Search Strategies for Self-driving Laboratories with Pending Experiments

Hao Wen
University of Surrey, UK
Matterhorn Studio
`hao@matterhorn.studio`

Jakob Zeitler
Matterhorn Studio
Oxford, United Kingdom
`jakob@matterhorn.studio`

Connor Rupnow
`connor@rupnow.com`

Abstract

Self-driving laboratories (SDLs) consist of multiple stations that perform material synthesis and characterisation tasks. To minimize station downtime and maximize experimental throughput, it is practical to run experiments in asynchronous parallel, in which multiple experiments are being performed at once in different stages. Asynchronous parallelization of experiments, however, introduces delayed feedback (i.e. "pending experiments"), which is known to reduce Bayesian optimiser performance. Here, we build a simulator for a multi-stage SDL and compare optimisation strategies for dealing with delayed feedback and asynchronous parallelized operation. Using data from a real SDL, we build a ground truth Bayesian optimisation simulator from 177 previously run experiments for maximizing the conductivity of functional coatings. We then compare search strategies such as expected improvement, noisy expected improvement, 4-mode exploration and random sampling. We evaluate their performance in terms of amount of delay and problem dimensionality. Our simulation results showcase the trade-off between the asynchronous parallel operation and delayed feedback.

1 Introduction

Self-driving laboratories (SDLs) utilize automation and artificial intelligence (AI) to accelerate the discovery of new mission critical materials such as perovskites [12], catalysts [5], nanoparticles [8], and more [10] [6] [13] [17]. SDLs consist of multiple stages that, step by step, perform different material synthesis and characterisation tasks (Figure 1A)[15].

As the SDL performs an experiment, the material moves between task stations much like a conveyor belt in a production line. The simplest way to make materials would be to run experiments one at a time in serial operation (Figure 1B). To minimize station downtime and maximize experimental throughput, it is practical to have all stations running continuously. This approach is known as *asynchronous parallel*, in which multiple experiments are being performed at once in different stages (Figure 1C). This increases experimental throughput by allowing the SDL to perform multiple experiments at once, rather than one at a time.

Asynchronous parallelization of experiments, however, introduces delayed feedback in which new experiments must be chosen while previous experiments are running. The optimisation algorithm must use incomplete information to choose new experimental conditions. Delayed feedback is known to reduce Bayesian optimiser performance [1]. Additionally, the number of processing variables (i.e. dimensions) in an experiment can increase the difficulty of finding an optimal material. To the best of our knowledge, the literature has no empirical results or discussion on the impact of pending experiments for SDLs.

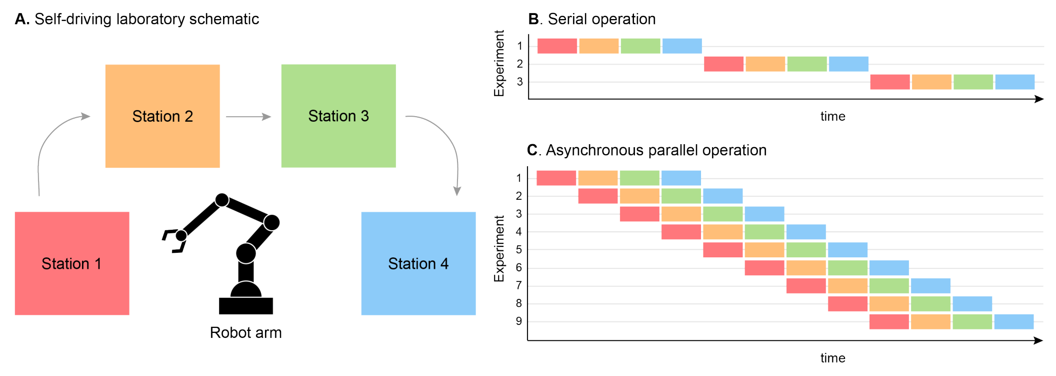


Figure 1: **(A)** Schematic of an SDL with four stages. The robot arm moves experiments from stage to stage. **(B)** In *serial* operation, the SDL performs each task in order and waits for one experiment to finish before starting the next. **(C)** In *asynchronous parallel* ("conveyor belt") operation, the SDL begins a new experiment as soon as the first stage is available. In this way, no stage is sitting idle for a significant amount of time and the experimental throughput is significantly increased. In the example above, 9 experiments are completed in asynchronous parallel operation in the same time it took to complete 3 in serial operation. Asynchronous parallel operation, however, causes a delay in the return of data as experiment N+3 must begin before experiment N is complete.

In this work, we fill this gap by building a simulator for a multi-stage SDL and comparing optimisation strategies for handling delayed feedback from asynchronous parallelized operation. We compare the performance of four optimisation strategies as well as compare their performance when subject to different amounts of delay and different number of dimensions on various test functions. Our simulation results showcase the trade-off between the efficiency gains of asynchronous parallel operation and performance loss due to delayed feedback.

2 Setup and Experiments

We built a simulator for a multi-stage SDL and compare optimisation strategies for dealing with delayed feedback and asynchronous parallelized operation. The simulator is based on the basic Bayesian optimisation example in BoTorch [2]. The simulated experiments are performed by testing a set of x-values on a synthetic experiment defined by a function. Noise is artificially added to the result to mimic experimental noise. To mimic the effects of using multiple stages in asynchronous parallel, the results are withheld from being returned until a predefined number of experiments have been completed. A delay of N would represent the asynchronous parallel operation of an SDL with N+1 stages. It should be noted that SDLs can typically operate with any number of experiments running in parallel, capped by the number of stages it has. With the simulator, we run synthetic experiments on three different test functions that are supposed to represent different material's behaviour:

(1) **Ackley test function:** an optimisation test function resembling a "needle-in-a-haystack" problem [16] whose dimensions can be scaled up and down (Appendix Figure 5A and equation 1).
(2) **Levy test function:** a reasonably challenging common optimisation test function whose dimensions can be scaled up and down (Appendix Figure 5B and equation 2).
(3) **SDL test function:** Using real SDL data from [15], we build a ground truth Gaussian Process (GP)[14] model from 177 previously run SDL experiments for maximizing the conductivity of functional coatings.

We choose the Ackley and Levy test functions as they are established benchmarks in the Bayesian Optimisation literature [4, 7, 20], and also allow evaluation across increasing numbers of dimensions. The SDL test function is designed to mimic the natural behaviour of functional coatings. In line with classic benchmarks, normally distributed noise is added with standard deviation of 0.5 and 2e5 for Ackley/Levy and SDL surfaces, respectively. Bounds for the x-dimensions of each surface can be found in the Appendix. We then compare the following search strategies:

(A) **Random:** The worst-case baseline any strategy needs to beat. For each input parameter x, draw a sample from a uniform probability distribution with upper and lower bounds determined by the problem parameter space.

(B) **Expected Improvement (EI):** This is the classic EI implementation, using the analytic formula for candidate calculation.

(C) **Noisy Expected improvement (qNEI):** This is the same as EI, but includes a model for the observed noise, as is commonly assumed in practical applications.

(D) **Mode cycling:** as proposed by [15] whose data we use for our SDL test function. This method cycles between the upper confidence bound (UCB) acquisition function and a space-filling algorithm that selects a points furthest from all other points.

We choose the random search strategy as a worst-case baseline that every strategy should be able to beat. EI is the most popular search strategy [9] and as such most relevant for comparison. qNEI is used to showcase a search strategy that accounts for noise, the standard setting SDLs operate in. The Mode-cycling is used to benchmark qNEI against a method that was used on a real-world SDL to handle asynchronous parallelisation. To counter the sub-optimal throughput of asynchronous parallel operation of a multi-stage SDL, we extend EI and qNEI with the implementation of a *pending points masks* available in BoTorch [3]. This strategy modification prevents recommendation of candidates that are running but have pending results. We expect EI and qNEI extended with the pending points mask to yield the best performance across all strategies. For each simulation, we average over 30 optimisation trials, using ten random initial points and a total of 100 observations. We look at delays $\in \{0, 1, 3, 5, 7\}$ and dimensions $\in \{3, 5, 7\}$, except for the SDL test function which is fixed at 7 dimensions. The length of the cycle in the Mode Cycle search strategy was varied depending on the amount of delay used in the simulation trial. For a delay of 0, 1, and 3, the mode cycle search strategy cycled between UCB with beta values 0.25, 2.5, and 25 and a space filling algorithm as described in [15]. For a delay of 5, the mode cycle search strategy cycled between UCB with beta values 0.1, 0.25, 1, 2.5, and 25 and the space filling algorithm. For a delay of 7, the mode cycle search strategy cycled between UCB with beta values 0.1, 0.25, 0.5, 1, 2, 4, and 10 and the space filling algorithm. These choices are based on a heuristic approach to balance exploration and exploitation. For each simulation, a dataset of all x-values and their resulting y-value found from the test function are saved and analyzed.

3 Results & Discussion

First, the performance loss of the simulated SDL due to delayed feedback was validated. The simulator was used to test the EI acquisition function on the SDL test function and the results are shown in figure 2. The running best y-value is averaged over thirty trials and standard deviation is shown as the shaded region. It can be observed that optimisations with no delay find the global maximum in the fewest iterations on average. As the amount of delay increases, the number of iterations on average required to reach the global max increases. It should be noted that observations occasionally exceed the test function global maximum due to the addition of noise within the simulation. Exceeding a theoretical limit may be possible in real experiments as there may be measurement noise which may cause the observations to exceed the theoretical maximum. The cumulative regret was then calculated and used as a metric of optimiser performance. The cumulative regret is calculated by calculating the area between the running best average curve and the global maximum from the first to the one hundredth observation. A lower cumulative regret is better because it means that the area is smaller and the optimiser was able to reach the global maximum faster. It can be seen in figure 2B that the cumulative regret increases as the amount of delay increases. Because delay is associated with running experiments in asynchronous parallel, increased delay also typically means increased experimental throughput. This figure clearly illustrate the trade-off between experimental throughput and optimiser performance for an SDL operating experiments in asynchronous parallel. An SDL operator must balance this trade-off. For an expensive experiment, an SDL operator may choose to prioritize minimizing the number of experiments performed by running the SDL in serial operation as this will increase the likelihood to reach the global maximum in fewer experiments. For an inexpensive or exploratory experiment, it may be more practical to get as much data as possible and prioritize experimental throughput by increasing the amount of experiments running in parallel.

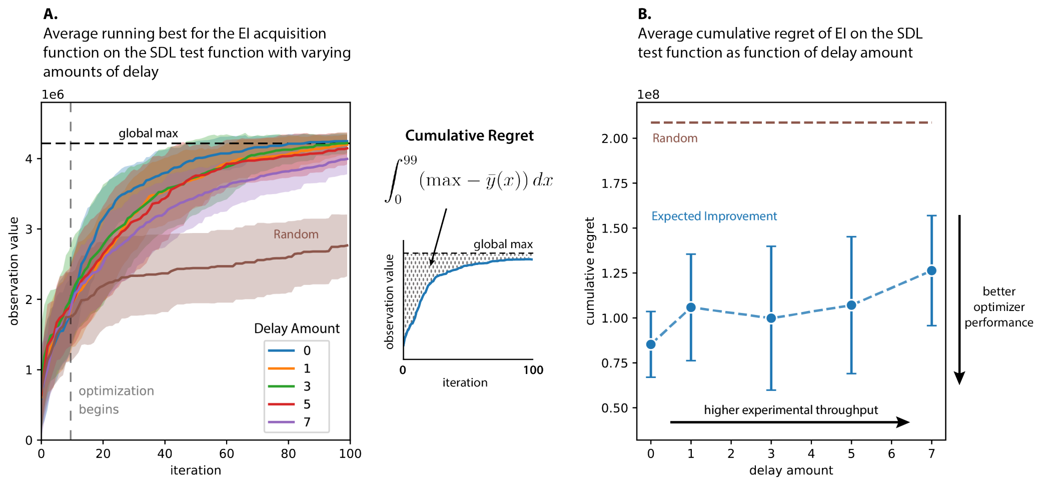


Figure 2: (A) The EI acqusition function was used to optimise the SDL test function modeled using real SDL data. The average running best y-value is shown up to 100 iterations and averaged over 30 trials. The shaded region represents the standard deviation of the running best value at each iteration. (B) Cumulative regret was calculated by taking the area between the curve and the global maximum from observation 0 to 99 for each running best in (A). Cumulative regret is analogous to average optimiser performance, as a lower cumulative regret means that the optimiser has reached the global maximum in less iterations. The cumulative regret increases as the amount of delay increases. The average cumulative regret of the random baseline acquisition strategy is shown for comparison. The error bars represent the standard deviation of the cumulative regret over all thirty iterations.

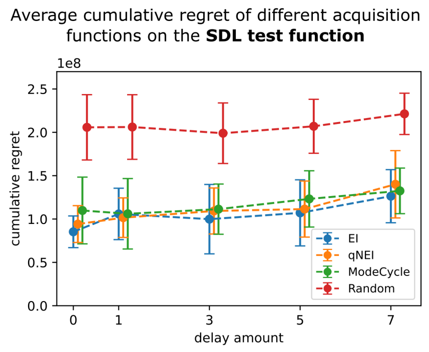


Figure 3: Cumulative regret for different optimisation strategies on the SDL test function. The error bars represent the standard deviation of the cumulative regret over all thirty trials. The x-position of the data have been staggered to improve readability of the error bars.

Two additional acquisition functions were tested and compared: qNEI, to evaluate whether the noisy version could perform better, and Mode-Cycling, as proposed by Rupnow et al.[15] as a way to reduce the repeated selection of identical experiments caused by delayed feedback. The cumulative regret of these tests are shown in 3. It can be seen that EI, qNEI, and ModeCycle all perform very similarly, with the cumulative regret increasing as delay amount increases. As expected, random sampling remains relatively constant since it does not depend on prior data to select new experiments. The average running best plots for each acquisition function and delay amount can be found in Appendix 7. To validate that delay remains an issue on other optimisation problems, the same simulations were performed on the Ackley test function. Additionally, the number of dimensions was varied to get a sense of how dimensionality affects performance due to delay. The cumulative regret for each search strategy on the Ackley test function are shown in Figure 4. It can be seen that as delay increases, the cumulative regret also increases linearly. As the dimensionality increases, the cumulative regret of EI, qNEI, and ModeCycle approach the cumulative regret of random sampling. The average running best plots for each dimension and delay amount for the Ackley function can be found in Appendix 8.

The cumulative regrets for each search strategy on the Levy test function are shown in Appendix Figure 6. The results are very similar to those of the Ackley function. However, increasing the number of dimensions in the Levy function significantly amplifies the cumulative regret. The average running best plots for each dimension and delay amount for the Levy function can be found in Appendix 9.

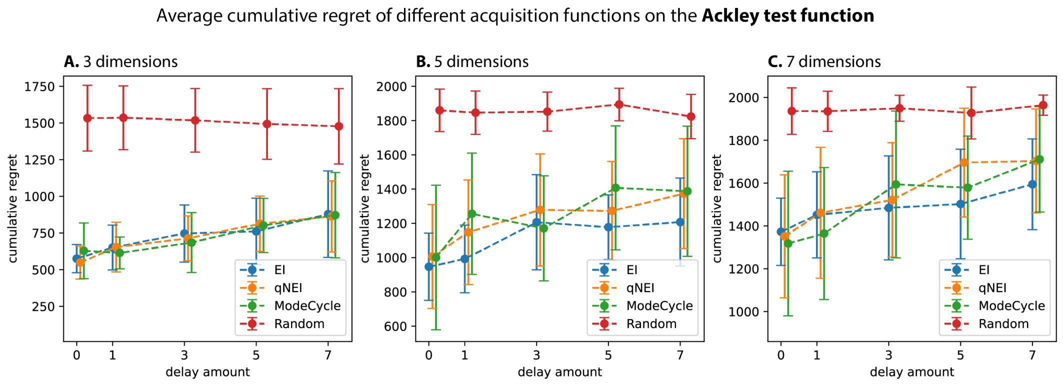


Figure 4: Cumulative regret for different optimisation strategies on the Ackley test function in (A) 3, (B) 5, and (D) 7 dimensions. The error bars represent the standard deviation of the cumulative regret over all thirty trials. The x-position of the data have been staggered to improve readability of the error bars.

Our simulation results showcase the trade-off between the efficiency gains of asynchronous parallel operation and performance loss due to delayed feedback. As expected, problems get harder with increasing dimensionality. Additionally, as the number of parallel stages (i.e. delay) increases, performance also decreases. As expected, all search strategies outperform the random strategy in all cases. However, there is not enough evidence in our analysis to suggest that one acquisition strategy performs better than another in the presence of delay, especially in higher dimensions. To find an acquisition strategy that performs better in the presence of delayed feedback would require more trials to be run, an increased maximum number of iterations, and compared to a wider range of acquisition strategies. In the context of SDL operation, the decision whether to add delay to increase throughput or to add a dimension to increase the search area seems problem dependent – both increase the cumulative regret, though our tests found that adding delay seems to produce a smaller impact then adding dimensions.

Limitations The results presented here represent a simulated SDL and not a real SDL. However, we expect these results to hold true for a real SDL. It is important to remember, though, that the results presented here represent the average performance of each strategy. An optimisation trial on a real SDL may perform better or worse than average. Additionally, cumulative regret is not the only metric for measuring optimiser performance. Optimisation algorithms have complicated behaviour that should not necessarily be quantified by a single number. These results are true only for the three test functions presented here. Results may vary when exploring other material spaces or synthetic surfaces. Our work intends to guide SDL operators to use their machine more effectively and efficiently. Improvements in optimisation strategies will lead to better mathematical optimisations, but may or may not necessarily lead to higher performance materials found in fewer experiments.

Conclusion Considering the limited experimentation budgets commonly encountered in SDLs, simulations are an effective way to test search strategies in advance of deployment on an SDL. In this work we show that increasing throughput by running experiments in asynchronous parallel comes with a trade-off of reducing the performance of Bayesian optimisation algorithms. We show that increasing delay by adding stages to an SDL produces a smaller impact to optimisation performance then adding dimensions to the materials problem space. We also compare a few different acquisition strategies, but were inconclusive in determining a champion strategy. These results allow an SDL operator to make informed recommendations on which search strategy to use and — on average — find the global optimum faster than random guessing. As a next step, we are looking forward to collaborating on on implementing and evaluating these algorithms in real-world SDLs, including globally interconnected labs such as [11, 18] which face similar asynchronous parallelization challenges, to further refine their search strategies and adjust them to specific applications. To facilitate this process, we are making a plug-in pipeline for the pending points implementation available online[1]. This will allow labs to easily access and integrate it into their SDLs.

[1]`https://matterhorn.studio/pages/seminars/search-strategies-with-pending-points/`

References

[1] Ahsan Alvi et al. "Asynchronous Batch Bayesian Optimisation with Improved Local Penalisation". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 253–262. URL: `https://proceedings.mlr.press/v97/alvi19a.html`.

[2] BoTorch authors. *Closed-loop batch, constrained BO in BoTorch with qEI and qNEI¶*. 2021. URL: `https://botorch.org/tutorials/closed_loop_botorch_only`.

[3] Maximilian Balandat et al. "BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization". In: *Advances in Neural Information Processing Systems 33*. 2020. URL: `http://arxiv.org/abs/1910.06403`.

[4] Poompol Buathong et al. *Bayesian Optimization of Function Networks with Partial Evaluations*. 2023. arXiv: `2311.02146 [stat.ML]`.

[5] Benjamin Burger et al. "A mobile robotic chemist". In: *Nature* 583.7815 (2020), pp. 237–241.

[6] Connor W. Coley et al. "A robotic platform for flow synthesis of organic compounds informed by AI planning". In: *Science* 365.6453 (2019), eaax1566. DOI: `10.1126/science.aax1566`. eprint: `https://www.science.org/doi/pdf/10.1126/science.aax1566`. URL: `https://www.science.org/doi/abs/10.1126/science.aax1566`.

[7] Mike Diessner et al. "Investigating Bayesian optimization for expensive-to-evaluate black box functions: Application in fluid dynamics". In: *Frontiers in Applied Mathematics and Statistics* 8 (2022). ISSN: 2297-4687. DOI: `10.3389/fams.2022.1076296`. URL: `https://www.frontiersin.org/articles/10.3389/fams.2022.1076296`.

[8] Robert W. Epps et al. "Artificial Chemist: An Autonomous Quantum Dot Synthesis Bot". In: *Advanced Materials* 32.30 (2020), p. 2001626. DOI: `https://doi.org/10.1002/adma.202001626`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/adma.202001626`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.202001626`.

[9] Peter I. Frazier. *A Tutorial on Bayesian Optimization*. 2018. arXiv: `1807.02811 [stat.ML]`.

[10] Aldair E. Gongora et al. "A Bayesian experimental autonomous researcher for mechanical design". In: *Science Advances* 6.15 (2020), eaaz1708. DOI: `10.1126/sciadv.aaz1708`. eprint: `https://www.science.org/doi/pdf/10.1126/sciadv.aaz1708`. URL: `https://www.science.org/doi/abs/10.1126/sciadv.aaz1708`.

[11] Dan Guevarra et al. "Orchestrating nimble experiments across interconnected labs". In: *ChemRxiv* (2023). DOI: `10.26434/chemrxiv-2023-jgc8g`.

[12] Jeffrey Kirman et al. "Machine-Learning-Accelerated Perovskite Crystallization". In: *Matter* 2.4 (2020), pp. 938–947. ISSN: 2590-2385. DOI: `https://doi.org/10.1016/j.matt.2020.02.012`. URL: `https://www.sciencedirect.com/science/article/pii/S2590238520300746`.

[13] B. P. MacLeod et al. "Self-driving laboratory for accelerated discovery of thin-film materials". In: *Science Advances* 6.20 (2020), eaaz8867. DOI: `10.1126/sciadv.aaz8867`. eprint: `https://www.science.org/doi/pdf/10.1126/sciadv.aaz8867`. URL: `https://www.science.org/doi/abs/10.1126/sciadv.aaz8867`.

[14] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning.* Adaptive computation and machine learning. MIT Press, 2006, pp. I–XVIII, 1–248. ISBN: 026218253X.

[15] Connor C Rupnow et al. "A self-driving laboratory optimizes a scalable process for making functional coatings". In: *Cell Reports Physical Science* 4.5 (2023).

[16] Alexander E Siemenn et al. "Fast Bayesian optimization of Needle-in-a-Haystack problems using zooming memory-based initialization (ZoMBI)". In: *npj Computational Materials* 9.1 (2023), p. 79.

[17] Kelsey L Snapp et al. "Autonomous Discovery of Tough Structures". In: *arXiv preprint arXiv:2308.02315* (2023).

[18] Felix Strieth-Kalthoff et al. "Delocalized, Asynchronous, Closed-Loop Discovery of Organic Laser Emitters". In: *ChemRxiv* (2023). DOI: `10.26434/chemrxiv-2023-wqp0d`.

[19] Sonja Surjanovic and Derek Bingham. "Virutal Library of Simulation Experiments". In: *Simon Fraser University* (2013). URL: `http://www.sfu.ca/~ssurjano/ackley.html`.

[20] Jialei Wang et al. “Parallel Bayesian global optimization of expensive functions”. In: *Operations Research* 68.6 (2020), pp. 1850–1865.

4 Appendix

Acknowledgements Thank you to the workshop reviewers for their detailed feedback that allowed us to improve and clarify our work. We are also grateful for the support for this work from the UKRI Innovate UK Transformative Technologies Grant 2023 Series.

Ackley test function

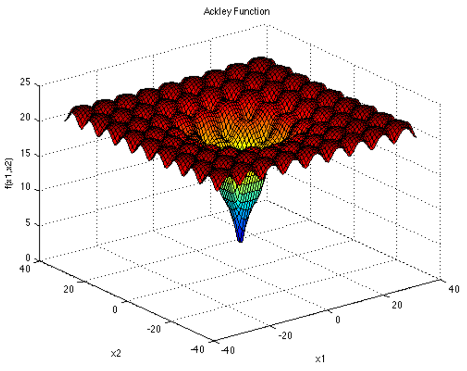
$$f(\mathbf{x}) = -a \cdot \exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + exp(1) \tag{1}$$

where a, b, c are constants that control the characteristics of the function and d controls the dimensionality of the input space. We used the packaged BoTorch version of the function which defines $a = 20$, $b = 0.2$, and $c = 2\pi$. The bounds for the Ackley test function are [-32.768, 32.768] for all d. The Ackley test function with $d = 2$ can be seen in Figure 5A

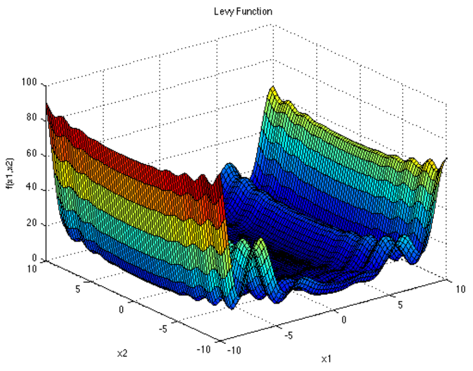
Levy test function

$$f(\mathbf{x}) = \sin^2(\pi w_1) + \sum_{i=1}^{d-1}(w_i - 1)^2\left[1 + 10\sin^2(\pi w_i + 1)\right] + (w_d - 1)^2\left[1 + \sin^2(2\pi w_d)\right] \tag{2}$$

where $w_i(x) = 1 + \frac{x_i - 1}{4}$, for all $i = 1, ..., d$. The bounds for the Levy test function are [-10, 10] for all d. The Levy test function with $d = 2$ can be seen in Figure 5B



(a) **Ackley**



(b) **Levy**

Figure 5: Both Ackley (A, eq. 1) and Levy (B, eq. 2) are comparatively challenging test functions that easily extend to higher dimensions [19]. Ackley is challenging for hill climbing algorithms that tend to get stuck in one of its many local minima, with its global minimum at $f(\mathbf{x} = 0) = 0$. Levy is similar to Ackley but not symmetric around its global minimum at $f(\mathbf{x} = 0) = 0$ and is often used to test algorithms aimed at handling high-dimensional problems with multiple local optima. In the context of our simulator, we chose to maximize the negative of each function.

SDL test function

The SDL test function was based on a GP model built using training data from Rupnow et al. [15] with a noise of 2e5 and fit using marginal log likelihood. The resulting model had cross validation r^2 of 0.9907. The bounds for the SDL test function shown below in Table 1:

Table 1: Bounds for the SDL test function

	input variable	lower bound	upper bound	units
x_1	DMSO content	0	0.3	v/v
x_2	precursor concentration	10	20	mg/mL
x_3	spray flow rate	2	8	µL/s
x_4	air flow rate	65	100	control valve %
x_5	number of passes	1	10	passes
x_6	spray height	10	25	mm
x_7	hotplate temperature	220	300	°C

Levy Cumulative Regret

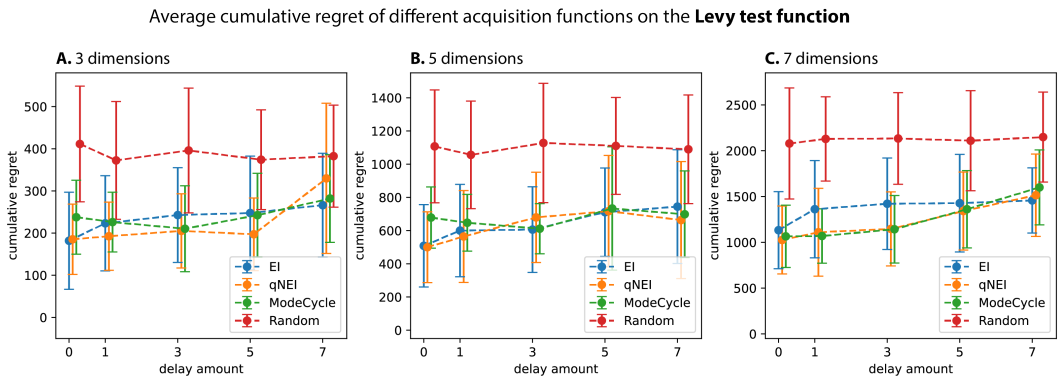


Figure 6: Cumulative regret for different optimisation strategies on the Levy test function in (A) 3, (B) 5, and (D) 7 dimensions. The error bars represent the standard deviation of the cumulative regret over all thirty trials. The x-position of the data have been staggered to improve readability of the error bars.

Average running best optimisations

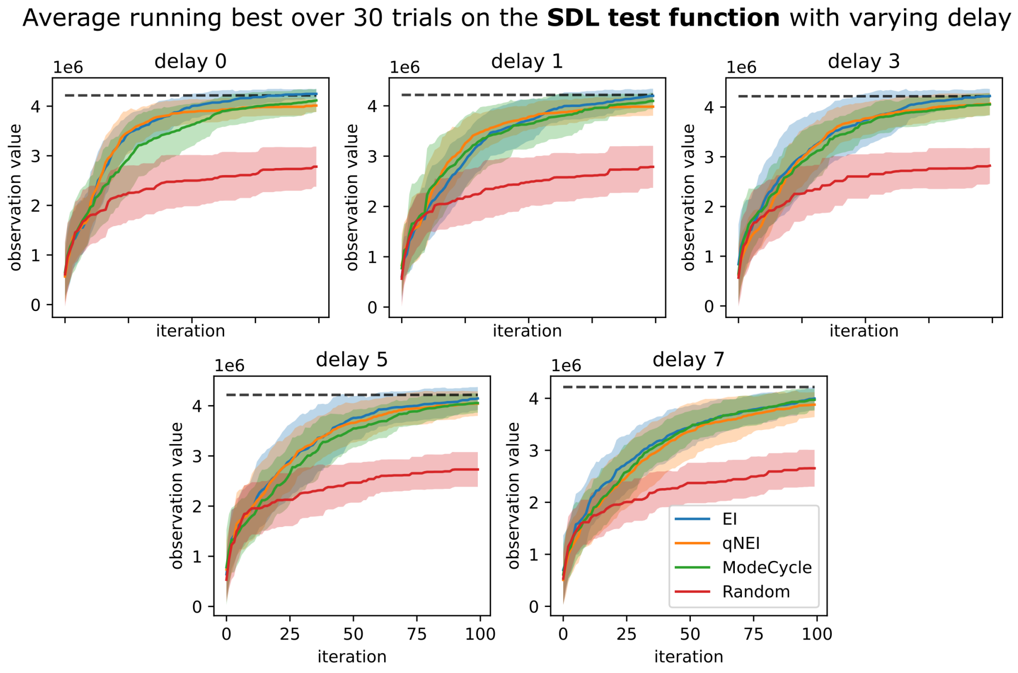


Figure 7: Average running best observation over 30 optimisation trials on the SDL test function for delays $\in \{0, 1, 3, 5, 7\}$. The line represents the mean running best observation while the shaded region represents the standard deviation. The dashed line represents the global maximum.

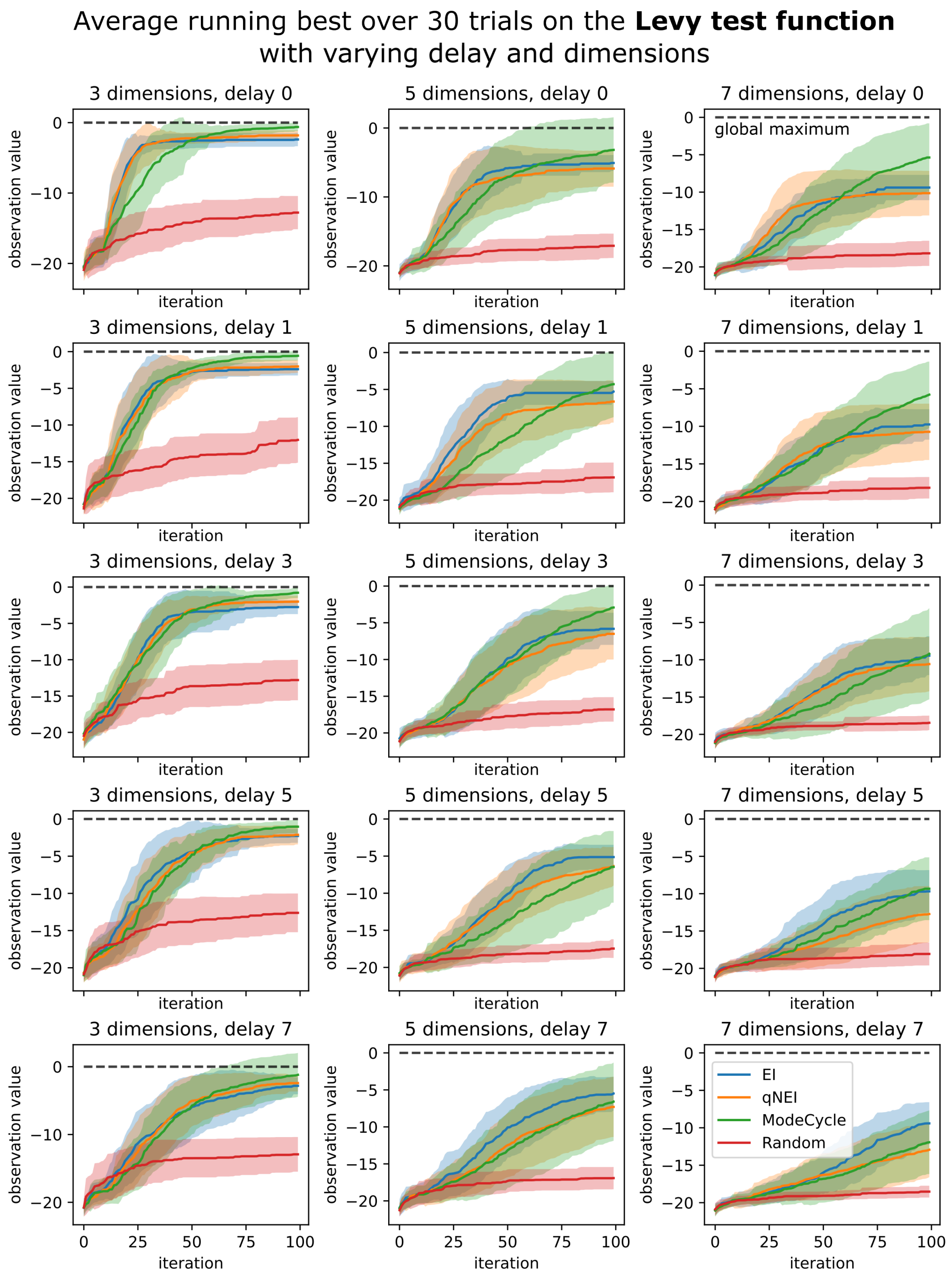


Figure 8: Average running best observation over 30 optimisation trials on the Ackley test function for delays $\in \{0, 1, 3, 5, 7\}$ and dimensions $\in \{3, 5, 7\}$. The line represents the mean running best observation while the shaded region represents the standard deviation. The dashed line represents the global maximum.

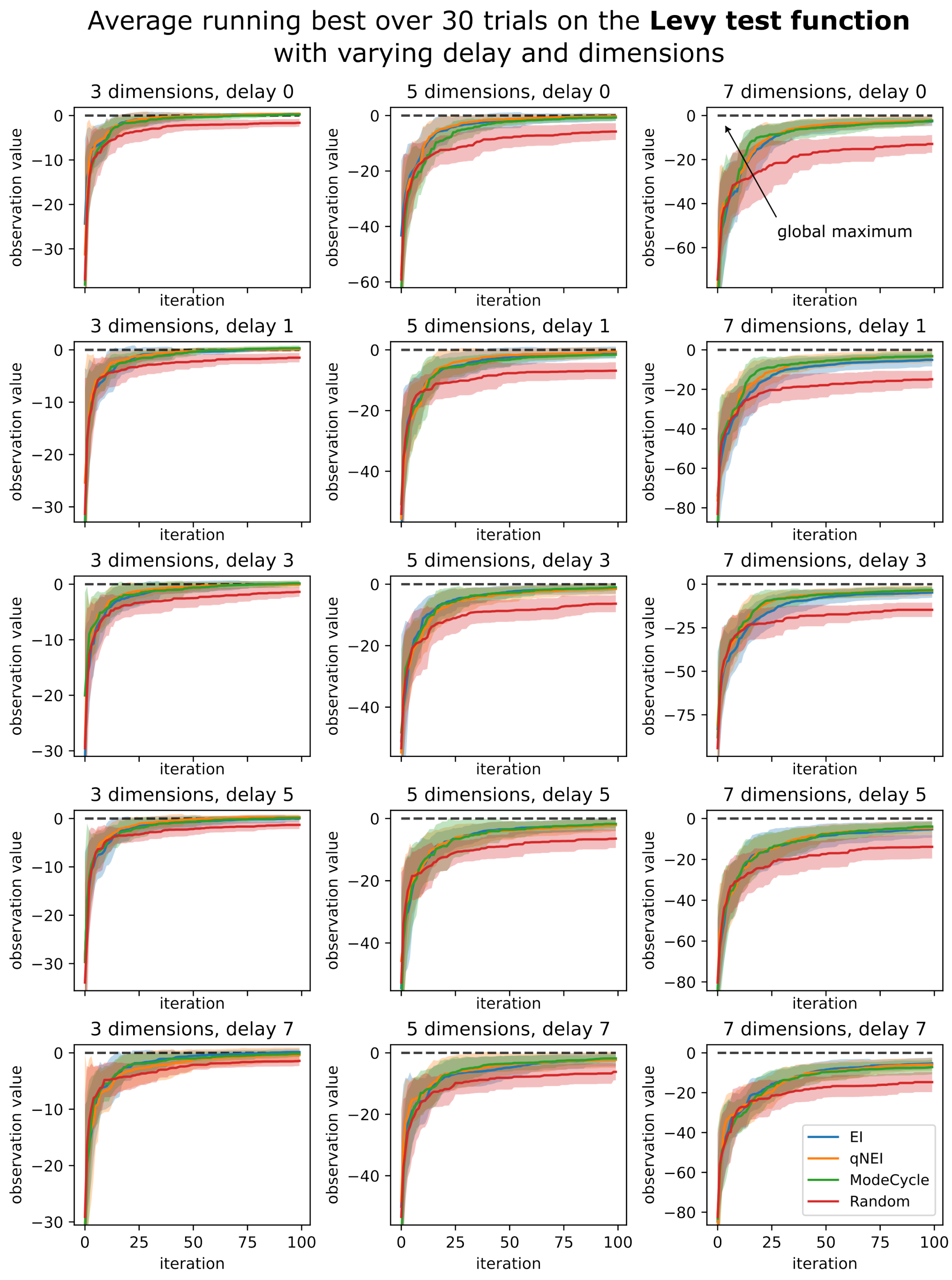


Figure 9: Average running best observation over 30 optimisation trials on the Levy test function for delays $\in \{0, 1, 3, 5, 7\}$ and dimensions $\in \{3, 5, 7\}$. The line represents the mean running best observation while the shaded region represents the standard deviation. The dashed line represents the global maximum.