ME	App. F.1	-	-	-	-	-	-	20	.14	15	.16	20	.24	18	.18
		NASA-ME	App. F.2	-	-	-	-	-	-	10	.35	10	.40	5	.52	8	.43
		Nested-MC-ME	App. F.3	-	-	-	-	-	-	0	.28	5	.29	5	.32	3	.29
		Subtot.		**80**	.12	**75**	.18	**55**	.28	**80**	.31	**65**	.34	45	.43	**67**	.28
	2	CL-BFGS-B	Sec. 3.2	0	.20	0	.28	0	.34	0	.36	0	.44	0	.50	0	.35
		Subtot.		0	.20	0	.28	0	.34	0	.36	0	.44	0	.50	0	.35
	Tot.			**80**	.17	**75**	.27	**55**	.39	**80**	.39	**65**	.43	45	.50	**67**	.36

Table 1: Marginal results over acquisition functions and synthetic black-box optimisation tasks (i.e., 20 tasks per dimension). For each dimension, the first column, #Best (%), indicates the percentage of tasks on which an optimiser yielded the lowest final regret, while the second column reports normalised final regret (NFR). ERM and FSM versions of first-order, non-compositional optimisers are grouped together. We mark the best percentage across each dimension (dim.) in red. Clearly, in 16, 40, 60, 80 and 100 dims, compositional solvers achieve the best performance in at least 55 % these tasks, with 80 % in 16 and 80 dims, while first-order non-compositional optimisers outperform others 55 % of the time in 120 dimensions.

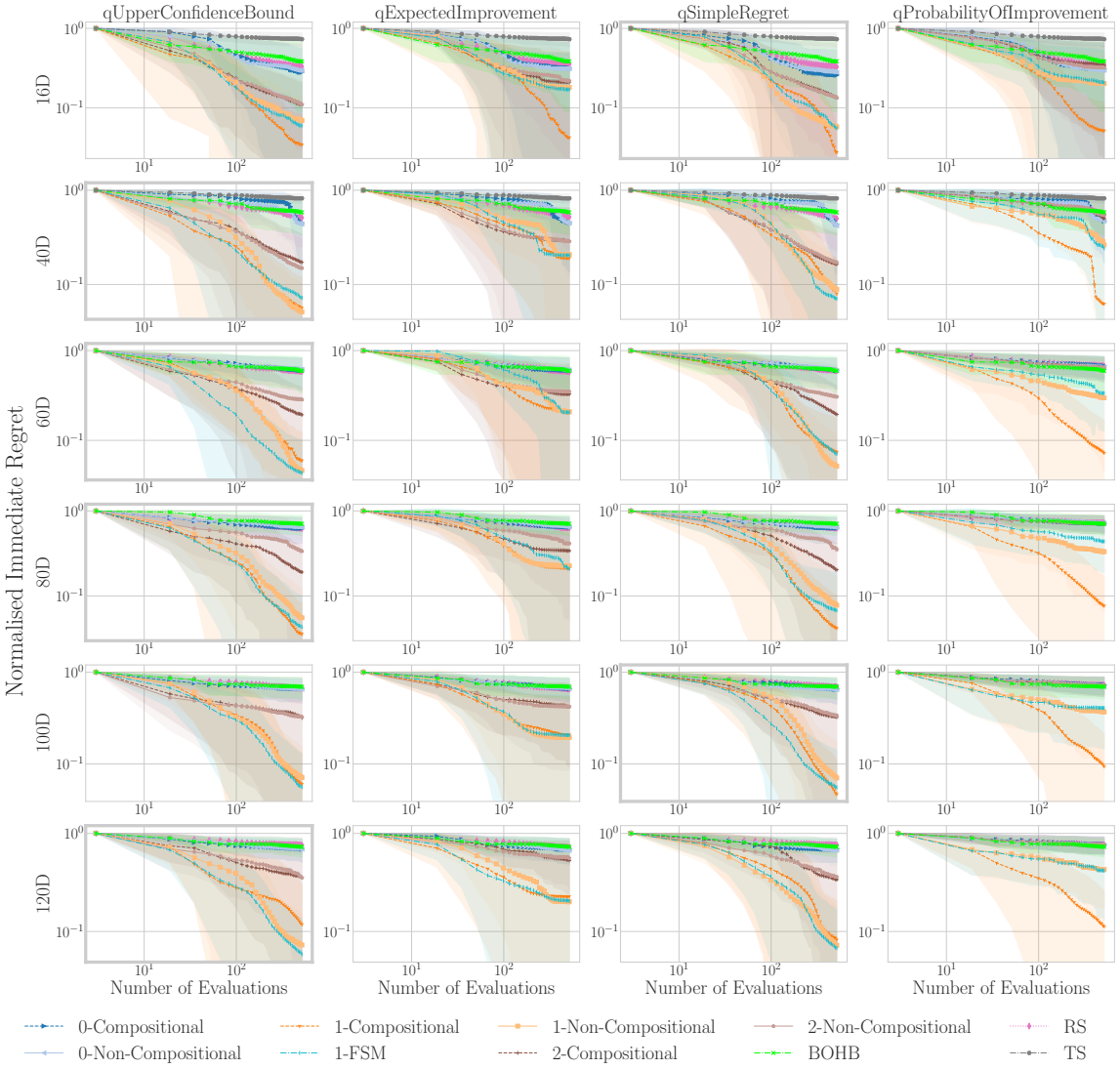

Figure 6: Detailed summary plot for synthetic black-box optimisation showing the best relative improvement for each category of optimiser. Each row corresponds to a domain dimension (16D, 40D, 60D, 80D, 100D and 120D) and each column is associated with an acquisition function (EI, PI, SR and UCB). Relative improvements yielded by BOHB and TS are also reported (there is no variation across columns as they do not depend on the acquisition function), leading the number of experiments aggregated on this figure to be 3100. On each row, the graph corresponding to the acquisition function that achieved the lowest regret for the given input dimension has a thick grey border. In 40, 60, 80 and 120 dimensions, the best performance is achieved using UCB with a first-order optimiser, while in 16 and 100 dims, it is SR with a first-order compositional optimiser that led to the largest relative improvement.

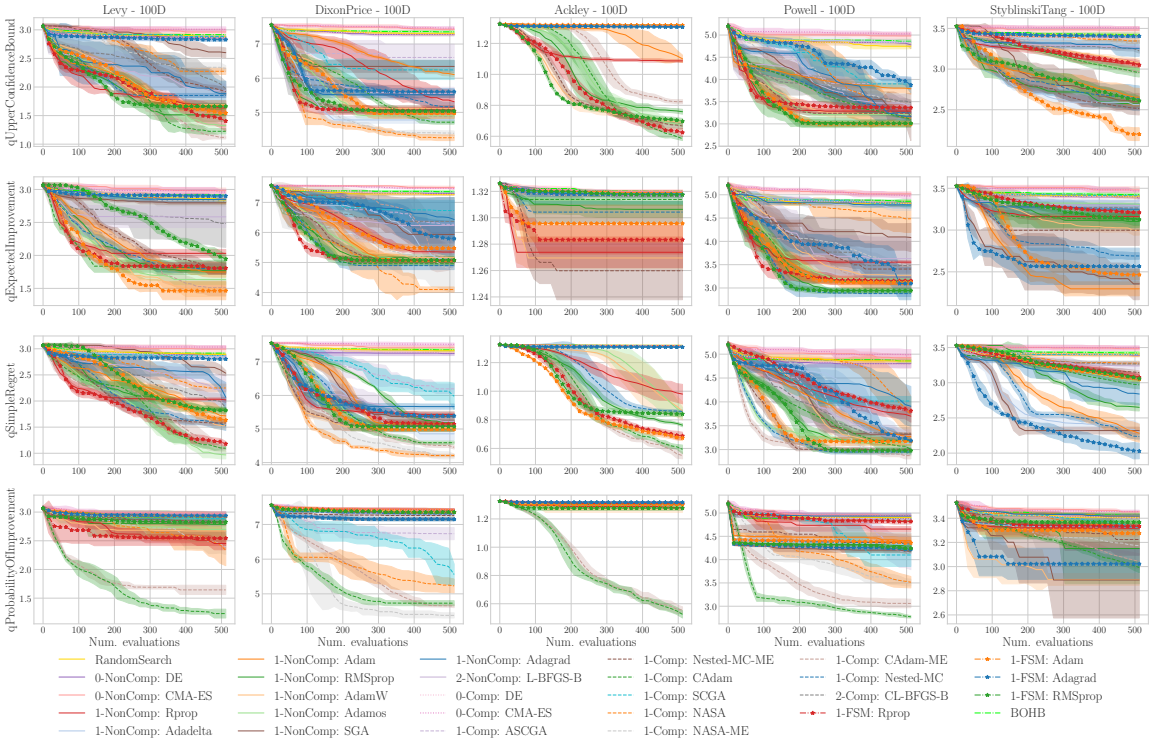

Figure 7: Evolution of immediate log-regret using various acquisition functions and optimisers on 100D synthetic black-box minimisation tasks. Each row is associated with an acquisition function, and each column corresponds to a black-box function. A total of 545 experiments have been carried out on 100 tasks to run BO with all combinations of optimisers, acquisition functions and black-box optimisation functions. We observe that the lowest regret is always achieved with a first-order optimiser, and notably 25% of the best performances are obtained when using CAdam.

outperforms the second-order methods, which achieve lower regret than zero-order ones. Aside from this trend at the level of the optimiser order, we do not notice any lower-level trend that may be driven by the input dimensionality.

An example of the most fine-grained level of analysis (all optimiser performances presented individually) is given in Figure 7. For each task-acquisition pair, we show the log regret over acquisition steps for each optimisation method introduced. We can see that in 65% of the experiments that a compositional optimiser outperforms all non-compositional optimisers. As shown in Table 1, the superior performance of compositional optimisers is observed across all task input dimensionalities except for 120D for which the best optimiser is compositional in only 45% of cases.

Moreover, Figure 6 provides some insight into the comparatively better performance of first-order compositional optimisers observed in the global summary Figure 1. Lower regrets are obtained when the PI acquisition function is used. Nevertheless, the shading of the graphs corresponding to the best acquisition function for each dimensionality indicates

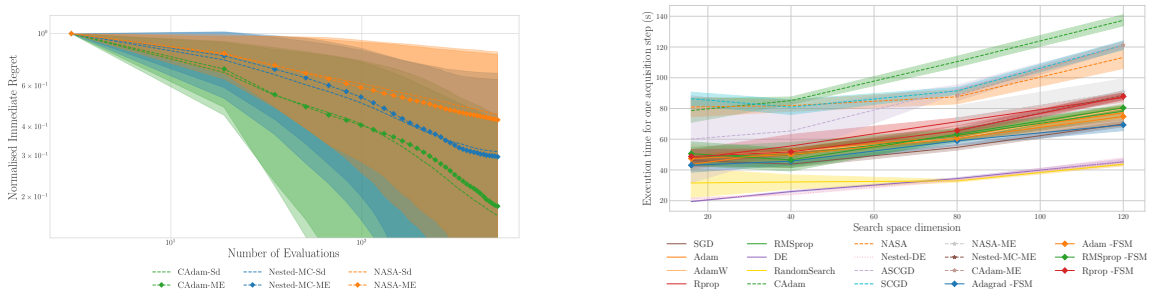

Figure 8: (8a) Summary plot comparing the evolution of the normalised immediate regret averaged over all considered acquisition functions, and optimisation tasks in 80, 100 and 120 dimensions, when using standard (Sd) and memory-efficient (ME) compositional first-order optimisers. From this figure, aggregating the results of 360 experiments, we can see that memory-efficient optimiser versions perform comparably to standard optimisers, thus making it worthwhile to use memory-efficient implementations due to the large memory savings. (8b) Execution time of UCB maximisation run on 4 CPUs. We report the time it takes an optimiser to carry out a single UCB maximisation, and we show the mean and standard deviation observed over 5 seeds, 32 acquisition steps and 2 synthetic black-box functions in 16, 40, 80 and 120 dims. From this figure, aggregating results of 152 experiments, we observe that compositional methods take about 1.5-2x the CPU time taken by non-compositional methods. We do not report the execution times measured for (C)L-BFGS-B and CMA-ES as they are an order of magnitude greater than those observed for non-compositional, first-order methods. We provide complementary results in Figure 19 in Appendix G.

that PI yields consistently higher regrets than UCB or SR, which encourages the use of these alternative acquisition functions in place of PI with a first-order compositional optimiser.

Returning to our second question:

> Question 2
>
> Do compositional optimisers provide any advantages over non-compositional optimisers?

The global summary Figure 1 in addition to Figure 6 indicate that there are a significant number of optimisation task and acquisition function pairs where a compositional optimiser is preferable and as such, compositional schemes warrant much more attention than they are currently receiving in the Bayesian optimisation community. We will now proceed to answer our third question.

4.3 Memory Efficiency

Compositional acquisition function maximisation requires considerably larger memory relative to ERM. However, by introducing a simple trick whereby we do not store all the

auxiliary variables and adopt an alternative sampling scheme, we can dramatically reduce the memory requirements to be equivalent to those of ERM. In answer to question 3:

Question 3

What are the practical savings for using memory-efficient implementations of compositional acquisition functions?

Figure 8a, which aggregates results obtained on tasks in 80, 100 and 120 dimensions using both memory-efficient and standard versions of CAdam, NASA and Nested-MC to maximise the acquisition function, shows that CAdam is negatively impacted by the ME implementation, whereas NASA and Nested-MC are positively impacted by memory efficiency. In all cases, the impact on going from standard to memory-efficient implementations is minor enough that we believe it warrants the use of the ME implementation as the de facto standard. We now proceed to answer Question 4:

4.4 Runtime Efficiency

Runtime efficiency is of great importance for many applications. As such, we wish to see how the execution time required for a single acquisition function optimisation varies across compositional optimisers and input dimensionality. We fix the acquisition function to UCB as this choice has negligible effect on overall timings and we run the BO algorithm for 32 acquisition steps on two black-box maximisation tasks using all available optimisers, repeating each experiment five times. In answer to Question 4:

Question 4

Are compositional methods more computationally expensive than non-compositional optimisation methods and how does runtime scale as a function of the input dimensionality?

There is a marked difference between the execution times reported in Figure 8b for compositional and non-compositional methods with compositional methods being slower relative to non-compositional. Additionally, ME methods are faster than standard compositional methods. We can also see that as the input dimensionality increases, a steeper incline in the execution time for compositional methods relative to non-compositional methods may be observed; a feature to be expected given the extra backward passes required by compositional optimisers. Due to these additional backward passes, compositional methods are 1.5-2 times slower per iteration in terms of wall-clock time. This being said, it should be noted that compositional optimisers may require fewer iterations in total to converge to a specified accuracy and in this case overall wall-clock time could be comparatively better for them. Finally, if the black-box system evaluation wall-clock time is factors larger than the optimisation wall-clock time, which is the case in many real-world problems such as molecule synthesis where a single query can take 2-3 weeks (Thawani et al., 2020), then the differences in runtime between compositional and non-compositional schemes becomes negligible. We now proceed to answer our final question.

4.5 Real-World Problems: Noisy Evaluations

We now examine the performance of optimisers on Bayesmark tasks. All tasks involve hyperparameter tuning for machine learning models. In contrast to the synthetic functions, the Bayesmark data sets possess noise in the evaluations of the black-box function, a feature inherent in the vast majority of real-world BBO problems. As such, these experiments are designed to assess whether the observations derived from the synthetic experiments are relevant for noisy problems.

Hyperparameter Tuning Tasks: The Bayesmark tasks consist of both regression and classification tasks on the Boston and Diabetes UCI data sets (Dua and Graff, 2017) respectively. In terms of hyperparameter tuning the following six models are considered: Decision Tree (DT), Random Forest (RF), K-Nearest Neighbours (kNN), Support Vector Machine (SVM), Linear and Lasso models. the dimensionality of each task varies from 2 to 9. In contrast to the synthetic functions, we only have access to noisy evaluation of the black-box functions in this instance. We apply Bayesian optimisation using 16 iterations of 8-batch acquisition steps, to optimise the validation loss, mean-squared error (MSE), mean absolute error (MAE), negative log likelihood (NLL) or accuracy depending on the task, plotting the normalised validation loss score (Eq 21) for performance comparison. We ran all six models on regression tasks (both MAE and MSE objectives) and we run three models (DT, RF and SVM) on classification tasks (both NLL and accuracy objectives) due to a limited computation budget. The score achieved after t acquisition steps is given by:

$$\mathbf{score}_t = \frac{\mathcal{L}_t - \mathcal{L}^*}{\mathcal{L}_t^{\text{rand}} - \mathcal{L}^*} \tag{21}$$

where $\mathcal{L}_t$ is the best-achieved loss at batch t. $\mathcal{L}^*$ is the estimated optimal loss for the task and $\mathcal{L}^{\text{rand}}$ is the mean loss (across multiple runs) acquired from random search at batch t.

Optimisers: The top three non-compositional optimisers (Adam, RMSprop, Rprop) were selected for performance comparison against compositional optimisers (NASA, CAdam, Nested-MC).

Acquisition Functions: We show results for the four top-performing acquisition functions (SR, EI, PI and UCB) from the synthetic function experiments.

Surrogate Model: We use the same GP surrogate model as in Sec 4.1, with rounding of integer values when either integer or categorical variables are present. Although more sophisticated methods exist to deal with categorical/integer variables (Ru et al., 2019; Daxberger et al., 2020; Garrido-Merchán and Hernández-Lobato, 2020) we do not consider them here as we are interested in solely in performance on acquisition function maximisation. We sample $2 \times D$ points uniformly at random to initialise the model. We run the same form of hyperparameter tuning for the initialisation as in the synthetic experiments, repeating each experiment 5 times in order to compute the variance for individual tasks.

Results Summary: In answer to our final question:

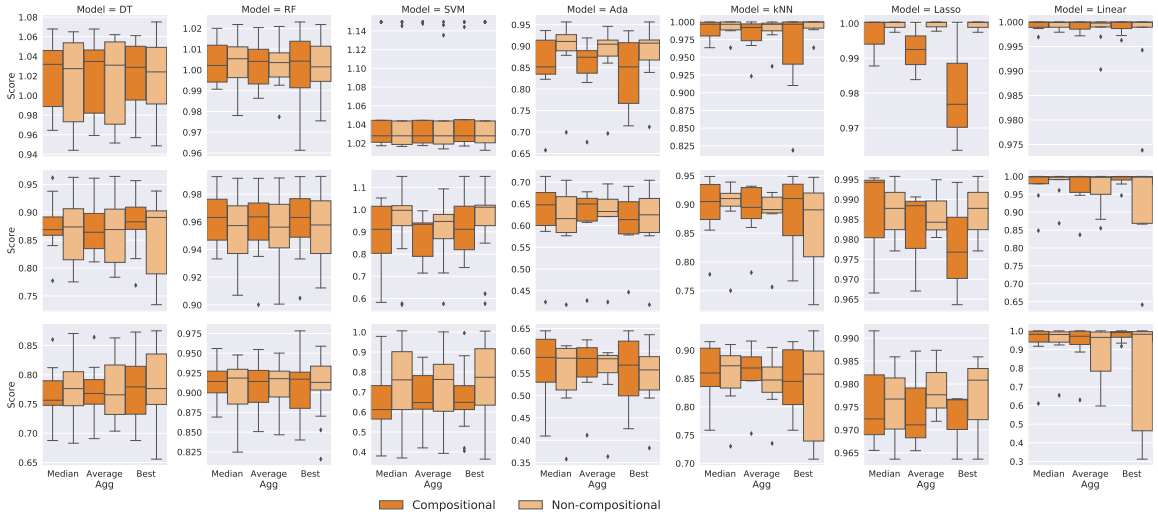

Figure 9: The boxplot shows the quartiles of compositional and non-compositional optimiser performance on the regression hyperparameter tuning task, where the performance metrics are MAE and MSE. For each model, we show a further split of the optimiser class for different aggregation methods. This plot summarises all 672 experiments conducted on regression tasks on the Bayesmark data set. We observe performance benefits for DT, RF and AdaBoost when using a compositional optimiser, with SVM and kNN showing performance benefits when using a non-compositional optimiser. When to use compositional and when to use non-compositional?

Question 5

How do compositional optimisers perform when optimising real-world black-box functions with noisy evaluations?

Figure 2 shows a high-level breakdown of compositional and non-compositional optimiser performance on the Bayesmark regression tasks. The best final scores for the model undergoing tuning are pooled across optimisers, tasks, loss functions and acquisition functions. We observe that compositional and non-compositional optimisers perform comparably, with compositional methods performing slightly better for DT, RF and SVM. We see that the mean scores are roughly equivalent for optimiser classes across the kNN, Lasso, linear and AdaBoost models. In an analogous fashion, Figure 3 pools the scores for all classification experiments. For the DT, and RF models, compositional methods achieve higher mean scores wheraeas comparable performance is observed when tuning the SVM model. In conclusion, compositional vs. non-compositional optimiser performance appears to vary depending on both the model class undergoing tuning as well as the performance metric.

Detailed Results: Figure 9 depicts a finer-grained breakdown of the pooled results for the Bayesmark regression tasks. Pooling in this case is carried out using the best, median and average optimiser performances across all intra-class optimisers and acquisition functions, where for example the best compositional optimiser for a given model would be

the top-scoring optimiser-acquisition pair. For DT and RF, the best results are produced from compositional optimisers, whereas for SVM, AdaBoost, kNN and the linear model, non-compositional methods exhibit better performance. For compositional optimisation of the Lasso model we observe better median performance for a higher number of black-box function evaluations, but deteriorating performance under the best grouping.

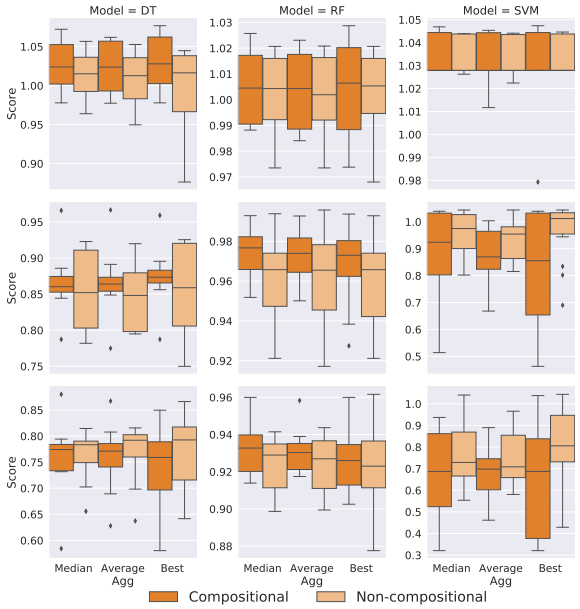

Figure 10: The boxplot shows the quartiles of compositional and non-compositional optimiser performance on the classification hyperparameter tuning task, where the performance metrics are NLL and accuracy. For each model, we show a further split of the optimiser class against different aggregation methods. This plot summarises all 288 experiments conducted on classification tasks for the Bayesmark data sets. We observe that for the DT and RF models, compositional optimisers offer modest performance gains relative to non-compositional optimisers, yet non-compositional optimisers perform better on SVM hyperparameter tuning.

Figure 10 similarly shows a finer-grained breakdown of the Bayesmark classification tasks. We observe that for certain models, such as RF, compositional methods perform better in each of best, median and average groupings at all steps in the optimisation, namely 8, 16 and 128 evaluations of the black-box system. In the DT experiments we again observe that compositional optimisers perform better in the latter optimisation steps (16 & 128 evaluations), but worse in the initial stages of the optimisation (8 evaluations). In summary, compositional methods yield better performance in two-thirds of the cases considered in Figure 10.

5. Conclusions & Future Directions

In this paper, we presented an in-depth study of acquisition function maximisation in Bayesian optimisation. Apart from conventional forms typically used in literature, we demonstrated that acquisition functions adhere to a compositional structure enabling numerous new algorithms that led to favourable empirical results. We verified our claims in a rigorous experimental study involving 3958 tasks and twenty-eight optimisers. We used both synthetic and real-world data gathered from Bayesmark. We demonstrated that compositional optimisers outperform traditional solvers in 67 % of the time. In the future, we plan to extend our analysis to cover non-myopic acquisition functions, constrained and safe BO, high-dimensional BO (Grosnit et al., 2021) as well as to investigate compositional structures of causal BO.

Appendix A. Compositional Construction of Acquisition Functions

Given a collection of M i.i.d samples $\{\mathbf{z}_m\}_{m=1}^M$ and following finite sum approximations for the acquisition functions given in Equations 10 - 13, we now provide detailed compositional reformulations for them (see Section 3.2.1). Let ω be a random variable distributed uniformly on a collection $\{1, \ldots, M\}$, i.e. $\omega \sim \text{Uniform}([1:M])$:

A.1 Expected Improvement

Consider an inner stochastic mapping $\mathbf{g}_\omega^{(\text{EI})} : \mathbb{R}^{dq} \to \mathbb{R}^{q \times M}$, such that:

$$\mathbf{g}_\omega^{(\text{EI})}(\mathbf{x}_{1:q}) = [\mathbf{0}_q, \ldots, \mathbf{v}_\omega^{(\text{EI})}, \ldots, \mathbf{0}_q].$$

where $\mathbf{v}_m^{(\text{EI})} = \text{ReLU}\left(\boldsymbol{\mu}_i(\mathbf{x}_{1:q}; \boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q}; \boldsymbol{\theta})\mathbf{z}_m - f(\mathbf{x}_i^+)\mathbf{1}_q\right) \in \mathbb{R}^q$ for $m = 1:M$. Hence, taking the expectation with respect to ω gives

$$\mathbb{E}_\omega[\mathbf{g}_\omega^{(\text{EI})}(\mathbf{x}_{1:q})] = \frac{1}{M}[\mathbf{v}_1^{(\text{EI})}, \ldots, \mathbf{v}_M^{(\text{EI})}]$$

Now let us consider an outer deterministic mapping $f^{(\text{EI})} : \mathbb{R}^{q \times M} \to \mathbb{R}$, such that for a given $q \times M$ input matrix:

$$f^{(\text{EI})}\left(\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1M} \\ a_{21} & a_{22} & \ldots & a_{2M} \\ \vdots & \vdots & \vdots & \vdots \\ a_{q1} & a_{q2} & \ldots & a_{qM} \end{bmatrix}\right) = \sum_{m=1}^M \max\{a_{1m}, \ldots, a_{qm}\}.$$

Therefore,

$$f^{(\text{EI})}(\mathbb{E}_\omega[\mathbf{g}_\omega^{(\text{EI})}(\mathbf{x}_{1:q})]) = \frac{1}{M}\sum_{m=1}^M \max_{j \in 1:q}\left\{\text{ReLU}\left(\boldsymbol{\mu}_i(\mathbf{x}_{1:q}; \boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q}; \boldsymbol{\theta})\mathbf{z}_m - f(\mathbf{x}_i^+)\mathbf{1}_q\right)\right\} = \alpha_{\text{rq-EI}}^{(\text{FSM})}.$$

A.2 Probability of Improvement

Consider an inner stochastic mapping $\mathbf{g}_\omega^{(\text{PI})} : \mathbb{R}^{dq} \to \mathbb{R}^{q \times M}$, such that:

$$\mathbf{g}_\omega^{(\text{PI})}(\mathbf{x}_{1:q}) = [\mathbf{0}_q, \ldots, \mathbf{v}_\omega^{(\text{PI})}, \ldots, \mathbf{0}_q].$$

where $\mathbf{v}_m^{(\text{PI})} = \frac{1}{\tau}\left[\boldsymbol{\mu}_i(\mathbf{x}_{1:q}; \boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q}; \boldsymbol{\theta})\mathbf{z}_m - f(\mathbf{x}_i^+)\mathbf{1}_q\right] \in \mathbb{R}^q$ for $m = 1:M$. Hence, taking the expectation with respect to ω gives

$$\mathbb{E}_\omega[\mathbf{g}_\omega^{(\text{PI})}(\mathbf{x}_{1:q})] = \frac{1}{M}[\mathbf{v}_1^{(\text{PI})}, \ldots, \mathbf{v}_M^{(\text{PI})}]$$

Now let us consider an outer deterministic mapping $f^{(\text{PI})} : \mathbb{R}^{q \times M} \to \mathbb{R}$, such that for a given $q \times M$ input matrix:

$$f^{(\text{PI})}\left(\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1M} \\ a_{21} & a_{22} & \ldots & a_{2M} \\ \vdots & \vdots & \vdots & \vdots \\ a_{q1} & a_{q2} & \ldots & a_{qM} \end{bmatrix}\right) = \frac{1}{M}\sum_{m=1}^M \max_{j \in 1:q}\{\text{Sig}(M[a_{1m}, \ldots, a_{qm}])\}$$

Therefore,

$$f^{(\mathrm{PI})}(\mathbb{E}_\omega[\mathbf{g}_\omega^{(\mathrm{PI})}(\mathbf{x}_{1:q})]) = \frac{1}{M}\sum_{m=1}^{M}\max_{j\in 1:q}\left\{\mathrm{Sig}\left(\frac{\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m - f(\mathbf{x}_i^+)\mathbf{1}_q}{\tau}\right)\right\} = \alpha_{\text{rq-PI}}^{(\mathrm{FSM})}.$$

A.3 Simple Regret

Consider an inner stochastic mapping $\mathbf{g}_\omega^{(\mathrm{PI})} : \mathbb{R}^{dq} \to \mathbb{R}^{q\times M}$, such that:

$$\mathbf{g}_\omega^{(\mathrm{SR})}(\mathbf{x}_{1:q}) = [\mathbf{0}_q, \ldots, \mathbf{v}_\omega^{(\mathrm{SR})}, \ldots, \mathbf{0}_q].$$

where $\mathbf{v}_m^{(\mathrm{SR})} = \boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m \in \mathbb{R}^q$ for $m = 1:M$. Hence, taking the expectation with respect to ω gives

$$\mathbb{E}_\omega[\mathbf{g}_\omega^{(\mathrm{SR})}(\mathbf{x}_{1:q})] = \frac{1}{M}[\mathbf{v}_1^{(\mathrm{SR})}, \ldots, \mathbf{v}_M^{(\mathrm{SR})}]$$

Now let us consider an outer deterministic mapping $f^{(\mathrm{SR})} : \mathbb{R}^{q\times M} \to \mathbb{R}$, such that for a given $q \times M$ input matrix:

$$f^{(\mathrm{SR})}\left(\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1M} \\ a_{21} & a_{22} & \ldots & a_{2M} \\ \vdots & \vdots & \vdots & \vdots \\ a_{q1} & a_{q2} & \ldots & a_{qM} \end{bmatrix}\right) = \sum_{m=1}^{M}\max\{a_{1m}, \ldots, a_{qm}\}.$$

Therefore,

$$f^{(\mathrm{SR})}(\mathbb{E}_\omega[\mathbf{g}_\omega^{(\mathrm{SR})}(\mathbf{x}_{1:q})]) = \frac{1}{M}\sum_{m=1}^{M}\max_{j\in 1:q}\{\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m\} = \alpha_{\text{rq-SR}}^{(\mathrm{FSM})}.$$

A.4 Upper Confidence Bound

Consider an inner stochastic mapping $\mathbf{g}_\omega^{(\mathrm{UCB})} : \mathbb{R}^{dq} \to \mathbb{R}^{q\times M}$, such that:

$$\mathbf{g}_\omega^{(\mathrm{UCB})}(\mathbf{x}_{1:q}) = [\mathbf{0}_q, \ldots, \mathbf{v}_\omega^{(\mathrm{UCB})}, \ldots, \mathbf{0}_q].$$

where $\mathbf{v}_m^{(\mathrm{UCB})} = \boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \sqrt{\beta\pi/2}\,|\mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m| \in \mathbb{R}^q$ for $m = 1:M$. Hence, taking the expectation with respect to ω gives

$$\mathbb{E}_\omega[\mathbf{g}_\omega^{(\mathrm{UCB})}(\mathbf{x}_{1:q})] = \frac{1}{M}[\mathbf{v}_1^{(\mathrm{UCB})}, \ldots, \mathbf{v}_M^{(\mathrm{UCB})}]$$

Now let us consider an outer deterministic mapping $f^{(\mathrm{UCB})} : \mathbb{R}^{q\times M} \to \mathbb{R}$, such that for a given $q \times M$ input matrix:

$$f^{(\mathrm{UCB})}\left(\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1M} \\ a_{21} & a_{22} & \ldots & a_{2M} \\ \vdots & \vdots & \vdots & \vdots \\ a_{q1} & a_{q2} & \ldots & a_{qM} \end{bmatrix}\right) = \sum_{m=1}^{M}\max\{a_{1m}, \ldots, a_{qm}\}.$$

Therefore,

$$f^{(\mathrm{UCB})}(\mathbb{E}_\omega[\mathbf{g}_\omega^{(\mathrm{UCB})}(\mathbf{x}_{1:q})]) = \frac{1}{M}\sum_{m=1}^{M}\max_{j\in 1:q}\left\{\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \sqrt{\beta\pi/2}|\mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m|\right\} = \alpha^{(\mathrm{FSM})}_{\text{rq-UCB}}.$$

A.5 Entropy Search

Consider an inner stochastic mapping $\mathbf{g}_\omega^{(\mathrm{ES})} : \mathbb{R}^{dq} \to \mathbb{R}^{u\times M}$ such that:

$$\mathbf{g}_{\boldsymbol{\omega}}^{(\mathrm{ES})}(\mathbf{x}_{1:q}) = [\mathbf{v}_{1,\boldsymbol{\omega}}^{(\mathrm{ES})}, \mathbf{v}_{2,\boldsymbol{\omega}}^{(\mathrm{ES})}, \ldots, \mathbf{v}_{M,\boldsymbol{\omega}}^{(\mathrm{ES})}].$$

where $\mathbf{v}_{m,\boldsymbol{\omega}}^{(\mathrm{ES})} = \mathrm{SM}\left(\frac{\boldsymbol{\mu}_i^{(\mathrm{g})}(\mathbf{x}_{1:u}^{(\mathrm{g})};\boldsymbol{\lambda}_{\boldsymbol{\theta}}^{(i)}(\mathbf{z}_m),\boldsymbol{\theta}) + \mathbf{L}_i^{(\mathrm{g})}(\mathbf{x}_{1:u}^{(\mathrm{g})};\boldsymbol{\theta})\boldsymbol{\omega}}{\tau}\right)$ for $m = 1,\ldots,M$ and $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}_u, \mathbf{I}_{u\times u})$. Hence, taking expectation with respect to $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}_u, \mathbf{I}_{u\times u})$ gives:

$$\mathbb{E}_{\boldsymbol{\omega}}\left[\mathbf{g}_{\boldsymbol{\omega}}^{(\mathrm{ES})}(\mathbf{x}_{1:q})\right] = \left[\mathbb{E}_{\boldsymbol{\omega}}\left[\mathbf{v}_{1,\boldsymbol{\omega}}^{(\mathrm{ES})}\right], \mathbb{E}_{\boldsymbol{\omega}}\left[\mathbf{v}_{2,\boldsymbol{\omega}}^{(\mathrm{ES})}\right], \ldots, \mathbb{E}_{\boldsymbol{\omega}}\left[\mathbf{v}_{M,\boldsymbol{\omega}}^{(\mathrm{ES})}\right]\right]$$

Now let us consider an outer stochastic mapping $f_\nu^{(\mathrm{ES})} : \mathbb{R}^{u\times M} \to \mathbb{R}$ such that for any given $u \times M$ input matrix:

$$f_\nu^{(\mathrm{ES})}\left(\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1M} \\ a_{21} & a_{22} & \ldots & a_{2M} \\ \vdots & \vdots & \vdots & \vdots \\ a_{u1} & a_{u2} & \ldots & a_{uM} \end{bmatrix}\right) = -\mathrm{H}\left[\begin{bmatrix} a_{1\nu} \\ a_{2\nu} \\ \vdots \\ a_{u\nu} \end{bmatrix}\right] = \sum_{j=1}^{u} a_{j\nu}\log a_{j\nu}.$$

where $\mathrm{H}[\cdot]$ is a Shannon entropy function. Taking expectation with respect to $\nu \sim \mathrm{Uniform}([1:M])$ gives:

$$\mathbb{E}_\nu\left[f_\nu^{(\mathrm{ES})}\left(\mathbb{E}_{\boldsymbol{\omega}}\left[\mathbf{g}_{\boldsymbol{\omega}}^{(\mathrm{ES})}(\mathbf{x}_{1:q})\right]\right)\right] = \\ -\frac{1}{M}\sum_{m=1}^{M}\mathrm{H}\left[\mathbb{E}_{\boldsymbol{\omega}}\left[\mathrm{SM}\left(\frac{\boldsymbol{\mu}_i^{(\mathrm{g})}(\mathbf{x}_{1:u}^{(\mathrm{g})};\boldsymbol{\lambda}_{\boldsymbol{\theta}}^{(i)}(\mathbf{z}_m),\boldsymbol{\theta}) + \mathbf{L}_i^{(\mathrm{g})}(\mathbf{x}_{1:u}^{(\mathrm{g})};\boldsymbol{\theta})\boldsymbol{\omega}}{\tau}\right)\right]\right] = \alpha^{(\mathrm{FSM})}_{\text{rq-ES}}$$

Appendix B. Zero-Order Optimisation Algorithms for ERM-BO

B.1 Random Search

The most simple zeroth-order strategy we attempted in our experiments was random search (RS), where a new batch of query points is constructed by sampling q candidates $\mathbf{x}_{1:q}$ uniformly at random from a bounded search domain. Though simple, RS has been shown to be an effective optimisation scheme in certain settings (Bergstra and Bengio, 2012; Li et al., 2017) and can serve as an essential low-memory, low-compute baseline for any acquisition optimiser.

B.2 CMA-ES

In the covariance matrix adaptation evolution strategy (CMA-ES) (van Rijn et al., 2016; Blank and Deb, 2020), a population of new search points is generated by sampling a multivariate normal distribution, which for generations $g = 0, 1, \ldots,$ can be written as:

$$\text{vec}\,(\mathbf{x})_l^{(g+1)} \sim \boldsymbol{\mu}_{\text{CMA-ES}}^{(g)} + \sigma_{\text{CMA-ES}}^{(g)} \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Sigma}_{\text{CMA-ES}}^{(g)}\right) \quad \text{for } l \in [1 : \#\text{off-springs}], \tag{22}$$

where $\boldsymbol{\mu}_{\text{CMA-ES}}^{(g)}, \sigma_{\text{CMA-ES}}^{(g)}$ and $\boldsymbol{\Sigma}_{\text{CMA-ES}}^{(g)}$ are the distribution's hyperparameters that will be updated based on function value information. Also, #off-springs > 2 represents the number of individuals sampled from a population, e.g., the number of optimiser restarts in our case. Moreover, the usage of the $\text{vec}(\mathbf{x}) \in \mathbb{R}^{dq}$ notation denotes a vector of inputs across all batches and dimensions.

Starting from an initialisation $\boldsymbol{\mu}_{\text{CMA-ES}}^{(0)}, \sigma_{\text{CMA-ES}}^{(0)}$ and $\boldsymbol{\Sigma}_{\text{CMA-ES}}^{(0)}$, CMA-ES updates each of the hyperparameters based on fitness or function values to improve the guess of $\mathbf{x}^\star$. At some generation $g+1$, the algorithm first samples $\text{vec}(\mathbf{x})_1^{(g+1)}, \ldots, \text{vec}(x)_{\#\text{off-springs}}^{(g+1)}$ according to Equation 22 and then ranks individual samples in a descending order based on their acquisition evaluation such that[6] $\alpha_{\text{rq-type}}(\text{vec}(\mathbf{x})_{1^\star}^{(g+1)}|\mathcal{D}_i) \leq \cdots \leq \alpha_{\text{rq-type}}(\text{vec}(\mathbf{x})_{\#\text{off-springs}^\star}^{(g+1)}|\mathcal{D}_i)$, where $\text{vec}(\mathbf{x})_{j^\star}^{(g+1)}$ is the j^{th} best sample vector (according to its acquisition value) from $\text{vec}(\mathbf{x})_1^{(g+1)}, \ldots, \text{vec}(\mathbf{x})_{\#\text{off-springs}}^{(g+1)}$. With samples ordered, the algorithm updates $\boldsymbol{\mu}_{\text{CMA-ES}}^{(g+1)}$ as an average of $\kappa \leq \#\text{off-springs}$ selected points:

$$\boldsymbol{\mu}_{\text{CMA-ES}}^{(g+1)} = \boldsymbol{\mu}_{\text{CMA-ES}}^{(g)} + \eta_{\boldsymbol{\mu}_{\text{CMA-ES}}} \sum_{i=1}^{\kappa} w_i \left(\text{vec}_{i^\star}^{(g+1)} - \boldsymbol{\mu}_{\text{CMA-ES}}^{(g)}\right), \tag{23}$$

with $\eta_{\boldsymbol{\mu}_{\text{CMA-ES}}} < 1$ being a learning rate, and $w_i \propto \kappa - i + 1$. In words, Equation 23 attempts to shift the distribution's mean closer to a weighted average of the best samples seen so far, which, in turn, can be reinterpreted as maximising a log-data-likelihood conditioned on $\boldsymbol{\mu}_{\text{CMA-ES}}^{(g)}$ as noted in (Hansen, 2016).

When it comes to $\sigma_{\text{CMA-ES}}^{(g)}$, a process of cumulative step-size adaptation (CSA) – also referred to as path length control – is applied to derive $\sigma_{\text{CMA-ES}}^{(g+1)}$. First, CSA computes an (isotropic) "evolutionary path" path_σ using:

$$\text{path}_\sigma = (1 - c_\sigma)\text{path}_\sigma + \sqrt{1 - (1 - c_\sigma)^2}\sqrt{\kappa_w}\boldsymbol{\Sigma}_{\text{CMA-ES}}^{(g),-\frac{1}{2}} \frac{\boldsymbol{\mu}_{\text{CMA-ES}}^{(g+1)} - \boldsymbol{\mu}_{\text{CMA-ES}}^{(g)}}{\sigma_{\text{CMA-ES}}^{(g)}}, \tag{24}$$

where c_σ is a constant typically set to $d/3$, and κ_w is a variance-related constant abiding by $1 \leq \kappa_w \leq \kappa$. Given Equation 24, CSA now updates $\sigma_{\text{CMA-ES}}^{(g+1)}$ by executing[7]:

$$\sigma_{\text{CMA-ES}}^{(g+1)} = \sigma_{\text{CMA-ES}}^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{||\text{path}_\sigma||}{\mathbb{E}[||\mathcal{N}(0,1)||]} - 1\right)\right), \quad \text{with } d_\sigma \text{ being a damping value.}$$

6. Please note that we use $\alpha_{\text{rq-type}}(\cdot)$ to denote one of the reparameterised acquisitions (i.e., EI, PI, UCB, and SR).

7. It is worth noting that the update of $\sigma_{\text{CMA-ES}}^{(g+1)}$ requires the computation of $\mathbb{E}[||\mathcal{N}(0,1)||]$. Such an expectation can be approximated using a Gamma distribution as shown in (van Rijn et al., 2016)

Similarly, $\mathbf{\Sigma}^{(g)}_{\text{CMA-ES}}$ is adapted by following a two-step process, where an (anisotropic) evolutionary path, $\text{path}_{\mathbf{\Sigma}_{\text{CMA-ES}}}$, is used to in $\mathbf{\Sigma}^{(g+1)}_{\text{CMA-ES}}$ as follows:

$$\text{path}_{\mathbf{\Sigma}_{\text{CMA-ES}}} = (1 - c_{\mathbf{\Sigma}_{\text{CMA-ES}}})\text{path}_{\mathbf{\Sigma}_{\text{CMA-ES}}} + \mathbb{1}_{[0,\eta\sqrt{d}]}(||\text{path}_{\sigma}||)\sqrt{1 - (1 - c_{\mathbf{\Sigma}_{\text{CMA-ES}}})^2}\sqrt{\kappa_w}\,\frac{\boldsymbol{\mu}^{(g+1)}_{\text{CMA-ES}} - \boldsymbol{\mu}^{(g)}_{\text{CMA-ES}}}{\sigma^{(g)}_{\text{CMA-ES}}}$$

$$\mathbf{\Sigma}^{(g+1)}_{\text{CMA-ES}} = \gamma\mathbf{\Sigma}^{(g)}_{\text{CMA-ES}} + c_1\text{path}_{\mathbf{\Sigma}_{\text{CMA-ES}}}\text{path}^{\mathsf{T}}_{\mathbf{\Sigma}_{\text{CMA-ES}}}, + \eta_{\mathbf{\Sigma}_{\text{CMA-ES}}}\sum_{i=1}^{\kappa} w_i \left(\frac{\text{vec}^{(g+1)}_{i\star} - \boldsymbol{\mu}^{(g)}_{\text{CMA-ES}}}{\sigma^{(g)}_{\text{CMA-ES}}}\right)\left(\frac{\text{vec}^{(g+1)}_{i\star} - \boldsymbol{\mu}^{(g)}_{\text{CMA-ES}}}{\sigma^{(g)}_{\text{CMA-ES}}}\right)^{\mathsf{T}},$$

where γ is a discount factor, $c_{\mathbf{\Sigma}_{\text{CMA-ES}}}$, c_1, and $\eta_{\mathbf{\Sigma}_{\text{CMA-ES}}}$ are tuneable hyperparameters. Finally, we used $\mathbb{1}_{[0,\eta\sqrt{d}]}(\cdot)$ to denote the indicator function with η typically set to ≈ 1.5.

B.3 DE

In differential evolution (DE) (Blank and Deb, 2020), a new set of input probes is generated from a previous population via component-wise mutation. The initial population $\mathcal{D}^{(0)} = \{\text{vec}(\mathbf{x})^{(0)}_1, \ldots, \text{vec}(\mathbf{x})^{(0)}_{\#\text{population}}\}$ is given as a collection of K vectors, where each $\text{vec}(\mathbf{x})^{(0)}_j \in \mathbb{R}^{dq}$. Each vector $\text{vec}(\mathbf{x})^{(g+1)}_j$ in the next population $\mathcal{D}^{(g+1)}$ undergoes a component-wise random mutation process consisting of three sequential steps. First, for each $\text{vec}(\mathbf{x})^{(g)}_j \in \mathcal{D}^{(g)}$, DE randomly picks a collection of three different candidates $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^{dq}$ that belong to the current population $\mathcal{D}^{(g)}$. These candidates will play the role of building blocks for a component-wise mutation process generating a candidate $\mathcal{C} \in \mathbb{R}^{dq}$ for the next population. In the second step, DE randomly picks a component $l \in [1, \ldots, dq]$ of $\text{vec}(\mathbf{x})^{(g)}_j$ which will be deterministically mutated with others undergoing a mutation with some fixed probability p_{mutation}:

$$[\mathcal{C}]_l = [\mathbf{a}]_l + \text{F}([\mathbf{b}]_l - [\mathbf{c}]_j), \quad \text{and} \quad [\mathcal{C}]_i = \begin{cases} [\mathbf{a}]_i + \text{F}([\mathbf{b}]_i - [\mathbf{c}]_i), & \text{w. p. } p_{\text{mutation}} \\ \left[\text{vec}(\mathbf{x})^{(g)}_j\right]_i & \text{w. p. } 1 - p_{\text{mutation}} \end{cases}$$

where $\text{F} \in [0, 2]$ is a scaling mutation parameter, and $[\mathbf{v}]_i$ is used to denote the i^{th} component of vector $\mathbf{v}$. In the last step the algorithm makes a choice on whether to add $\mathcal{C}$ to the new population based on the acquisition function value information. In case the mutated vector achieves a better solution than $\text{vec}(\mathbf{x})^{(g)}_j$, then $\mathcal{C}$ is added to a the new population $\mathcal{D}^{(g+1)}$, otherwise $\text{vec}(\mathbf{x})^{(g)}_j$ is preserved. After the algorithm terminates, DE reports the best solution out of all constructed populations $\cup_{g\geq 0}\mathcal{D}^{(g)}$.

Appendix C. First-Order Optimisers for ERM-BO

First-order optimisation techniques rely on gradient information to compute updates of $\mathbf{x}$. They are iterative in nature, running for a total of T iterations and executing a variant of

the following rule at each step:

$$\mathbf{x}_{1:q,t+1} = \delta_t \mathbf{x}_{1:q,t} + \eta_t \frac{\boldsymbol{\phi}_t^{(1)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}, \left\{\beta_k^{(1)}\right\}_{k=0}^{t}\right)}{\boldsymbol{\phi}_t^{(2)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}^2, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}^2, \left\{\beta_k^{(2)}\right\}_{k=0}^{t}, \epsilon\right)} \text{ (Generalised update)}, \tag{25}$$

where δ_t is a weighting that depends on the class of algorithm used, η_t is a (typically) decaying learning rate, $\boldsymbol{\phi}_t^{(1)}(\cdot)$ and $\boldsymbol{\phi}_t^{(2)}(\cdot)$ are history-dependent mappings that differ between algorithms with the ratio between them computed element-wise. $\left\{\beta_k^{(1)}\right\}_{k=0}^{t}$ and $\left\{\beta_k^{(2)}\right\}_{k=0}^{t}$ are history weighting parameters, and ϵ is a small positive constant used to avoid division by zero. Additionally, $\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}$ represent sub-sampled gradient estimators that are acquired using Monte Carlo samples of $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. It is also worth noting that differentiating through the max operator that appears in all acquisitions can be performed either using sub-gradients or by propagating through the max value of the corresponding vector.

C.1 SGA

Stochastic gradient ascent (SGA) is a cornerstone of the optimisation algorithm literature (Robbins and Monro, 1951), simply using gradients to ascend the objective function. Though it requires a large number of iterations to converge, recent studies demonstrate that stochastic gradients (Wilson et al., 2018a) exhibit better generalisation capabilities when compared to other methods in machine learning applications. We can attain SGA's update from Equation 25 by setting constant weightings $\delta_1 = \cdots = \delta_T = 1$, $\left\{\beta_k^{(1)}\right\}_{k=0}^{t} = \emptyset$, $\left\{\beta_k^{(2)}\right\}_{k=0}^{t} = \emptyset$, and defining $\boldsymbol{\phi}_t^{(1)}(\cdot)$ and $\boldsymbol{\phi}_t^{(2)}(\cdot)$ as:

$$\boldsymbol{\phi}_t^{(1)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}\right) = \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)},$$
$$\boldsymbol{\phi}_t^{(2)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}^2, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}^2\right) = \mathbf{1}_{dq}.$$

C.2 AdaGrad

In adaptive gradients (AdaGrad), SGA is modified so as to exhibit per-parameter learning rates (Duchi et al., 2011). Intuitively, AdaGrad increases learning rates for sparse parameters and decreases them for denser ones. Such a strategy has been shown to be successful in settings where the data is sparse, and where sparse parameters convey more information (e.g., natural language processing (Pennington et al., 2014) and image recognition tasks (Song et al., 2020)). AdaGrad's update can also be extracted from Equation 25 by

choosing $\delta_1 = \cdots = \delta_T = 1$, $\left\{\beta_k^{(1)}\right\}_{k=0}^{t} = \emptyset$, $\left\{\beta_k^{(2)}\right\}_{k=0}^{t} = \emptyset$, and:

$$\boldsymbol{\phi}_t^{(1)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}\right) = \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)},$$

$$\boldsymbol{\phi}_t^{(2)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}^2, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}^2, \epsilon\right) = \sqrt{\sum_{k=0}^{t} \overline{\nabla\alpha(\mathbf{x}_{1:q,k}|\mathcal{D}_i)}^2 + \epsilon}\,.$$

C.3 RMSprop

In root mean-square propagation (RMSprop), learning rates are also adapted to each of the parameters. Here, the idea is to divide the learning rate for a parameter by a running average of the magnitudes of recent gradients for that specific parameter (Tieleman and Hinton, 2012). RMSprop has enjoyed considerable success in machine learning (Khosla et al., 2020; Wu et al., 2020). To arrive at its update rule, we set $\delta_1 = \cdots = \delta_T = 1$, $\left\{\beta_k^{(1)}\right\}_{k=0}^{t} = \emptyset$, $\beta_1^{(2)} = \cdots = \beta_T^{(2)} = \gamma$ with γ denoting a forgetting factor. Furthermore, a constant learning rate η is typically adopted in RMSprop, i.e., $\eta_1 = \cdots = \eta_T = \eta$, and $\boldsymbol{\phi}_t^{(1)}(\cdot)$ and $\boldsymbol{\phi}_t^{(2)}(\cdot)$ defined as:

$$\boldsymbol{\phi}_t^{(1)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}\right) = \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)},$$

$$\boldsymbol{\phi}_t^{(2)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}^2, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}^2, \gamma, \epsilon\right) = \sqrt{(1-\gamma)\sum_{k=0}^{t} \gamma^k \overline{\nabla\alpha(\mathbf{x}_{1:q,k}|\mathcal{D}_i)}^2 + \epsilon}\,.$$

C.4 Adam

Adam (Kingma and Ba, 2015) is one of the most successful and widely-used algorithms in machine learning applications. The method computes individual adaptive learning rates for different parameters from estimates of the first and second moments of the gradients. In terms of Equation 25, we can derive Adam's update as a special case using the following settings: 1) constant weightings $\delta_1 = \cdots = \delta_T = 1$, $\beta_1^{(1)} = \cdots = \beta_T^{(1)} = \beta_1$, $\beta_1^{(2)} = \cdots = \beta_T^{(2)} = \beta_2$, and 2) $\boldsymbol{\phi}_t^{(1)}$ and $\boldsymbol{\phi}_t^{(2)}$ defined as:

$$\boldsymbol{\phi}_t^{(1)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}, \beta_1\right) = \frac{1-\beta_1}{1-\beta_1^t}\sum_{k=0}^{t} \beta_1^k \overline{\nabla\alpha(\mathbf{x}_{1:q,t-k}|\mathcal{D}_i)},$$

$$\boldsymbol{\phi}_t^{(2)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}^2, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}^2, \beta_2, \epsilon\right) = \sqrt{\frac{1-\beta_2}{1-\beta_2^t}\sum_{k=0}^{t} \beta_2^k \overline{\nabla\alpha(\mathbf{x}_{1:q,t-k}|\mathcal{D}_i)}^2} + \epsilon.$$

C.5 AdaDelta

The AdaDelta algorithm can be viewed as a robust extension of the AdaGrad method (Zeiler, 2012). AdaDelta adapts learning rates based on a moving window of gradient updates. This window-based modification is implemented in an efficient manner by recursively

defining the sum of the gradients as a decaying average of all past squared gradients. Following the general update rule introduced in Equation 25, AdaDelta can be formulated by setting $\delta_1 = \cdots = \delta_T = 1$, $\beta_1^{[1]} = \epsilon$, $\beta_2^{[1]} = \eta$, $\beta_3^{[1]} = \cdots = \beta_T^{[1]} = \beta_1^{[2]} = \cdots = \beta_T^{[2]} = \gamma$, $\eta_1 = \cdots = \eta_T = \eta$, and

$$\boldsymbol{\phi}_t^{(1)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}, \gamma, \eta, \epsilon\right) =$$
$$\overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}\sqrt{\sum_{k=0}^{t-1}\frac{\gamma^k\overline{\nabla\alpha(\mathbf{x}_{1:q,t-k-1}|\mathcal{D}_i)}^2}{\sum_{j=0}^{t-k-1}\gamma^j\overline{\nabla\alpha(\mathbf{x}_{1:q,t-k-1-j}|\mathcal{D}_i)}^2 + \frac{\epsilon}{(1-\gamma)}} + \frac{\epsilon}{\eta^2}},$$
$$\boldsymbol{\phi}_t^{(2)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}^2, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}^2, \gamma, \epsilon\right) = \sqrt{(1-\gamma)\sum_{k=0}^{t}\gamma^k\overline{\nabla\alpha(\mathbf{x}_{1:q,t-k}|\mathcal{D}_i)}^2 + \epsilon}.$$

C.6 RProp

To overcome the inherent disadvantages of pure gradient descent/ascent techniques in terms of tuning the learning rate, (Riedmiller and Braun, 1993) propose RProp, an algorithm that takes into account only the sign of the corresponding partial derivative value. In terms of Equation 25, RProp can be defined by choosing $\delta_1 = \cdots = \delta_T = 1$, $\left\{\beta_k^{(1)}\right\}_{k=0}^{t} = \emptyset$, $\left\{\beta_k^{(2)}\right\}_{k=0}^{t} = \emptyset$, and $\boldsymbol{\phi}_t^{(1)}(\cdot)$ and $\boldsymbol{\phi}_t^{(2)}(\cdot)$ as:

$$\boldsymbol{\phi}_t^{(1)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}\right) = \text{sign}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}\right),$$
$$\boldsymbol{\phi}_t^{(2)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}^2, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}^2\right) = \mathbf{1}_{dq}.$$

C.7 AdamW

(Loshchilov and Hutter, 2019) propose a variation of Adam optimisation algorithm with decoupled weight decay regularisation to improve its generalisation properties. AdamW can be written in the form of Equation 25 by specifying $\delta_t = (1 - \lambda\eta_t)$, $\beta_1^{[1]} = \cdots = \beta_T^{[1]} = \beta_1$, $\beta_1^{[2]} = \cdots = \beta_T^{[2]} = \beta_2$, and

$$\boldsymbol{\phi}_t^{(1)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}, \beta_1\right) = \frac{(1-\beta_1)\sum_{k=0}^{t}\beta_1^k\overline{\nabla\alpha(\mathbf{x}_{1:q,t-k}|\mathcal{D}_i)}}{1-\beta_1^t},$$
$$\boldsymbol{\phi}_t^{(2)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}^2, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}^2, \beta_2, \epsilon\right) = \sqrt{\frac{(1-\beta_2)\sum_{k=0}^{t}\beta_2^k\overline{\nabla\alpha(\mathbf{x}_{1:q,t-k}|\mathcal{D}_i)}^2}{1-\beta_2^t}} + \epsilon.$$

C.8 AdamOs

To isolate the effect of our compositional reformulation, we consider a variation of the standard Adam optimiser with the parameter setup adopted from its compositional counterpart CAdam. In terms of Equation 25, AdamOs can be formulated by setting $\delta_1 = \cdots = \delta_T = 1$,

$\beta_t^{[1]} = \mathcal{O}(\mu^t)$, $\beta_t^{[2]} = 1 - \frac{(1-\mathcal{O}(\mu^t))^2}{t^{\eta_\gamma}}$, $\eta_t = \mathcal{O}\left(\frac{\sqrt{1-\beta_t^{[2]}}}{(1-\mathcal{O}(\mu^t))t^{\eta_\eta}}\right)$, and

$$\boldsymbol{\phi}_t^{(1)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}, \dots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}, \beta_1\right) = \frac{1-\beta_1}{1-\beta_1^t}\sum_{k=0}^{t}\beta_1^k\overline{\nabla\alpha(\mathbf{x}_{1:q,t-k}|\mathcal{D}_i)},$$

$$\boldsymbol{\phi}_t^{(2)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}^2, \dots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}^2, \beta_2, \epsilon\right) = \sqrt{\frac{1-\beta_2}{1-\beta_2^t}\sum_{k=0}^{t}\beta_2^k\overline{\nabla\alpha(\mathbf{x}_{1:q,t-k}|\mathcal{D}_i)}^2} + \epsilon.$$

Appendix D. Second-Order Optimisers in ERM-BO

Second-order optimisation methods along with gradients utilise second-order information of the objective function, typically[8] encoded in the Hessian matrix $\nabla^2\alpha(\cdot|\mathcal{D}_i)$. The general iterative update for second-order methods is given by:

$$\mathbf{x}_{1:q,t+1} = \mathbf{x}_{1:q,t} - \eta_t\left[\overline{\nabla^2\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}\right]^{-1}\overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)} \qquad \text{(Generalised update)},$$

where $\overline{\nabla^2\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}$ is an approximation for the Hessian matrix evaluated at a current iterate $\mathbf{x}_{1:q,t}$. This approximation is needed due to the size of the real Hessian matrix (in our case $\overline{\nabla^2\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)} \in \mathbb{R}^{dq\times dq}$) as well as the necessity to compute its inverse at each iteration of the above generalised update.

The BFGS algorithm (Kelley, 1999) and its memory-efficient version (Byrd et al., 1995) are the most commonly-used second-order techniques for high-dimensional, non-convex optimisation and are based on the Sherman-Morison formulae for recursive computation of the approximated Hessian inverse:

$$\left[\overline{\nabla^2\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}\right]^{-1} = \left[\mathbf{I} - \frac{\mathbf{s}_t\mathbf{h}_t^\mathsf{T}}{\mathbf{h}_t^\mathsf{T}\mathbf{s}_t}\right]\left[\overline{\nabla^2\alpha(\mathbf{x}_{1:q,t-1}|\mathcal{D}_i)}\right]^{-1}\left[\mathbf{I} - \frac{\mathbf{h}_t\boldsymbol{s}_t^\mathsf{T}}{\mathbf{h}_t^\mathsf{T}\mathbf{s}_t}\right] + \frac{\mathbf{s}_t\mathbf{s}_t^\mathsf{T}}{\mathbf{h}_t^\mathsf{T}\mathbf{s}_t},$$

where $\left[\overline{\nabla^2\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}\right]^{-1} = \mathbf{I}$ and curvature pairs $\mathbf{h}_t = \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)} - \overline{\nabla\alpha(\mathbf{x}_{1:q,t-1}|\mathcal{D}_i)}$, $\mathbf{s}_t = \mathbf{x}_{1:q,t} - \mathbf{x}_{1:q,t-1}$. The recursive expression is beneficial for two reasons: 1) it admits computation of the Hessian inverse approximation while avoiding the inversion of large matrices and 2) it is formulated in terms of curvature pairs $\mathbf{s}_t, \mathbf{y}_t$ and hence permits computation of the descent direction efficiently with respect to both time and memory.

Appendix E. First-Order Compositional Optimisers

As discussed in Section 3.2.1, first-order compositional methods depend on a stochastic approximation of the gradient of a compositional function $\alpha^{(\text{Comp})}(\mathbf{x}_{1:q}|\mathcal{D}_i) = f(\mathbb{E}_\omega[\mathbf{g}_\omega(\mathbf{x}_{1:q})])$

8. An alternative is the Fischer Information Matrix (Amari and Nagaoka, 2007) used in the natural gradient decent update equation (Amari, 2012).

given by:

$$\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q}|\mathcal{D}_i)} = \left[\frac{1}{K_2}\sum_{m=1}^{K_2}\nabla_{\text{vec}(\mathbf{x}_{1:q})}\mathbf{g}_{\omega_m}(\mathbf{x}_{1:q})\right]^{\mathsf{T}}\nabla_{\boldsymbol{\zeta}}f(\boldsymbol{\zeta})$$

where $\mathbf{y}$ is an iterative auxiliary variable introduced to approximate the expectation of the inner mapping $\mathbb{E}_\omega[\mathbf{g}_\omega(\mathbf{x}_{1:q})]$ in a momentum-based fashion. Generalised update rules for first-order compositional optimisers are iterative in nature and have the following form:

Main variable update:

$$\mathbf{x}_{1:q,t+1} = \mathbf{x}_{1:q,t} + \eta_t \frac{\boldsymbol{\phi}_t^{(1)}\left(\left\{\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}\right\}_{k=0}^{t},\left\{\gamma_k^{(1)}\right\}_{k=0}^{t}\right)}{\boldsymbol{\phi}_t^{(2)}\left(\left\{\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}^2\right\}_{k=0}^{t},\left\{\gamma_k^{(2)}\right\}_{k=0}^{t},\epsilon\right)},$$

The second auxiliary variable update:

$$\mathbf{u}_{t+1} = \boldsymbol{\phi}_{t+1}^{(3)}\left(\mathbf{x}_{1:q,0},\ldots,\mathbf{x}_{1:q,t+1},\{\beta_k\}_{k=0}^{t}\right),$$

The first auxiliary variable update:

$$\boldsymbol{\zeta}_{t+1} = \boldsymbol{\phi}_{t+1}^{(4)}\left(\overline{\mathbf{g}(\mathbf{u}_1)},\ldots,\overline{\mathbf{g}(\mathbf{u}_{t+1})},\{\beta_k\}_{k=0}^{t},\boldsymbol{\zeta}_0,\mathbf{u}_0\right).$$

where $\overline{\mathbf{g}(\mathbf{u})} = \frac{1}{K_1}\sum_{m=1}^{K_1}\mathbf{g}_{\omega_m}(\mathbf{u})$ is a Monte Carlo approximation of $\mathbb{E}_\omega[\mathbf{g}_\omega(\mathbf{x}_{1:q})]$. Next, we show how different first-order compositional optimisers can be formulated in terms of the above generalised iterative updates.

E.1 SCGA

Stochastic Compositional Gradient Ascent (Wang et al., 2017a) is the first algorithm which focuses on a quasi-gradient computation and a momentum-based approximation of the inner mapping $\mathbb{E}_\omega[\mathbf{g}_\omega(\mathbf{x})]$. Following the generalised update scheme, SCGA can be accessed by setting:

$$\boldsymbol{\phi}_t^{(1)}\left(\left\{\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}\right\}_{k=0}^{t},\left\{\gamma_k^{(1)}\right\}_{k=0}^{t}\right) = \overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,t},\boldsymbol{\zeta}_t|\mathcal{D}_i)},$$
$$\boldsymbol{\phi}_t^{(2)}\left(\left\{\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}^2\right\}_{k=0}^{t},\left\{\gamma_k^{(2)}\right\}_{k=0}^{t},\epsilon\right) = \mathbf{1}_{dq},$$
$$\boldsymbol{\phi}_t^{(3)}\left(\mathbf{x}_{1:q,0},\ldots,\mathbf{x}_{1:q,t},\{\beta_k\}_{k=0}^{t-1}\right) = \mathbf{x}_{1:q,t},$$
$$\boldsymbol{\phi}_t^{(4)}\left(\overline{\mathbf{g}(\mathbf{u}_1)},\ldots,\overline{\mathbf{g}(\mathbf{u}_t)},\{\beta_k\}_{k=0}^{t-1},\boldsymbol{\zeta}_0,\mathbf{u}_0\right) = \sum_{k=1}^{t}\beta_{k-1}\prod_{j=k}^{t-1}(1-\beta_j)\overline{\mathbf{g}(\mathbf{u}_k)}.$$

E.2 ASCGA

(Wang et al., 2017a) propose an accelerated stochastic compositional gradient algorithm by evaluating compositional gradients via two-timescale iteration updates. We can attain ASCGA from the generalised update equations by defining $\boldsymbol{\phi}_t^{(1)}$, $\boldsymbol{\phi}_t^{(2)}$, $\boldsymbol{\phi}_t^{(3)}$, $\boldsymbol{\phi}_t^{(4)}$ as:

$$\boldsymbol{\phi}_t^{(1)}\left(\left\{\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}\right\}_{k=0}^{t},\left\{\gamma_k^{(1)}\right\}_{k=0}^{t}\right)=\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp})}(\mathbf{x}_{1:q,t},\boldsymbol{\zeta}_t|\mathcal{D}_i)},$$
$$\boldsymbol{\phi}_t^{(2)}\left(\left\{\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}^2\right\}_{k=0}^{t},\left\{\gamma_k^{(2)}\right\}_{k=0}^{t},\epsilon\right)=\mathbf{1}_{dq},$$
$$\boldsymbol{\phi}_t^{(3)}\left(\mathbf{x}_{1:q,0},\ldots,\mathbf{x}_{1:q,t},\{\beta_k\}_{k=0}^{t-1}\right)=(1-\beta_{t-1}^{-1})\mathbf{x}_{1:q,t-1}+\beta_{t-1}^{-1}\mathbf{x}_{1:q,t},$$
$$\boldsymbol{\phi}_t^{(4)}\left(\overline{\mathbf{g}(\mathbf{u}_1)},\ldots,\overline{\mathbf{g}(\mathbf{u}_t)},\{\beta_k\}_{k=0}^{t-1},\boldsymbol{\zeta}_0,\mathbf{u}_0\right)=\sum_{k=1}^{t}\beta_{k-1}\prod_{j=k}^{t-1}(1-\beta_j)\overline{\mathbf{g}(\mathbf{u}_k)}.$$

E.3 CAdam

As mentioned in the main body of the paper, one can recover CAdam (Tutunov et al., 2020) by instantiating the above as follows:

$$\boldsymbol{\phi}_t^{(1)}\left(\left\{\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}\right\}_{k=0}^{t},\left\{\gamma_k^{(1)}\right\}_{k=0}^{t}\right)=$$
$$\sum_{k=0}^{t}(1-\gamma_k^{[1]})\prod_{j=k+1}^{t}\gamma_j^{[1]}\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)},$$
$$\boldsymbol{\phi}_t^{(2)}\left(\left\{\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}^2\right\}_{k=0}^{t},\left\{\gamma_k^{(2)}\right\}_{k=0}^{t},\epsilon\right)=$$
$$\sqrt{\sum_{k=0}^{t}(1-\gamma_k^{[2]})\prod_{j=k+1}^{t}\gamma_j^{[2]}\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}^2}+\epsilon,$$

$$\boldsymbol{\phi}_t^{(3)}\left(\mathbf{x}_{1:q,0},\ldots,\mathbf{x}_{1:q,t},\{\beta_k\}_{k=0}^{t-1}\right)=(1-\beta_{t-1}^{-1})\mathbf{x}_{1:q,t-1}+\beta_{t-1}^{-1}\mathbf{x}_{1:q,t},$$
$$\boldsymbol{\phi}_t^{(4)}\left(\overline{\mathbf{g}^{(\mathrm{type})}(\mathbf{u}_1)},\ldots,\overline{\mathbf{g}^{(\mathrm{type})}(\mathbf{u}_t)},\{\beta_k\}_{k=0}^{t-1},\boldsymbol{\zeta}_0,\mathbf{u}_0\right)=\sum_{k=1}^{t}\beta_{k-1}\prod_{j=k}^{t-1}(1-\beta_j)\overline{\mathbf{g}^{(\mathrm{type})}(\mathbf{u}_k)}.$$

E.4 NASA

Nested Averaged Stochastic Approximation (Ghadimi et al., 2020) is a single time-scale stochastic approximation algorithm whereby the problem is transformed to a high-dimensional space and together with the main variable $\mathbf{x}$, the behaviour of the gradient of the compositional function $\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp})}(\mathbf{x}_{1:q}|\mathcal{D}_i)$ as well as the value of the inner mapping $\mathbb{E}_\omega[\mathbf{g}_\omega(\mathbf{x})]$ are studied. In terms of generalised update rules, the NASA algorithm

can be formulated by the following setup:

$$\boldsymbol{\phi}_t^{(1)}\left(\left\{\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}\right\}_{k=0}^{t},\{\rho\tau_k\}_{k=0}^{t}\right)=$$
$$\rho\sum_{k=0}^{t-1}\tau_{k-1}\prod_{j=k}^{t}(1-\rho\tau_j)\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)},$$
$$\boldsymbol{\phi}_t^{(2)}\left(\left\{\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}^2\right\}_{k=0}^{t},\left\{\gamma_k^{(2)}\right\}_{k=0}^{t},\epsilon\right)=\mathbf{1}_{dq},$$
$$\boldsymbol{\phi}_t^{(3)}\left(\mathbf{x}_{1:q,0},\ldots,\mathbf{x}_{1:q,t},\{\beta_k\}_{k=0}^{t-1}\right)=\mathbf{x}_{1:q,t},$$
$$\boldsymbol{\phi}_t^{(4)}\left(\overline{\mathbf{g}(\mathbf{u}_1)},\ldots,\overline{\mathbf{g}(\mathbf{u}_t)},\{\beta_k\}_{k=0}^{t-1},\boldsymbol{\zeta}_0,\mathbf{u}_0\right)=\beta\sum_{k=1}^{t}\tau_{k-1}\prod_{j=k}^{t-1}(1-\beta\tau_j)\overline{\mathbf{g}(\mathbf{u}_k)}.$$

E.5 Nested-MC

To emphasise the effect of a momentum-based update for the auxiliary variable $\mathbf{y}$, we also consider a compositional variation of the Adam optimiser, where all involved expectation operators are approximated by corresponding Monte Carlo estimates. In terms of the generalised update scheme, Nested-MC can be formulated as follows:

$$\boldsymbol{\phi}_t^{(1)}\left(\left\{\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}\right\}_{k=0}^{t},\beta_1\right)=$$
$$\frac{(1-\beta_1)}{1-\beta_1^t}\sum_{k=0}^{t}\beta_1^k\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp})}(\mathbf{x}_{1:q,t-k},\boldsymbol{\zeta}_{t-k}|\mathcal{D}_i)},$$
$$\boldsymbol{\phi}_t^{(2)}\left(\left\{\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}^2\right\}_{k=0}^{t},\beta_2,\epsilon\right)=$$
$$\sqrt{\frac{(1-\beta_2)}{1-\beta_2^t}\sum_{k=0}^{t}\beta_2^k\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp})}(\mathbf{x}_{1:q,t-k},\boldsymbol{\zeta}_{t-k}|\mathcal{D}_i)}^2}+\epsilon,$$

$$\boldsymbol{\phi}_t^{(3)}\left(\mathbf{x}_{1:q,0},\ldots,\mathbf{x}_{1:q,t},\{\beta_k\}_{k=0}^{t-1}\right)=\mathbf{x}_{1:q,t},$$
$$\boldsymbol{\phi}_t^{(4)}\left(\overline{\mathbf{g}(\mathbf{u}_1)},\ldots,\overline{\mathbf{g}(\mathbf{u}_t)},\{\beta_k\}_{k=0}^{t-1},\boldsymbol{\zeta}_0,\mathbf{u}_0\right)=\overline{\mathbf{g}(\mathbf{u}_t)}.$$

Appendix F. Memory-Efficient Adaptations for Compositional Optimisers

As described in Section 3.2.2, the necessity of storing all M samples of the reparameterisation random variables $\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$ makes compositional optimisers cumbersome with respect to memory capacity. For example, an inner mapping $\mathbf{g}^{(\mathrm{type})}(\mathbf{x}_{1:q})=\mathbb{E}_\omega\left[\mathbf{g}^{(\mathrm{type})}(\mathbf{x}_{1:q})\right]\in\mathbb{R}^{q\times M}$, where $\mathrm{type}\in\{\mathrm{EI},\mathrm{PI},\mathrm{SR},\mathrm{UCB}\}$ and each stochastic instance $\mathbf{g}_\omega^{(\mathrm{type})}(\mathbf{x}_{1:q})$ is defined

as:

$$\mathbf{g}_{\omega}^{(\text{type})}(\mathbf{x}_{1:q}) = [\mathbf{0}_q, \ldots, \mathbf{v}_{\omega}^{(\text{type})}, \ldots, \mathbf{0}_q] \in \mathbb{R}^{q \times M}$$

where each $\mathbf{v}_{\omega}^{(\text{type})} \in \mathbb{R}^q$ is formulated in terms of an associated vector $\mathbf{z}$ sampled uniformly from a fixed collection, as described in Section 3.2.1. As a result, the construction of a Monte Carlo estimate for $\mathbf{g}^{(\text{type})}(\mathbf{x}_{1:q})$ involves storing all $\{\mathbf{z}_1, \ldots, \mathbf{z}_M\}$ and therefore gives rise to high memory consumption. In the memory-efficient adaptation however, we remedy this problem by sampling a set of reparameterisation random variables $\mathbf{z}_1, \ldots, \mathbf{z}_K$ directly from a distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ rather then from a large fixed collection. As a result, a stochastic instance of the inner mapping can be written as a q by K matrix:

$$\overline{\mathbf{g}^{(\text{type}),(\text{ME})}(\cdot)} = \left[\mathbf{v}_{\mathbf{z}_1}^{(\text{type})}(\cdot), \ldots, \mathbf{v}_{\mathbf{z}_K}^{(\text{type})}(\cdot)\right] \in \mathbb{R}^{q \times K}$$

where the j^{th} column is defined via the associated $\mathbf{v}_{\mathbf{z}_j}^{(\text{type})}(\cdot)$ in an analogous fashion to Section 3.2.1. This adjustment immediately allows us to compute stochastic estimates for the Jacobian $\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\mathbf{g}^{(\text{type}),(\text{ME})}(\cdot)} = \nabla_{\text{vec}(\mathbf{x}_{1:q})}\overline{\mathbf{g}^{(\text{type}),(\text{ME})}(\cdot)}$ of the inner mapping in a memory-efficient manner. Finally, the gradient of the compositional objective $\alpha^{(\text{Comp})}(\star|\mathcal{D}_i)$ can be estimated as follows:

$$\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp}),(\text{ME})}(\star, *|\mathcal{D}_i)} = \left[\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\mathbf{g}^{(\text{type}),(\text{ME})}(\star)}\right]^{\mathsf{T}} \nabla_{\boldsymbol{\zeta}} f^{(\text{type})}(*).$$

where $*$ represents the value of the first auxiliary variable $\boldsymbol{\zeta}$ obtained via the exponentially-weighted average of estimates $\overline{\mathbf{g}^{(\text{type}),(\text{ME})}(\cdot)}$ (see Section 3.2.1). The generalised iterative update equations for memory-efficient compositional optimisers already have a familiar form:

Main variable update:

$$\mathbf{x}_{1:q,t+1} = \mathbf{x}_{1:q,t} + \eta_t \frac{\boldsymbol{\phi}_t^{(1)}\left(\left\{\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp}),(\text{ME})}(\mathbf{x}_{1:q,k}, \boldsymbol{\zeta}_k|\mathcal{D}_i)}\right\}_{k=0}^{t}, \left\{\gamma_k^{(1)}\right\}_{k=0}^{t}\right)}{\boldsymbol{\phi}_t^{(2)}\left(\left\{\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp}),(\text{ME})}(\mathbf{x}_{1:q,k}, \boldsymbol{\zeta}_k|\mathcal{D}_i)}^2\right\}_{k=0}^{t}, \left\{\gamma_k^{(2)}\right\}_{k=0}^{t}, \epsilon\right)},$$

The second auxiliary variable update:

$$\mathbf{u}_{t+1} = \boldsymbol{\phi}_{t+1}^{(3)}\left(\mathbf{x}_{1:q,0}, \ldots, \mathbf{x}_{1:q,t+1}, \{\beta_k\}_{k=0}^{t}\right),$$

The first auxiliary variable update:

$$\boldsymbol{\zeta}_{t+1} = \boldsymbol{\phi}_{t+1}^{(4)}\left(\overline{\mathbf{g}^{(\text{type}),(\text{ME})}(\mathbf{u}_1)}, \ldots, \overline{\mathbf{g}^{(\text{type}),(\text{ME})}(\mathbf{u}_{t+1})}, \{\beta_k\}_{k=0}^{t}, \boldsymbol{\zeta}_0, \mathbf{u}_0\right).$$

Next, we show how memory-efficient compositional optimisers can be formulated in terms of the above generalised iterative updates.

F.1 CAdam-ME

A memory-efficient version of the CAdam optimiser in terms of the generalised update:

$$\boldsymbol{\phi}_t^{(1)}\left(\left\{\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp}),(\mathrm{ME})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}\right\}_{k=0}^{t},\left\{\gamma_k^{(1)}\right\}_{k=0}^{t}\right)=$$
$$\sum_{k=0}^{t}(1-\gamma_k^{[1]})\prod_{j=k+1}^{t}\gamma_j^{[1]}\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp}),(\mathrm{ME})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)},$$
$$\boldsymbol{\phi}_t^{(2)}\left(\left\{\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp}),(\mathrm{ME})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}^2\right\}_{k=0}^{t},\left\{\gamma_k^{(2)}\right\}_{k=0}^{t},\epsilon\right)=$$
$$\sqrt{\sum_{k=0}^{t}(1-\gamma_k^{[2]})\prod_{j=k+1}^{t}\gamma_j^{[2]}\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp}),(\mathrm{ME})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}^2}+\epsilon,$$
$$\boldsymbol{\phi}_t^{(3)}\left(\mathbf{x}_{1:q,0},\ldots,\mathbf{x}_{1:q,t},\{\beta_k\}_{k=0}^{t-1}\right)=(1-\beta_{t-1}^{-1})\mathbf{x}_{1:q,t-1}+\beta_{t-1}^{-1}\mathbf{x}_{1:q,t},$$
$$\boldsymbol{\phi}_t^{(4)}\left(\left\{\overline{\mathbf{g}^{(\mathrm{type}),(\mathrm{ME})}(\mathbf{u}_k)}\right\}_{k=1}^{t},\{\beta_k\}_{k=0}^{t-1},\boldsymbol{\zeta}_0,\mathbf{u}_0\right)=\sum_{k=1}^{t}\beta_{k-1}\prod_{j=k}^{t-1}(1-\beta_j)\overline{\mathbf{g}^{(\mathrm{type}),(\mathrm{ME})}(\mathbf{u}_k)}.$$

F.2 NASA-ME

The NASA algorithm also has a memory-efficient adaptation:

$$\boldsymbol{\phi}_t^{(1)}\left(\left\{\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp}),(\mathrm{ME})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}\right\}_{k=0}^{t},\{\rho\tau_k\}_{k=0}^{t}\right)=$$
$$\rho\sum_{k=0}^{t-1}\tau_{k-1}\prod_{j=k}^{t}(1-\rho\tau_j)\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp}),(\mathrm{ME})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)},$$
$$\boldsymbol{\phi}_t^{(2)}\left(\left\{\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp}),(\mathrm{ME})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}^2\right\}_{k=0}^{t},\left\{\gamma_k^{(2)}\right\}_{k=0}^{t},\epsilon\right)=\mathbf{1}_{dq},$$
$$\boldsymbol{\phi}_t^{(3)}\left(\mathbf{x}_{1:q,0},\ldots,\mathbf{x}_{1:q,t},\{\beta_k\}_{k=0}^{t-1}\right)=\mathbf{x}_{1:q,t},$$
$$\boldsymbol{\phi}_t^{(4)}\left(\left\{\overline{\mathbf{g}^{(\mathrm{type}),(\mathrm{ME})}(\mathbf{u}_k)}\right\}_{k=1}^{t},\{\beta_k\}_{k=0}^{t-1},\boldsymbol{\zeta}_0,\mathbf{u}_0\right)=\beta\sum_{k=1}^{t}\tau_{k-1}\prod_{j=k}^{t-1}(1-\beta\tau_j)\overline{\mathbf{g}^{(\mathrm{type}),(\mathrm{ME})}(\mathbf{u}_k)}.$$

F.3 Nested MC-ME

Finally, the Nested MC optimiser can also be converted to its memory-efficient form:

$$\boldsymbol{\phi}_t^{(1)}\left(\left\{\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp}),(\mathrm{ME})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}\right\}_{k=0}^{t},\beta_1\right)=$$
$$\frac{(1-\beta_1)}{1-\beta_1^t}\sum_{k=0}^{t}\beta_1^k\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp}),(\mathrm{ME})}(\mathbf{x}_{1:q,t-k},\boldsymbol{\zeta}_{t-k}|\mathcal{D}_i)},$$

$$\boldsymbol{\phi}_t^{(2)}\left(\left\{\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp}),(\mathrm{ME})}(\mathbf{x}_{1:q,k},\boldsymbol{\zeta}_k|\mathcal{D}_i)}^2\right\}_{k=0}^{t},\beta_2,\epsilon\right)=$$

$$\sqrt{\frac{(1-\beta_2)}{1-\beta_2^t}\sum_{k=0}^{t}\beta_2^k\overline{\nabla_{\mathrm{vec}(\mathbf{x}_{1:q})}\alpha^{(\mathrm{Comp}),(\mathrm{ME})}(\mathbf{x}_{1:q,t-k},\boldsymbol{\zeta}_{t-k}|\mathcal{D}_i)}^2}+\epsilon,$$

$$\boldsymbol{\phi}_t^{(3)}\left(\mathbf{x}_{1:q,0},\ldots,\mathbf{x}_{1:q,t},\{\beta_k\}_{k=0}^{t-1}\right)=\mathbf{x}_{1:q,t},$$

$$\boldsymbol{\phi}_t^{(4)}\left(\left\{\overline{\mathbf{g}^{(\mathrm{type}),(\mathrm{ME})}(\mathbf{u}_k)}\right\}_{k=1}^{t},\{\beta_k\}_{k=0}^{t-1},\boldsymbol{\zeta}_0,\mathbf{u}_0\right)=\overline{\mathbf{g}^{(\mathrm{type}),(\mathrm{ME})}(\mathbf{u}_t)}.$$

Appendix G. Extended Results

G.1 Non-myopic acquisition function: Entropy Search

Although our main focus was set on myopic acquisition functions, we showed that compositional optimisation can also be applied to maximise non-myopic acquisition functions as illustrated with Entropy Search. As noted in Section 2.2 and contrary to myopic acquisition functions, Entropy Search directly comes in a compositional form making it naturally amenable to compositional optimisation, but can also be optimised via nested-MC approach as the other compositional forms we derived for myopic acquisition functions. Nonetheless, this compositional structure requires, for the inner expectation, the evaluation of GP posteriors on a set of points covering the search space, making this information-efficient acquisition function scale poorly with the dimensionality of the problem.

Owing to poor scalability, we adapted the experimental setup described in Section 4 for synthetic experiments, considering bayesian optimisation of the 3D versions of *Levy* and *Ackley* synthetic functions. We compare the use of zero'th, first and second-order optimisers for the acquisition function optimisation (with a budget of 64 optimisation steps, similar to other experiments). At each inner BO step, three points are acquired until reaching a total of 60 evaluations (on top of the three initial random ones). To reduce memory usage we applied memory-efficient strategy for compositional optimisation and also followed (Hennig and Schuler, 2012) for the discretisation strategy, sampling 100 points based on Expected Improvement values (avoiding the burden of considering a regular grid over the search space which would require taking more points to get a good approximation of the entropy as discussed in (Hennig and Schuler, 2012)).

The evolution of immediate log-regrets shown on Figure 11 have been obtained after applying the hyperparameter tuning procedure presented in Section 4 to assess performance of each optimiser in a favourable setting. We see that for this non-myopic acquisition function, first-order optimisers outperforms second and zeroth-order approaches that are comparable to Random Search. These observations aligns with the ones made when studying myopic cases. Nevertheless, this set of experiments fails to establish a clear advantage of the use of compositional optimisers compared to nested-MC approach (based on Adam optimiser in this case) as NASA permits the lowest regret when minimising *Levy* while nested-MC permits the lowest regret for *Ackley* minimisation. We believe that this preliminary result

should open the door to a more in-depth study of non-myopic acquisition functions, which would be beyond the scope of this paper.

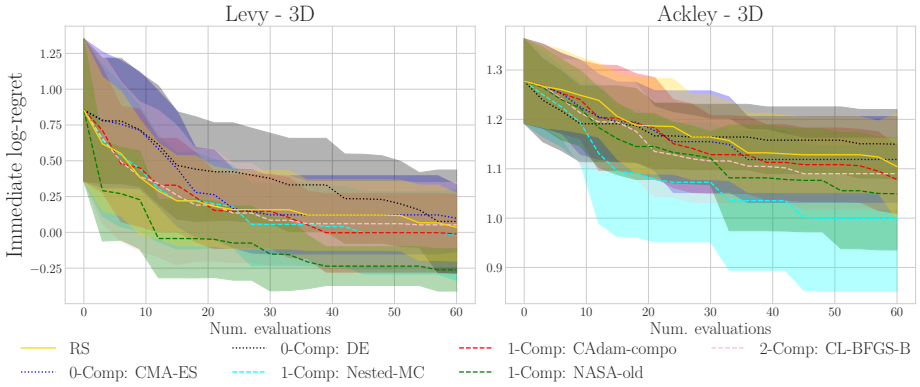

Figure 11: Evolution of immediate log-regret obtained when running Bayesian optimisation with Entropy Search to minimise Levy (left) and Ackley (right) functions. Each line is associated with an optimiser used to perform the acquisition function maximisation step that produces a new batch of 3 points to evaluate. The immediate log-regret averaged over 10 seeds appears in solid (Random Search), dotted (0-order optimisers) or dashed (order-1 and order-2 optimisers) line and the standard deviation appears as shaded area. We see that first-order optimisers tend to attain lower regrets.

G.2 Results presentation

When comparing acquisition function optimisation approaches on synthetic BBO tasks, we showed regret evolutions aggregating results obtained across several synthetic objective functions, acquisition functions and optimisers and plotted the mean surrounded by one standard deviation. Due to the variability of the tasks over which we aggregated the results this variance is usually very large (see Figures 1, 5) leading to uncertainty as whether the discrepancies observed between the mean curves corresponds to statistically significant performance gaps. On the other hand, we note that we could alternatively aggregate these experiments by first averaging regrets across the BBO tasks and therefore taking into account solely the variability induced by the random seeds. We compare these two aggregation methods in Figure 12, which shows normalised regrets across all synthetic tasks, and in Figure 12m which reports performances achieved when using ERM and FSM formulations of the acquisition functions. In both cases the variability is drastically reduced when only randomness from the seeds is considered, which indicates that most of the variability comes from the variety of the tasks.

From the summary results of Figure 12 we can make several observations regarding the final normalised log-regret achieved by each type of optimisers, notably that (c.1) on average the best first-order compositional optimiser outperforms best first-order non-compositional optimiser. Secondly, we observe that (c.2) the latter achieves lower regrets than second-order optimisers. We test the statistical significance of these claims along with similar comparisons between categories of optimisers by conducting one-tailed *Z-test* when we take into account task variability and one-tailed *t-test* when we only take seed variability into

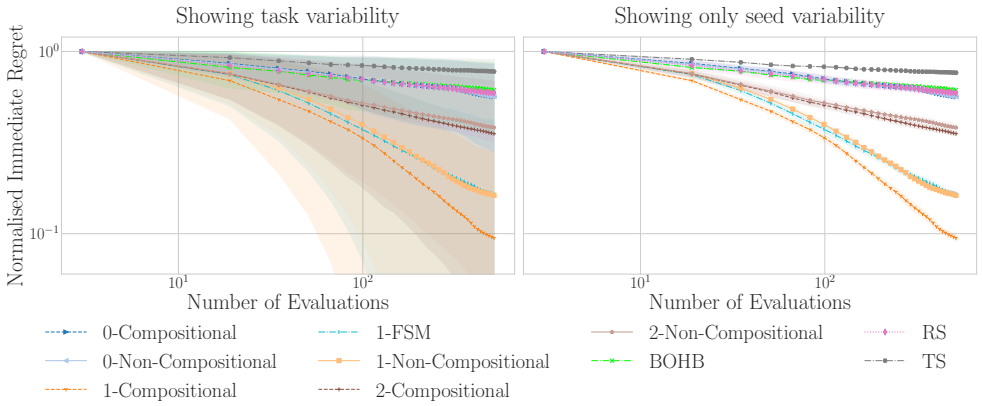

Figure 12: Summary plots for 3100 synthetic BBO experiments. On the left, the variability across the BBO tasks is taken into account to compute the standard deviation inducing a large shaded areas (showing ±1 std). On the right, normalised immediate regrets are averaged across all the tasks and only the variability across the 5 random seeds is shown as a shaded area surrounding the mean curve.

H_0	H_a	p-value (w/ task var.)	p-value (w/o task var.)
1-Comp = 1-Non-Comp	1-Comp < 1-Non-Comp	**8.28e-03**	**4.45e-08**
1-Non-Comp = 2-Comp	1-Non-Comp < 2-Comp	**2.14e-07**	**6.69e-10**
2-Comp = 2-Non-Comp	2-Comp < 2-Non-Comp	2.62e-01	**1.51e-02**
2-Non-Comp = 0-Comp	2-Non-Comp < 0-Comp	**4.74e-06**	**2.75e-07**
0-Comp = 0-RS	0-Comp < RS	2.28e-01	5.71e-02

Table 2: Statistical significance of the observations made on summary Figures 12: the p-values computed from upper-tailed *Z-test* when task variability is considered and from upper-tailed *t-test* when only seed variability is measured. The null-hypothesis H_0 (resp. alternative hypothesis H_a) formulated as $\text{Category}_a = \text{Category}_b$ (resp. $\text{Category}_a < \text{Category}_b$) reads "On average, final normalised log-regrets achieved using best optimiser of Category_a is equal (resp. is lower) than the one obtained using best optimiser of Category_b". We highlight the *p-values* lower than 0.05 in **bold font** denoting rejection of H_0.

account, as we only observe 5 samples for each category). The *p-values* are reported in Table 2 for several hypotheses and allow to conclude to the statistical significance of the claims (c.1) and (c.2) even when task variability is considered.

Similarly we assess statistical significance of the outperformance of FSM over ERM versions of the acquisition functions which we studied in Section 4.1. Again, we perform one-tailed *Z-tests* and *t-tests* on hypotheses comparing final regrets achieved when using ERM and FSM forms with four first-order optimisers (Adam, Adagrad, RMSprop, and Rprop) and reports the associated *p-values* in Table 3. Based on the 960 synthetic BBO experiments carried out, we can conclude that for half of the optimisers (Adagrad and Rprop) when we include task variability in the analysis. For all optimisers, if we focus on

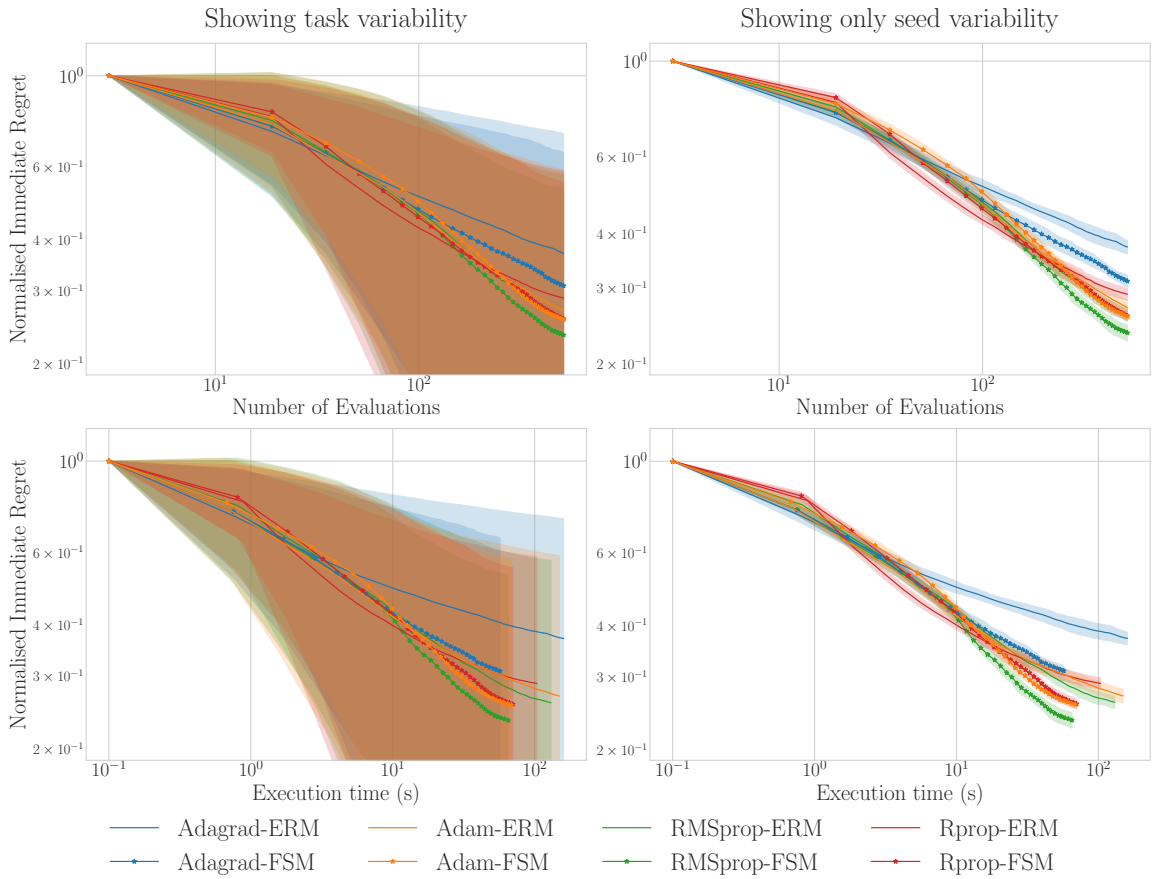

Figure 13: Summary plots comparing the evolution of the normalised immediate regret obtained across all synthetic tasks when using first-order methods with either the ERM or FSM formulation of the acquisition function. First column displays the standard deviation obtained when considering the variability across the BBO tasks, while on the second column, normalised immediate regrets, is averaged across tasks and only the variability across the 5 random seeds is shown as a shaded area surrounding the mean curve. Normalised regret is plotted as a function of the number of evaluations on the top row, while on the bottom row the x-axis corresponds to the execution time. The plot highlights that FSM incurs lower execution runtime than its ERM counterpart.

the average performance across tasks, FSM allows achieving significantly lower log-regret than ERM. Nevertheless, this significant benefit of FSM over ERM may be mitigated by a longer execution time.

To address this potential concern, we showed evolution of normalised log-regret as a function of the execution time on the two bottom plots of Figure 13. We notice that using FSM form does not induce any penalty in terms of run-time and even allows to marginally speed-up the optimisation process, making the case for FSM form even stronger.

Optimiser (op.)	H_0	H_a	p-value (w/ task var.)	p-value (w/o task var.)
Adam	op-ERM = op-FSM	op-ERM > op-FSM	2.53e-1	**3.14e-02**
Adagrad	op-ERM = op-FSM	op-ERM > op-FSM	**1.25e-03**	**3.42e-05**
RMSprop	op-ERM = op-FSM	op-ERM > op-FSM	8.27e-02	**3.15e-03**
Rprop	op-ERM = op-FSM	op-ERM > op-FSM	**3.57e-02**	**3.86e-04**

Table 3: Statistical significance of the observations made on summary Figures 13: the *p-values* are computed from lower-tailed *Z-test* when task variability is considered and from lower-tailed *t-test* when only seed variability is measured. The null-hypothesis H_0 (resp. alternative hypothesis H_a) formulated as op-ERM = op-FSM (resp. op-ERM < op-FSM) reads "On average, final normalised log-regrets achieved using best optimiser of op-ERM is equal (resp. is greater) than the one obtained using best optimiser of op-FSM". We highlight the *p-values* lower than 0.05 in **bold font** denoting rejection of H_0

G.3 Synthetic tasks

G.3.1 Immediate log-regret across tasks

We provide in Figures 14, 15, 16, 17 and 18 the evolution of immediate regrets obtained using each optimiser and acquisition function on synthetic black-box maximisation tasks in 16, 40, 60, 80 and 120 dimensions. These results are summarised in Table 1.

G.3.2 Inner optimisation

Despite our interest in acquisition function optimisation, we compare performance based on the evolution of immediate regrets achieved in the course of BO rather than focusing on the visualisation of the acquisition function optimisation trajectories. This choice is motivated by the fact that this interest for the BO inner optimisation step is meaningful when it is intimately linked to the global BO process, since in practice the goal will always be to achieve the lowest possible regret for a given black-box function. Therefore analysing regret curves when using different inner optimisation processes allows to make comparisons in line with the core motivation of this work — that is to shed light on the impact of the choice of the inner optimisation process to get the best BO performance (which of course should be highly correlated to the quality of the inner optimisation). Moreover, having this objective in mind, it then becomes difficult to compare inner optimisation performances directly, since a valid comparison requires sharing similar initial points which is generally not possible beyond the first acquisition step. An alternative consists in probing some points in the search space on which we fit one surrogate model that is used to test all acquisition optimisation methods. We provide an example of such experiment on Figure 20 showing acquisition values trajectories across 64 optimisation steps when taking 3, 50, 250 and 500 initial random points (we use *Powell* test function in 16D to get the corresponding blackbox values and fit the surrogate model common to all optimisers). As 32 starting points have been used for each optimisation, only the trajectories leading to the highest acquisition values are shown. Although some observations can be made from this figure

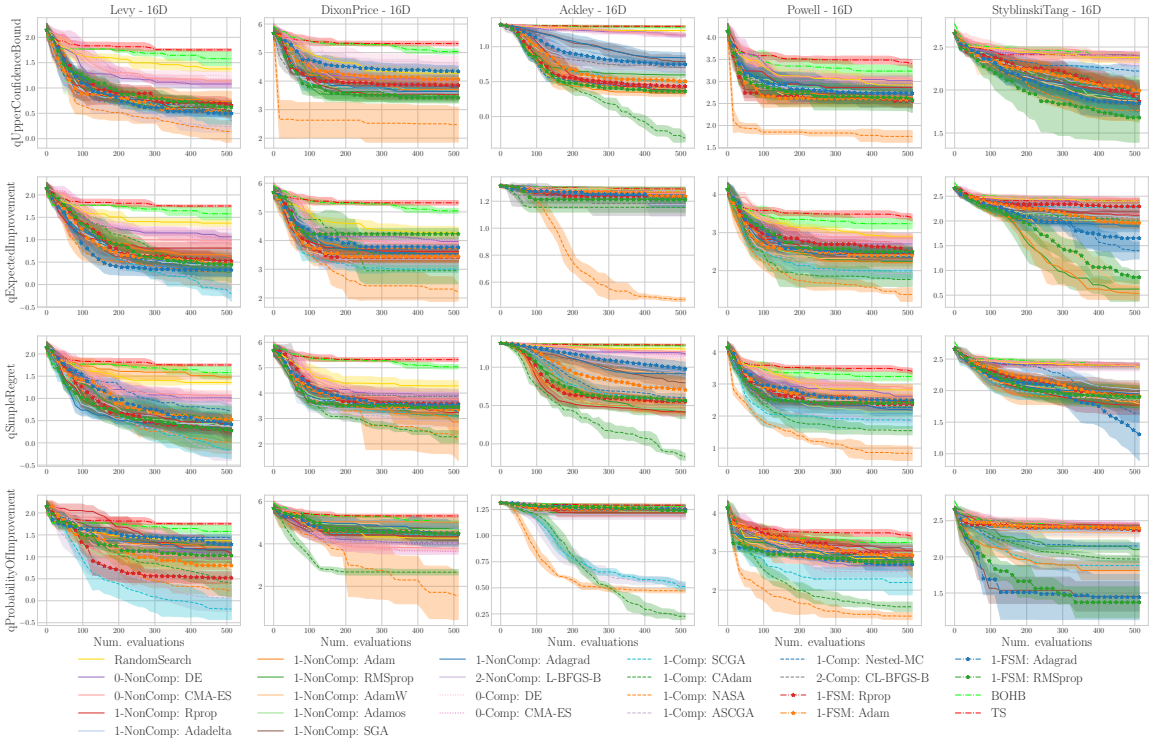

Figure 14: Evolution of immediate log-regret for various acquisition functions and optimisers on 16D synthetic function maximisation tasks. Results of 490 experiments are shown on this graph.

as we notice that compositional first-order forms tend to outperform when working with PI (which corroborates former observations, e.g. regret curves of Figure 6) and achieve highest acquisition values half of the time when optimising EI. We believe that no solid conclusions can be drawn from such graph, as the surrogate models built from random points are certainly not representative of the surrogate models encountered in the course of BO where evaluated points have been properly selected.

G.4 Hyperparameter tuning tasks

As described on Section 4.5, we compared optimisers performances on a set of real-world tasks taken from the Bayesmark data sets. Figures 21, 22, 23, 24 show show the evolution of the scores obtained by each optimiser for all tasks.

Appendix H. Hyperparameter Settings

As explained in Section 4, the performances of first-order optimisers that are reported have been obtained after a hyperparameter tuning phase. We show in Table 4 and 5 the hyperparameters that have been tuned for each optimiser, along with their tuning domains.

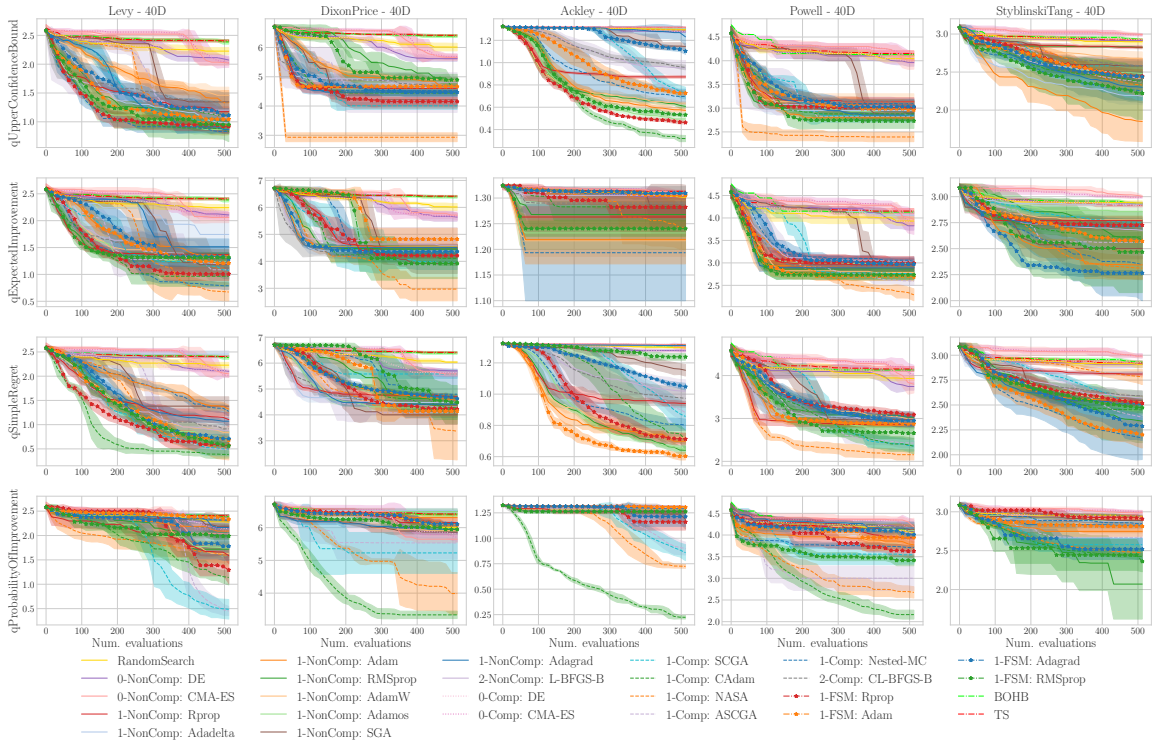

Figure 15: Evolution of immediate log-regret for various acquisition functions and optimisers on 40D synthetic function maximisation tasks. Results of 490 experiments are shown on this graph.

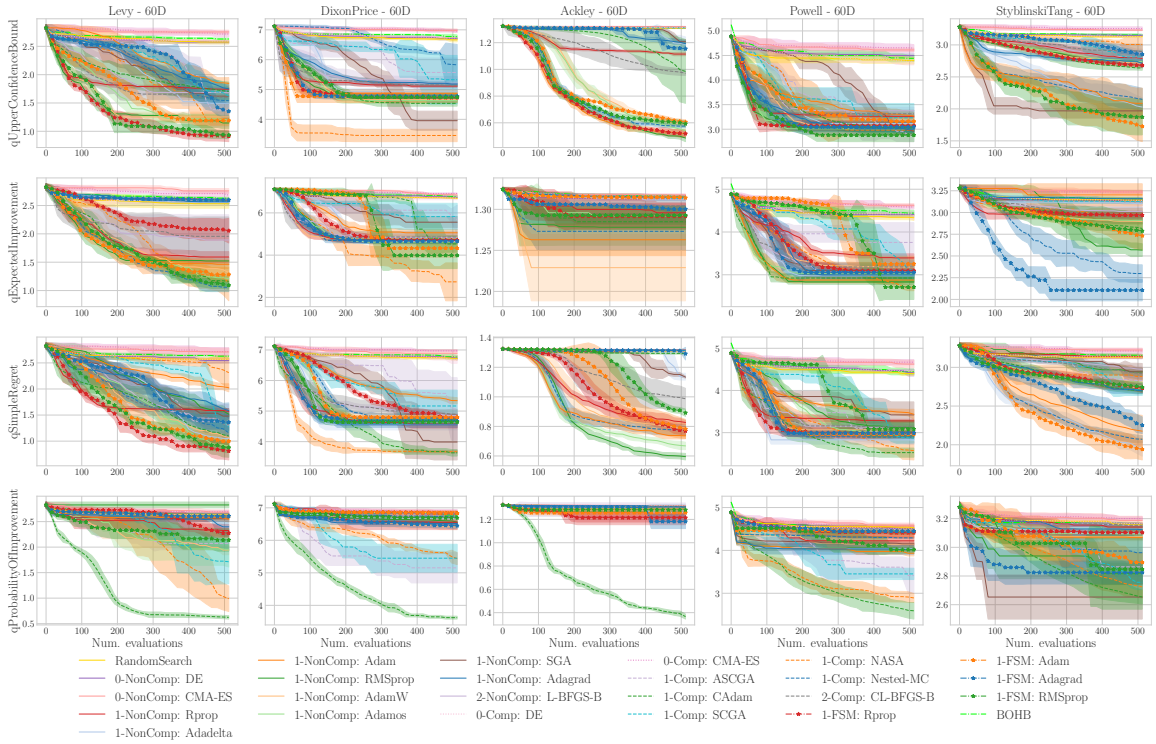

Figure 16: Evolution of immediate log-regret for various acquisition functions and optimisers on 60D synthetic function maximisation tasks. Results of 485 experiments are shown on this graph.

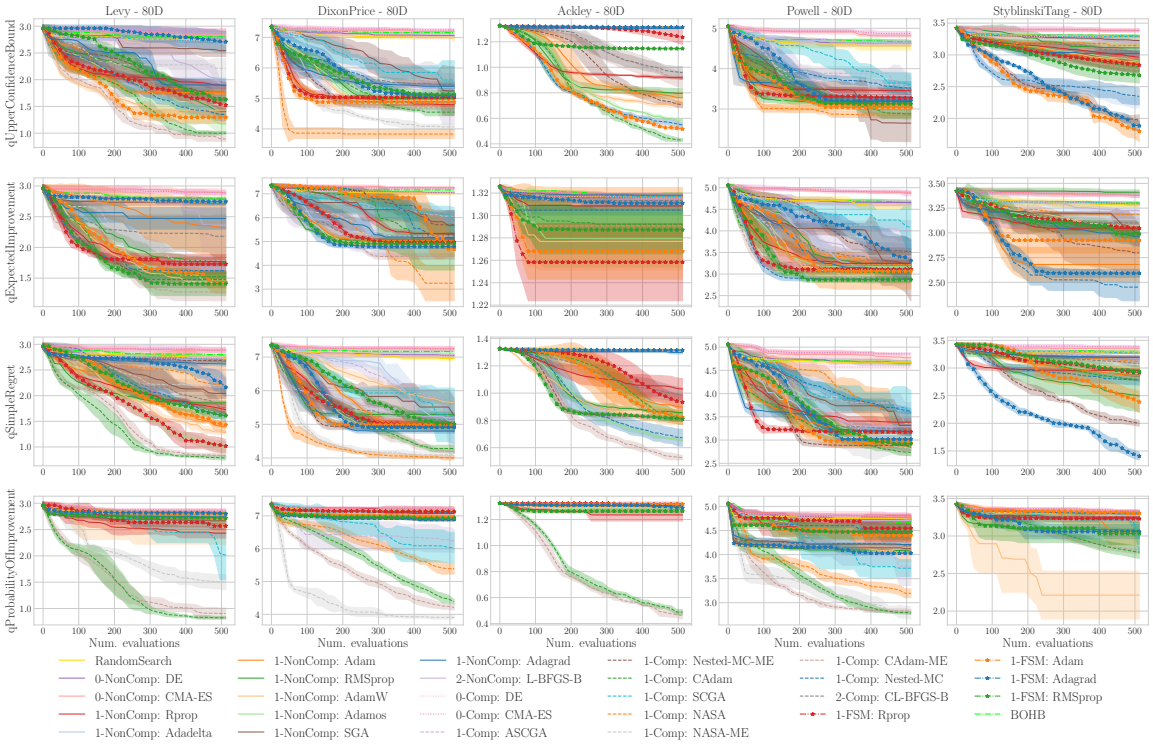

Figure 17: Evolution of immediate log-regret for various acquisition functions and optimisers on 80D synthetic function maximisation tasks. Results of 545 experiments are shown on this graph.

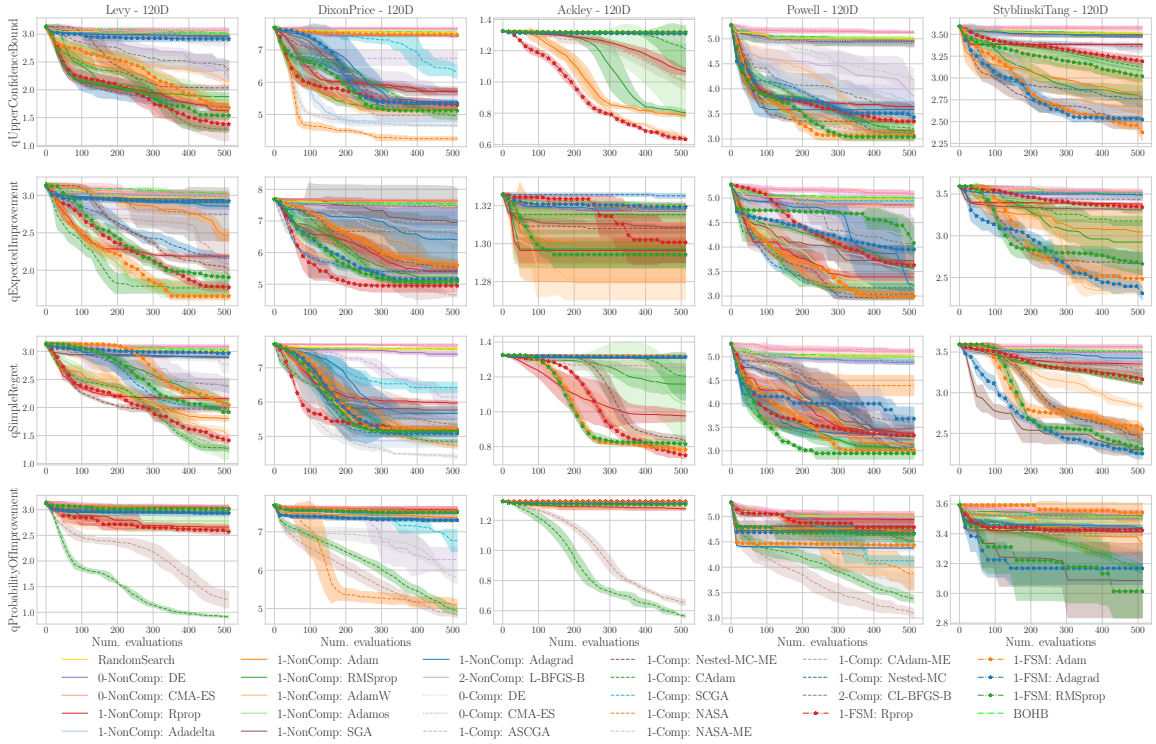

Figure 18: Evolution of immediate log-regret for various acquisition functions and optimisers on 120D synthetic function maximisation tasks. Results of 545 experiments are shown on this graph.

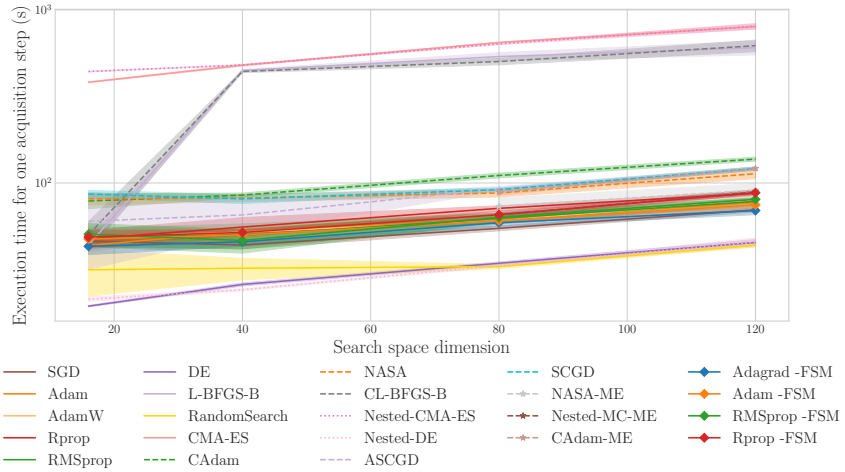

Figure 19: Execution time of UCB maximisation run on 4 CPUs measured for each optimiser on 2 synthetic black-box maximisation tasks in 16, 40, 80 and 120 dims, amounting to 184 BO experiments.

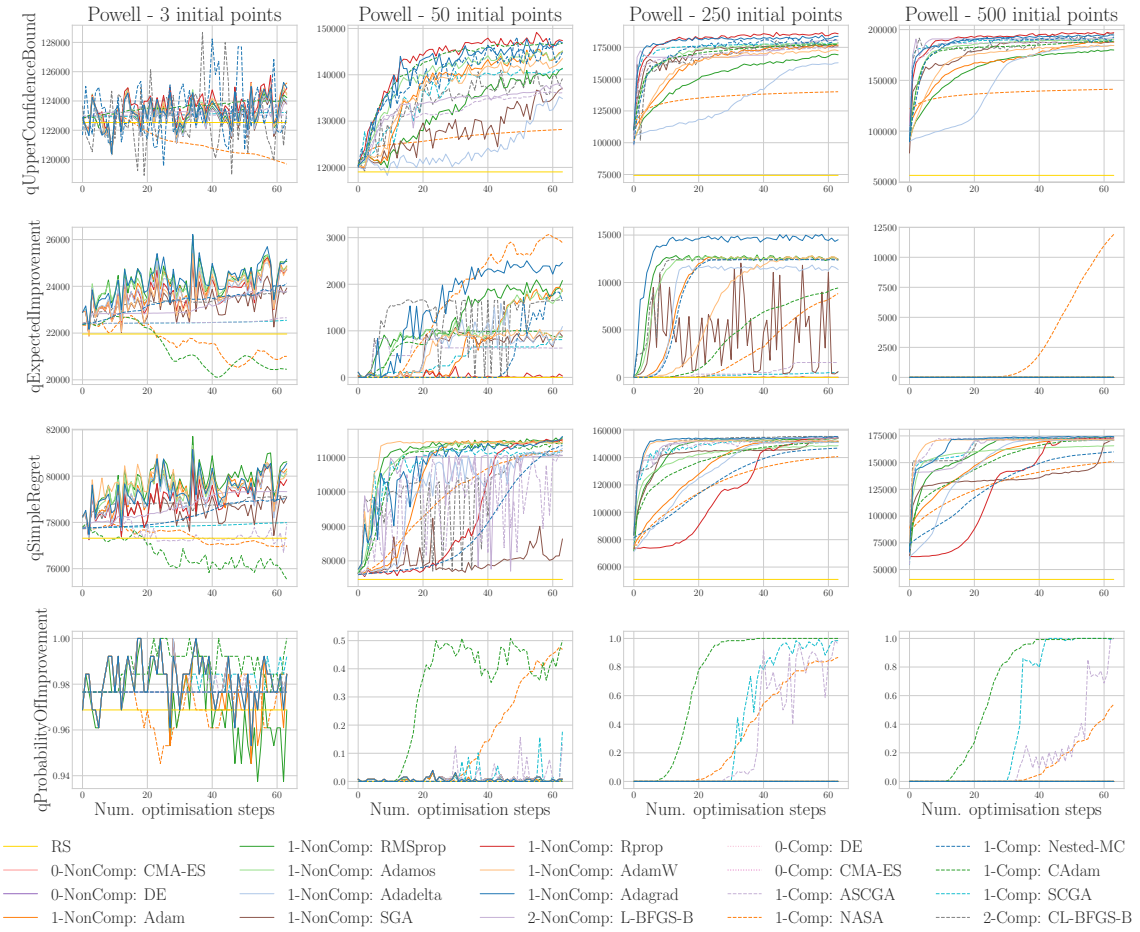

Figure 20: **Powell 16D.** Evolution of the acquisition function values over the course of 64 optimisation steps. Each columns corresponds to a number of random initial points considered to fit the surrogate model (these points are the same for all optimisers and does not necessarily reflects the distribution of the points that would be observed if these points had been selected through BO) and each row is associated to an acquisition function. As 32 starting points are initially selected to perform acquisition function optimisation, we only show for each optimiser the trajectory yielding the highest acquisition value at the end of the 64 steps.

Tuning has been performed using the BO library GPyOpt with a total of 33 configurations tested for each optimiser.

Optimiser	Reference	Parameters	Tuning domain	Scheduled
Adam	App. C.4	lr	$\mathcal{LU}(10^{-5}, 0.3)$	✓
		β_1	$\mathcal{U}(0.05, 0.999)$	
		β_2	$\mathcal{LU}(0.9, 1-10^{-6})$	
		w_d	$\mathcal{LU}(10^{-8}, 0.1)$	
AdamW	App. C.7	lr	$\mathcal{LU}(10^{-5}, 0.3)$	✓
		β_1	$\mathcal{U}(0.05, 0.999)$	
		β_2	$\mathcal{LU}(0.9, 1-10^{-6})$	
		w_d	$\mathcal{LU}(10^{-8}, 0.1)$	
Adadelta	App. C.5	lr	$\mathcal{LU}(10^{-5}, 0.3)$	✓
		ρ	$\mathcal{LU}(0, 0.999)$	
		w_d	$\mathcal{LU}(10^{-8}, 0.1)$	
Adagrad	App. C.2	lr	$\mathcal{LU}(10^{-5}, 0.3)$	
		lr_d	$\mathcal{LU}(10^{-7}, 10)$	
		δ	$\mathcal{LU}(10^{-8}, .3)$	
		w_d	$\mathcal{LU}(10^{-8}, 0.1)$	
SGA	App. C.1	lr	$\mathcal{LU}(10^{-5}, 0.3)$	✓
		ρ	$\mathcal{U}(0, 1)$	
		Δ	$\mathcal{U}(0, 1)$	
		nesterov	$\mathcal{U}\{0, 1\}$	
		w_d	$\mathcal{LU}(10^{-8}, 0.1)$	
Rprop	App. C.6	lr	$\mathcal{LU}(10^{-5}, 0.3)$	✓
		η_1	$\mathcal{U}(0, 1)$	
		η_2	$\mathcal{U}(1, 3)$	
RMSprop	App. C.3	lr	$\mathcal{LU}(10^{-5}, 0.3)$	✓
		ρ	$\mathcal{U}(0, 1)$	
		α	$\mathcal{LU}(10^{-6}, .3)$	
		centering	$\mathcal{U}\{0, 1\}$	
		w_d	$\mathcal{LU}(10^{-8}, 0.1)$	
Adamos	App. C.8	lr	$\mathcal{LU}(10^{-5}, 1.)$	
		μ	$\mathcal{LU}(0.1, 0.999)$	
		C_γ	$\mathcal{U}(0.5, 1)$	
		α_d	$\mathcal{U}(0.02, 0.5)$	
		μ_d	$\mathcal{U}(0.8, 1.2)$	
		γ_{2d}	$\mathcal{U}(0.2, 0.8)$	

Table 4: Selected first-order non-compositional optimisers used in our experiments together with their tuning domains. Searches in a log-uniform domain are denoted by $\mathcal{LU}$, while searches in a uniform continuous (resp. discrete) domain is denoted by $\mathcal{U}()$ (resp. $\mathcal{U}\{\}$). The learning rate scheduling, marked with the ✓ sign in the last column, is an exponential decay schedule with a multiplicative factor γ tuned in $\mathcal{LU}(10^{-7}, 0.3)$ along with the other optimiser hyperparameters.

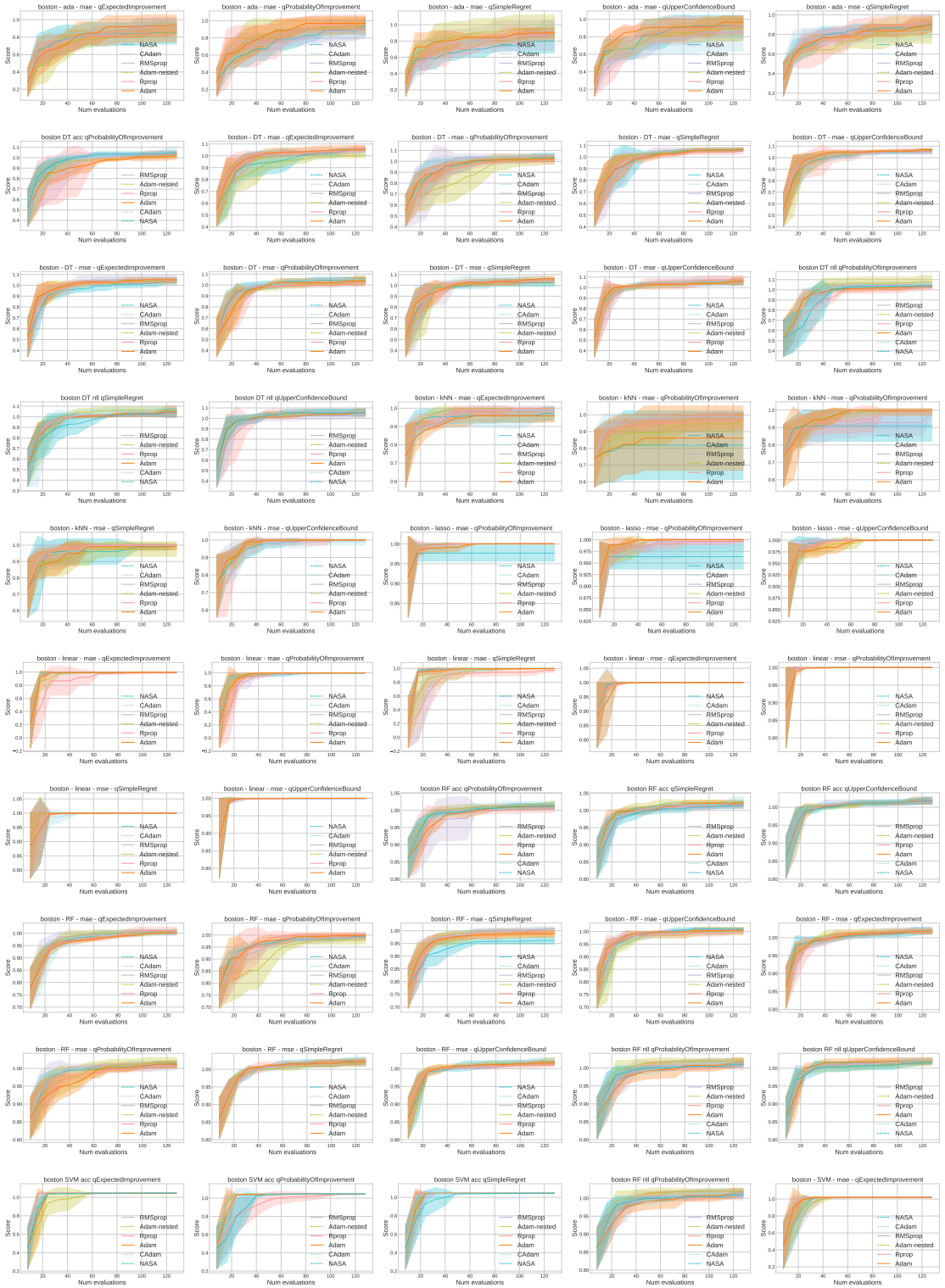
boston - ada - mae - qExpectedImprovement
boston - ada - mae - qProbabilityOfImprovement
boston - ada - mae - qSimpleRegret
boston - ada - mae - qUpperConfidenceBound
boston - ada - mse - qSimpleRegret
boston DT acc qProbabilityOfImprovement
boston - DT - mae - qExpectedImprovement
boston - DT - mae - qProbabilityOfImprovement
boston - DT - mae - qSimpleRegret
boston - DT - mae - qUpperConfidenceBound
boston - DT - mse - qExpectedImprovement
boston - DT - mse - qProbabilityOfImprovement
boston - DT - mse - qSimpleRegret
boston - DT - mse - qUpperConfidenceBound
boston DT nll qProbabilityOfImprovement
boston DT nll qSimpleRegret
boston DT nll qUpperConfidenceBound
boston - kNN - mae - qExpectedImprovement
boston - kNN - mae - qProbabilityOfImprovement
boston - kNN - mse - qProbabilityOfImprovement
boston - kNN - mse - qSimpleRegret
boston - kNN - mse - qUpperConfidenceBound
boston - lasso - mae - qProbabilityOfImprovement
boston - lasso - mse - qProbabilityOfImprovement
boston - lasso - mse - qUpperConfidenceBound
boston - linear - mae - qExpectedImprovement
boston - linear - mae - qProbabilityOfImprovement
boston - linear - mae - qSimpleRegret
boston - linear - mse - qExpectedImprovement
boston - linear - mse - qProbabilityOfImprovement
boston - linear - mse - qSimpleRegret
boston - linear - mse - qUpperConfidenceBound
boston RF acc qProbabilityOfImprovement
boston RF acc qSimpleRegret
boston RF acc qUpperConfidenceBound
boston - RF - mae - qExpectedImprovement
boston - RF - mae - qProbabilityOfImprovement
boston - RF - mae - qSimpleRegret
boston - RF - mae - qUpperConfidenceBound
boston - RF - mse - qExpectedImprovement
boston - RF - mse - qProbabilityOfImprovement
boston - RF - mse - qSimpleRegret
boston - RF - mse - qUpperConfidenceBound
boston RF nll qProbabilityOfImprovement
boston RF nll qUpperConfidenceBound
boston SVM acc qExpectedImprovement
boston SVM acc qProbabilityOfImprovement
boston SVM acc qSimpleRegret
boston RF nll qProbabilityOfImprovement
boston - SVM - mae - qExpectedImprovement
Score
Num evaluations
NASA
CAdam
RMSprop
Adam-nested
Rprop
Adam

Figure 21

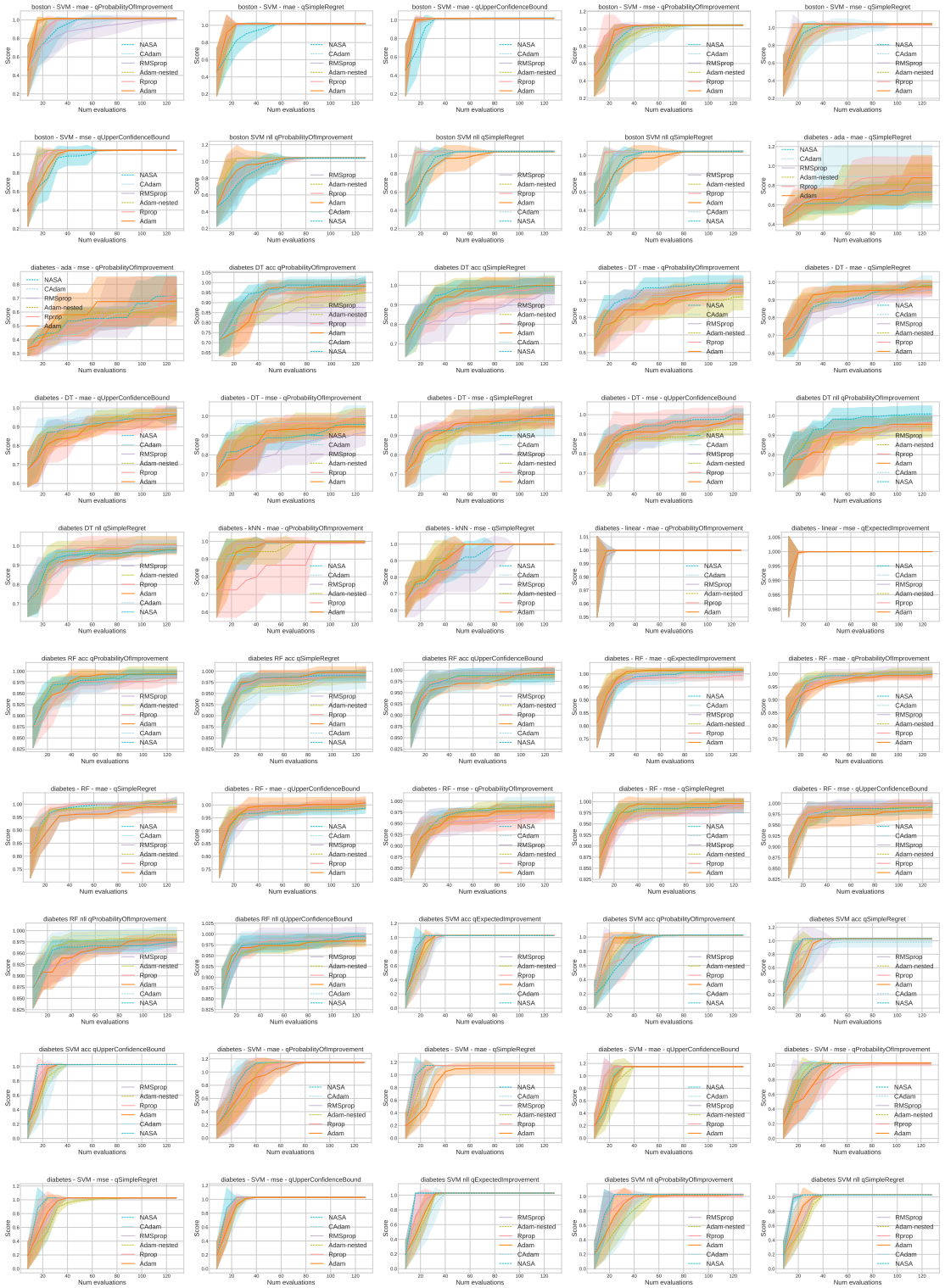
boston - SVM - mae - qProbabilityOfImprovement
boston - SVM - mae - qSimpleRegret
boston - SVM - mae - qUpperConfidenceBound
boston - SVM - mse - qProbabilityOfImprovement
boston - SVM - mse - qSimpleRegret
boston - SVM - mse - qUpperConfidenceBound
boston SVM nll qProbabilityOfImprovement
boston SVM nll qSimpleRegret
boston SVM nll qSimpleRegret
diabetes - ada - mae - qSimpleRegret
diabetes - ada - mse - qProbabilityOfImprovement
diabetes DT acc qProbabilityOfImprovement
diabetes DT acc qSimpleRegret
diabetes - DT - mae - qProbabilityOfImprovement
diabetes - DT - mae - qSimpleRegret
diabetes - DT - mae - qUpperConfidenceBound
diabetes - DT - mse - qProbabilityOfImprovement
diabetes - DT - mse - qSimpleRegret
diabetes - DT - mse - qUpperConfidenceBound
diabetes DT nll qProbabilityOfImprovement
diabetes DT nll qSimpleRegret
diabetes - kNN - mae - qProbabilityOfImprovement
diabetes - kNN - mse - qSimpleRegret
diabetes - linear - mae - qProbabilityOfImprovement
diabetes - linear - mse - qExpectedImprovement
diabetes RF acc qProbabilityOfImprovement
diabetes RF acc qSimpleRegret
diabetes RF acc qUpperConfidenceBound
diabetes - RF - mae - qExpectedImprovement
diabetes - RF - mae - qProbabilityOfImprovement
diabetes - RF - mae - qSimpleRegret
diabetes - RF - mae - qUpperConfidenceBound
diabetes - RF - mse - qProbabilityOfImprovement
diabetes - RF - mse - qSimpleRegret
diabetes - RF - mse - qUpperConfidenceBound
diabetes RF nll qProbabilityOfImprovement
diabetes RF nll qUpperConfidenceBound
diabetes SVM acc qExpectedImprovement
diabetes SVM acc qProbabilityOfImprovement
diabetes SVM acc qSimpleRegret
diabetes SVM acc qUpperConfidenceBound
diabetes - SVM - mae - qProbabilityOfImprovement
diabetes - SVM - mae - qSimpleRegret
diabetes - SVM - mae - qUpperConfidenceBound
diabetes - SVM - mse - qProbabilityOfImprovement
diabetes - SVM - mse - qSimpleRegret
diabetes - SVM - mse - qUpperConfidenceBound
diabetes SVM nll qExpectedImprovement
diabetes SVM nll qProbabilityOfImprovement
diabetes SVM nll qSimpleRegret
Score
Num evaluations
NASA
CAdam
RMSprop
Adam-nested
Rprop
Adam

Figure 22

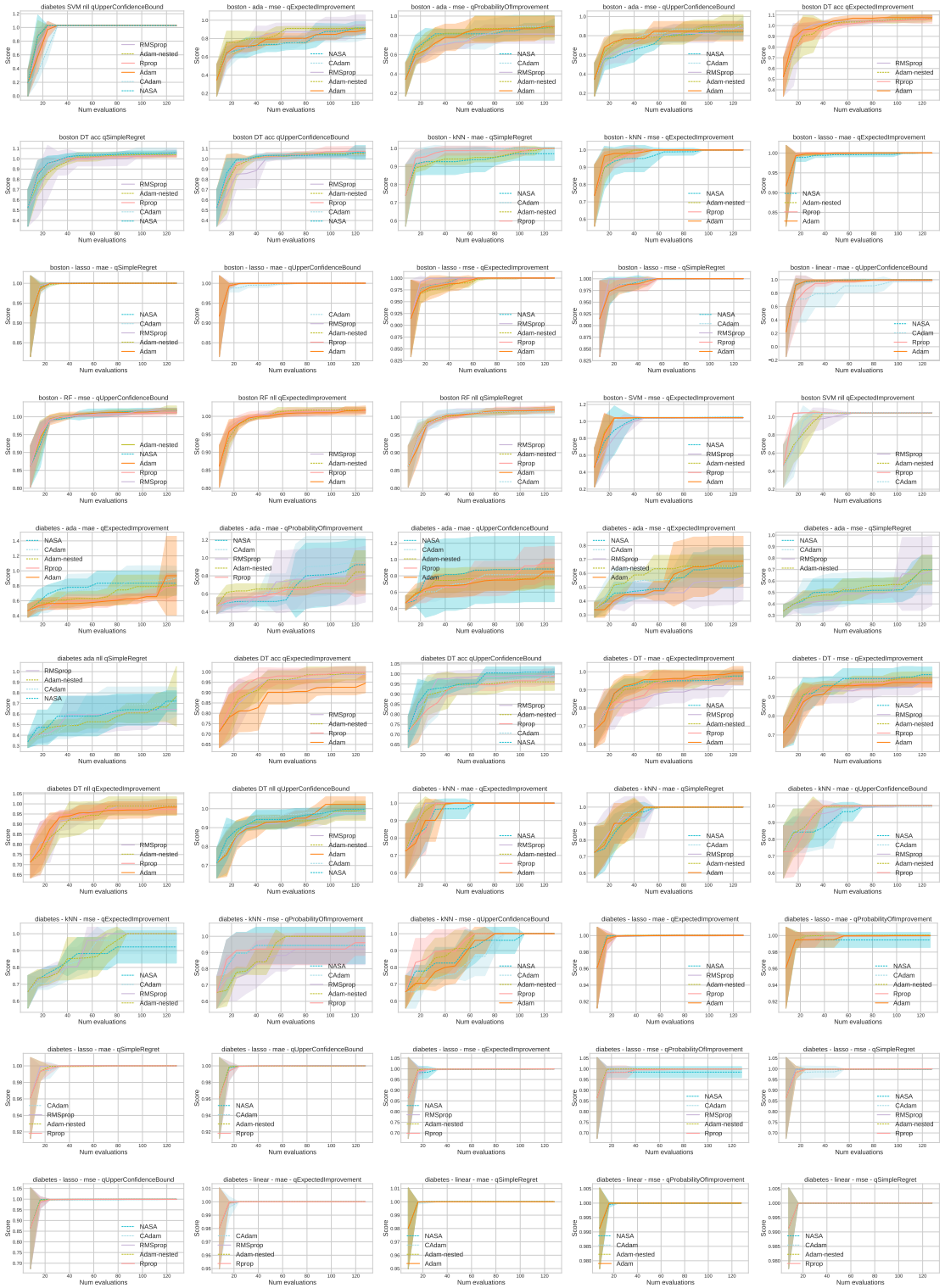
diabetes SVM nll qUpperConfidenceBound
boston - ada - mse - qExpectedImprovement
boston - ada - mse - qProbabilityOfImprovement
boston - ada - mse - qUpperConfidenceBound
boston DT acc qExpectedImprovement
boston DT acc qSimpleRegret
boston DT acc qUpperConfidenceBound
boston - kNN - mae - qSimpleRegret
boston - kNN - mse - qExpectedImprovement
boston - lasso - mae - qExpectedImprovement
boston - lasso - mae - qSimpleRegret
boston - lasso - mae - qUpperConfidenceBound
boston - lasso - mse - qExpectedImprovement
boston - lasso - mse - qSimpleRegret
boston - linear - mae - qUpperConfidenceBound
boston - RF - mse - qUpperConfidenceBound
boston RF nll qExpectedImprovement
boston RF nll qSimpleRegret
boston - SVM - mse - qExpectedImprovement
boston SVM nll qExpectedImprovement
diabetes - ada - mae - qExpectedImprovement
diabetes - ada - mae - qProbabilityOfImprovement
diabetes - ada - mae - qUpperConfidenceBound
diabetes - ada - mse - qExpectedImprovement
diabetes - ada - mse - qSimpleRegret
diabetes ada nll qSimpleRegret
diabetes DT acc qExpectedImprovement
diabetes DT acc qUpperConfidenceBound
diabetes - DT - mae - qExpectedImprovement
diabetes - DT - mse - qExpectedImprovement
diabetes DT nll qExpectedImprovement
diabetes DT nll qUpperConfidenceBound
diabetes - kNN - mae - qExpectedImprovement
diabetes - kNN - mae - qSimpleRegret
diabetes - kNN - mae - qUpperConfidenceBound
diabetes - kNN - mse - qExpectedImprovement
diabetes - kNN - mse - qProbabilityOfImprovement
diabetes - kNN - mse - qUpperConfidenceBound
diabetes - lasso - mae - qExpectedImprovement
diabetes - lasso - mae - qProbabilityOfImprovement
diabetes - lasso - mae - qSimpleRegret
diabetes - lasso - mae - qUpperConfidenceBound
diabetes - lasso - mse - qExpectedImprovement
diabetes - lasso - mse - qProbabilityOfImprovement
diabetes - lasso - mse - qSimpleRegret
diabetes - lasso - mse - qUpperConfidenceBound
diabetes - linear - mae - qExpectedImprovement
diabetes - linear - mae - qSimpleRegret
diabetes - linear - mse - qProbabilityOfImprovement
diabetes - linear - mse - qSimpleRegret
Score
Num evaluations
RMSprop
Adam-nested
Rprop
Adam
CAdam
NASA

Figure 23

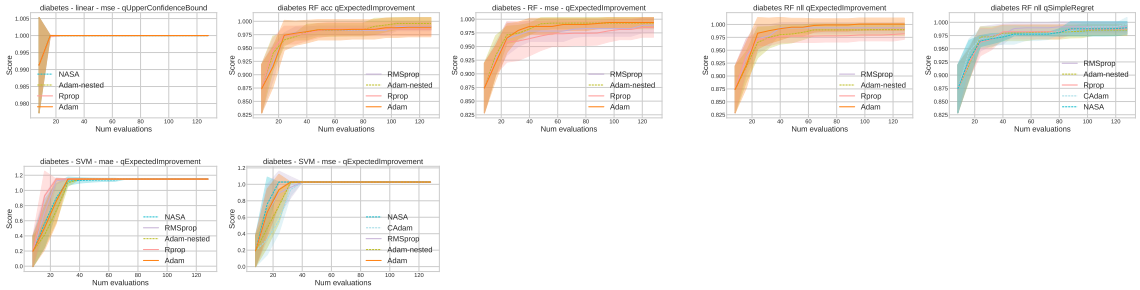

Figure 24

Optimiser	Reference	Parameters	Tuning domain	Scheduled
CAdam	(Tutunov et al., 2020)	lr	$\mathcal{LU}(10^{-5}, 1.)$	
		β	$\mathcal{LU}(0.001, 0.999)$	
		μ	$\mathcal{LU}(0.1, 0.999)$	
		C_γ	$\mathcal{U}(0.5, 1)$	
		α_d	$\mathcal{U}(0.02, 0.5)$	
		μ_d	$\mathcal{U}(0.8, 1.2)$	
		$\gamma 2_d$	$\mathcal{U}(0.2, 0.8)$	
NASA	(Ghadimi et al., 2020)	a	$\mathcal{U}(0.1, 10)$	
		b	$\mathcal{U}(0.1, 10)$	
		β	$\mathcal{U}(0.1, 10)$	
		γ	$\mathcal{U}(0.5, 1.)$	
SCGD	(Wang et al., 2017a)	lr	$\mathcal{LU}(-4, 1)$	
		lr_d	$\mathcal{U}(0.4, .95)$	
		β	$\mathcal{LU}(0.1, 0.999)$	
		β_d	$\mathcal{U}(0.2, 0.8)$	
ASCGD	(Wang et al., 2017a)	lr	$\mathcal{LU}(-4, 1)$	
		lr_d	$\mathcal{U}(0.4, .95)$	
		β	$\mathcal{LU}(0.1, 0.999)$	
		β_d	$\mathcal{U}(0.25, 0.85)$	

Table 5: Selected first-order non-compositional optimisers used in our experiments together with their tuning domains.

References

Mohammed Amin Abdullah, Hang Ren, Haitham Bou Ammar, Vladimir Milenkovic, Rui Luo, Mingtian Zhang, and Jun Wang. Wasserstein robust reinforcement learning. *arXiv preprint arXiv:1907.13196*, 2019.

Shun-ichi Amari. Neural learning in structured parameter spaces-natural Riemannian gradient. In *Advances In Neural Information Processing Systems*, pages 127–133, 1997.

Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10 (2):251–276, 1998.

Shun-ichi Amari. *Differential-Geometrical Methods in Statistics*. Lecture Notes in Statistics. Springer New York, 2012. ISBN 9781461250562. URL `https://books.google.co.uk/books?id=XiDnBwAAQBAJ`.

Shun-ichi Amari and Hiroshi Nagaoka. *Methods of Information Geometry*. Translations of mathematical monographs. American Mathematical Society, 2007. ISBN 9780821843024. URL `https://books.google.co.uk/books?id=vc2FWSo7wLUC`.

Anup Aprem and Stephen Roberts. A Bayesian optimization approach to compute Nash equilibrium of potential games using bandit feedback. *The Computer Journal*, 2018.

Kaito Ariu, Narae Ryu, Se-Young Yun, and Alexandre Proutière. Regret in online recommendation systems. *Advances in Neural Information Processing Systems*, 33, 2020.

Raul Astudillo and Peter Frazier. Bayesian optimization of composite functions. In *International Conference on Machine Learning*, pages 354–363, 2019.

Ajmal Aziz, Edward Elson Kosasih, Ryan-Rhys Griffiths, and Alexandra Brintrup. Data considerations in graph representation learning for supply chain networks, 2021.

Marco Baioletti, Gabriele Di Bari, Alfredo Milani, and Valentina Poggioni. Differential evolution for neural networks optimization. *Mathematics*, 8(1):69, 2020.

Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew Gordon Wilson, and Eytan Bakshy. Botorch: A framework for efficient Monte-Carlo Bayesian optimization. *Advances in Neural Information Processing Systems*, 33, 2020.

Peter J Bentley. *Evolutionary design by computers*. Morgan Kaufmann, 1999.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

Julian. Blank and Kalyanmoy. Deb. Pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. *Advances in Neural Information Processing Systems*, 20:161–168, 2007.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

Anthony Bourached, Ryan-Rhys Griffiths, Robert Gray, Ashwani Jha, and Parashkev Nachev. Generative model-enhanced human motion prediction. *arXiv preprint arXiv:2010.11699*, 2020.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Guy Bresler, Devavrat Shah, and Luis Filipe Voloch. Collaborative filtering with low regret. In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*, pages 207–220, 2016.

Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208, 1995.

Richard H Byrd, Samantha L Hansen, Jorge Nocedal, and Yoram Singer. A stochastic quasi-Newton method for large-scale optimization. *SIAM Journal on Optimization*, 26 (2):1008–1031, 2016.

Roberto Calandra. *Bayesian modeling for optimization and control in robotics*. PhD thesis, Darmstadt, Technische Universität, 2017.

Xi-Ren Cao. Convergence of parameter sensitivity estimates in a stochastic experiment. *IEEE Transactions on Automatic Control*, 30(9):845–853, 1985.

Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework and applications. In *International Conference on Machine Learning*, pages 151–159, 2013.

Bingqing Cheng, Ryan-Rhys Griffiths, Simon Wengert, Christian Kunkel, Tamas Stenczel, Bonan Zhu, Volker L Deringer, Noam Bernstein, Johannes T Margraf, Karsten Reuter, et al. Mapping materials and molecules. *Accounts of Chemical Research*, 53(9):1981–1991, 2020.

Clément Chevalier and David Ginsbourger. Fast computation of the multi-points expected improvement with applications in batch selection. In *International Conference on Learning and Intelligent Optimization*, pages 59–69. Springer, 2013.

Emile Contal, David Buffoni, Alexandre Robicquet, and Nicolas Vayatis. Parallel Gaussian process optimization with upper confidence bound and pure exploration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 225–240. Springer, 2013.

Alberto Costa, Giacomo Nannicini, Thomas Schroepfer, and Thomas Wortmann. Black-box optimization of lighting simulation in architectural design. In *Complex Systems Design & Management Asia*, pages 27–39. Springer, 2015.

Alexander I Cowen-Rivers, Wenlong Lyu, Rasul Tutunov, Zhi Wang, Antoine Grosnit, Ryan Rhys Griffiths, Hao Jianye, Jun Wang, and Haitham Bou Ammar. An empirical study of assumptions in Bayesian optimisation. *arXiv preprint arXiv:2012.03826*, 2020a.

Alexander I Cowen-Rivers, Daniel Palenicek, Vincent Moens, Mohammed Abdullah, Aivar Sootla, Jun Wang, and Haitham Ammar. Samba: Safe model-based & active reinforcement learning. *arXiv preprint arXiv:2006.09436*, 2020b.

John P Cunningham, Philipp Hennig, and Simon Lacoste-Julien. Gaussian probabilities and expectation propagation. *arXiv preprint arXiv:1111.6832*, 2011.

Erik Daxberger, Anastasia Makarova, Matteo Turchetta, and Andreas Krause. Mixed-variable Bayesian optimization. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 2633–2639, 7 2020.

Alexander G De G. Matthews, Mark Van Der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. Gpflow: A Gaussian process library using TensorFlow. *The Journal of Machine Learning Research*, 18(1):1299–1304, 2017.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

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

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61): 2121–2159, 2011.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pages 1437–1446, 2018.

Peter I Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Victor Gabillon, Rasul Tutunov, Michal Valko, and Haitham Bou-Ammar. Derivative-free & order-robust optimisation. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108, pages 2293–2303. PMLR, 2020.

Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems*, 2018.

Eduardo C Garrido-Merchán and Daniel Hernández-Lobato. Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes. *Neurocomputing*, 380:20–35, 2020.

Alan Genz. Numerical computation of multivariate normal probabilities. *Journal of computational and graphical statistics*, 1(2):141–149, 1992.

Alan Genz. Numerical computation of rectangular bivariate and trivariate normal and t probabilities. *Statistics and Computing*, 14(3):251–260, 2004.

Saeed Ghadimi, Andrzej Ruszczynski, and Mengdi Wang. A single timescale stochastic approximation method for nested stochastic optimization. *SIAM Journal on Optimization*, 30(1):960–979, 2020.

David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. A Multi-points Criterion for Deterministic Parallel Global Optimization based on Gaussian Processes. Technical report, March 2008. URL `https://hal.archives-ouvertes.fr/hal-00260579`.

Paul Glasserman. Performance continuity and differentiability in Monte Carlo optimization. In *1988 Winter Simulation Conference Proceedings*, pages 518–524. IEEE, 1988.

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.

Alon Gonen and Shai Shalev-Shwartz. Fast rates for empirical risk minimization of strict saddle problems. In *Conference on Learning Theory*, pages 1043–1063, 2017.

Javier González, Michael Osborne, and Neil Lawrence. Glasses: Relieving the myopia of Bayesian optimisation. In *Artificial Intelligence and Statistics*, pages 790–799. PMLR, 2016.

Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained Bayesian optimization for automatic chemical design using variational autoencoders. *Chemical Science*, 11 (2):577–586, 2020.

Ryan-Rhys Griffiths, Miguel Garcia-Ortegon, Alexander A Aldrick, and Alpha A Lee. Achieving robustness to aleatoric uncertainty with heteroscedastic Bayesian optimisation. *arXiv preprint arXiv:1910.07779*, 2019.

Ryan-Rhys Griffiths, Jiachen Jiang, Douglas J. K. Buisson, Dan Wilkins, Luigi C. Gallo, Adam Ingram, Alpha A. Lee, Dirk Grupe, Erin Kara, Michael L. Parker, William Alston, Anthony Bourached, George Cann, Andrew Young, and S. Komossa. Modeling the multiwavelength variability of Mrk 335 using Gaussian processes. *The Astrophysical Journal*, 914(2):144, jun 2021. doi: 10.3847/1538-4357/abfa9f. URL `https://doi.org/10.3847/1538-4357/abfa9f`.

Jean-Bastien Grill, Michal Valko, and Rémi Munos. Black-box optimization of noisy functions with unknown smoothness. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 667–675, 2015.

Antoine Grosnit, Rasul Tutunov, Alexandre Max Maraval, Ryan-Rhys Griffiths, Alexander I Cowen-Rivers, Lin Yang, Lin Zhu, Wenlong Lyu, Zhitang Chen, Jun Wang, et al. High-dimensional Bayesian optimisation with variational autoencoders and deep metric learning. *arXiv preprint arXiv:2106.03609*, 2021.

Nadav Hallak, Panayotis Mertikopoulos, and Volkan Cevher. Regret minimization in stochastic non-convex learning via a proximal-gradient approach. *arXiv preprint arXiv:2010.06250*, 2020.

Nikolaus Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of IEEE international conference on evolutionary computation*, pages 312–317. IEEE, 1996.

Elad Hazan. Introduction to online convex optimization. *Found. Trends Optim.*, 2(3-4): 157–325, 2016.

Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *The Journal of Machine Learning Research*, 13(1):1809–1837, 2012.

James Hensman, Nicolò Fusi, and Neil D Lawrence. Gaussian processes for Big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 282–290, 2013.

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, 2017.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. Coursera: Neural Networks for Machine Learning, 2012.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, pages 507–523. Springer, 2011a.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*, LION'05, page 507–523, Berlin, Heidelberg, 2011b. Springer-Verlag. ISBN 9783642255656. doi: 10.1007/978-3-642-25566-3_40. URL `https://doi.org/10.1007/978-3-642-25566-3_40`.

Christian Igel, Thorsten Suttorp, and Nikolaus Hansen. A computational efficient covariance matrix update and a (1+ 1)-CMA for evolution strategies. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, pages 453–460, 2006.

Momin Jamil and Xin-She Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 2013.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. In *International Conference on Learning Representations*, 2017.

Grahame A Jastrebski and Dirk V Arnold. Improving evolution strategies through active covariance matrix adaptation. In *2006 IEEE International Conference on Evolutionary Computation*, pages 2814–2821. IEEE, 2006.

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.

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

C.T. Kelley. *Iterative Methods for Optimization*. Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics, 1999. ISBN 9780898714333. URL `https://books.google.co.uk/books?id=Bq6VcmzOe1IC`.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning, 2020.

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *International Conference on Learning Representations*, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Nicolas Knudde, Joachim van der Herten, Tom Dhaene, and Ivo Couckuyt. Gpflowopt: A Bayesian optimization library using TensorFlow, 2017.

Ksenia Korovina, Sailun Xu, Kirthevasan Kandasamy, Willie Neiswanger, Barnabas Poczos, Jeff Schneider, and Eric Xing. Chembo: Bayesian optimization of small organic molecules with synthesizable recommendations. In *International Conference on Artificial Intelligence and Statistics*, pages 3393–3403. PMLR, 2020.

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(1):97–106, 03 1964.

Manuel Laguna and Rafael Marti. Experimental testing of advanced scatter search designs for global optimization of multimodal functions. *Journal of Global Optimization*, 33: 235–255, 10 2005. doi: 10.1007/s10898-004-1936-z.

Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017.

Prasant Kumar Mahapatra, Susmita Ganguli, and Amod Kumar. A hybrid particle swarm optimization and artificial immune system algorithm for image enhancement. *Soft Computing*, 19(8):2101–2109, 2015.

Mitchell McIntire, Daniel Ratner, and Stefano Ermon. Sparse Gaussian processes for Bayesian optimization. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pages 517–526, 2016.

Thomas P Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 362–369, 2001a.

Thomas P Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001b.

Jonas Močkus. On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404. Springer, 1975.

Aryan Mokhtari and Alejandro Ribeiro. RES: Regularized stochastic BFGS algorithm. *IEEE Transactions on Signal Processing*, 62(23):6089–6104, 2014.

Aryan Mokhtari and Alejandro Ribeiro. Global convergence of online limited memory BFGS. *The Journal of Machine Learning Research*, 16(1):3151–3181, 2015.

Henry Moss, David Leslie, Daniel Beck, Javier Gonzalez, and Paul Rayson. Boss: Bayesian optimization over string spaces. *Advances in Neural Information Processing Systems*, 33, 2020a.

Henry B Moss and Ryan-Rhys Griffiths. Gaussian process molecule property prediction with flowmo. *arXiv preprint arXiv:2010.01118*, 2020.

Henry B Moss, Vatsal Aggarwal, Nishant Prateek, Javier González, and Roberto Barra-Chicote. Boffin tts: Few-shot speaker adaptation by bayesian optimization. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7639–7643. IEEE, 2020b.

Manfred Opper, Ole Winther, et al. From naive mean field theory to the tap equations. *Advanced mean field methods: theory and practice*, pages 7–20, 2001.

Michael A Osborne, Roman Garnett, and Stephen J Roberts. Gaussian processes for global optimization. In *3rd International Conference on Learning and Intelligent Optimization (LION3)*, pages 1–15, 2009.

Art B Owen. Quasi-Monte Carlo sampling. *Monte Carlo Ray Tracing: Siggraph 2003 Course 44*, pages 69–88, 2003.

Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. In *International Conference on Learning Representations*, 2014.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Bo Peng and Lei Li. An improved localization algorithm based on genetic algorithm in wireless sensor networks. *Cognitive Neurodynamics*, 9(2):249–256, 2015.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL `https://www.aclweb.org/anthology/D14-1162`.

Nikolaos Ploskas, Christopher Laughman, Arvind U Raghunathan, and Nikolaos V Sahinidis. Optimization of circuitry arrangements for heat exchangers using derivative-free optimization. *Chemical Engineering Research and Design*, 131:16–28, 2018.

Kenneth V Price. Differential evolution: a fast and simple numerical optimizer. In *Proceedings of North American Fuzzy Information Processing*, pages 524–527. IEEE, 1996.

Carl Edward Rasmussen and Christopher KI Williams. *Gaussian Processes for Machine Learning*, volume 2. MIT press Cambridge, MA, 2006.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning-Volume 32*, pages II–1278, 2014.

Kurt S Riedel. A Sherman–Morrison–Woodbury identity for rank augmenting matrices with application to centering. *SIAM Journal on Matrix Analysis and Applications*, 13 (2):659–662, 1992.

Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The Rprop algorithm. In *IEEE International Conference on Neural Networks*, 1993.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

Binxin Ru, Ahsan S Alvi, Vu Nguyen, Michael A Osborne, and Stephen J Roberts. Bayesian optimisation over multiple continuous and categorical inputs. *arXiv preprint arXiv:1906.08878*, 2019.

Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

Robin M Schmidt, Frank Schneider, and Philipp Hennig. Descending through a crowded valley–benchmarking deep learning optimizers. *arXiv preprint arXiv:2007.01547*, 2020.

Günther Schrack and Mark Choit. Optimized relative step size random searches. *Mathematical Programming*, 10(1):230–244, 1976.

MA Schumer and Kenneth Steiglitz. Adaptive step size random search. *IEEE Transactions on Automatic Control*, 13(3):270–276, 1968.

Shweta B Shah and Nikolaos V Sahinidis. SAS-Pro: Simultaneous residue assignment and structure superposition for protein structure alignment. *PloS one*, 7(5):e37493, 2012.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 1(104):148–175, 2016.

Shai Shalev-Shwartz and Yoram Singer. Online learning: Theory, algorithms, and applications. 2007.

David F Shanno. Conditioning of quasi-Newton methods for function minimization. *Mathematics of computation*, 24(111):647–656, 1970.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959, 2012.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable Bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180. PMLR, 2015a.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2171–2180, Lille, France, 07–09 Jul 2015b. PMLR. URL `http://proceedings.mlr.press/v37/snoek15.html`.

A. Song, Q. Hu, X. Ding, X. Di, and Z. Song. Similar face recognition using the ie-cnn model. *IEEE Access*, 8:45244–45253, 2020. doi: 10.1109/ACCESS.2020.2978938.

Jialin Song, Yuxin Chen, and Yisong Yue. A general framework for multi-fidelity Bayesian optimization with Gaussian processes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3158–3167. PMLR, 2019.

Dmitrii V Speranskii. Ant colony optimization algorithms for digital device diagnostics. *Automatic Control and Computer Sciences*, 49(2):82–87, 2015.

Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust Bayesian neural networks. In *Advances in Neural Information Processing Systems*, pages 4134–4142, 2016.

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

Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 2012.

David G. Stork, Anthony Bourached, George H. Cann, and Ryan-Rhys Griffiths. Computational identification of significant actors in paintings through symbols and attributes, 2021.

Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao. A survey of optimization methods from a machine learning perspective. *IEEE Transactions on Cybernetics*, 50(8):3668–3681, 2019.

Aditya R Thawani, Ryan-Rhys Griffiths, Arian Jamasb, Anthony Bourached, Penelope Jones, William McCorkindale, Alexander A Aldrick, and Alpha A Lee. The photoswitch dataset: A molecular machine learning benchmark for the advancement of synthetic chemistry. *arXiv preprint arXiv:2008.03226*, 2020.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

T. Tieleman and G. Hinton. Lecture 6.5—RMSprop: Divide the gradient by a running average of its recent magnitude. Coursera: Neural Networks for Machine Learning, 2012.

Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574, 2009.

Ryan Turner, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. *arXiv preprint arXiv:2104.10201*, 2021.

Rasul Tutunov, Haitham Bou-Ammar, and Ali Jadbabaie. Distributed SDDM solvers: Theory & applications. *arXiv preprint arXiv:1508.04096*, 2015.

Rasul Tutunov, Haitham Bou-Ammar, and Ali Jadbabaie. Distributed Newton method for large-scale consensus optimization. *IEEE Transactions on Automatic Control*, 64(10): 3983–3994, 2019.

Rasul Tutunov, Minne Li, Jun Wang, and Haitham Bou-Ammar. Compositional ADAM: An adaptive compositional solver. *arXiv preprint arXiv:2002.03755*, 2020.

Michal Valko, Alexandra Carpentier, and Rémi Munos. Stochastic simultaneous optimistic optimization. In *International Conference on Machine Learning*, pages 19–27, 2013.

Sander van Rijn, Hao Wang, Matthijs van Leeuwen, and Thomas Bäck. Evolving the structure of evolution strategies. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2016.

Paolo Viappiani and Craig Boutilier. Regret-based optimal recommendation sets in conversational recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, pages 101–108, 2009.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

Jialei Wang, Scott C Clark, Eric Liu, and Peter I Frazier. Parallel Bayesian global optimization of expensive functions. *Operations Research*, 2020.

Mengdi Wang and Ji Liu. A stochastic compositional gradient method using Markov samples. In *2016 Winter Simulation Conference (WSC)*, pages 702–713. IEEE, 2016.

Mengdi Wang, Ethan X Fang, and Han Liu. Stochastic compositional gradient descent: algorithms for minimizing compositions of expected-value functions. *Mathematical Programming*, 161(1-2):419–449, 2017a.

Mengdi Wang, Ji Liu, and Ethan X Fang. Accelerating stochastic composition optimization. *The Journal of Machine Learning Research*, 18(1):3721–3743, 2017b.

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

Colin White, Willie Neiswanger, and Yash Savani. BANANAS: Bayesian optimization with neural architectures for neural architecture search. *arXiv preprint arXiv:1910.11858*, 2019.

Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning, 2018a.

James Wilson, Frank Hutter, and Marc Deisenroth. Maximizing acquisition functions for Bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 9884–9895, 2018b.

James T Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Peter Deisenroth. Efficiently sampling functions from Gaussian process posteriors. In *International Conference on Machine Learning*, 2020.

Peilun Wu, Hui Guo, and Nour Moustafa. Pelican: A deep residual network for network intrusion detection, 2020.

Yaodong Yang, Rasul Tutunov, Phu Sakulwongtana, and Haitham Bou Ammar. α^α-Rank: Practically scaling α-rank through stochastic optimisation. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1575–1583, 2020.

Mingzhang Yin and Mingyuan Zhou. Semi-implicit variational inference. In *International Conference on Machine Learning*, pages 5660–5669, 2018.

Kwang-Seon Yoo and Seog-Young Han. Modified ant colony optimization for topology optimization of geometrically nonlinear structures. *International Journal of Precision Engineering and Manufacturing*, 15(4):679–687, 2014.

Matthew D Zeiler. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Zhuhong Zhang, Lei Wang, and Fei Long. Immune optimization approach solving multi-objective chance-constrained programming. *Evolving Systems*, 6(1):41–53, 2015.

Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560, 1997.