GLASSES: Relieving The Myopia Of Bayesian Optimisation

Javier González
University of Sheffield
Dept. of Computer Science &
Chem. and Biological Engineering
j.h.gonzalez@sheffield.ac.uk

Michael Osborne
University of Oxford
Dept. of Engineering Science
mosb@robots.ox.ac.uk

Neil D. Lawrence
University of Sheffield
Dept. of Computer Science
n.lawrence@sheffield.ac.uk

Abstract

We present GLASSES: Global optimisation with Look-Ahead through Stochastic Simulation and Expected-loss Search. The majority of global optimisation approaches in use are myopic, in only considering the impact of the next function value; the non-myopic approaches that do exist are able to consider only a handful of future evaluations. Our novel algorithm, GLASSES, permits the consideration of dozens of evaluations into the future. This is done by approximating the ideal *look-ahead* loss function, which is expensive to evaluate, by a cheaper alternative in which the future steps of the algorithm are simulated beforehand. An Expectation Propagation algorithm is used to compute the expected value of the loss. We show that the far-horizon planning thus enabled leads to substantive performance gains in empirical tests.

1 Introduction

Global optimisation is core to any complex problem where design and choice play a role. Within Machine Learning, such problems are found in the tuning of hyperparameters [?], sensor selection [?] or experimental design [??]. Most global optimisation techniques are myopic, in considering no more than a single step into the future. Relieving this myopia requires solving the *multi-step lookahead* problem: the global optimisation of an function by considering the significance of the next function evaluation on function evaluations (steps) further into the future. It is clear that a solution to the problem would offer performance gains. For example, consider the case in which we have a budget of two evaluations with which to optimise a function $f(x)$ over the domain $\mathcal{X} = [0, 1] \subset \mathbb{R}$. If we are strictly myopic, our first evaluation will likely be at $x = 1/2$, and our second then at only one of $x = 1/4$ and $x = 3/4$. This myopic strategy will thereby result in ignoring half of the domain $\mathcal{X}$, regardless of the second choice. If we adopt a two-step lookahead approach, we will select function evaluations that will be more evenly distributed across the domain by the time the budget is exhausted. We will consequently be better informed about f and its optimum.

There is a limited literature on the multi-step lookahead problem. **?** perform multi-step lookahead by optimising future evaluation locations, and sampling over future function values. This approach scales poorly with the number of future evaluations considered, and the authors present results for no more than two-step lookahead. [?] reframe the multi-step lookahead problem as a partially observed Markov decision process, and adopt a Monte Carlo tree search approach in solving it. Again, the scaling of the approach permits the authors to consider no more than six steps into the future. In the past, the multi-step look ahead problem was studied by **?** proposing a utility function that maximizes the total 'reward' of the algorithm by taking into account the cost of future computations, rather than trying to find the optimum after a fixed number of evaluations.

Interestingly, there exists a link between the multi-step lookahead problem and *batch* Bayesian optimisation [??]. In this later case, batches of locations rather than individual observations are selected in each iteration of the algorithm and evaluated in parallel. When such locations are selected *greedily*, that is, one after the other, the key to selecting good batches relies on the ability of the batch criterion of predicting future steps of the algorithm. In this work we will exploit this

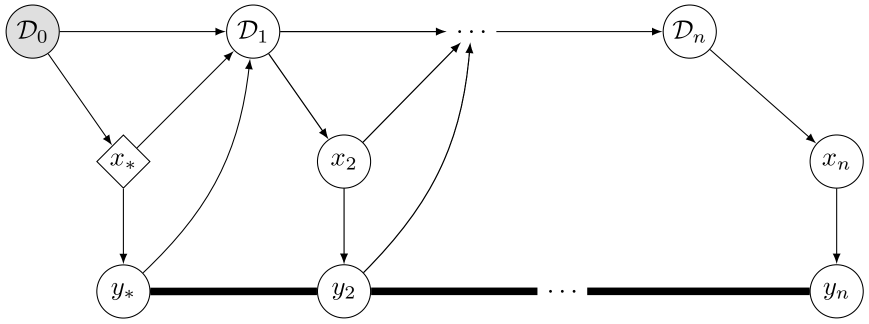


Figure 1: A Bayesian network describing the n-step lookahead problem. The shaded node ($\mathcal{D}_0$) is known, and the diamond node (x_*) is the current decision variable. All y nodes are correlated with one another under the GP model. Note that the nested maximisation problems required for x_i and integration problems required for y_* and y_i (in either case for $i = 2, \ldots, n$) render inference in this model prohibitively computationally expensive.

parallelism to compute a non-myopic loss for Bayesian optimisation.

This paper is organised as follows. In Section 2 we formalise the problem and describe the contributions of this work. Section 3 describe the details of the proposed algorithm. Section 4 illustrates the superior performance of GLASSES in a variety of test functions and we conclude in Section 5 with a discussion about the most interesting results observed in this work.

2 Background and challenge

2.1 Bayesian optimisation with one step look-ahead

Let $f : \mathcal{X} \to \Re$ be well behaved function defined on a compact subset $\mathcal{X} \subseteq \Re^q$. We are interested in solving the global optimization problem of finding

$$\mathbf{x}_M = \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}).$$

We assume that f is a *black-box* from which only perturbed evaluations of the type $y_i = f(\mathbf{x}_i) + \epsilon_i$, with $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, are available. Bayesian Optimization (BO) is an heuristic strategy to make a series of evaluations $\mathbf{x}_1, \ldots, \mathbf{x}_n$ of f, typically very limited in number, such that the the minimum of f is evaluated as soon as possible [**????**].

Assume that N points have been gathered so far, having a dataset $\mathcal{D}_0 = \{(\mathbf{x}_i, y_i)\}_{i=1}^N = (\mathbf{X}_0, \mathbf{y}_0)$. Before collecting any new point, a surrogate probabilistic model for f is calculated. This is typically a Gaussian Process (GP) $p(f) = \mathcal{GP}(\mu; k)$ with mean function μ and a covariance function k, and whose parameters will be denoted by $\boldsymbol{\theta}$. Let $\mathcal{I}_0$ be the current available information: the conjunction of $\mathcal{D}_0$, the model parameters and the model likelihood type. Under Gaussian likelihoods, the predictive distribution for y_* at $\mathbf{x}_*$ is also Gaussian with mean posterior mean and variance

$$\mu(\mathbf{x}_*|\mathcal{I}_0) = \mathbf{k}_{\boldsymbol{\theta}}(\mathbf{X}_*)^\top[\mathbf{k}_{\boldsymbol{\theta}} + \sigma^2\mathbf{I}]^{-1}\mathbf{y} \text{ and}$$

$$\sigma^2(\mathbf{x}_*|\mathcal{I}_0) = k_{\boldsymbol{\theta}}(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_{\boldsymbol{\theta}}(\mathbf{x}_*)^\top[\mathbf{K}_{\boldsymbol{\theta}} + \sigma^2\mathbf{I}]^{-1}\mathbf{k}_{\boldsymbol{\theta}}(\mathbf{x}_*),$$

where $\mathbf{K}_{\boldsymbol{\theta}}$ is the matrix such that $(\mathbf{K}_{\boldsymbol{\theta}})_{ij} = k_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{k}_{\boldsymbol{\theta}}(\mathbf{x}_*) = [k_{\boldsymbol{\theta}}(\mathbf{x}_1, \mathbf{x}_*), \ldots, k_{\boldsymbol{\theta}}(\mathbf{x}_n, \mathbf{x}_*)]^\top$ [**?**].

Given the GP model, we now need to determine the best location to sample. Imagine that we only have one remaining evaluation ($n = 1$) before we need to report our inferred location about the minimum of f. Denote by $\eta = \min\{\mathbf{y}_0\}$, the current best found value. We can define the loss of evaluating f this last time at $\mathbf{x}_*$ assuming it is returning y_* as

$$\lambda(y_*) \triangleq \begin{cases} y_*; & \text{if} \quad y_* \leq \eta \\ \eta; & \text{if} \quad y_* > \eta. \end{cases}$$

Its expectation is

$$\Lambda_1(\mathbf{x}_*|\mathcal{I}_0) \triangleq \mathbb{E}[\min(y_*, \eta)] = \int \lambda(y_*) p(y_*|\mathbf{x}_*, \mathcal{I}_0) \mathrm{d}y_*$$

where the subscript in Λ refers to the fact that we are considering one future evaluation. Giving the properties of the GP, $\Lambda_1(\mathbf{x}_*|\mathcal{I}_0)$ can be computed in closed form for any $\mathbf{x}_* \in \mathcal{X}$. In particular, for Φ the usual Gaussian cumulative distribution function, we have that

$$\begin{aligned} \Lambda_1(\mathbf{x}_*|\mathcal{I}_0) &\triangleq \eta \int_\eta^\infty \mathcal{N}(y_*; \mu, \sigma^2) \mathrm{d}y_* \qquad (1) \\ &+ \int_{-\infty}^\eta y_* \mathcal{N}(y_*; \mu, \sigma^2) \mathrm{d}y_* \\ &= \eta + (\mu - \eta)\Phi(\eta; \mu, \sigma^2) - \sigma^2 \mathcal{N}(\eta, \mu, \sigma^2), \end{aligned}$$

where we have abbreviated $\sigma^2(y_*|\mathcal{I}_0)$ as σ^2 and $\mu(y_*|\mathcal{I}_0)$ as μ. Finally, the next evaluation is located where $\Lambda_1(\mathbf{x}_*|\mathcal{I}_0)$ gives the minimum value. This point can be obtained by any gradient descent algorithm since analytical expressions for the gradient and Hessian of $\Lambda_1(\mathbf{x}_*|\mathcal{I}_0)$ exist [**?**].

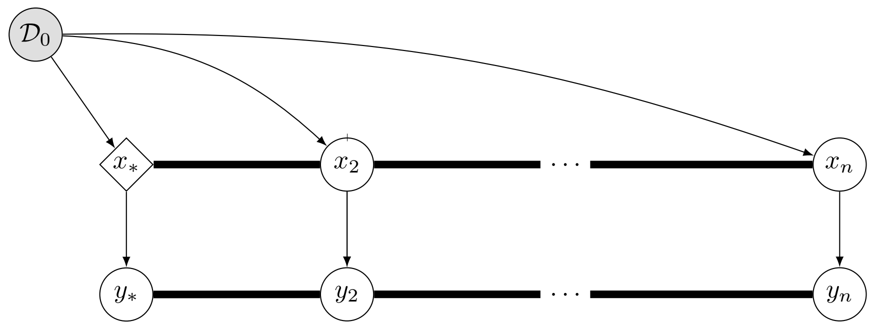


Figure 2: A Bayesian network describing our approximation to the n-step lookahead problem. The shaded node ($\mathcal{D}_0$) is known, and the diamond node (x_*) is the current decision variable, which is now directly connected with all future steps of the algorithm. Compare with Figure 1: the sparser structure renders our approximation computationally tractable.

2.2 Looking many steps ahead

Expression (1) can also be used as a myopic approximation to the optimal decision when n evaluations of f remain available. Indeed, most BO methods are myopic and ignore the future decisions that will be made by the algorithm in the future steps.

Let $\{(\mathbf{x}_j, y_j)\}$ for $j = 1, \ldots, n$ be the remaining n available evaluations and by $\mathcal{I}_j$ the available information after the data set $\mathcal{D}_0$ has been augmented with $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_j, y_j)$ and the parameters $\boldsymbol{\theta}$ of the model updated. We use $\Lambda_n(\mathbf{x}_*|\mathcal{I}_0)$ to denote the expected loss of selecting $\mathbf{x}_*$ given $\mathcal{I}_0$ and considering n future evaluations. A proper Bayesian formulation allows us to define this *long-sight* loss [?] as[1]

$$\Lambda_n(\mathbf{x}_*|\mathcal{I}_0) = \int \lambda(y_n) \prod_{j=1}^{n} p(y_j|\mathbf{x}_j, \mathcal{I}_{j-1}) p(\mathbf{x}_j|\mathcal{I}_{j-1}) \, \mathrm{d}y_* \ldots \mathrm{d}y_n \mathrm{d}\mathbf{x}_2 \ldots \mathrm{d}\mathbf{x}_n \tag{2}$$

where

$$p(y_j|\mathbf{x}_j, \mathcal{I}_{j-1}) = \mathcal{N}\left(y_j; \mu(\mathbf{x}_j; \mathcal{I}_{j-1}), \sigma^2(\mathbf{x}_j|\mathcal{I}_{j-1})\right)$$

is the predictive distribution of the GP at $\mathbf{x}_j$ and

$$p(\mathbf{x}_j|\mathcal{I}_{j-1}) = \delta\big(\mathbf{x}_j - \arg\min_{\mathbf{x}_* \in \mathcal{X}} \Lambda_{n-j+1}(\mathbf{x}_*|\mathcal{I}_{j-1})\big)$$

reflects the optimization step required to obtain $\mathbf{x}_j$ after all previous the evaluations f have been iteratively optimized and marginalized. The graphical probabilistic model underlying (2) is illustrated in Figure 1.

To evaluate Eq. (2) we can successively sample from y_1 to y_{j-1} and optimize for the appropriate $\Lambda_{n-j+1}(\mathbf{x}_*|\mathcal{I}_{j-1})$. This is in done in [?] for only two steps look ahead. The reason why further horizons remain unexplored is the computational burden required to compute this loss for many steps ahead. Note that analytical expression are only available in the myopic case $\Lambda_1(\mathbf{x}_*|\mathcal{I}_0)$.

[1]We assume that $p(\mathbf{x}_*|\mathcal{I}_0) = 1$.

2.3 Contributions of this work

The goal of this work is *to propose a computationally efficient approximation to Eq. (2) for many steps ahead able to relieve the myopia of classical Bayesian optimization.* The contributions of this paper are:

- A new algorithm, GLASSES, to relieve the myopia of Bayesian optimisation that is able to efficiently take into account dozens of steps ahead. The method is based on the prediction of the future steps of the myopic loss to efficiently integrate out a long-side loss.

- The key aspect of our approach is to split the recursive optimization marginalization loop in Eq. (2) into two independent optimisation-marginalization steps that jointly act on all the future steps. We propose an Expectation-Propagation formulation for the joint marginalisation and we discuss different strategies to carry out the optimisation step.

- Together with this work, we deliver a *open source Python code framework*[2] containing a fully functional implementation of the method useful to reproduce the results of this work and applicable in general global optimisation problems. As we mentioned in the introduction of this work, there exist a limited literature in BO non-myopic methods and, to our knowledge, none available generic BO package can be used with myopic loss functions.

[2]http://sheffieldml.github.io/GPyOpt/

- Simulations: New practical experiments and insights that show that non-myopic methods outperform myopic approaches in a benchmark of optimisation problems.

3 The GLASSES algorithm

As detailed in the previous section, a proper multi-step look ahead loss function requires the iterative optimization-marginalization of the future steps, which is computationally intractable. A possible way of dealing with this issue is to jointly model our epistemic uncertainty over the future locations $\mathbf{x}_2, \dots, \mathbf{x}_n$ with a joint probability distribution $p(\mathbf{x}_2, \dots, \mathbf{x}_n | \mathcal{I}_0, \mathbf{x}_*)$ and to consider the expected loss

$$\Gamma_n(\mathbf{x}_*|\mathcal{I}_0) = \int \lambda(y_n) p(\mathbf{y}|\mathbf{X}, \mathcal{I}_0, \mathbf{x}_*) p(\mathbf{X}|\mathcal{I}_0, \mathbf{x}_*) \mathrm{d}\mathbf{y}\mathrm{d}\mathbf{X} \tag{3}$$

for $\mathbf{y} = \{y_*, \dots, \dots, y_n\}$ the vector of future evaluations of f and $\mathbf{X}$ the $(n-1) \times q$ dimensional matrix whose rows are the future evaluations $\mathbf{x}_2, \dots, \mathbf{x}_n$. $p(\mathbf{y}|\mathbf{X}, \mathcal{I}_0, \mathbf{x}_*)$ is multivariate Gaussian, since it corresponds to the predictive distribution of the GP at $\mathbf{X}$. The graphical probabilistic model underlying (2) is illustrated in Figure 2. $\Gamma_n(\mathbf{x}_*|\mathcal{I}_0)$ differs from $\Lambda_n(\mathbf{x}_*|\mathcal{I}_0)$ in the fact that all future evaluations are modelled jointly rather then sequentially. A proper choice of $p(\mathbf{X}|\mathcal{I}_0, \mathbf{x}_*)$ is crucial here. An interesting option would be to choose $p(\mathbf{X}|\mathcal{I}_0, \mathbf{x}_*)$ to be some determinantal point process (DPP)[3] defined on $\mathcal{X}$ [?] and integrate Eq. (3) with respect to $\mathbf{x}_2, \dots, \mathbf{x}_n$ by averaging over multiple samples [??]. DPPs provide a density over sample locations that induces them to be dissimilar to each other (as well-spaced samples should), but that can be concentrated in chosen regions (such as regions of low myopic expected loss). However, although DPPs have nice computational properties in discrete sets, here we would need to take samples from the DPP by conditioning on $\mathbf{x}_*$ and the number of steps ahead. Although this is possible in theory, the computational burden of generating these samples will make this strategy impractical.

An alternative and more efficient approach, that we explore here, is to work with a fixed $\mathbf{X}$, which we assume it is computed beforehand. As we show in this section, although this approach does not make use of our epistemic uncertainty on the future steps, it drastically reduces the computational burden of approximating $\Lambda_n(\mathbf{x}_*|\mathcal{I}_0)$.

[3] A determinantal point process is a probability measure over sets that is entirely characterised by the determinant of some (kernel) function.

3.1 Oracle multiple steps look-ahead expected loss

Suppose that we had access to an oracle function $\mathcal{F}_n : \mathcal{X} \to \mathcal{X} \times \overset{n}{\cdots} \times \mathcal{X}$ able to predict the n future locations that the loss $\Lambda_n(\cdot)$ would suggest if we started evaluating f at $\mathbf{x}_*$. In other words, $\mathcal{F}_n$ takes the putative location $\mathbf{x}_*$ as input and it returns $\mathbf{x}_*$ and the predicted future locations $\mathbf{x}_2, \dots, \mathbf{x}_n$. We work here under the assumption that the oracle has perfect information about the future locations, in the same way we have have perfect information about the locations that the algorithm already visited. This is an unrealistic assumption in practice, but it will help us to set-up our algorithm. We leave for the next section the details of how to marginalise over $\mathcal{F}_n$.

Assume, for now, that $\mathcal{F}_n$ exists and that we have access to it. We it and denote by $\mathbf{y} = (y_*, \dots, \dots, y_n)^T$ the vector of future locations evaluations of f at $\mathcal{F}_n(\mathbf{x}_*)$. Assuming that $\mathbf{y}$ is known, it is possible to rewrite the expected loss in Eq. (2) as

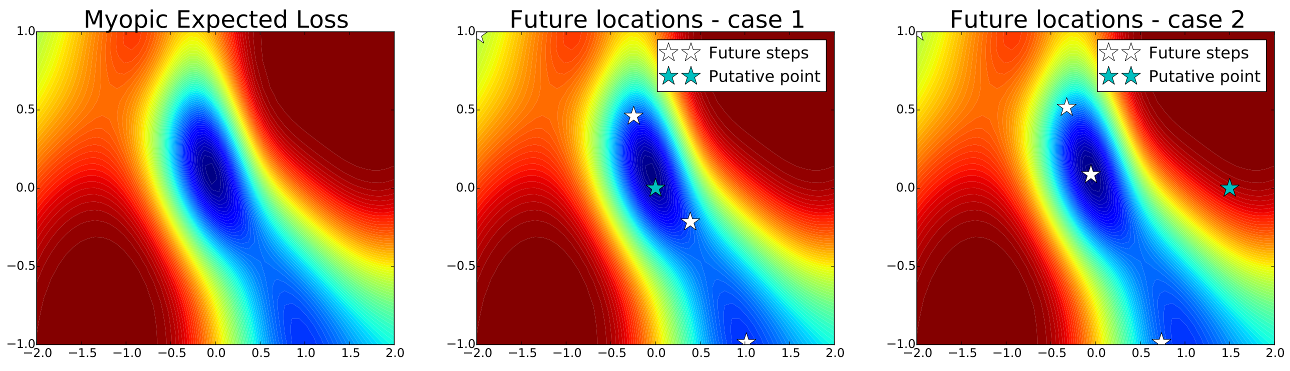
$$\Lambda_n\big(\mathbf{x}_* \mid \mathcal{I}_0, \mathcal{F}_n(\mathbf{x}_*)\big) = \mathbb{E}\big[\min(\mathbf{y}, \eta)\big], \tag{4}$$

where the expectation is taken over the multivariate Gaussian distribution, with mean vector μ and covariance matrix Σ, that gives rise after marginalizing the posterior distribution of the GP at $\mathcal{F}_n(\mathbf{x}_*)$. Note that under a fixed $\mathcal{F}_n(\mathbf{x}_*)$, it also holds that $\Lambda_n\big(\mathbf{x}_* \mid \mathcal{I}_0, \mathcal{F}_n(\mathbf{x}_*)\big) = \Gamma_n\big(\mathbf{x}_* \mid \mathcal{I}_0, \mathcal{F}_n(\mathbf{x}_*)\big)$. See supplementary materials for details.

The intuition behind Eq. (4) is as follows: the expected loss at $\mathbf{x}_*$ is the best possible function value that we expect to find in the next n steps, conditional on the first evaluation being made at $\mathbf{x}_*$. Rather than merely quantifying the benefit provided by the next evaluation, this loss function accounts for the expected gain in the whole future sequence of evaluations of f. As we analyse in the experimental section of this work, the effect of this is an adaptive loss that tends to be more explorative when more remaining evaluations are available and more exploitative as soon as we approach the final evaluations of f.

To compute Eq. (4) we use Expectation Propagation, EP, [?]. This turns out to be a natural operation by observing that

$$\begin{aligned}\mathbb{E}[\min(\mathbf{y}, \eta)] \quad &= \quad \eta \int_{\mathbb{R}^n} \prod_{i=1}^{n} h_i(\mathbf{y}) \mathcal{N}(\mathbf{y}; \mu, \Sigma) \mathrm{d}\mathbf{y} \qquad (5)\\ &+ \quad \sum_{j=1}^{n} \int_{\mathbb{R}^n} y_j \prod_{i=1}^{n} t_{j,i}(\mathbf{y}) \mathcal{N}(\mathbf{y}; \mu, \Sigma) \mathrm{d}\mathbf{y}\end{aligned}$$



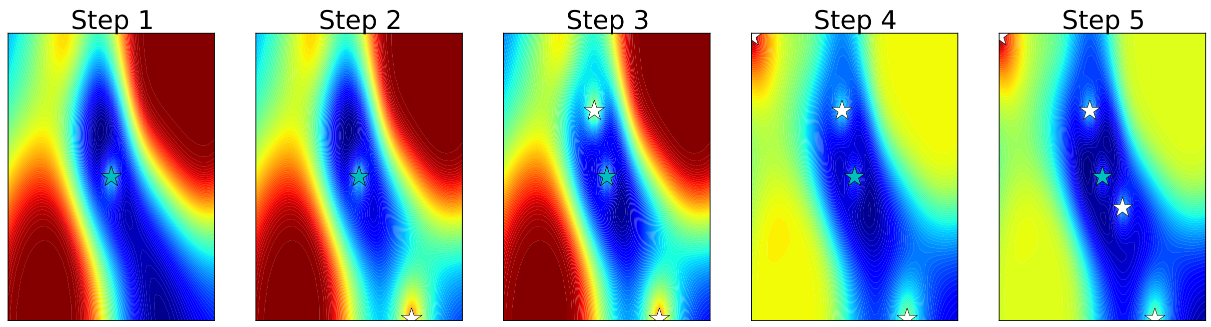


Figure 3: *Top row*: (left) Myopic expected loss computed after 10 observations of the Six-Hump Camel function; (center, case 1) predicted steps ahead when the putative point, grey star, is close to the global optimum of the acquisition; (right, case 2) predicted steps ahead when the putative input is located far from the optimum of the acquisition. *Bottom row*: iterative decision process that predicts that 5 steps look ahead of case 2. Every time a point is selected, the loss is penalised in a neighbourhood, encouraging the next location to be selected far from any previous location point but still in a region where the value of the loss is low.

Algorithm 1 Decision process of the GLASSES algorithm.

```
**Input:** dataset $\mathcal{D}_0 = \{(\mathbf{x}_0, y_0)\}$, number of remaining evaluations ($n$), look-ahead predictor $\mathcal{F}$.
**for** $j = 0$ **to** $n$ **do**
    1. Fit a GP with kernel $k$ to $\mathcal{D}_j$.
    2. Build a predictor of the future $n - l$ evaluations: $\hat{\mathcal{F}}_{n-j}$.
    3. Select next location $\mathbf{x}_j$ by taking $\mathbf{x}_j = \arg\min_{\mathbf{x}\in\mathcal{X}} \Lambda_{n-j}(\mathbf{x}_*|\mathcal{I}_0, \mathcal{F}_{n-j}(\mathbf{x}_*))$.
    4. Evaluate $f$ at $\mathbf{x}_j$ and obtain $y_j$.
    5. Augment the dataset $\mathcal{D}_{j+1} = \{\mathcal{D}_j \cup (\mathbf{x}_j, y_j)\}$.
**end for**
Fit a GP with kernel $k$ to $\mathcal{D}_n$
**Returns**: Propose final location at $\arg\min_{\mathbf{x}\in\mathcal{X}} \{\mu(\mathbf{x}; \mathcal{D}_n)\}$.
```

where $h_i(\mathbf{y}) = \mathbb{I}\{y_i > \eta\}$ and

$$t_{j,i}(\mathbf{y}) = \begin{cases} \mathbb{I}\{y_j \leq \eta\} & \text{if i=j} \\ \mathbb{I}\{0 \leq y_i - y_j\} & \text{otherwise.} \end{cases}$$

See supplementary materials for details. The first term in Eq. (5) is a Gaussian probability on an unbounded polyhedron in which the limits are aligned with the axis. The second term is the sum of the Gaussian expectations on different non-axis-aligned different polyhedra defined by the indicator functions. Both terms can be computed with EP using the approach proposed in [?]. In a nutshell, to compute the integrals it is possible to replace the indicator functions with univariate Gaussians that play the role of *soft-indicators* in the EP iterations. This method is computationally efficient and scales well for high dimensions. Note that when $n = 1$, Eq. (4) reduces to Eq. (1).

Under the hypothesis of this section, the next evaluation is located where $\Lambda_n(\mathbf{x}_*|\mathcal{I}_0, \mathcal{F}_n(\mathbf{x}_*))$ gives the minimum value. We still need, however, to propose a

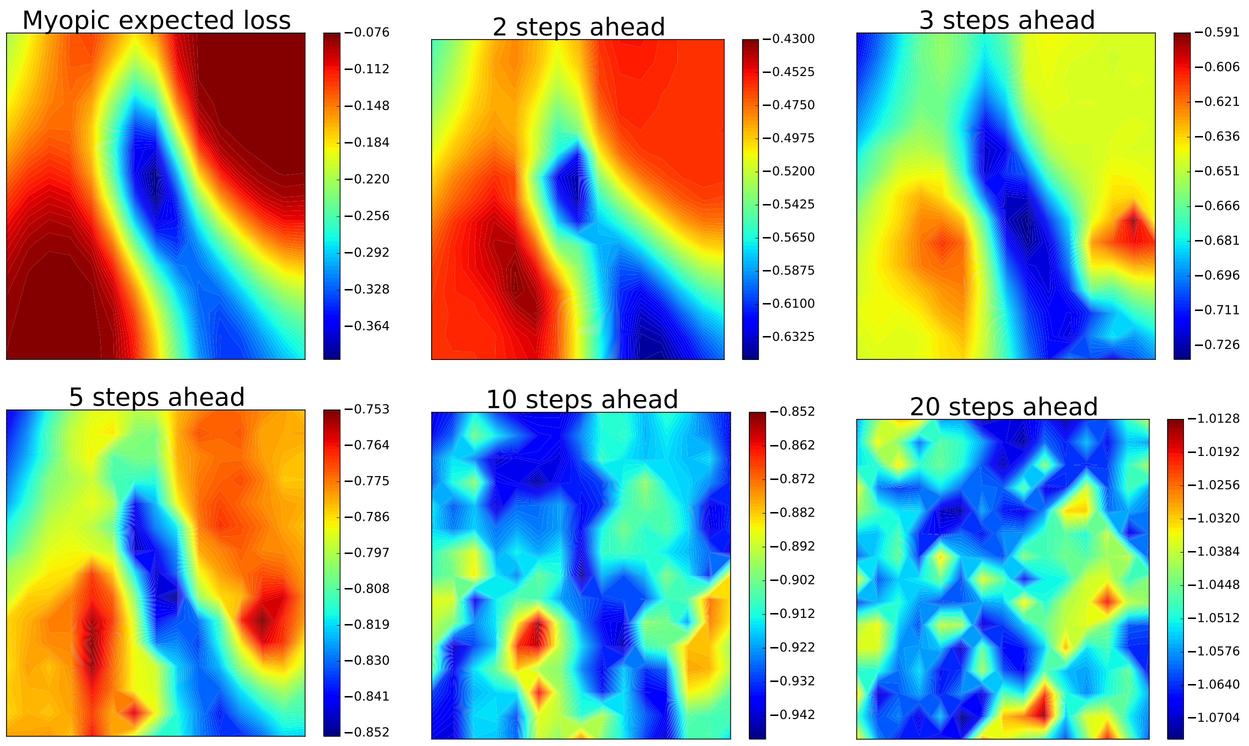


Figure 4: Expected loss for different number of steps ahead in an example with 10 data points and the Six-hump Camel function. Increasing the number of steps-ahead flattens down the loss since it is likely for the algorithm to hit a good location irrespective of the initial point (all candidate points look better because of the future chances of the algorithm to be in a good location).

way to approximate the oracle $\hat{\mathcal{F}}_n(\mathbf{x}_*)$. We do in next section.

3.2 Local Penalisation to Predicting the Steps Ahead

This section proposes an empirical surrogate $\hat{\mathcal{F}}_n(\mathbf{x}_*)$ for $\mathcal{F}_n(\mathbf{x}_*)$. A sensible option would be to use the *maximum a posteriori probability*, MAP, of the above-mentioned DPP. However, as it is the generation of DPP samples, to compute the MAP of a DPP is an expensive operation [?]. Alternatively, here we use some ideas that have been recently developed in the batch Bayesian optimisation literature. In a nutshell, batch BO methods aim to define sets of points in $\mathcal{X}$ where f should be evaluated in parallel, rather than sequentially. In essence, a key aspect to building a 'good' batch is the same as to computing a good approximation for $\Lambda_n(\mathbf{x}_*|\mathcal{I}_0)$: to find a set of good locations at which to evaluate the objective.

In this work we adapt to our context the batch BO idea proposed by **?**. Inspired by the repulsion properties of DPP, **?** propose to build each batch by iteratively optimising and penalising the acquisition function in a neighbourhood of the already collected points by means of some local penalisers $\varphi(\mathbf{x};\mathbf{x}_j)$. Note that any other batch method could be used here, but we consider this approach since it is computationally tractable and scales well with the size of the batches (steps ahead in our context) and the dimensionality of the problem.

More formally, assume that the objective function f is L-Lipschitz continuous, that is, it satisfies that $|f(\mathbf{x}_1) - f(\mathbf{x}_2)| \leq L\|\mathbf{x}_1 - \mathbf{x}_2\|$, $\forall\ \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$. Take $M = \min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$ and valid Lipschitz constant L. It is possible to show that the ball

$$B_{r_j}(\mathbf{x}_j) = \{\mathbf{x} \in \mathcal{X} : \|\mathbf{x}_j - \mathbf{x}\| \leq r_j\} \tag{6}$$

where $r_j = \frac{f(\mathbf{x}_j)-M}{L}$, doesn't contain the minimum of f. Probabilistic versions of these balls are used to define the above mentioned penalisers by noting that $f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$. In particular, $\varphi(\mathbf{x};\mathbf{x}_j)$ is chosen as the probability that $\mathbf{x}$, any point in $\mathcal{X}$ that is a potential candidate to be a minimum, does not belong to $B_{r_j}(\mathbf{x}_j)$: $\varphi(\mathbf{x};\mathbf{x}_j) = 1 - p(\mathbf{x} \in B_{r_j}(\mathbf{x}_j))$. As detailed in **?**, these functions have a closed form and create an exclusion zone around the point $\mathbf{x}_j$. The predicted k-th location when looking at n step ahead and using $\mathbf{x}_*$ as putative point is defined as

$$[\hat{\mathcal{F}}_n(\mathbf{x}_*)]_k = \arg\min_{x\in\mathcal{X}} \left\{ g\big(\Lambda_1(\mathbf{x}_* \mid \mathcal{I}_0)\big) \prod_{j=1}^{k-1} \varphi(\mathbf{x};\hat{\mathbf{x}}_j) \right\}, \tag{7}$$

for $k = 2,\dots,n$ and where $\varphi(\mathbf{x};\mathbf{x}_j)$ are local local penalizers centered at $\mathbf{x}_j$ and $g:\Re\to\Re^+$ is the *soft-plus* transformation $g(z)=\ln(1+e^z)$.

To illustrate how $\hat{\mathcal{F}}_n$ computes the steps ahead we include Figure (3). We show the myopic loss in a BO experiment together with predicted locations by $\hat{\mathcal{F}}_n$ for two different putative points. In case 1, the putative point $\mathbf{x}_*$ (grey star) is close to the location of the minimum of the myopic loss (blue region). In case 2, $\mathbf{x}_*$ is located in an uninteresting region. In both cases the future locations explore the interesting region determined for the myopic loss while the locations of the points are conditioned to the first location. In the bottom row of the figure we show how the points are selected in Case 1. The first putative point creates an exclusion zone that shifts the minimum of the acquisition, where the next location is selected. Iteratively, new locations are found by balancing diversity (due to the effect of the exclusion areas) and quality (exploring locations where the loss is low), similarly to the way samples the probabilities over subsets can be characterised in a DPP [?].

3.3 Algorithm and computational costs

All the steps of GLASSES are detailed in Algorithm 1. The main computational cost is the calculation of the steps ahead, which is done using a sequence of L-BFGS optimizers at $\mathcal{O}(Pq^2)$ complexity for P, the maximum number of L-BFGS updates. The use of EP to compute the value of the expected loss at each location requires a quadratic run time factor update in the dimensionality of each Gaussian factor.

4 Experiments

4.1 Interpreting the non-myopic loss

The goal of this experiment is to visualise the effect on the expected loss of considering multiple steps ahead. To this end, we use six-hump camel function (see Table 4 for details). We fit a GP with a square exponential kernel and we plot the myopic expected loss together with 5 variants that consider 2, 3, 5, 10 and 20 steps ahead. Increasing the steps ahead decreases the optimum value of the loss: the algorithm can visit more locations and the expected minimum is lower. Also, increasing the steps ahead flattens down the loss: it is likely to hit a good location irrespective of the initial point so all candidate looks better because of the future chances of the algorithm to be in a good location. In practice this behaviour translates into an acquisition function that becomes more explorative as we look further ahead.

Name	Function domain	q
SinCos	$[0,10]$	1
Cosines	$[0,1]\times[0,1]$	2
Branin	$[-5,10]\times[-5,10]$	2
Sixhumpcamel	$[-2,2]\times[-1,1]$	2
McCormick	$[-1.5,4]\times[-3,4]$	2
Dropwave	$[-1,1]\times[-1,1]$	2
Beale	$[-1,1]\times[-1,1]$	2
Powers	$[-1,1]\times[-1,1]$	2
Alpine2-q	$[-10,10]^q$	2, 5, 10
Ackley-q	$[-5,5]^q$	2, 5

Table 1: Details of the functions used in the experiments. The explicit form of these functions can be found at `http://www.sfu.ca/~ssurjano/optimization.html`, [?] and the supplementary materials of this work.

4.2 Testing the effect of considering multiple steps ahead

To study the validity of our approximation we choose a variety of functions with a range of dimensions and domain sizes. See Table 1 for details. We use the full GLASSES algorithm (in which at each iteration the number of remaining iterations is used as the number of steps-ahead) and we show the results when 2, 3, 5, and 10 steps look-ahead are used to compute the loss function. Each problem is solved 5 times with different random initialisations of 5 data points. The number of allows evaluations is 10 times the dimensionality of the problem. This allows us to compare the average performance of each method on each problem. As baseline we use the myopic expected loss, EL. For comparative purposes we used other two loss functions are commonly used in the literature. In particular, we use the Maximum Probability of Improvement, MPI, and the GP lower confidence bound, GP-LCB. In this later case we set the parameter that balances exploration and exploitation to 1. See [?] for details on these loss functions. All acquisition functions are optimised using the dividing rectangles algorithm DIRECT [?]. As surrogate model for the functions we used a GP with a squared exponential kernel plus a bias kernel [?]. The hyper-parameters of the model were optimised by the standard method of maximising the marginal likelihood, using L-BFGS [?] for 1,000 iterations and selected the best of 5 random restarts. To compare the methods we used the 'gap' measure of performance [?],

	MPI	GP-LCB	EL	EL-2	EL-3	EL-5	EL-10	GLASSES
SinCos	0.7147	0.6058	0.7645	*0.8656*	0.6027	0.4881	*0.8274*	***0.9000***
Cosines	0.8637	0.8704	0.8161	*0.8423*	*0.8118*	0.7946	0.7477	***0.8722***
Branin	0.9854	0.9616	**0.9900**	0.9856	0.9673	0.9824	0.9887	0.9811
Sixhumpcamel	0.8983	**0.9346**	0.9299	0.9115	0.9067	0.8970	0.9123	0.8880
Mccormick	**0.9514**	0.9326	0.9055	*0.9139*	*0.9189*	*0.9283*	*0.9389*	*0.9424*
Dropwave	0.7308	0.7413	0.7667	0.7237	0.7555	0.7293	0.6860	***0.7740***
Powers	0.2177	0.2167	0.2216	*0.2428*	*0.2372*	*0.2390*	*0.2339*	***0.3670***
Ackley-2	0.8230	**0.8975**	0.7333	0.6382	0.5864	0.6864	0.6293	0.7001
Ackley-5	0.1832	0.2082	0.5473	*0.6694*	0.3582	0.3744	***0.6700***	0.4348
Ackley-10	0.9893	0.9864	0.8178	*0.9900*	*0.9912*	***0.9916***	*0.8340*	*0.8567*
Alpine2-2	**0.8628**	0.8482	0.7902	0.7467	0.5988	0.6699	0.6393	0.7807
Alpine2-5	0.5221	0.6151	**0.7797**	0.6740	0.6431	0.6592	0.6747	0.7123

Table 2: Results for the average 'gap' measure (5 replicates) across different functions. EL-k is the expect loss function computed with k steps ahead at each iteration. GLASSES is the GLASSES algorithm, MPI is the maximum probability of improvement and GP-LCB is the lower confidence bound criterion. The best result for each function is bolded. In italic, the cases in which a non-myopic loss outperforms the myopic loss are highlighted.

which is defined as

$$G \triangleq \frac{y(\mathbf{x}_{first}) - y(\mathbf{x}_{best})}{y(\mathbf{x}_{first}) - y(\mathbf{x}_{opt})},$$

where $y(\cdot)$ represents the evaluation of the objective function, $y(\mathbf{x}_{opt})$ is the global minimum, and $\mathbf{x}_{first}$ and $\mathbf{x}_{best}$ are the first and best evaluated point, respectively. To avoid gap measures larger that one due to the noise in the data, the measures for each experiment are normalized across all methods. The initial point $\mathbf{x}_{first}$ was chosen to be the best of the original points used to initialise the models.

Table 2 shows the comparative across different functions and methods. None of the methods used is universally the best but a non myopic loss is the best in 6 of the 11 cases. In 3 cases the full GLASSES approach is the best of all methods. Specially interesting is the case of the McCormick and the Powers function. In these two cases, to increase the number of steps ahead used to compute the loss consistently improve the obtained results. Note as well that when the GLASSES algorithm is not the best global method it typically performs closely to the best alternative which makes it a good 'default' choice if the function to optimise is expensive to evaluate.

5 Conclusions

In this paper we have explored the myopia in Bayesian optimisation methods. For the first time in the literature, we have proposed an non-myopic loss that allows taking into account dozens of future evaluations before making the decision of where to sample the objective function. The key idea is to jointly model all future evaluations of the algorithm with a probability distribution and to compute the expected loss by marginalising them out. Because this is an expensive step, we avoid it by proposing a fixed prediction of the future steps. Although this doesn't make use of the epistemic uncertainty on the steps ahead, it drastically reduces the computation burden of approximating the loss. We made use of the connection of the multiple steps ahead problem with some methods proposed in the batch Bayesian optimisation to solve this issue. The final computation of the loss for each point in the domain is carried out by adapting EP to our context. As previously suggested in **?**, our results confirm that using a non-myopic loss helps, in practice, to solve global optimisation problems. Interestingly, and as happens with any comparison of loss functions across many objective functions, there is not a universal best method. However, in cases in which GLASSES is not superior, it performs very closely to the myopic loss, which makes it an interesting default choice in most scenarios.

Some interesting challenges will be addressed in the future such as making the optimisation of the loss more efficient (for which DIRECT is employed in this work): although the smoothness of the loss is guaranteed if the steps ahead are consistently predicted for points close in the space, if the optimisation of the steps ahead fails, the optimisation of the loss may be challenging. Also, the use of non-stationary kernels, extensions to deal with to high dimensional problems and finding efficient was of sampling many steps ahead will also be analysed.

Supplementary materials for: 'GLASSES: Relieving The Myopia Of Bayesian Optimisation'

S1 Oracle Multiple Steps look-ahead Expected Loss

Denote by $\eta_n = \min\{\mathbf{y}_0, y_*, y_2 \dots, y_{n-1}\}$ the value of the best visited location when looking at n evaluations in the future. Note that η_n reduces to the current best lost η in the one step-ahead case. It is straightforward to see that

$$\min(y_n, \eta_n) = \min(\mathbf{y}, \eta).$$

It holds hat

$$\Lambda_n(\mathbf{x}_*|\mathcal{I}_0, \mathcal{F}_n(\mathbf{x}_*)) \quad = \quad \int \min(\mathbf{y}, \eta) \prod_{j=1}^{n} p(y_j|\mathcal{I}_{j-1}, \mathcal{F}_n(\mathbf{x}_*))\mathrm{d}y_* \dots \mathrm{d}y_n$$

where the integrals with respect to $\mathbf{x}_2 \dots \mathrm{d}\mathbf{x}_n$ are $p(\mathbf{x}_j|\mathcal{I}_{j-1}, \mathcal{F}_n(\mathbf{x}_*)) = 1$, $j = 2, \dots, n$ since we don't need to optimize for any location and $p(y_j|\mathbf{x}_j, \mathcal{I}_{j-1}, \mathcal{F}_n(\mathbf{x}_*)) = p(y_j|\mathcal{I}_{j-1}, \mathcal{F}_n(\mathbf{x}_*))$. Notice that

$$\begin{aligned}
\prod_{j=1}^{n} p(y_j|\mathcal{I}_{j-1}, \mathcal{F}_n(\mathbf{x}_*)) &= p(y_n|\mathcal{I}_{n-1}, \mathcal{F}_n(\mathbf{x}_*)) \prod_{j=1}^{n-1} p(y_j|\mathcal{I}_{j-1}\mathcal{F}_n(\mathbf{x}_*)) \\
&= p(y_n, y_{n-1}|\mathcal{I}_{n-2}, \mathcal{F}_n(\mathbf{x}_*)) \prod_{j=1}^{n-2} p(y_j|\mathcal{I}_{j-1}\mathcal{F}_n(\mathbf{x}_*)) \\
&\quad \dots \\
&= p(y_n, y_{n-1}, \dots, y_2|\mathcal{I}_1, \mathcal{F}_n(\mathbf{x}_*)) \prod_{j=1}^{2} p(y_j|\mathcal{I}_{j-1}\mathcal{F}_n(\mathbf{x}_*)) \\
&= p(\mathbf{y}|\mathcal{I}_0, \mathcal{F}_n(\mathbf{x}_*))
\end{aligned}$$

and therefore

$$\Lambda_n(\mathbf{x}_*|\mathcal{I}_0, \mathcal{F}_n(\mathbf{x}_*)) = \mathbb{E}[\min(\mathbf{y}, \eta)] = \int \min(\mathbf{y}, \eta) p(\mathbf{y}|\mathcal{I}_0, \mathcal{F}_n(\mathbf{x}_*)) d\mathbf{y}$$

S2 Formulation of the Oracle Multiple Steps loook-ahead Expected Loss to be computed using Expectation Propagation

Assume that $\mathbf{y} \sim \mathcal{N}(\mathbf{y}; \mu, \Sigma)$. Then we have that

$$\begin{aligned}
\mathbb{E}[\min(\mathbf{y}, \eta)] &= \int_{\mathbb{R}^n} \min(\mathbf{y}, \eta)\mathcal{N}(\mathbf{y}; \mu, \Sigma) d\mathbf{y} \\
&= \int_{\mathbb{R}^n - (\eta, \infty)^n} \min(\mathbf{y})\mathcal{N}(\mathbf{y}; \mu, \Sigma) d\mathbf{y} + \int_{(\eta, \infty)^n} \eta \mathcal{N}(\mathbf{y}; \mu, \Sigma) d\mathbf{y}.
\end{aligned}$$

The first term can be written as follows:

$$\int_{\mathbb{R}^n - (\eta, \infty)^n} \min(\mathbf{y})\mathcal{N}(\mathbf{y}; \mu, \Sigma) d\mathbf{y} = \sum_{j=1}^{n} \int_{P_j} y_j \mathcal{N}(\mathbf{y}; \mu, \Sigma) \mathrm{d}\mathbf{y}$$

where $P_j := \{\mathbf{y} \in \mathbb{R}^n - (\eta, \infty)^n : y_j \leq y_i, \ \forall i \neq j\}$. We can do this because the regions P_j are disjoint and it holds that $\cup_{j=1}^{n} P_j = \mathbb{R}^n - (\eta, \infty)^n$. Also, note that the $\min(\mathbf{y})$ can be replaced within the integrals since within each P_j it holds that $\min(\mathbf{y}) = y_j$. Rewriting the integral in terms of indicator functions we have that

$$\sum_{j=1}^{n} \int_{P_j} y_j \mathcal{N}(\mathbf{y}; \mu, \Sigma) \mathrm{d}\mathbf{y} = \sum_{j=1}^{n} \int_{\mathbb{R}^n} y_j \prod_{i=1}^{n} t_{j,i}(\mathbf{y}) \mathcal{N}(\mathbf{y}; \mu, \Sigma) \mathrm{d}\mathbf{y} \tag{S.1}$$

where $t_{j,i}(y) = \mathbb{I}\{y_i \leq \eta\}$ if $j = i$ and $t_{j,i}(y) = \mathbb{I}\{y_j \leq y_i\}$ otherwise.

The second term can be written as

$$\int_{(\eta,\infty)^n} \eta \mathcal{N}(\mathbf{y}; \mu, \Sigma) d\mathbf{y} = \eta \int_{\mathbb{R}^n} \prod_{i=1}^{n} h_i(\mathbf{y}) \mathcal{N}(\mathbf{y}; \mu, \Sigma) d\mathbf{y} \tag{S.1}$$

where $h_i(\mathbf{y}) = \mathbb{I}\{y_i > \eta\}$. Merge (S.1) and (S2) to obtain Eq. (5).

S2.1 Synthetic functions

In this section we include the formulation of the objective functions used in the experiments that are not available in the references provided.

Name	Function
SinCos	$f(x) = x\sin(x) + x\cos(2x)$
Alpine2-q	$f(\mathbf{x}) = \prod_{i=1}^{q} \sqrt{x_i}\sin(x_i)$
Cosines	$f(\mathbf{x}) = 1 - \sum_{i=1}^{2}(g(x_i) - r(x_i))$ with $g(x_i) = (1.6x_i - 0.5)^2$ and $r(x_i) = 0.3\cos(3\pi(1.6x_i - 0.5))$.

Table 3: Functions used in the experimental section.