Proceedings of the 2018 Winter Simulation Conference
M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, eds.

ON PARALLELIZING MULTI-TASK BAYESIAN OPTIMIZATION

Matthew Groves
Michael Pearce
Juergen Branke

University of Warwick
Gibbet Hill Road
Coventry, CV4 7AL, UK

ABSTRACT

Parallelizing Bayesian Optimization has recently attracted a lot of attention. The challenge is usually to estimate the effect multiple new samples will have on the posterior distribution of the objective function, and the combinatorial explosion of the possible sample locations. In this paper, we show that at least for multi-task Bayesian Optimization, parallelization is straightforward because the benefit of samples is independent as long as they are sufficiently far apart in the task space. We propose a simple penalization approach proportional to correlation based on the kernel and demonstrate that for the problem settings we considered, the efficacy of our parallel multi-task Bayesian Optimization algorithm is close to the sequential version, while being able to exploit parallel computation to speed up optimization. Depending on the setup considered, we observe efficiencies of parallelization between 69% and 100%.

1 INTRODUCTION

Bayesian Optimization (BO) is a powerful and popular simulation optimization tool. Based on a few evaluated solutions, it builds a surrogate model of the response surface (usually a Gaussian Process, GP, regression model). Then an acquisition function (or infill criterion) uses the regression model to quantify the benefit of a new objective function sample (simulation run) for a given input, balancing exploration and exploitation. In each iteration, the process determines the next input for sampling with the maximum expected benefit and updates the regression model based on the observation. The process is usually sequential, with one new sample allocated per iteration. However, there is increased interest in parallelizing BO, given that parallel processors are now standard in most computers, cloud computing gives everyone easy access to massive parallelization, and because physical experiments can often be parallelized at no additional cost. The challenges of parallelization are usually to estimate the posterior distribution from multiple new samples, as well as the combinatorial explosion of the possible sample locations.

In this paper, we consider multi-task optimization, i.e., the case where multiple simulation optimization problems have to be solved, and correlations among these problems may be exploited to speed up convergence. An example could be when we would like to optimize the parameters of a traffic light controller, for different (but related) traffic scenarios. We show that multi-task BO is much easier to parallelize, as the expected value of information of sampling the performance of a solution on two different tasks may be considered independent if the tasks are sufficiently distant to be considered uncorrelated. We propose a simple penalization mechanism based on the GP kernel that defines correlation between tasks. Basically, we penalize the expected improvement of the *i*-th sample to be taken in a batch proportionally to the correlation between it and all previously chosen samples in the task space. We show that this simple mechanism allows parallelization with an excellent speed-up within the settings tested.

The paper is structured as follows. First, we review the literature on parallel BO in Section 2. In Section 3 we define the considered multi-task problem, followed by the suggested parallelization strategy in Section 4. Empirical results are reported in Section 5. The paper concludes with a summary and some ideas for future work.

2 RELATED WORK

The development of parallel sampling techniques for Bayesian Optimization with Gaussian Process models has recently attracted significant attention. In a key earlier work, Ginsbourger et al. (2007) present a generalization of the popular Efficient Global Optimization (EGO) method (Jones et al. 1998) to the parallel setting, proposing the $q-$EI acquisition function, extending the Expected Improvement acquisition function to batches of q samples. Unfortunately, optimizing $q-$EI is difficult as computing both the function and its gradient is highly expensive. To avoid this issue, Ginsbourger et al. (2010) propose two simple heuristic methods, the *Kriging Believer* and the *Constant Liar*. The former builds batches by assuming that at each step, the previous sample added to the batch will return the expected value of the GP model at that point, whereas the latter assumes that a constant L is returned from each sample. The value of L can be selected to generate the desired level of exploration in sample selection. In later work, Chevalier and Ginsbourger (2013) suggest a closed-form approximation for $q-$EI, which performs well for small batch sizes ($q < 10$).

Snoek et al. (2012) also propose a method for approximating $q-$EI, sequentially constructing batches by attempting to maximize the expected acquisition function over all possible sample result of points currently added to the batch, which they estimate through Monte Carlo sampling. The authors demonstrate that parallel sampling, with batch sizes of 3 and 5, can improve on expert-level human hyper-parameter optimization for several well-known machine learning test problems.

González et al. (2016) also consider batch sample allocation for optimization with Gaussian Process models. They propose a heuristic method that approximates the effects of interactions between allocated samples within a batch by penalizing the acquisition function around points selected for the batch to model the local repulsion effect. In their method, samples are sequentially allocated to the batch by choosing points to maximize the acquisition function penalized at all previously selected points. They discuss the design of suitable penalization functions under the assumption of Lipschitz continuity of the acquisition function, calculated using the maximum observed value of the target function, and an estimate of its maximum gradient.

Rontsis et al. (2017) propose an alternative method for efficiently approximating $q-$EI. They show that the best-case expectation over all probability distributions consistent with the model forms a lower bound for $q-$EI, which can be calculated exactly as the solution to a convex optimization problem. They compare this method empirically to other $q-$EI approximation methods, including the *Constant Liar* approach from Ginsbourger et al. (2010) and Local Penalisation with EI from González et al. (2016), with their lower bound method achieving best performance on a 1-dimensional test example for batch sizes of 4 and 5.

Several of the above works share a common principle to batch selection; they construct the batch sequentially, iteratively optimizing over the acquisition function. While this may help to ensure only highly relevant points are added, the requirement of repeated optimization can limit the scalability of the parallelization. In contrast, Hernández-Lobato et al. (2017) propose a fully parallel approach for large scale batch sampling with Gaussian Process models, whereby each point in the batch is independently chosen using Thompson sampling. In this approach, each parallel thread collects a single Monte Carlo sample from the posterior distribution of the model parameters conditioned on all previously observed data, before choosing to sample the point that maximizes the expected function value given previous data and conditioned on the obtained parameter value. They compare performance of this method against the Monte Carlo based $q-$EI method of Snoek et al. (2012), demonstrating improved performance on large-scale problems.

Wang et al. (2016) present an alternative method for avoiding the computational expense of optimizing the $q-$EI acquisition function exactly. They propose using infinitesimal perturbation analysis to construct an unbiased stochastic gradient estimator for $\nabla q-$EI, which can be used in stochastic gradient ascent optimization. They show in numerical experiments that, for $q > 4$, using this approximation can lead to faster convergence than calculating $\nabla q-$EI through closed-form integration.

Wu and Frazier (2016) adapt the well-known Knowledge Gradient (KG) (Frazier et al. 2009) method for the parallel setting, proposing the q-KG acquisition function for choosing Bayes-optimal batches of q samples under the assumption that these will be the final samples allocated. q-KG is similar to parallel EI, and is equivalent when function evaluations are noise free and the final solution is restricted to be from the previously sampled points. As such, computing q-KG and its derivative analytically is very expensive. The paper presents an alternative method for approximating q-KG by discretizing the search space.

To the best of our knowledge, no one has yet looked into the parallelization of multi-task Bayesian Optimization.

3 CONTINUOUS MULTI-TASK OPTIMIZATION

In this paper, we consider the multi-task problem also considered by Ginsbourger et al. (2014) and Pearce and Branke (2018). We assume that there exists a (discrete or continuous) set of tasks described by $x \in X$, and the tasks are distributed according to a known density $\mathbb{P}[x]$. There are tunable parameters $a \in A$. Running a simulation with parameters a on a task with features x yields a performance measurement $Y_{x,a} = \theta(x,a) + \varepsilon$ where $\theta : X \times A \to \mathbb{R}$ is a deterministic latent function from (task, parameter) to expected performance and ε is independent and identically distributed observation noise $\varepsilon \sim N(0, \sigma_\varepsilon^2)$. Our aim is to find a mapping $S : X \to A$ from a given task to the optimal parameter setting for this task that approximates the true optimal mapping $S^*(x) = \underset{a}{\text{argmax}}\, \theta(x,a)$ with incomplete information. The quality of our derived mapping $S(x)$ at the end of sampling is the corresponding true expected performance over all tasks:

$$\int_{x \in X} \theta(x, S(x)) \mathbb{P}[x] dx. \tag{1}$$

We assume we have a fixed budget of N samples (simulating a parameter setting on a task), and that we can sample iteratively, i.e., in every iteration, we can select the task x and the parameter setting a from which to sample performance $Y_{x,a}$ based on the information collected so far.

Given this formulation, if x is constant, the problem reduces to a single global optimization over A. However, because there is a range of x, one must find the global optimal $\max_a \theta(x,a)$ for each x.

Our formulation also accounts for task distribution because in practice, under a constrained budget, finding the optimal parameters for tasks with unusual outlying features (low $\mathbb{P}[x]$) is less useful than finding the optimal parameters for common tasks (high $\mathbb{P}[x]$). Note that if the set of tasks is finite, the integral in Equation 1 is replaced with a summation and $\mathbb{P}[x]$ is a probability mass function.

The above problem is similar to other multi-task BO settings, e.g., Hu and Ludkovski (2017) or Pearce and Branke (2017b) where the space of alternatives/parameters is discrete, or Morales-Enciso and Branke (2015) and Poloczek et al. (2016) who assume that tasks arrive sequentially and exploit the information from previous tasks to solve the current task as efficiently as possible. Rather than trying to find the best solution for each task, Swersky et al. (2013), Pearce and Branke (2017a) and Toscano-Palmerin and Frazier (2018) look at the problem of finding one solution that works well on average over all tasks. Further multi-task BO algorithms include Bonilla et al. (2007), Poloczek et al. (2017).

In this paper, we use the LEVI acquisition function, which is the simplest of the methods proposed by Pearce and Branke (2018), and very similar also to the method proposed by Ginsbourger et al. (2014). Following Pearce and Branke (2018), we use the notation $\tilde{x} = (x,a) \in X \times A$ and write functions of both input variables as $\mu^n(x,a) = \mu^n(\tilde{x})$. We define the set of sampled task feature values $\{x^1, ..., x^n\} = X^n$, parameters $\{a^1, ..., a^n\} = A^n$, the set of input pairs $\{(x^1, a^1), ..., (x^n, a^n)\} = \tilde{X}^n$ and column vector of responses $(y^1, ..., y^n) = Y^n$. We define the sequence of filtrations, $\mathscr{F}^n$, as sigma algebras generated by the data collected up to time n, $\mathscr{F}^n = \sigma\{(x^1,a^1,y^1),\dots,(x^n,a^n,y^n)\}$. Given prior mean and covariance functions $\mu^0(x,a)$, $k^0((x,a),(x',a'))$, one may use the Matrix Inversion Lemma (Hager 1989) to condition the prior functions yielding the posterior mean and covariance functions as follows (Rasmussen and Williams 2004),

$$\mathbb{E}[\theta(x,a)|\mathscr{F}^n] = \quad \mu^n(\tilde{x}) = \quad \mu^0(\tilde{x}) + k^0(\tilde{x},\tilde{X}^n)(K^n+\sigma_\varepsilon^2 I)^{-1}(Y^n - \mu^0(\tilde{X}^n)) \tag{2}$$

$$Cov[\theta(x,a),\theta(x',a')|\mathscr{F}^n] = \quad k^n(\tilde{x},\tilde{x}') = \quad k^0(\tilde{x},\tilde{x}') - k^0(\tilde{x},\tilde{X}^n)(K^n+\sigma_\varepsilon^2 I)^{-1}k^0(\tilde{X}^n,\tilde{x}') \tag{3}$$

where $K^n_{ij} = k^0((x^i,a^i),(x^j,a^j))$ is the $n\times n$ matrix composed of the prior covariance function evaluated for all the sampled input points $\tilde{X}^n$. We have written $k^0(\tilde{x},\tilde{X}^n)$ to denote the $1\times n$ matrix of prior covariance between $\tilde{x}$ and all points in $\tilde{X}^n$ and likewise $k^0(\tilde{X}^n,\tilde{x}')$ is the $n\times 1$ matrix.

If the mapping is made by using the best predicted parameter $S(x) = argmax_a \mu^n(x,a)$, the predicted performance of the mapping summed over all tasks, after a sample $\tilde{x}^{n+1}$ has been simulated, can be calculated as

$$\mathbb{E}\left[P^{n+1}|\mathscr{F}^n,\tilde{x}^{n+1}\right] \quad = \quad \int_{x\in X}\mathbb{E}\left[\max_a\{\mu^n(x,a)+\tilde{\sigma}^n((x,a);\tilde{x}^{n+1})Z^{n+1}\}\right]\mathbb{P}[x]\,dx, \tag{4}$$

where Z^{n+1} is the stochastic z-score of the new response value on the prior predictive distribution and is given by

$$Z^{n+1} = \frac{y^{n+1}-\mu^n(\tilde{x}^{n+1})}{\sqrt{k^n(\tilde{x}^{n+1},\tilde{x}^{n+1})+\sigma_\varepsilon^2}},$$

which is a standard normal random variable $Z^{n+1}\sim N(0,1)$ when conditioned on $\mathscr{F}^n$.

The incremental gain, or value of information, is the difference in predicted performance between consecutive samples and can be expressed as

$$\begin{aligned}\mathscr{I}(\tilde{x}) \quad &= \quad \mathbb{E}\left[P^{n+1}-P^n|\mathscr{F}^n,\tilde{x}^{n+1}=\tilde{x}\right] \\ &= \quad \int_{x'\in X}\mathbb{E}\left[\max_{a'}\{\mu^n(x',a')+\tilde{\sigma}^n((x',a');\tilde{x})Z^{n+1}\} - \max_{a'}\mu^n(x',a')\right]\mathbb{P}[x']\,dx'. \end{aligned} \tag{5}$$

The Local Expected Value of Information acquisition function (LEVI) (Pearce and Branke 2018) sequentially allocates samples to maximize the improvement at the sampled task only. For a given task x, the contribution to the improvement in total performance of that task alone is given by the integrand of Equation 5

$$\mathbb{E}\left[\max_{a'}\{\mu^n(x,a')+\tilde{\sigma}^n((x,a');(x,a))Z^{n+1}\} - \max_{a'}\mu^n(x,a')\right]\mathbb{P}[x]. \tag{6}$$

Since the expectation over Z cannot be solved for continuous $a\in A$, we propose to replace A with a finite set A_D of n_A points which are randomly distributed according to a Latin Hypercube over the space A (scaled by $\mathbb{P}[x]$). The random elements of the set A_D are held constant for each time step and are re-generated between time steps such that the mapping does not overfit to a single discretization. Replacing A with A_D, Equation 6 can then be approximated by

$$\mathbb{E}\left[\max\{\mu^n(x,A_D)+\tilde{\sigma}^n((x,A_D);\tilde{x})\,Z^{n+1}\} - \max\mu^n(x,A_D)\right]\mathbb{P}[x], \tag{7}$$

where $\mu^n(x,A_D) = (\mu^n(x,a^{n+1}),\mu^n(x,a_1),\dots)\in\mathbb{R}^{n_A}$ is the vector of means and similarly for $\tilde{\sigma}^n((x,A_D);(x,a^{n+1}))$. Gathering terms, Equation 7 is thus of the form

$$LEVI(x,a) \quad = \quad \mathbb{E}[\max\{\mu_1+\sigma_1 Z,\dots,\mu_{n_A}+\sigma_{n_A}Z\}]\,\mathbb{P}[x], \tag{8}$$

which is the normal expectation of the maximum of linear functions. This expectation can be cheaply evaluated using Algorithm 1 in Knowledge Gradient for Correlated Normal Beliefs (Frazier et al. 2009).

4 PARALLELIZING CONTINUOUS MULTI-TASK OPTIMIZATION

Ideally when parallelizing BO, one would like to look several samples ahead and allocate k samples to maximize the expected improvement $\mathbb{E}\left[P^{n+k}-P^{n}\right]$. Unfortunately this is difficult, because the expected value of information of several samples is not just the sum of the values of information of each individual sample due to the correlation between the values of neighboring samples, and also because we are only interested in the maximum for a particular task. LEVI (Equation 6) approximates Equation 5 by only considering the improvement at the sampled location, but even under this approximation, the value of information of multiple samples is not just the sum of LEVI values at each sampled location, because we are only interested in the maximum for a task. In the multi-task setting, however, the goal is to maximize performance for *all* tasks. Thus, if the samples are sufficiently far apart in task space, the value of information from multiple samples can indeed be approximated just as the sum of the values of information of the individual samples, in our case as the sum of the different LEVI values.

Based on this observation, we propose a simple penalty approach similar to González et al. (2016). Let $\tilde{x}_{t,k}$ denote the k-th sample to be chosen in the t-th batch (iteration). Then, the first sample is placed at the (numerically approximated) global maximum of the LEVI acquisition function, and thus identical to the case of taking only a single sample per iteration. For batch sizes larger than one, the LEVI acquisition function is modified by a penalization scheme in the neighborhood of tasks that have already been sampled.

As penalization, we propose the following function based on the kernel in task space:

$$\phi((x,a),(x',a')) = 1-k^{0}(a,a'). \tag{9}$$

The acquisition function for the k-th sample in the t-th batch is then

$$LEVI^{k}(x,a) = LEVI(x,a)\prod_{j=1}^{k-1}\phi(\tilde{x},\tilde{x}_{t,j}). \tag{10}$$

This strategy has the additional advantage that it doesn't require to update the GP after each element in the batch of samples is determined, an operation which has a computational complexity of $\mathcal{O}(n^3)$.

5 EMPIRICAL RESULTS

In this section, we evaluate the performance of our parallelization strategy on some artificial benchmark problems.

We generate test functions from a Gaussian Process prior where the task space X as well as the action space A is $[0,100]$. Unless specified otherwise, the length scale in each dimension is 10, and the process variance is $\sigma^2 = 10^2$. We initialize each sampling procedure with 20 samples in a latin hypercube. In all experiments, we use a uniform task distribution and all function evaluations have a noise added, $\varepsilon \sim N(0,1)$. To remove the influence of hyperparameter tuning we assume length scales are known.

To measure the quality of a mapping learnt by each method, for each experiment, a test set of 250 tasks values, X_{test}, are generated from $\mathbb{P}[x]$, and the difference in performance between the true optimal a and the performance of the a value determined by the mapping (referred to as the Opportunity Cost of the mapping) is averaged over all $x_i \in X_{test}$. This is calculated using:

$$\text{Opportunity Cost} = \frac{1}{250}\sum_{x_i\in X_{test}} \max_{a}\theta(x_i,a)-\theta(x_i,S^{N}(x_i)). \tag{11}$$

All results are averaged over 100 runs. All experiments were performed on a compute node with two Intel Xeon E5-2630 v3 8-core processors and 64GB RAM.

Figure 1 shows the reduction in opportunity cost per iteration gained by batch sample allocation for different batch sizes. At each iteration, Parallel MaxiLevi (PML) selects and performs a batch of (either 4,8

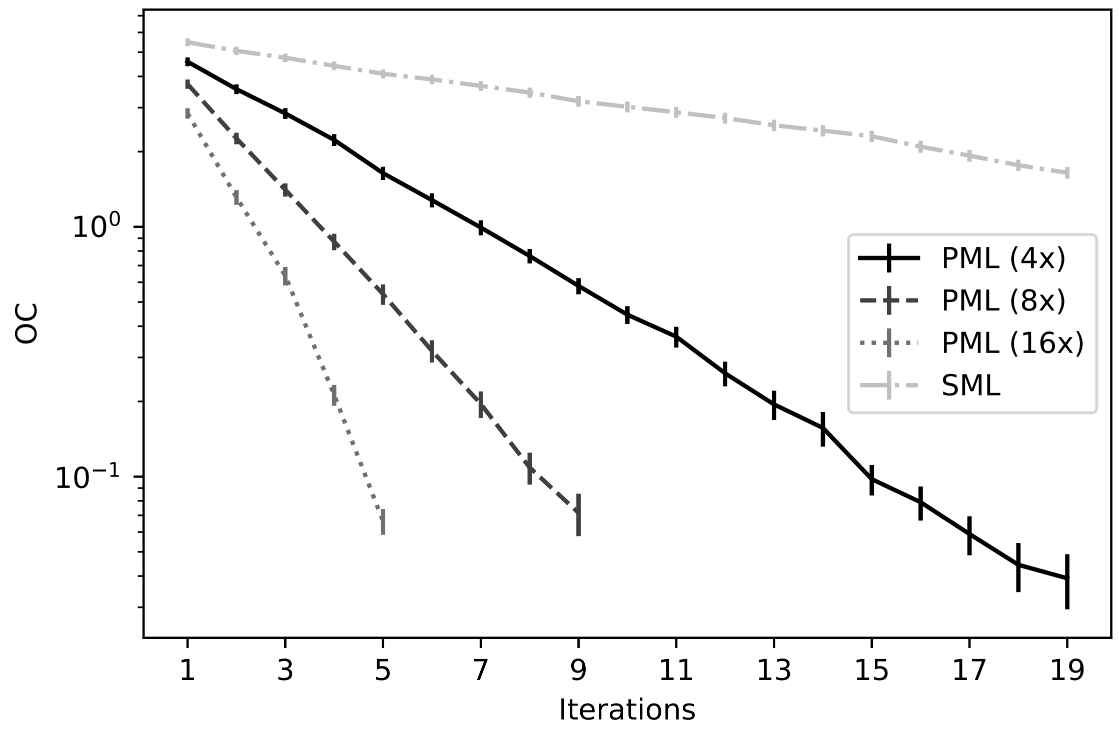


Figure 1: The performance gain from parallelization for the PML algorithm for batch sizes 4, 8 and 16. At each iteration, PML selects and samples a batch, compared to a single sample for SML.

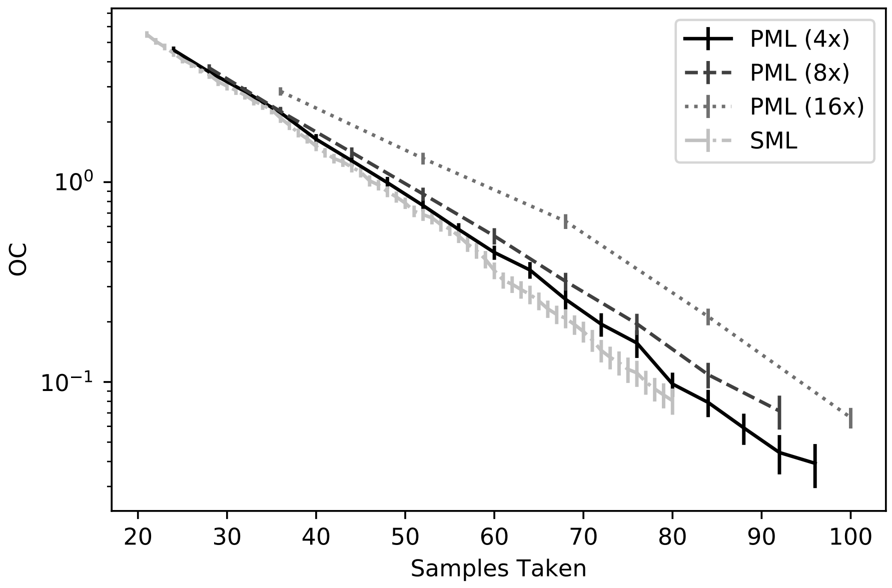
or 16) samples, compared to a single sample for fully sequential sampling (SML). Perhaps unsurprisingly, the performance gain increases for larger batch sizes, although the gain is sub-linear. The speed-up factor due to parallelization for these PML batches is summarized in Table 1 .

Figure 2a shows the efficacy of different batch sizes compared to the same total sampling budget allocated sequentially. As expected, the efficacy suffers slightly with increasing batch size, but the deterioration is very moderate.

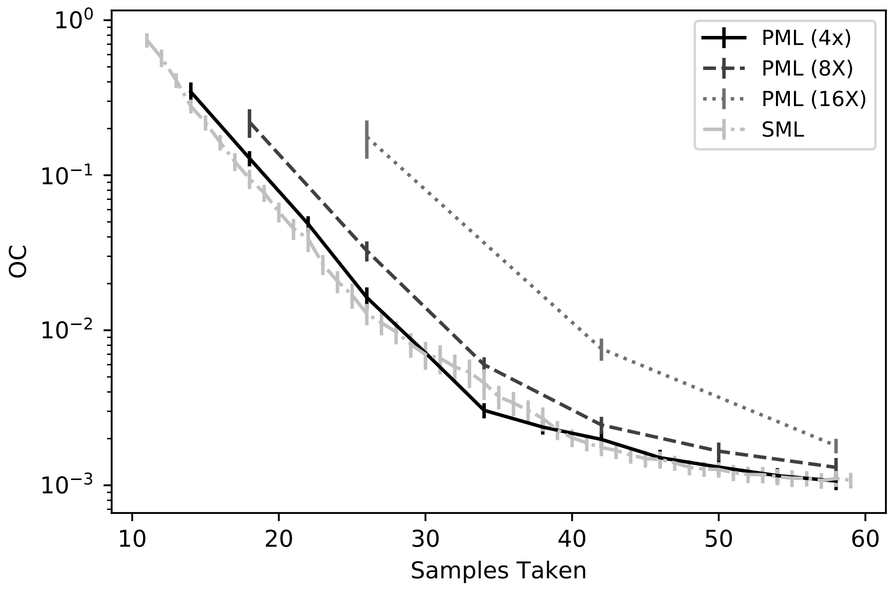
Figure 2b shows a similar result, but using a length scale of 30 in each dimension for the underlying Gaussian Process. A larger length scale makes the overall learning problem simpler, because the underlying functions become smoother and a sample carries information about a larger neighborhood, which is why all sampling schemes reduce opportunity cost much more quickly than for the case of smaller length scale. We expected that the efficiency of parallelization would suffer due to the larger length scale, because it would be more difficult to identify several samples that are interesting to evaluate but also sufficiently far apart in the task space to be uncorrelated. As Table 2 shows, the effect was only noticeable for batch size

Table 1: Number of iterations and samples needed to achieve an opportunity cost of 0.1.

Batch Size	Samples	Iterations	Speedup	Efficiency
1	77	77	-	-
4	80	20	3.85	0.96
8	90	11.3	6.81	0.85
16	95	6	12.83	0.80



(a) Length Scale 10



(b) Length Scale 30

Figure 2: The efficiency of various PML batch sizes compared to sequentially allocating the same total number of samples, for different underlying GP length scales.

Table 2: Number of iterations and samples needed to achieve an opportunity cost of 0.01 for length scale of 30.

Batch Size	Samples	Iterations	Speedup	Efficiency
1	28	28	-	-
4	28	7	3.95	0.99
8	32	4	8	1
16	40	2.5	11.1	0.69

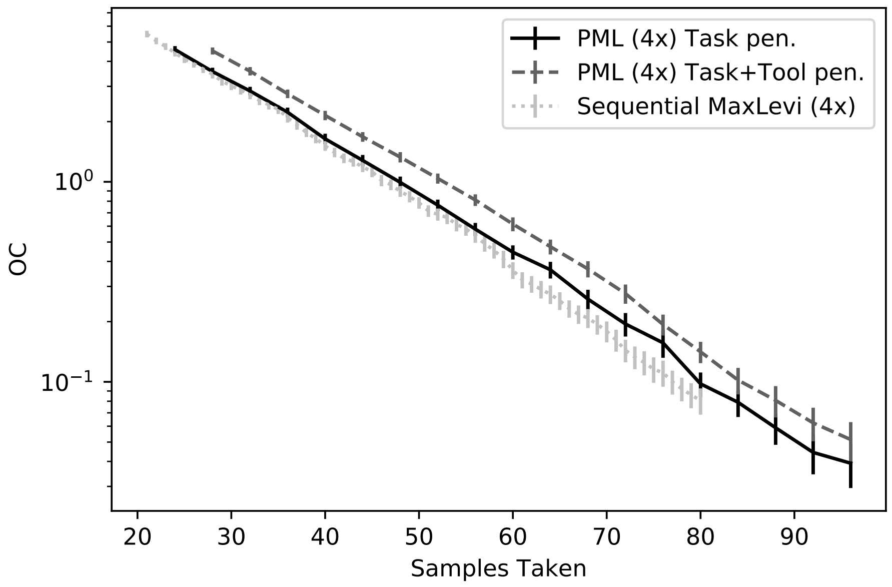


Figure 3: The performance of Parallel MaxLevi (batch size 4) for two different penalization functions.

of 16, where the efficiency dropped from 80% to 69%, but the results still confirm that the approach loses efficiency for large batch sizes and length scales.

Finally, Figure 3 compares our proposed approach, which only penalizes based on the task dimension, to an approach that penalizes in both dimensions, i.e., tasks and parameter. As can be seen, results are better if the penalty is only applied in the task dimension, as it is perfectly fine to sample similar parameter settings for different tasks, because their expected improvements are still independent.

6 CONCLUSION

In this paper, we have considered parallelization of multi-task Bayesian Optimization. Because for multi-task optimization the total expected value of information for samples sufficiently apart in task space to be uncorrelated is just the sum of the individual expected values of information, we have proposed a simple penalization scheme that penalizes the acquisition function proportional to the correlation with already selected samples in the same batch. Initial empirical results are very promising, indicating that this simple scheme may achieve an efficiency of approximately 69-80% for a case of 16 processors, and higher efficiencies for smaller numbers of processors or smaller length scales.

Clearly, this approach should be tested more widely, on a wider range of test problems with different length scales, in combination with other multi-task Bayesian Optimization techniques (including multiobjective optimization), and when hyperparameters have to be learned. Also, we plan to compare the mechanism with other parallelization mechanisms from the literature.

REFERENCES

Bonilla, E. V., K. M. Chai, and C. Williams. 2007. "Multi-Task Gaussian Process Prediction". In *Advances in Neural Information Processing Systems 20*, edited by J. Platt et al., 153–160.

Chevalier, C., and D. Ginsbourger. 2013. "Fast Computation of the Multi-Points Expected Improvement with Applications in Batch Selection". In *Learning and Intelligent Optimization*, edited by N. G. et al., 59–69. Heidelberg, Germany: Springer.

Frazier, P., W. Powell, and S. Dayanik. 2009. "The Knowledge-Gradient Policy for Correlated Normal Beliefs". *INFORMS Journal on Computing* 21(4):599–613.

Ginsbourger, D., R. Le Riche, and L. Carraro. 2007. "A Multi-Points Criterion for Deterministic Parallel Global Optimization Based on Kriging". In *Proceedings of the International Conference on Non-Convex Programming*. December 17$^{\text{th}}$-21$^{\text{st}}$, Rouen, France.

Ginsbourger, D., R. Le Riche, and L. Carraro. 2010. "Kriging is Well-Suited to Parallelize Optimization". In *Computational Intelligence in Expensive Optimization Problems*, 131–162. Heidelberg, Germany: Springer.

Ginsbourger, D., J. Baccou, C. Chevalier, F. Perales, N. Garland, and Y. Monerie. 2014. "Bayesian Adaptive Reconstruction of Profile Optima and Optimizers". *SIAM/ASA Journal on Uncertainty Quantification* 2(1):490–510.

González, J., Z. Dai, P. Hennig, and N. Lawrence. 2016. "Batch Bayesian Optimization via Local Penalization". In *Artificial Intelligence and Statistics*, 648–657.

Hager, W. W. 1989. "Updating the Inverse of a Matrix". *SIAM review* 31(2):221–239.

Hernández-Lobato, J. M., J. Requeima, E. O. Pyzer-Knapp, and A. Aspuru-Guzik. 2017. "Parallel and Distributed Thompson Sampling for Large-Scale Accelerated Exploration of Chemical Space". In *Proceedings of Machine Learning Research 70*, edited by D. Precup et al., 1470–1479.

Hu, R., and M. Ludkovski. 2017. "Sequential Design for Ranking Response Surfaces". *SIAM/ASA Journal on Uncertainty Quantification* 5(1):212–239.

Jones, D. R., M. Schonlau, and W. J. Welch. 1998. "Efficient Global Optimization of Expensive Black-Box Functions". *Journal of Global Optimization* 13(4):455–492.

Morales-Enciso, S., and J. Branke. 2015. "Tracking Global Optima in Dynamic Environments with Efficient Global Optimization". *European Journal of Operational Research* 242:744–755.

Pearce, M., and J. Branke. 2017a. "Bayesian Simulation Optimization with Input Uncertainty". In *Proceedings of the 2017 Winter Simulation Conference*, edited by W. K. V. Chan et al., 2268–2278. Piscataway, New Jersey: IEEE.

Pearce, M., and J. Branke. 2017b. "Efficient Expected Improvement Estimation for Continuous Ranking and Selection". In *Proceedings of the 2017 Winter Simulation Conference*, edited by W. K. V. Chan et al., 2161–2172. Piscataway, New Jersey: IEEE.

Pearce, M., and J. Branke. 2018. "Continuous Multi-Task Bayesian Optimisation with Correlation". *European Journal of Operational Research* 270(3):1074–1085.

Poloczek, M., J. Wang, and P. Frazier. 2016. "Warm Starting Bayesian Optimization". In *Proceedings of the 2016 Winter Simulation Conference*, edited by T. M. K. Roeder et al., 770–781. Piscataway, New Jersey: IEEE.

Poloczek, M., J. Wang, and P. Frazier. 2017. "Multi-Information Source Optimization". In *Advances in Neural Information Processing Systems 30*, edited by I. Guyon et al., 4289–4299.

Rasmussen, C. E., and C. K. I. Williams. 2004. *Gaussian Processes for Machine Learning*. MIT Press.

Rontsis, N., M. A. Osborne, and P. J. Goulart. 2017. "Distributionally Robust Optimization Techniques in Batch Bayesian Optimization". *arXiv preprint arXiv:1707.04191*.

Snoek, J., H. Larochelle, and R. P. Adams. 2012. “Practical Bayesian Optimization of Machine Learning Algorithms”. In *Advances in Neural Information Processing Systems 25*, edited by F. Pereira et al., 2951–2959.

Swersky, K., J. Snoek, and R. P. Adams. 2013. “Multi-Task Bayesian Optimization”. In *Advances in Neural Information Processing Systems 26*, edited by C. Burges et al., 2004–2012.

Toscano-Palmerin, S., and P. Frazier. 2018. “Bayesian Optimization with Expensive Integrands”. *arXiv preprint arXiv:1803.08661*.

Wang, J., S. C. Clark, E. Liu, and P. I. Frazier. 2016. “Parallel Bayesian Global Optimization of Expensive Functions”. *arXiv preprint arXiv:1602.05149*.

Wu, J., and P. Frazier. 2016. “The Parallel Knowledge Gradient Method for Batch Bayesian Optimization”. In *Advances in Neural Information Processing Systems 29*, edited by D. Lee et al., 3126–3134.

AUTHOR BIOGRAPHIES

JUERGEN BRANKE is Professor of Operational Research and Systems of Warwick Business School, University of Warwick, UK. He is Area Editor for the Journal of Heuristics, and Associate Editor for IEEE Transaction on Evolutionary Computation, for the Evolutionary Computation Journal, and for the Journal on Multi Criteria Decision Analysis. His research interests include metaheuristics, multiobjective optimization and decision making, optimization in the presence of uncertainty, and simulation-based optimization, and he has published over 170 peer-reviewed papers in international journals and conferences. His e-mail address is Juergen.Branke@wbs.ac.uk.

MICHAEL PEARCE is currently a PhD student at the University of Warwick’s Complexity Science Centre. He graduated from the University of Bristol in 2009 with MSci. in Mathematics and in 2015 with an MSc in Complexity Science from the University of Warwick. His e-mail address is m.a.l.pearce@warwick.ac.uk.

MATTHEW GROVES is currently a PhD student at the Mathematics for Real-World Systems Centre, University of Warwick. His research interests include optimization, ranking and selection in noisy environments, and reinforcement learning in stochastic 2-player games. His email address is M.J.Groves@warwick.ac.uk.