Imperial College of Science, Technology and Medicine
Department of Mathematics

Towards improving Bayesian optimisation for the physical sciences via long-term planning

Jose Pablo Folch

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Modern Statistics and Statistical Machine Learning
of the University of London and the Diploma of Imperial College,
September 2024

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC). Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose. When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes. Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Abstract

Design of experiments is a vital area of research in all sciences. Initially an arduous and expensive task due to the need to manually do experiments, there has been massive opportunity to advance scientific research through the use of automation. Bayesian optimisation has emerged as the leading tool for expensive black-box maximisation, in particular with regards to selecting the configuration of expensive-to-train machine learning models.

In the physical sciences we face a much more complicated task, where the dynamics of experimentation are subject to strict physical constraints, time-delays, and varying accuracy of measurements. In order to unleash the potential of Bayesian optimisation for scientific discovery it is vital to adapt the algorithms to all the constraints faced in real-world scenarios. In this thesis we will tackle three of those problems through *planning*.

We look at multi-fidelity and asynchronous batch optimisation. These refer to the setting where we can use cheap measurements to approximate expensive experiments which may take a long time to run. The desynchronisation between experiments and the potential to parallelise them results in the *asynchronous* setting, where we must choose new experiments while taking into account other yet-unfinished experiments. We show there is a close relationship between both areas and propose an algorithm for combining them efficiently.

Later, we look at two related physical problems. Traditionally Bayesian optimisation can freely query any experiment in the search space. However, in the physical sciences we often cannot make large changes between experiments as it may be costly or break the system dynamics. By planning full trajectories through the search space we are able to efficiently optimise the underlying functions, and extend Bayesian optimisation to a whole range of physical problems. We showcase connections with previous Bayesian optimisation approaches, real-world applications, and experimentally validate the benefits of the proposed methods.

Acknowledgements

The list of people who have had an important impact in my life during my PhD is far too long to be included in such a short document, however, even at the risk of forgetting someone and then feeling terrible, I will do my best to include as many people as possible – so forgive me if the section is rather long.

- I would like to begin by thanking my supervisors, Ruth and Mark. Having been chosen by both of you in the random selection process that is the StatML application pool was perhaps the luckiest thing that happened during my PhD. I would like to thank you for the frequent advice, help, friendship, and also patience whenever I needed it. You made my PhD journey extremely rewarding, opening so many doors and opportunities, while also allowing me to have fun and remain positive throughout. I hope you keep working together for many years as you make a fantastic duo.

- On a related note, I would like to thank Calvin. Even though it is not official, I will always consider you as my third supervisor. Your help was invaluable and is present in every piece of work I have done in the last four years. It has been great to see you go from post-doc to lecturer, and now an amazing supervisor to other students. They are very lucky to have you.

- Additionally, I would like to thank the BASF team who supported me the last four years; Behrang, Robert, and David. Thank you for allowing me to always chase my curiosity and for providing invaluable feedback across the many projects we worked on together. After some years of working together, finally getting to meet you in person when I visited BASF was one of the highlights of my PhD.

- I would like to thank everyone at StatML who provided support and allowed me to thrive in a research environment, particularly the management team Axel, Judith, Vanessa, Joanna, and later Sarah and Ed. Also to my friends and cohort who somehow managed to make the whole COVID time fun: Ben H, Ben T, Andrea, Xing, Tresnia, Mike, Alex L, Oscar, Lucile, Tom, Jake, Desi, Andrew, Zoi, Emmeran, Efthymios, Alex B, Anna,

Bryan, Antoine, Daniel, Enrico, Jase, Adam, James, to name a few... our trips to Berlin, Como and Finland will remain some of the biggest highlights of my PhD.

- The Computational Optimisation Group was the group I would frequent most on a daily basis. I thank everyone for making the day-to-day of the last four years so enjoyable and fun; I would not have been able to achieve what I have without every single one of you. I really appreciated everything I experienced and all the support and love I felt from the group. I'd like to thank Alex for all the laughs and guidance, James who helped me improve every single idea I brought to him, Ruby whose advice and perspective was always the most helpful, Daniel who always joined me on crazy adventures and reminded me the standard of what a researcher should be, Shiqiang for being the most fun and helpful person to work with, Juan who I consider my life coach, Chryssa who is one of the most caring people I know. Thanks to Aras who unfortunately never sat in the same office but was always a good friend. I would also like to mention the visiting researchers, Amalia, Mujin, and Antonio who I started missing as soon as they left. Finally, I would like to thank the newest members of the group, and who I wish I had overlapped with for longer: Toby, Becky, Phil, Yilin, and Jixiang; the future of COG could not look brighter.

- The other group I frequented, OccaMLab, was just as important in my PhD journey. It would be difficult to find a group of more curious and talented people in academia. I learnt so much from every person in the group and so much of my learning and understanding of machine learning comes from casual conversations in the group. Anish taught me everything I know about causality, Artem I will have to thank when I inevitably end up training GPs on large data-sets, Ruby whose transfer learning work I am trying to implement as I write this, Seth who taught me that belief propagation is not just something I should practice for my self-esteem, and Tycho who is the only person I know that can consistently train a Bayesian neural network (while playing jazz). I am also grateful for having shared some time with Becky, Guiomar, Jake, Marcel, and Stefano and I wish we had overlapped longer.

- I would also like to thank other researchers and colleagues who had an important impact in my PhD. Firstly Mojmír, I feel very privileged to have had the opportunity to work

with you and I will forever be extremely grateful for having invited me to Zurich and for having been such a good friend and collaborator. I would also like to thank Andreas for the opportunity, my time at ETH has been one of the best learning experiences of my PhD. Moving forward I would like to thank Clara and Stefan for all the patience given to me while I was learning how to be a supervisor. Finally, I would like to thank Linden for always being open to chat to me and help me understand the applications of my work – I am very excited for what is coming next in SOLVE and look forward to working with you for many years. I'd also like to thank Tom and Sarah for making the viva preparation a fun journey we did together.

- None of my work would have been possible without all the friends who supported me outside of my academic life. I would like to thank Chidubem, Chow, Lombe, Jay, Issy, Tomiwa, Jean-Luc, Seb, Louis, Alex M, Victor, Joe, Iker, Pablo, Zandi, Hannah, Carlota, Beatty, Juan, Jorge, Jed, Shona, Jamie, Vincenzo, Carlos, Alex O, and anyone else who I might have missed, for being my family away from home.

- Igual me gustaría darle las gracias a todos mis amigos en México: Armella, Pol, Bolio, Caraza, Patxi, Iñigo, Pablito, Memo, Iván, Fran, Manuel, Kiko, Gozgo, Maldo, Jero, Diego, Majo, Lu, Fer, Pop, Carla, Dani, Vero, Marina, Estefi, Paloma, Ren, Borja, Ana Pau, Paloma, Duba, Pier, Chava, Fabio, Castillo, Emi, Rangel, Jimena, Andrea, Rocío y también a cualquier amigo o amiga que se me haya olvidado incluir. Me hacen sentir como si nunca me hubiera ido, y siempre les voy a agradecer que siento su cariño y apoyo todos los días desde Londres.

- De mi familia me gustaría agradecer a todos mis tíos, tías y tía-abuela: Alejandro, Erik, Magali, Manuel, Mario, Yeye, Nathalie, William, Lore, Denis, Federico, Memo, Joseline, Martha, Alejandra, Eric y Jerry. Igualmente a mis primos y primas Ale, Pau, Isa, Mateo, María, Ramón, Roberta, Cami, Humberto, Mari José, y Memo. Siempre me dan mucho cariño e interés en lo que he estado haciendo y me inspiran a seguir dando todo mi esfuerzo. Quiero dar un agradecimiento especial a mi hermano, Alberto, perdón si a veces me he equivocado en nuestra relación, quiero que sepas que te tengo muchísima admiración y que todos los días me inspiras a ser una mejor persona. A mi hermana Ligia, te quiero agradecer por siempre ser mi soporte cuando las cosas han sido más difíciles, y por siempre

hacer el esfuerzo para visitarme. A todos los quiero muchísimo.

- Creo que se puede decir que nadie ha sacrificado más por mi doctorado que mis abuelas. Lo que les debo a ustedes es algo que nunca voy a poder pagar – no hay nadie que me dé más apoyo que ustedes y no hay nadie que me extrañe más. Las porras que me llevan echando los últimos 8 años son las más importantes para mí, me dan tranquilidad y fuerza dando mi todo y aunque estemos lejos siempre las siento presentes.

- Para acabar le quiero dar las gracias a mi mamá y a mi papá. Son la definición de lo que es el amor y apoyo incondicional. Gracias por dejarme seguir mis sueños siempre apoyándome en todo. Me han dado todo el amor, enseñanzas y recursos sin nunca pedir nada a cambio. No existen mejores papás en el mundo, este logro es igual suyo que mío.

Dedication

This thesis is dedicated to anyone who leaves their home to chase their ambitions.

"How the hell did I get here?"
"Sometimes I sit and I wonder?"
"I'm the version of me I always imagined when I was younger"
Little Simz

Contents

Abstract **i**

Acknowledgements **iii**

1 Introduction **1**

1.1 Motivation and Objectives 1

1.2 Contributions 5

1.3 Statement of Originality 6

1.4 Publications 6

2 Background Theory **8**

2.1 Introduction 8

2.2 Gaussian Processes 8

2.2.1 Kernel functions 10

2.2.2 Kernel hyper-parameters and the marginal-likelihood 11

2.2.3 Finite dimensional approximations and sampling 13

2.3 Bayesian Optimisation 14

2.3.1 Acquisition functions 15

2.3.2 Batch Bayesian Optimisation 18

2.4 Non-myopic Bayesian Optimisation 19

2.4.1 The myopia of Bayesian optimisation 19

2.4.2 Markov Decision Processes 20

2.4.3 Full horizon problem formulation 21

2.4.4 Look-ahead Bayesian optimisation 21

2.4.5 The failures of look-ahead Bayesian optimisation 22

3 Combining Multi-fidelity Modelling and Asynchronous Batch Optimisation 24

3.1 Preliminaries 24

3.2 Related Work 26

3.3 Multi-fidelity Modelling 28

3.3.1 Independent Gaussian Processes 30

3.3.2 Multi-task Gaussian Processes 30

3.3.3 Further Extensions 32

3.4 Multi-fidelity Bayesian Optimisation 32

3.4.1 Multi-fidelity Upper Confidence Bound 33

3.4.2 Multi-fidelity Max Entropy Search 36

3.5 Asynchronous Batch Bayesian Optimisation 37

3.5.1 Thompson Sampling 37

3.5.2 Parallel Querying Through Fantasies 38

3.5.3 Local Penalisation 39

3.6 Relationship between multi-fidelity and asynchronous batch methods 40

3.7 Asynchronous Multi-fidelity Batch Optimisation 44
3.7.1 Problem Setting 44
3.7.2 A general algorithm for any acquisition function 45
3.7.3 Parameter Estimation 46
3.8 Experimental Results 50
3.8.1 Synthetic Benchmarks 50
3.8.2 High-dimensional optimisation 54
3.8.3 Battery Material Design 56
3.9 Discussion 59
3.10 Looking back... 61

4 SnAKe: Bayesian Optimisation with Pathwise Exploration 62

4.1 Preliminaries 62
4.1.1 Problem Set-up 64
4.2 Related Work 65
4.3 An ordering-based algorithm 67
4.3.1 General Approach 67
4.3.2 Creating a Batch Through Thompson Sampling 67
4.3.3 Creating a Path via the Travelling Salesman Problem 68
4.3.4 Naively Updating the optimisation path 69
4.4 Escape Analysis 69
4.4.1 Areas without stationary points 71
4.4.2 Areas with stationary points 72

4.4.3 Escaping with ϵ-Point Deletion 73

4.5 The SnAKe algorithm 75

4.6 Computational Considerations 75

4.7 Practical extensions of SnAKe 77

4.7.1 Self-stopping SnAKe 78

4.7.2 Truncated SnAKe 79

4.7.3 Multi-objective SnAKe 79

4.8 Experimental Results 80

4.8.1 Synthetic Functions 81

4.8.2 Reaction Control on SnAr Benchmark 82

4.8.3 Finding Contamination Sources in Ypacarai Lake 85

4.8.4 Simultaneous optimisation of multiple objectives 86

4.8.5 Results on practical extensions 88

4.9 Discussion 90

4.10 Looking back... 91

5 Transition Constrained Bayesian Optimisation via Markov Decision Processes 92

5.1 Preliminaries 92

5.1.1 Related works 94

5.2 Visual abstract of MDP-BO 97

5.3 Deriving a utility for maximiser identification 97

5.3.1 Optimum identification via hypothesis testing 97

5.3.2 Utility with Embeddings . . . 101

5.4 The State-Action Visitation Polytope . . . 102

5.5 Experiment Design in Markov Chains . . . 103

5.6 Transition Constrained Bayesian Optimisation . . . 104

5.6.1 Expected Utility for Maximiser Identification . . . 105

5.6.2 Set of potential maximisers $\mathcal{Z}$. . . . 105

5.6.3 Discrete vs Continuous MDPs. . . . 106

5.6.4 General algorithm and Theory . . . 107

5.7 Solving the planning problem . . . 108

5.7.1 Adaptive Resampling: Non-Markovian policies. . . . 109

5.8 Continuous MDPs: Model Predictive Control . . . 110

5.8.1 Linearising the objective . . . 110

5.8.2 Practical Planning for Continuous MDPs . . . 112

5.9 Experiments . . . 112

5.9.1 Knorr pyrazole synthesis . . . 113

5.9.2 Monitoring Lake Ypacarai . . . 115

5.9.3 Free-electron laser: Transition-driven corruption . . . 115

5.9.4 Synthetic Benchmarks . . . 116

5.9.5 Ablation Study . . . 117

5.9.6 Computational study . . . 119

5.10 Discussion . . . 120

5.11 Looking back... . . . 120

6 Conclusion **122**

6.1 Summary of Thesis Achievements 122

6.2 Applications 123

6.3 Future Work 124

Bibliography **125**

A Implementation Details: **147**

A.1 Combining Multi-fidelity and Asynchronous Batch Bayesian Optimisation 147

A.1.1 Gaussian Processes 147

A.1.2 Methods 148

A.1.3 MF-MES 149

A.1.4 Benchmark Details 149

A.2 SnAKe: Bayesian Optimization via Pathwise Exploration 152

A.2.1 Gaussian Processes 152

A.2.2 Training the hyper-parameters of the Gaussian Processes 152

A.2.3 SnAKe 153

A.2.4 Classical Bayesian Optimization 154

A.2.5 Asynchronous Bayesian Optimization 155

A.2.6 Description of Benchmark Functions 155

A.2.7 Ypacarai Implementation Details 158

A.3 Transition Constrained Bayesian Optimisation via Markov Decision Processes . 159

A.3.1 Benchmark Details 159

A.3.2 Free-electron Laser 160

B Derivations **161**

B.1 Transition Constrained Bayesian Optimisation 161

B.1.1 Objective reformulation and linearization 161

B.1.2 Theory: convergence to the optimal policy 164

B.2 Kernel for ODE Knorr pyrazole synthesis 168

List of Tables

3.1 Comparison table for multi-fidelity methods, highlighting and summarising the strengths and weaknesses of each method. 42

3.2 Comparison table for asynchronous batch methods, highlighting and summarising the strengths and weaknesses of each method. 43

3.3 Summary of all the methods compared in this section. We split the table into three sections: the first one refers to UCB-based methods, the second to the our modifications of TuRBO, and for the third we include multi-fidelity Max Entropy Search (MF-MES) as a general baseline. Variance-based and Information-based multi-fidelity refers to using Equations (3.20) and (3.21) (respectively) to choose which fidelity to query. In the multi-fidelity column, V stands for "Variance-based" and I stands for "Information-based". In the batching column, LP stands for "Local Penalisation", TS stands for "Thompson Sampling. In the Modelling Column, GP stands for standard single GP, IGPs stands for "Independent GPs", and MOGP stands for "Multi-output GP". Local penalisation is implemented as described in [Alvi et al., 2019], using hard local penalisers and a local estimate of the Lipschitz constant. Thompson Sampling is carried out across a fixed grid in TuRBO trust regions. In all cases, for TuRBO, we use a single trust region. For all non-batching methods, we fill the batch through random sampling as to avoid idle resources. We used GPyTorch and BoTorch to implement the methods [Paszke et al., 2019, Gardner et al., 2018, Balandat et al., 2020]. 51

4.1 Ypacarai results. We present the average regret from 10 runs ± the standard deviation, multiplied by 10^3. O1, O2, O3 correspond to the different objectives. All runs are terminated after the cost exceeds 10 units (approximately 100 km). SnAKe is the best performer in 2 out of 3 objectives. 86

4.2 Ypacarai experimental results. We present the average regret from 10 runs ± the standard deviation multiplied by 10^3. MO1, MO2, MO3 represent the different maximums (one for each function). All runs are terminated after the cost exceeds 10 units (approximately 100 km). SnAKe is the only method with good regret in all metrics, as TrEI performs poorly in the first objective, and EIpu performs poorly in the second. 87

4.3 Results of multi-objective optimisation experiments on four dimensional benchmarks. Red and blue indicate linear and Tchebyshev scalarazations respectively. 89

5.1 Average acquisition function solving times for each practical benchmark. We give the solving times to the nearest second, and provide the size of the state-space, $|\mathcal{S}|$, the maximum number of actions one can take from a specific state, $|\mathcal{A}(S)|$, and the planning horizon. In all benchmarks we are able to solve the problem in a few seconds. 120

A.1 Benchmark and hyper-parameter information. Δ_{max} represents the size of the box constraints in the traditional benchmarks. For the synchronous benchmarks and for SnAr we used a noise level of $\sigma^2 = 0.001$. For the asynchronous benchmarks, and the knorr pyrazole example we used $\sigma^2 = 0.0001$. For the Ypacarai example we used $\sigma^2 = 0.001$ and $\sigma^2 = 0.01$ for the episodic and immediate feedback respectively. 160

List of Figures

2.1 Two GPs fit on the same data, with different hyper-parameters. We show the data as black dots, and the model's mean and confidence intervals in blue, as well as displaying the corresponding marginal log-likelihood of each model. . . . 11

2.2 Example of myopic against non-myopic decision making with a budget of $T = 2$. We plot the queries chosen by each strategy as vertical lines. In a single step, BO will choose to maximise the expected improvement by querying a point near the middle of the search space. At the next iteration, BO will be forced to choose between using the final query in either the right part of the space or the left, leaving one unexplored. A non-myopic decision maker can recognise that taking a sub-optimal step and searching the right hand side of the search space is correct, since we have one more query left to search the left hand side, which results in a more complete search. A conveniently hand-picked, bi-modal, underlying function plotted in black helps illustrate why this is important. . . . 20

3.1 Motivating example. On the left is pouch-cell battery and on the right are coin cell batteries [Chen et al., 2019, Dörfler et al., 2020, Liu et al., 2021]. Using coin cells, which are cheaper and faster to make than pouch batteries, we can approximate the behaviour of our target. Usually these experiments are carried out in large batches. 26

3.2 Building a multi-fidelity UCB, in the case where $M = 2$. The dotted lines represent upper confidence bounds based on bias and variances. In the first figure we can see the bounds for each independent fidelity, in the second figure both bounds are combined to obtain the final acquisition function. 35

3.3 An example of the local penalisation process. We selected $x = 0.378$ as a point to evaluate. The acquisition function is then penalised as shown above until we obtain a new observation. This allows us to select new queries without receiving an observation, as the penalised acquisition function is guaranteed to have a different maximum. The blue and red dots represent low and high fidelity observations, respectively. 41

3.4 Results of synthetic experiments. We see that simply using UCB with our algorithm consistently achieves good results, always being one of the top performs. TuRBO also tends to perform fairly well, especially strongly in Hartmann6D, however, it also struggles badly in some benchmarks (Park4D and Borehole8D). MF-MES is not present in the Hartmann experiments due to the computational expense of testing the function when there are three fidelities. 53

3.5 Results on Ackley 40D function. We can see that TuRBO is constantly improving, and outperforms other functions considerably. MF-MES and UCB-V-LP have good initial performance but begin to struggle after due to the over-exploration of classical Bayesian optimisation. 56

3.6 Results on battery design experiment. We can see that the asynchronous batching and multi-fidelity methods are significantly outperforming simple baselines (UCB), multi-fidelity without batching (MF-GP-UCB), and single-fidelity methods (PLAyBOOK). The performance of all batching multi-fidelity methods is similarly strong. 57

3.7 Experimental querying frequency comparison. The histogram above shows how often we queried the low and high fidelities for different time-intervals. We compare the querying frequency for the methods using UCB and Local Penalisation, with different ways of choosing the fidelity – variance-based (left) against information-based (right). The information-based fidelity criteria chooses to query the lower-fidelity more frequently, which suggests it could be the reason why it slightly outperforms the variance-based criteria. 58

4.1 Motivating example. Droplets flow into the micro-reactor where we control conditions such as temperature and flow rate. The cost to change function inputs arises from how adjacent droplets are coupled, e.g. rapidly changing the temperature after droplet K means waiting for system equilibration before taking new measurements. Asynchronicity arises from choosing drops $2, 3, ..., N$ before getting the results of droplet 1. 63

4.2 Effect of Point Deletion for ordering-based Bayesian optimisation. (a) and (b) show the maximization objective and the underlying surrogate model, with black crosses representing queries and observations. (c) shows the optimisation paths. Naive resampling gets stuck in the local optimum ($x = 0.16$) whereas Point Deletion escapes the local solution. We also show the predicted escape iteration, Tp (after estimating $p \approx 0.74$ in Figure 4.4), which accurately predicts the algorithm behavior in this simple example. 70

4.3 We estimate the probability of non-escape by taking 5000 independent Thompson Samples and counting the number of samples inside A (i.e. the MLE estimator of the Bernoulli distribution). We do this for increasing number of training points in A (which are chosen randomly with a uniform distribution in A). We repeat the experiment 10 times. The left plot shows the underlying function and the Gaussian Process for 15 training points. The right plot shows the evolution of our estimate as we increase training points inside $A = [0, 0.1]$ (we plot the mean of each run $\pm$ the standard deviation). We can see that p_t quickly converges to (almost) zero. We almost guaranteed to fully escape the area after 15 time-steps, even for very large budgets. 72

4.4 We estimate the probability of non-escape by taking 5000 independent Thompson Samples and counting the number of samples inside A (i.e. the MLE estimator of the Bernoulli distribution). We do this for increasing number of training points in A (which are chosen randomly with a uniform distribution in A). We repeat the experiment 10 times. The left plot shows the underlying function and the Gaussian Process for 15 training points. The right plot shows the evolution of our estimate as we increase training points inside $A = [0.1, 0.2]$ (we plot the mean of each run $\pm$ the standard deviation). This example makes it clear that p_t does not converge to zero. Furthermore, it seems to converge to just over 0.7 which a very large probability. This will make fully escaping the local minimum very difficult without Point Deletion. 73

4.5 Graphical example of SnAKe behavior in a full iteration if new information is available. The underlying function is Branin 2D. The feasible set is $[0, 1]^2$, so we do not see all the samples or the complete path with this zoomed-in view. (a) The red line shows the already-queried path. The blue path shows our future plans. (b) For each query, we can see the ϵ-ball under which the deletion step is deterministic. We plot the Thompson Samples as dots, the accepted ones in black and the deleted ones in green. (c) The new path is in blue and the points ignored (due to Point Deletion) are in green. Note the higher concentration of samples in what the model considers a promising area. 77

4.6 Synthetic experimental results. The first row represents three of the synchronous benchmarks. The last two rows correspond to asynchronous experiments with $t_{delay} = 25$. We plot the average log(regret) achieved against the cost spent. For every experiment, $T = 250$, and we limit the x-axis to the maximum cost achieved by SnAKe or Random. ℓ-SnAKe is the parameter-free version. SnAKe consistently achieves good regret against at cost. The bounds are $\pm$ half the standard deviation of 25 different runs. 82

4.7 Hartmann6D. Here we showcase the effect of increasing the budget size, showing SnAKe performs increasingly better as the budget grows. Each row represents a different budget. The left column shows the evolution of regret against the cost used. The middle column shows the evolution of regret with iterations, and the right columns show the evolution of the 2-norm cost. A high-dimensional example where SnAKe performs exceedingly well, giving the best regret at low costs for all budgets except $T = 15$. The final cost is considerably lower for SnAKe than any other method. 83

4.8 Perm10D. Here we showcase how SnAKe can fail to optimise well in higher dimensions. Each row represents a different budget. The left column shows the evolution of regret against the cost used. The middle column shows the evolution of regret with iterations, and the right columns show the evolution of the 2-norm cost. SnAKe struggles in this benchmark, however, EI also struggles. As an interesting observation, if we *did not* update the model, we would achieve a much better performance (as it would be equivalent to Random). EIpu and TrEI perform well in this example, we conjecture this is because they are doing far more localised searches while SnAKe is trying to cover all the space available (which is very difficult in higher dimensions). 84

4.9 Results of experiments on the SnAr chemistry Benchmark. SnAKe achieves the best regret out of all low-cost methods. The bounds are created from ± half the standard deviation of all runs. The best performer, ℓ-SnAKe, is parameter-free. 85

4.10 Visualization of Ypacarai Lake, and the objectives we optimise over. We show an example of a SnAKe optimisation path on the simultaneous optimisation problem (see Section 4.8.4). 86

4.11 Self-stopping optimisation of Hartmann 6D with different maximum budgets. Vanilla EIpu stops too early due to model misspecification; our modification that includes uncertainty criteria greatly improves performance. The average budget used (a.b.) is robust to maximum budget choice. 89

4.12 Results for truncated Branin2D with a budget of 250 queries. The cost of non-truncated methods is very high due to penalization for breaking movement constraints. TrSnAKe achieves the best performance for low cost. 90

5.1 Representative task: finding pollution in a river while following the current. (a) Problem formulation: The star represents the maximiser and the arrows the Markov dynamics. (b) Objective formulation: Orange balls represent potential maximisers, with size corresponding to model uncertainty. (c) Optimisation: Deploy a potentially stochastic policy that minimises our objective. 95

5.2 Visual abstract of the work. In black we show the method presented in this paper, with literature connections shown in blue. In red we show solutions which we did not pursue due to intractability. The problem creates the **(a) need to plan ahead**. To do this, we take inspiration from hypothesis testing and focus on **(b) the variance reduction in a set of maximisers**, which leads to our **(c) acquisition function**. The objective is the same as Fiez et al. [2019] introduced in the linear bandits literature from a frequentist perspective. To optimise it, we follow developments in Hazan et al. [2019], Mutný et al. [2023] by **(d) relaxing the acquisition function to the space of state-action distributions** and **(e) solving the planning problem using the Frank-Wolfe algorithm**. This consists of iteratively solving tractable **(f) reinforcement learning sub-problems** which give us optimal Markov policies. We then apply adaptive resampling to obtain **(g) non-Markovian policies**. 98

5.3 We can see the state space as black dots, the current state as a black star and the next state as a green star. In a transient flow reactor we can preload the reactor with different equivalents, resulting in differing sets of measurements when doing residence time ramps. The blue and red paths represent two different ways of moving from the black star to the green star, and the resulting measurements are displayed as crosses. For this action space we choose, not only the next state (green star) but which path to take there (blue or red path). 103

5.4 The Knorr pyrazole synthesis experiment. On the left, we show the quantitative results. The line plots denote the best prediction regret, while the bar charts denote the percentage of runs that correctly identify the best arm at the end of each episode. On the right, we show ten paths in different colours chosen by the algorithm. The underlying black-box function is shown as the contours, and we can see the discretisation as dots. We can see four remaining potential maximisers (in orange), which includes the true one (star). *Notice all paths are non-decreasing in residence time, following the transition constraints.* 114

5.5 Lake Ypacari with the added movement constraints. We show one local optimum and one global one. The constraints of the problem requiring beginning and ending the optimisation in the dark square. 115

5.6 Results for Ypacarai and free electron-laser tuning experiments. On the left, the line plots denote the best prediction regret, while the bar charts denote the percentage of runs that correctly identify the best arm at the end of each episode. On the right, We plot the regret and compare against standard BO without accounting for movement-dependent noise. 116

5.7 Results of experiments on the asynchronous and synchronous benchmarks. We plot the median predictive regret and the 10% and 90% quantiles. For the asynchronous experiments, we can see that the paths taken by MDP-BO-TS are more consistent, and the final performance is comparable to TrSnAKe. While in the asynchronous setting, we found creating the maximisation set using Thompson Sampling gave a stronger performance, in the synchronous setting, UCB is preferred. LSR gives a very strong performance, comparable to MDP-BO-UCB in almost all benchmarks. 117

5.8 Ablation study on the number of mixture components on the Knorr pyrazole synthesis benchmark 118

5.9 Ablation study on the number of mixture components on the Ypacarai benchmark118

5.10 Ablation study into the size of the Thompson Sampling maximisation set in the asynchronous Hartmann3D function. We can see that the performance of the algorithm is very similar for all values of $K = 25, 50, 100$. 119

5.11 Ablation study into the size of the UCB maximisation set in a variety of benchmarks. We can see that the performance of the algorithm is very similar for all values of $K = 10, 25, 100$. 119

5.12 Example paths planned for data-collection in a transient flow reactor, the experiments were then carried out in a laboratory at SOLVE Chemistry. 121

B.1 Comparing the numerical solution against the solutions found in equation (B.12). 174

Chapter 1

Introduction

1.1 Motivation and Objectives

The concept of finding the optimal conditions of a system is something every person faces daily. *What is the fastest route to work? How can I best prepare a meal? What is the most efficient way of getting my daily tasks done?* Humans have developed an intuition for these kind of optimisation problems, and are very capable of learning through experience. *I took that route last time and the traffic was impossible, I will take a detour instead. This pasta with that sauce was very tasty, I'll repeat the recipe and add some onions. I am more efficient at work if I exercise in the mornings.* Every single one of us is constantly solving optimisation problems. We do not optimise in the sense of Newton or Fermat – we do not have access to gradients. Instead we have developed an intuition for optimising through interactions and subsequent observations of our environment, an efficient trial-and-error strategy.

The mathematical area of black-box optimisation seeks to formalise such problems. Assume we have a system that we can interact with, and a quantity of interest represented by a *function* f. Example of such functions include how much time it will take us to get home from work, the flavour of a cookie recipe, or the time at which we finish all our daily tasks. For each interaction, we can modify the system according to some input space $\mathcal{X}$ which corresponds to actions we can choose: what route to take in traffic, what ratio of the recipe's ingredients will

we use, in what order do we go about our daily tasks. Whenever we choose an input $x \in \mathcal{X}$ we obtain a noisy observation of the environment:

$$y = f(x) + \epsilon$$

where ϵ represents the random effects out of our control, for example, a traffic accident. We are able to interact with the environment many times and therefore build experience on what the black-box function f looks like, storing all our experience into a data-set $D_t = \{(x_0, y_0), (x_1, y_1)..., (x_{t-1}, y_{t-1})\}$. We can use this data-set to inform future decisions when interacting with the environment. Our final goal is to find the optimal conditions of the function, x_*, which, without loss of generality, is simply a maximization problem:

$$x_* = \arg\max_{x \in \mathcal{X}} f(x). \tag{1.1}$$

Finding the optimum can be done simply via randomly obtaining so much information that we know the location of x_* with certainty. However, it is typically the case that querying the black-box function is *expensive* with respect to time, money or other resources. This is an important problem which arises, not only in our every-day lives, but in almost all sciences.

In computer science, we might be interested in tuning the parameters of a machine learning model which requires many days of heavy compute to train. In chemistry, it is common to want to find the optimal conditions of a reaction whose kinetic dynamics are poorly understood, this is done in a lab via physically carrying our the reactions and observing the results. In aerodynamics, we may want to minimise the wind resistance of a car's chassis which requires expensive testing in a wind tunnel. In physics, we may want to optimise the final state of a dynamical system given the initial conditions, however, each simulation of the system can take hours or days.

In order to save time and money, it is necessary to develop efficient algorithms that optimise black-box functions for the smallest number of iterations as possible. Here we can use our human intuition of how we go about solving such problems in our every-day life. Consider the

example of baking cookies. We might begin with no information of what recipe we want and bake a few recipes at random. After this we will begin baking good recipes that are similar to our favourite one so far, i.e., we *exploit* the knowledge we have. However, every now and then we might bake something completely different from anything we have done before, just out of curiosity to see what it will taste like, i.e., we *explore* to gain new knowledge. This leads to potentially finding a very good or very bad recipe, however, it is still an important part of the process. Indeed the random recipes we bake at the begging correspond to an explorative step. This trade-off between exploration and exploitation is what we as humans have learned to balance through experience. Note that there are two important factors to balance such trade-offs: (i) we are able to predict how the underlying unknown function f looks like, but also (ii) we are also able to quantify our uncertainty about certain inputs – *we know how much we don't know*.

Bayesian optimisation is an area of research following the simple ideas just introduced but in a rigorous mathematical manner. By using a Bayesian model of the unknown function f we are able to make predictions while also quantifying the uncertainty in our predictions. Therefore, we are able to come-up with strategies that balance the *exploitation-exploration trade-off* and efficiently optimise black-box functions through a small number of observations. The interest in Bayesian optimisation emerged early on from choosing the best treatment in clinical trials [Thompson, 1933], and grew when looking at global optimisation techniques [Mockus, 1974] in the 1970s. Recently, however, we have observed a large surge of interest due to the need to tune the hyper-parameter of expensive-to-train machine learning models (as well as targeted advertising in online platforms). This has led to a myriad of new efficient methods for global optimisation of black-box functions.

Inspired by this surge, the physical sciences have emerged as a very promising area for the application of Bayesian optimisation techniques [Mowbray et al., 2021, Thebelt et al., 2022c, Van De Berg et al., 2022]. However, each new application comes with new problem structures which may be incompatible with current state-of-the-art techniques. Indeed the vanilla sequential black-box optimisation problem which we just described represents a massive over-simplification of most real-world systems. To give a few examples of these:

(I) In many cases we are able to estimate the expensive black-box function at lower cost. This can be through interaction with other environments whose observations give us a possibly noisier or biased approximation of the true optimisation objective. We humans do this often, for example, by having a small taste of the food before it is finished cooking and adjusting accordingly. A second example arises in chemistry where we can carry out tests on a reaction at low residence time to estimate the effects when the residence time is high.

(II) We may be able to interact with the environment simultaneously to obtain many parallel observations at the same cost. Furthermore, it might be the case that measurements return at uneven time-frames and we might need to make decisions while other experiments are running. For example, when baking cookies we can make each of them a different recipe and obtain results on many observations. Similarly, we may be able to test various reactions in the lab by using reactors in parallel.

(III) There may be a *switching cost* associated with subsequent experiments. For example consider environmental monitoring; assume that we have an automated vehicle looking for a source of maximal pollution, indeed to take a measurement we need to wait for the vehicle to reach the requested location. Experiments close to each other will incur a much smaller cost than experiments that are further away.

(IV) Sometimes switching costs may not be enough, and we might have strong constraints on the changes we are allowed to make. For example, an automated vehicle *must return to vehicle bay* in order to avoid running out of power in the middle of the environment. Another example are flow reactors, where small changes are needed to maintain the steady-state of the reaction and therefore obtain accurate measurements.

A common theme to all the above problems is the need to ***plan*** in order to have efficient optimisation. Indeed we must consider *how will my current actions affect which observations I will obtain in the future?* In this thesis, we investigate all the problems described above and propose Bayesian optimisation solutions as to efficiently optimise systems which fall into those categories.

1.2 Contributions

In this thesis we present a comprehensive review of Bayesian optimisation and how it has developed with respect to physical science applications. In particular, in Section 2 we introduce the building blocks of Bayesian optimisation, Gaussian Processes. We then formally introduce Bayesian optimisation's fundamental concepts and give a thorough description of how it can be used to optimise black-box functions. We further describe how it can be described as a myopic approximation with respect to a decision making problem, and how many authors have attempted to tackle this myopia – and why current methods fall short in many applications.

In Section 3 we describe a novel algorithm that allows the combination of multi-fidelity methods and asynchronous batch Bayesian optimisation in general settings. Motivated by efficient battery design, we showcase how the existing literature can fit together into a two-step process: first all points are chosen using standard batch Bayesian optimisation and then multi-fidelity methods are applied to choose whether to evaluate the expensive black-box function or whether to obtain a cheap approximation instead. This extends previous literature as the acquisition method for the points is fully general, and therefore specialist acquisition functions can be used. We showcase this by giving an example in high-dimensional optimisation, and also include general results that give empirical evidence of the benefits of our approach.

In Section 4, we introduce the problem of Bayesian optimisation under switching or movement costs. This problem has seen little research and we provide the first algorithm that tackles the problem in generality for any type of cost. Our approach, dubbed SnAKe, is shown to perform as strong as traditional Bayesian optimisation methods while keeping movement costs much smaller. We motivate the importance of the work by looking at applications in chemical reactor optimisation and environmental monitoring. Finally, we present extensions of the algorithm to many practical setting, including multi-objective optimisation and the addition of strict local constraints.

In Section 5, we consider the problem of Bayesian optimisation under transition constraints. Such problems arise in many settings, but we motivate it primarily by looking at optimisation

in flow reactors where the experiments are carried out in episodes and certain variables must be monotonically increasing in the optimisation process. By considering advances in exploration in Markov chains and linear bandit optimisation, we develop the first algorithm that tackles maximizer identification with general transition constraints in discrete search spaces, and show competitive performance in continuous spaces under box constraints. We give examples of three real-world applications: flow reactor optimisation, environmental monitoring, and the tuning of an electron laser.

Finally in Section 6 we provide a discussion and conclusion of the work, and present the possible impact of the work, and directions of future research.

1.3 Statement of Originality

All the ideas presented in this thesis are my own work, unless stated otherwise. All concepts which are not original are properly cited and credited.

1.4 Publications

The work presented in this thesis has results in three publications, and one further paper under review. These are:

- Folch et al. [2023a] which was published in the Journal of Computers & Chemical Engineering in 2023, where it was given the Best Paper Award for 2023. This paper formed the basis for Chapter 3.

- Folch et al. [2022] which was published in Neural Information Processing Systems in 2022. This paper formed the basis for Chapter 4.

- Folch et al. [2023b] which was published in the NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World in 2023. This short paper formed part of Chapter 4.

- Folch et al. [2024] which has been accepted for publication in Neural Information Processing Systems for 2024. This paper formed the basis for Chapter 5.

A further work, Stoddart et al. [2023] was published in the NeurIPS 2023 Workshop: Tackling Climate Change with Machine Learning, which the author of this thesis supervised.

Chapter 2

Background Theory

2.1 Introduction

We will now review the necessary technical background that will form the basis of the thesis. In Section 2.2 we take a look at Gaussian Processes which are the basic model which we will use to build our predictions and quantify uncertainty. We then introduce Bayesian optimisation formally in Section 2.3 and describe vital concepts such as acquisition functions and batch Bayesian optimisation. In Section 2.4, we explain how Bayesian optimisation can be reformulated as a myopic approximation of a difficult sequential decision making problem. We further describe the non-myopic problem in detail and explain why tackling the problem directly is computationally infeasible.

2.2 Gaussian Processes

Gaussian Processes (GP) [Rasmussen and Williams, 2005] are an stochastic processes such that every finite dimensional distribution of the process is Gaussian. In other words, given an index set $\mathcal{X}$, then a stochastic process $f(x)$ is said to be a Gaussian process, $\mathcal{GP}$, if for any

$x_1, x_2, ..., x_n \in \mathcal{X}$ then:

$$(f(x_1), ..., f(x_n)) \sim \text{MultivariateNormal}(m, \mathbf{K})$$

where m is a mean vector and $\mathbf{K}$ a covariance matrix. In fact, for an arbitrary mean function $m_f : \mathcal{X} \to \mathbb{R}$ and given any positive definite function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, we can build a Gaussian process that follows a mean defined by $m_i = m_f(x_i)$ and a covariance structure defined by $\mathbf{K}_{ij} = k(x_i, x_j)$. The covariance function k is usually known as the *kernel* function and it encodes important attributes of the underlying stochastic process.

Using GPs we are able to model unknown functions via taking a Bayesian perspective and assuming it is a realisation of a Gaussian processes. In other words, we assume that $f \sim \mathcal{GP}$ as a prior. We then assume our observations of the underlying function are noisy realisations of the process:

$$y = f(x) + \epsilon$$

where ϵ follows some noise distribution. Throughout this thesis we follow the most simple and common assumption that the noise is zero-mean Gaussian with variance σ^2. Given data $\mathcal{D} = \{(x_1, y_1), ..., (x_n, y_n)\}$ we can then obtain the posterior of the process through Bayes' Theorem:

$$p(f(X)|Y, X) = \frac{p(Y|f(X), X)p(f(X)|X)}{p(Y|X)}.$$

Note that we know the from our assumptions both $p(Y|f(X), X)$ and $p(f(X)|X)$ follow a Gaussian distribution and the posterior can be found to follow a Gaussian distribution as well. This holds more generally, and it can be shown that the posterior stochastic processes is also a Gaussian processes with mean and covariance functions given by:

$$\mu_{\mathcal{D}}(x) = m_0(x) + k(x)^T(\mathbf{K}_{XX} + \sigma^2 I)^{-1}(y - m_0(X)) \tag{2.1}$$

$$k^2_{\mathcal{D}}(x, x') = k(x, x') - k(x)^T(\mathbf{K}_{XX} + \sigma^2 I)^{-1}k(x') \tag{2.2}$$

where $m_0(\cdot)$ and k represents the prior mean function and the kernel, $k(x)_i = k(x, x_i)$, and $\mathbf{K}_{ij} = k(x_i, x_j)$. While the computations are analytical, note that they come at the cost of an

$N \times N$ matrix inversion which leads to a $\mathcal{O}(N^3)$ computational cost.

2.2.1 Kernel functions

Perhaps the most important feature of a Gaussian Process is the kernel. It defines important properties of the underlying random functions, such as smoothness, differentiability and variability. The most commonly used kernel is the radial basis function or squared exponential kernel, defined as:

$$k_{RBF}(x, x') = \exp\left(-\frac{||x - x'||^2}{2\ell^2}\right)$$

where the parameter ℓ is known as the *lengthscale* of the process. This is an important hyper-parameter that defines how close two points have to be (in the L^2 sense) to be correlated. This notion can be extended so that variability is independent for each input dimension [Williams and Rasmussen, 1995]:

$$k_{ARD}(x, x') = \exp\left(-\frac{1}{2}(x - x')^T \Lambda^{-2}(x - x')\right) \tag{2.3}$$

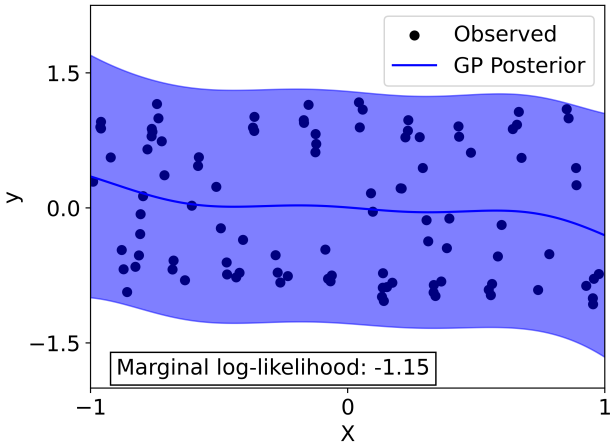
where $\Lambda = \text{diagonal}((\ell_1, ..., \ell_d))$. Each individual length-scale ℓ_i represents the variability of the function in the ith dimension. An important attribute of kernels is that we can build new kernels via multiplication or addition. Indeed, the ARD kernel can be obtained as the product of dimension-independent standard one-dimensional RBF kernels:

$$k_{ARD}(x, x') = \prod_{i=1}^{d} \exp\left(-\frac{(x_i - x_i')^2}{2\ell_i^2}\right) = \prod_{i=1}^{d} k_{RBF}(x_i, x_i'|\ell_i).$$

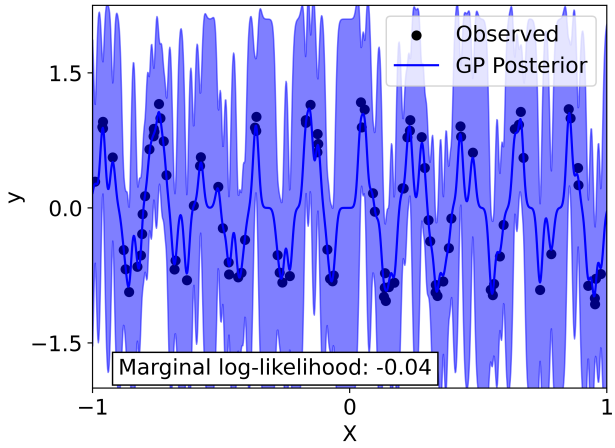
We will later show how this can be useful to build new kernels for models that have multiple information sources (Section 3) and to incorporate prior knowledge into the Gaussian process (Section 5). It also means that we can scale the kernel, e.g., $k(x, x') = \nu^2 k_{RBF(x,x')}$. Here ν is commonly known as the *outputscale* hyper-parameter of the kernel and it defines the variance of the prior GP functions. It is also common to think of the noise level, σ, as a model hyper-parameter as it is commonly unknown.

2.2.2 Kernel hyper-parameters and the marginal-likelihood

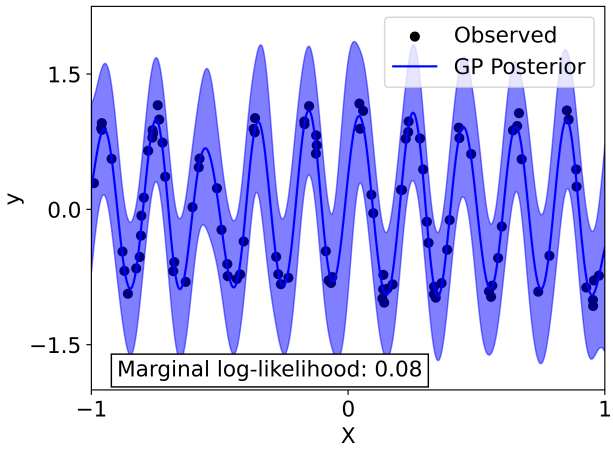
We have introduced kernels and their corresponding hyper-parameters, however, it is very important to choose them correctly. Indeed, consider two cases of fitting a model as shown in Figure 2.1. In the first case, Figure 2.1a we fit the model using a large lengthscale and noise and it can be seen that the model simply predicts everything to be a realisation of an almost constant but very noisy process. In the second case, Figure 2.1c, we instead fit using more reasonable (and much smaller) hyper-parameters for the noise and lengthscale – it is clear that the model fit is much better.



(a) Gaussian process fit with large noise and lengthscale hyper-parameters.



(b) Gaussian process fit with very small lengthscale.



(c) Gaussian process fit with small noise and lengthscale hyper-parameters.

Figure 2.1: Two GPs fit on the same data, with different hyper-parameters. We show the data as black dots, and the model's mean and confidence intervals in blue, as well as displaying the corresponding marginal log-likelihood of each model.

Therefore it becomes vital to choose the right hyper-parameters for the model to be useful. A simple way of choosing the hyper-parameters could be prior knowledge about the underlying function, however, it is possible that no prior knowledge is available, or at least it may be concrete enough to give us good hyper-parameter estimates. Luckily, Bayesian model selection gives us a way of automatically choosing (or *learning*) the hyper-parameter values.

Given a model parameterised by $\theta \in \Theta$, the *marginal-likelihood*, also known as the Bayesian evidence, is defined as the probability that the data is randomly generated by the model's prior distribution:

$$p(y|X,\theta) = \int p(y|f,X,\theta)p(f|X,\theta)df.$$

It can be written in closed form due to the properties of Gaussian distributions, and it is more common to use the marginal log-likelihood since it is more stable to optimise:

$$\log p(y|X,\theta) = -\frac{1}{2}y^T(\mathbf{K}_{XX} + \sigma^2 I)y - \frac{1}{2}\log|\mathbf{K}_{XX} + \sigma^2 I| - \frac{n}{2}\log 2\pi. \tag{2.4}$$

This quantity is very useful as it allows us to differentiate models without over-fitting. Indeed if we chose the hyper-parameters that maximised the likelihood we would be highly prone to over-fitting.

Instead, the marginal likelihood will have a preference towards choosing the simplest model which is consistent with the data. To see this, consider that simpler models are able to generate "less" data-sets, therefore if two models fit a data-set, the simpler one must assign higher prior probability to it. It is said that the marginal-likelihood encodes Occam's Razor in this sense. The marginal likelihood has also been shown to be equivalent to doing exhaustive cross-validation [Fong and Holmes, 2020]. See again Figure 2.1 showcasing three different models. Figure 2.1a shows the simplest model, however, it does not fit the data well. Figure 2.1b shows a model that fits the data, however, it has a lengthscale that is too small making the model far too complex. Figure 2.1c shows the model with the correct hyper-parameters. We can see that the best model has the largest marginal log-likelihood. This gives us a method for learning the

hyper-parameters simply by maximizing the marginal likelihood:

$$\hat{\theta} = \arg\max_{\theta \in \Theta} \log p(y|X, \theta). \tag{2.5}$$

The optimisation is non-convex and difficult to solve exactly, however, it can be easily approximated via gradient methods with multi-starts. It is important to note that there are some drawbacks to using the marginal-likelihood for choosing the hyper-parameters. In particular, calculating the log-determinant in Eq.(2.4) is of order $\mathcal{O}(N^3)$, and in addition the marginal log-likelihood can still be prone to over-fitting in many cases [Lotfi et al., 2022]. These disadvantage can be mitigated, for example, by using sparse approximations to GPs [Snelson and Ghahramani, 2005, Van der Wilk, 2019] or by setting reasonable priors on the hyper-parameters.

2.2.3 Finite dimensional approximations and sampling

Gaussian processes define a distribution over functions. Across this thesis we will often be interested in creating sample functions from this distribution, however, doing this is non-trivial. Given a specific set of target inputs, x_*, it is easy enough as the distribution of $f(x_*|\mathcal{D})$ is Gaussian and we can use standard techniques. However, assuming we want to solve optimisation problems in a sample function, for example, maximizing it: $\max_{x \in \mathcal{X}} \tilde{f}(x)$ such that $\tilde{f} \sim f|\mathcal{D}$; then we need a way of sampling the function itself. While this cannot be done exactly, we can use properties of kernel methods to obtain approximate samples.

This follows from the use of Mercer's Theorem [König, 2013] which states that under some regularity assumptions a kernel k can be decomposed as:

$$k(x, x') = \sum_{i=1}^{\infty} \lambda_i \phi_i(x) \phi_i(x')$$

where λ_i and ϕ_i are eigenvalues and eigenfunctions of the kernel operator $T_k(f)(x) = \int_{\mathcal{X}} k(x, x') f(x') dx'$.

In particular, this means we can rewrite the Gaussian process as:

$$f(x) = \sum_{i=1}^{\infty} \frac{1}{\sqrt{\lambda_i}} \phi_i(x) \omega_i \qquad \omega_i \sim \mathcal{N}(0, 1) \tag{2.6}$$

and we can build a *finite-dimensional approximation* by simply truncating the sum at term M. This has been explored in the context of reducing the computational cost of GPs by considering an M-rank approximation to the kernel matrix.

The decomposition is non necessarily unique, and there are many methods to create the eigenfunctions, such as using random Fourier features [Rahimi and Recht, 2007, Mutný and Krause, 2018] or Nyström features [Williams and Seeger, 2000]. Recently, it has been proposed to use separate methods for prior sampling and posterior updating which leads to more efficient sampling [Wilson et al., 2020]. Once the features have been chosen, we can create function samples via the weight vector ω whose posterior is Gaussian and well known from Bayesian linear regression.

2.3 Bayesian Optimisation

As discussed in the introduction, our goal is to optimise expensive black-box functions. In recent years Bayesian Optimisation [Jones et al., 1998, Shahriari et al., 2016] has emerged as one of the leading solutions for such problems. The methods go through the sequential querying of the black-box function to obtain a sequence of experiments and noisy observations $\mathcal{D}_t = (x_1, y_1), ..., (x_t, y_t)$ where the sequence is carefully chosen as to optimise the function. Generally the metric that we are trying to optimise is simple regret, defined as:

$$SR_t = f(x^*) - \arg\max_{i=1,...,t} f(x_i) \tag{2.7}$$

which measures how close our best observation is to the true maximum. While this metric is not generally observable in real-world experiments (due to experimental noise and lack of knowledge of the optimal value), it can be used to benchmark algorithms in synthetic experiments, as well

as being the subject of many theoretical results.

Queries are chosen through the use of an *acquisition function*, α, which assign a measure of experimental value to each possible experiment. These functions tend to be easy and cheap to optimise, and will define an inner optimisation loop in the Bayesian optimisation process:

$$x_{t+1} = \arg\max_{x \in \mathcal{X}} \alpha(x|\mathcal{D}) \tag{2.8}$$

where the dependence of the acquisition function on the Gaussian Process is implied by the dependence on the data-set. From this basic idea we obtain the Bayesian optimisation loop, which is described in generality in Algorithm 1.

Algorithm 1 General Bayesian Optimisation Algorithm

INPUT: Acquisition function choice, α; budget, T; initial data-set $\mathcal{D}_0$
for $t = 1, 2, 3, ..., T$ **do**
  $x_t = \arg\max_{x \in \mathcal{X}} \alpha(x|\mathcal{D}_{t-1})$ *// Choose next experiment*
  $y_t = x_t + \epsilon_t$ *// Obtain noisy observation*
  $D_t = D_{t-1} \cup (x_t, y_t)$ *// Update data-set*
  Update Gaussian Process posterior

2.3.1 Acquisition functions

Perhaps the most important aspect of a successful Bayesian optimisation implementation is to make sure the choice of acquisition function is appropriate. Many have been developed and they continue to be the subject of ongoing research. In this section we will describe a few of the most important ones.

Expected Improvement

One of the simplest, yet most widely used, acquisition functions is expected improvement (EI) [Mockus, 1974, Jones et al., 1998]. It is based on the simple idea that we want to choose the experiment which will improve out current best observation the most *in expectation*. That is:

$$\alpha_{EI} = \mathbb{E}_{f(x)} \left[\max(f(x) - y_{best}, 0)\right] \tag{2.9}$$

where is the best observation we have seen so far, $y_{best} = \max_{y \in \mathcal{D}_t} y$. In particular, due to the properties of Gaussian processes, this can easily be shown to have the analytical expression:

$$\alpha_{EI} = (\mu_t(x) - y_{best})\Phi\left(\frac{\mu_t(x) - y_{best}}{\sigma_t(x)}\right) + \sigma_t(x)\phi\left(\frac{\mu_t(x) - y_{best}}{\sigma_t(x)}\right)$$

where $\mu_t(x)$ and $\sigma_t(x)$ represent the mean and variance functions of the posterior Gaussian process, and Φ and ϕ are the cumulative density and probability density functions of a standard Gaussian.

Probability of Improvement

Instead of considering the best improvement in expectation, we can simply consider which query $x \in \mathcal{X}$ has the largest probability of improving on our best observation. This leads to the probability of improvement (PI) [Kushner, 1964] acquisition function:

$$\alpha_{PI}(x) = \mathcal{P}_{f(x)}\left(f(x) > y_{best}\right) \tag{2.10}$$

which again can be easily calculated analytically:

$$\alpha_{PI}(x) = 1 - \Phi\left(\frac{\mu_t(x) - y_{best}}{\sigma_t(x)}\right).$$

While intuitive, note that this time we do not consider the magnitude of the improvement. In addition, this acquisition function is known to be a little too exploitative.

Upper Confidence Bound

Taking inspiration from multi-armed bandits [Lai and Robbins, 1985], we can instead choose to be optimistic about our surrogate model. Indeed we can build confidence intervals around each value $f(x) \forall x \in \mathcal{X}$ and choose the experiment with the largest possible value. This is achieved by the Upper Confidence Bound [Srinivas et al., 2010] acquisition function:

$$\alpha_{UCB}(x) = \mu_t(x) + \beta\sigma_t(x) \tag{2.11}$$

where β is a hyper-parameter which controls how optimistic we want to be about the function.

Due to the underlying Gaussian distribution this can be chosen according to the Bayesian credible intervals, e.g., a 95% interval corresponds to $\beta = 1.96$.

Thompson Sampling

Thompson Sampling [Thompson, 1933] introduces the idea of sampling-based optimisation, whose theoretical and empirical performance has been investigated in broader contexts such as multi-armed bandits [Scott, 2010, May et al., 2012] and more recently Bayesian optimisation [Russo and Van Roy, 2014, Kandasamy et al., 2018, Grant and Leslie, 2020]. Due to the randomness of the underlying Gaussian process, we also obtain a plethora of induced distributions on properties of the function itself. For example, since the function follows a distribution, then the derivatives also follow a stochastic process. Most importantly, the location of the maximum $x_* = \arg\max_{x \in \mathcal{X}} f(x)$ also has an induced distribution. Unfortunately the distribution is incredibly complex and even estimating it can be difficult [Hennig and Schuler, 2012].

However, it is relatively easy to *sample* the distribution. Indeed by creating a sample of the Gaussian process and then maximizing it, we obtain a sample of the true distribution of the maximum, $\hat{x}_* \sim p(x_*)$. Alternatively, we can think of the selection process where we optimise a random acquisition function where:

$$\alpha_{TS}(x) = \hat{f}(x) \qquad \hat{f} \sim f|\mathcal{D}. \tag{2.12}$$

The sampling can be done exactly over a discrete space of target inputs, or alternatively, we can create continuous samples via the finite-dimensional approximations described in Section 2.2.3.

Entropy Search

Finally, we can instead take an information theoretic view of the problem. Indeed, as discussed for Thompson Sampling, the Gaussian process induces a distribution on the location of the maximum, x_*. An intuitive avenue for optimisation is to try to reduce the uncertainty in x_*. The notion of the uncertainty for a random variable is formalised in information theory under the guise of *entropy*. The entropy search acquisition function [Villemonteix et al., 2009] chooses

the experiment that will lead to the maximum expected reduction in entropy:

$$\alpha_{ES} = H(x_*|\mathcal{D}) - \mathbb{E}_{y|\mathcal{D}}\left[H(x_*|\mathcal{D} \cup \{(x,y)\}\right] \tag{2.13}$$

where H denotes the differential entropy of the posterior distribution of x_*:

$$H(x_*|\mathcal{D}) = \int p(x_*|\mathcal{D}) \log p(x_*|\mathcal{D}) dx_*.$$

This acquisition function, while theoretically grounded, is not tractable to compute and Monte Carlo approximations have to be made. Predictive Entropy Search [Hennig and Schuler, 2012] shows how to do this efficiently using Thompson Sampling. Other advances include focusing on the information gain of the value of the optimum, $f(x_*)$, instead of on the location [Wang and Jegelka, 2017], to reduce the dimensionality of integrals that need to be estimated.

2.3.2 Batch Bayesian Optimisation

So far we have considered the standard Bayesian optimisation formulation, where a single experiment is chosen and observed before choosing the next one. An important variant of the problem considers the case when many experiments can be run in parallel. In such cases, we are interested in choosing a *set* of experiments to be run, instead of a single query. In generality, the Bayesian optimisation solution of the problem can be formulated as for the standard case by using acquisition functions which are now a function of a set of q points lying in $\mathcal{X}$:

$$\{x_{t,k}\}_{k=1}^{q} = \underset{x_s \in \mathcal{X}^k}{\arg\max}\, \alpha(x_s|\mathcal{D}_{t-1}). \tag{2.14}$$

The resulting general algorithm is detailed in Algorithm 2. Acquisition functions in such a space can be more complicated, indeed consider the batched version of expected improvement [Ginsbourger et al., 2008]:

$$\alpha_{qEI}(x_1, ..., x_q) = \mathbb{E}\left[\max\left(\max(f(x_1), ..., f(x_q)) - y_{best}, 0\right)\right].$$

The acquisition function cannot be calculated analytically and Monte Carlo simulations are required instead. In addition, the space over which the acquisition function must be optimised grows exponentially in q. Therefore it is also common for batch Bayesian optimisation algorithm to mimic the sequential setting instead, i.e., the batch is created by sequentially adding points in q iterations. Details of such methods can be found in Chapter 3. An alternative method which will be frequently used in this thesis will be to use the inherent randomness in Thompson Sampling to create batches – the main benefit being that we need to optimise q independent functions which can be done in parallel.

Algorithm 2 Batch Bayesian Optimisation Algorithm

INPUT: Acquisition function choice, α; budget, T; initial data-set $\mathcal{D}_0$; batch size q
for $t = 1, 2, 3, ..., T$ **do**
  $\{x_{t,k}\}_{k=1}^{q} = \arg\max_{x_s \in \mathcal{X}^q} \alpha(x_s|\mathcal{D}_{t-1})$ *// Choose next set of experiments*
  $y_{t,k} = x_{t,k} + \epsilon_{t,k} \quad \forall k = 1, ..., K$ *// Obtain noisy observations*
  $D_t = D_{t-1} \cup \{(x_{t,k}, y_{t,k})\}_{k=1}^{K}$ *// Update data-set*
  Update Gaussian Process posterior

2.4 Non-myopic Bayesian Optimisation

2.4.1 The myopia of Bayesian optimisation

So far we have described Bayesian optimisation in its simplest and most usual form. However, note that when making decisions we are lacking one vital piece of information: the total optimisation budget remaining. Indeed all acquisition functions we have introduced so far are independent of the budget, however, the optimal decision can vary depending on how many observations are remaining. We give an example of such a case in Figure 2.2.

In all our applications it is important to consider how our current actions will have an effect in the future of the optimisation, therefore in this section we explore the current solutions for non-myopic Bayesian optimisation.

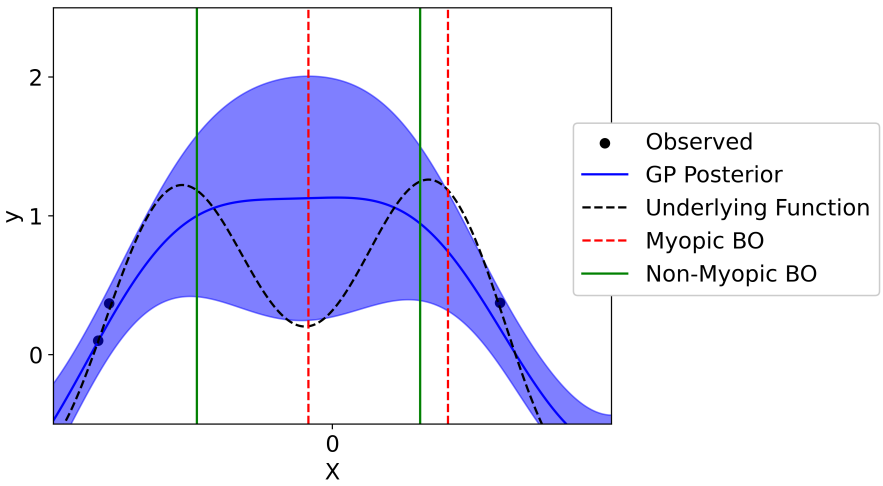


Figure 2.2: Example of myopic against non-myopic decision making with a budget of $T = 2$. We plot the queries chosen by each strategy as vertical lines. In a single step, BO will choose to maximise the expected improvement by querying a point near the middle of the search space. At the next iteration, BO will be forced to choose between using the final query in either the right part of the space or the left, leaving one unexplored. A non-myopic decision maker can recognise that taking a sub-optimal step and searching the right hand side of the search space is correct, since we have one more query left to search the left hand side, which results in a more complete search. A conveniently hand-picked, bi-modal, underlying function plotted in black helps illustrate why this is important.

2.4.2 Markov Decision Processes

In order to formulate the Bayesian optimisation problem in full generality, we will require to introduce the Markov Decision Process (MDP) framework. An MDP is defined by a state space $\mathcal{X}$, an action space $\mathcal{A}$, and the system dynamics defined by a transition operator $P(x_{t+1}|x_t, a_t)\ \forall x_{t+1}, x_t \in \mathcal{X}, a_t \in \mathcal{A}$ which defines the probability of going to state x_{t+1} given we chose action a_t in state x_t. In such processes, we can define a policy $\pi(a|x)$ as a mapping dictating the probability of taking action a in state x. We can then define decision making problems where we seek to find a policy that achieves a certain objective within the environment.

2.4.3 Full horizon problem formulation

Bayesian optimisation can be reformulated as a sequential decision making problem in an MDP where we try to optimise a reward function of a *decision policy*, π:

$$\mathcal{U}(\pi) = \sum_{i=t}^{T} U_t(\pi(D_{t-1})) \tag{2.15}$$

where π is a policy depending on all past observations. If we consider simple regret as the final objective, we simply have improvement on best observation as the instantaneous reward:

$$U_t(x_t) = \max_{x \in D_t} f(x) - \max_{x \in D_{t-1}} f(x).$$

Two things that are important to notice. First, the quantity we are looking to maximise is a random variable whose distribution is defined by the Gaussian process, even the expectation will have a highly complicated form which will require expensive numerical methods to estimate. Secondly, the state space is the space of all possible data-sets of size at most T, and the action space is the search space $\mathcal{X}$. This makes the corresponding optimisation problem, $\pi^* = \arg\max_\pi \mathbb{E}[\mathcal{U}(\pi)]$ extremely intractable due to the size of the search space.

Classical Bayesian optimisation can be rediscovered by taking a pragmatic approach to this problem, and considering a one-step approximation of the objective and optimizing simply $U_t(\pi(D_{t-1}))$ at every time-step, i.e., assume we only have one observation left and then solve the decision making problem. Indeed if we seek to optimise the expected reward under a single time-step, we recover Expected Improvement as the optimal policy. Alternatively, if we seek to optimise the worst-case reward under a single time-step, we recover Upper Confidence Bound.

2.4.4 Look-ahead Bayesian optimisation

Many attempts have been made to solve the issue of myopia in Bayesian optimisation. Ideas primarily revolve around making the look-ahead horizon larger, i.e., solving the problem under the assumption that there are H observations left instead of just one. This is because even

for small look-ahead horizons, finding the optimal policy is very computationally challenging. Indeed, Ginsbourger and Riche [2010] initially explore the problem by examining look-ahead Expected Improvement for the case $h = 2$, and conclude "dimension reduction techniques and well-tuned heuristics may be required to allow the computation of reasonable estimates for the EI of a strategy with horizon larger than 2".

Further work in look-ahead Bayesian optimisation has looked at approximating solutions through dynamic programming approaches [Bertsekas, 2012], particularly the roll-out algorithm [Lam et al., 2016]. In particular, Monte Carlo simulations are ran using a 'base policy' which allows an approximation of the optimal reward – the most common example is to use a myopic Bayesian optimisation policy for this. Yue and Kontar [2020] explore the effect of the look-ahead horizon and show that at most 2 or 3 look-ahead steps are required to get the benefit of non-myopia before the computational costs grow too large.

Of more interest to our work, González et al. [2016b] and Jiang et al. [2020a] propose using batch Bayesian optimisation to estimate look-ahead objectives. This is achieved by showing that under access to an oracle that gives all future query locations, the expectation can be cheaply calculated. While such an oracle does not exist, it can be approximated through Batch Bayesian optimisation. Such techniques have been used to look up to 15-20 steps ahead.

2.4.5 The failures of look-ahead Bayesian optimisation

For the problems considered in this thesis we will need to plan and make non-myopic Bayesian optimisation decisions. However, look-ahead Bayesian optimisation techniques currently fall short in providing solutions to our problems due to the computational expense of solving the full horizon problem formulation. For example, consider a problem where we must choose if we want to carry out a cheap experiment that takes 1 minutes, or an expensive one that takes 100 minutes – this requires knowing the effect that an action at our current time-step will have 100 time-steps into the future. Horizons that long cannot be considered by current state-of-the-art techniques.

As a second example, consider having local search constraints in our Bayesian optimisation set-up. For example, if we are optimising temperature in a reaction, assume we cannot change it by more than 0.5°C per experiment – if we are optimising in a search range from room temperature to 200°C then we may need to plan hundreds of steps into the future to avoid local optima. Paulson et al. [2022] investigated using look-ahead Bayesian optimisation for locally constrained problems, however, the method was only applied for horizons smaller than 5 due to the computational expense. For very restrictive applications, such as those we will consider in Chapters 4 and 5, we will be need to come up with different solutions which will leverage classical optimisation problems, batch Bayesian optimisation, and eventually a return to Markov decision processes.

Chapter 3

Combining Multi-fidelity Modelling and Asynchronous Batch Optimisation

This chapter is associated with the publication Folch et al. [2023a].

3.1 Preliminaries

The optimal design of many engineering processes can be subject to expensive and time-consuming experimentation. For efficiency, we seek to avoid wasting valuable resources in testing sub-optimal designs. One way to achieve this is by obtaining cheaper approximations of the desired system, which allow us to quickly explore new regimes and avoid areas that are clearly sub-optimal. As an example, consider the case diagrammed in Figure 3.1 from battery materials research with the goal of designing electrode materials for optimal performance in pouch cells. We can use experiments with cheaper coin cells and shorter test procedures to approximate the behaviour of the material in longer stability tests in pouch cells, which is in turn closer to the expected performance in electric car applications [Chen et al., 2019, Dörfler et al., 2020, Liu et al., 2021]. Similarly, design goals regarding battery life such as discharge capacity retention can be approximated using an early prediction model on the first few charge cycles rather than running aging and stability tests to completion [Attia et al., 2020].

Furthermore, in most applications it is infeasible to wait for a single experiment to end before planning new ones, due to the long waiting times for the experiments to finish (from a few weeks to months). This means that in practice experiments will be carried out in batch or in parallel. However, owing to the introduction of cheaper approximations and variations in material degradation rates, experiments may not finish simultaneously. For example, if our experimental batch consists of a mixture of cells tested with slower and accelerated test procedures, some results will become available in a few weeks, whereas others can take *months*. In the meantime, we do not want to leave resources idle while we wait for all experiments to be finished. Therefore, we must select new experiments before receiving the full results from those still in progress. This inspires the question: *How can we automatically and continuously design new experiments when the prior experimental data returns on different (and uncertain) time-frames and the type of measurements may vary?*

This chapter focuses on recent advances in multi-fidelity Bayesian optimisation [Kandasamy et al., 2016a, Takeno et al., 2020, Moss et al., 2021b] and batch Bayesian optimisation [González et al., 2016a, Alvi et al., 2019, Kandasamy et al., 2018, Snoek et al., 2012]. Multi-fidelity Bayesian optimisation uses cheap approximations of the objective function to speed up the optimisation process. Batch Bayesian optimisation selects multiple experiments at the same time. We explore the close relationship between both research areas, provide an overview of the current state-of-the-art, and introduce a general algorithm that allows practitioners to *simultaneously* take advantage of multi-fidelity and batching while using any acquisition function of their choice, significantly speeding up Bayesian optimisation procedures. We show the usefulness of the method by applying it to synthetic benchmarks, a high-dimensional example, and real life-experiments relating to coin and pouch cell batteries tested with different measurement procedures.

The contributions of this chapter include:

I. A brief review of state-of-the-art multi-fidelity and batch Bayesian optimisation methods.

II. We propose an algorithm that allows us to incorporate multi-fidelity and batch Bayesian optimisation to any acquisition function. Allowing many acquisition functions extends

the current literature, because previous contributions are limited to a single acquisition function each.

III. We provide an empirical study, that showcases the benefit of adding multi-fidelity and parallelisation to black-box optimisation.

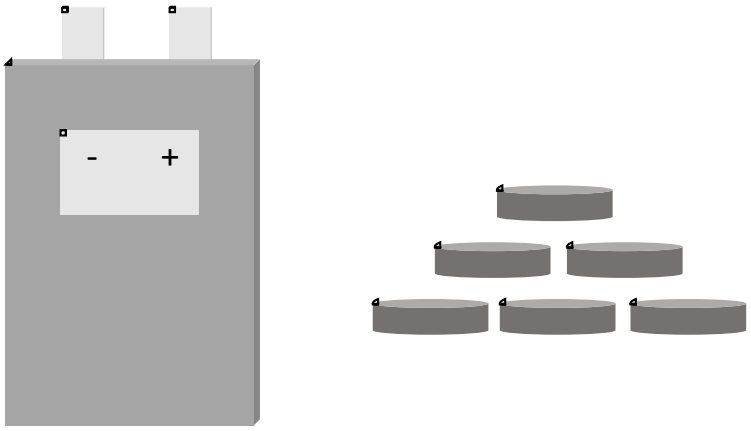


Figure 3.1: Motivating example. On the left is pouch-cell battery and on the right are coin cell batteries [Chen et al., 2019, Dörfler et al., 2020, Liu et al., 2021]. Using coin cells, which are cheaper and faster to make than pouch batteries, we can approximate the behaviour of our target. Usually these experiments are carried out in large batches.

3.2 Related Work

Bayesian optimisation has been well researched in the domain of tuning the hyper-parameters of computationally expensive machine learning models [Bergstra et al., 2011, Li et al., 2017]. However, to be efficiently applicable to chemical engineering there is a need to adapt and improve such methods [Amaran et al., 2016, Thebelt et al., 2022c, Wang and Dowling, 2022]. For example, in Chapter 4 we consider the physical cost of changing from one experimental set-up to another and Thebelt et al. [2021, 2022b,a] allow input constraints in addition to the black-box functions. The related area of *grey-box optimisation* seeks to solve optimal decision-making problems incorporating both black-box, data-driven models and mechanistic, equation-based models [Cozad et al., 2014, Boukouvala and Floudas, 2017, Olofsson et al., 2018, Park et al., 2018, Paulson and Lu, 2022, Kudva et al., 2022].

The multi-fidelity setting has been gaining attention in machine learning research recently. Multi-fidelity Bayesian optimisation uses cheap approximations of the objective function to explore the input space more efficiently, and saves full evaluations for more promising areas. Different approaches for modelling the multi-fidelity using Gaussian Processes (GPs) have been developed [Kennedy and O'Hagan, 2000, Gratiet, 2013, Cutajar et al., 2019]. In terms of Bayesian optimisation, there has been research from the basic k-armed bandit problem [Kandasamy et al., 2016b] to more general non-GP based methods [Sen et al., 2018]. This chapter addresses the setting and method introduced in Kandasamy et al. [2016a], where we use an independent GP to model each fidelity. Each fidelity is linked by a bias assumption which allows the transfer of information by taking the tightest of many upper confidence bounds. As a more efficient alternative, we will also explore multi-task Gaussian Processes [Journel and Huijbregts, 1976, Goovaerts et al., 1997, Alvarez et al., 2012], and information-based multi-fidelity [Takeno et al., 2020, Moss et al., 2021b].

Multi-fidelity shares a close relationship with transfer learning [Zhuang et al., 2021, Pan and Yang, 2009], which has seen many successful engineering applications [Li et al., 2020, Li and Rangarajan, 2022, Jia et al., 2020, Rogers et al., 2022]. In transfer learning, we seek to get improved performance in a target task by using data from similar tasks that have been carried out in the past. Multi-fidelity optimisation can be thought of as transfer learning, where we can actively *choose* to increase the data of auxiliary data-sets.

Asynchronous Bayesian optimisation chooses queries while waiting for delayed observations. Kandasamy et al. [2018] proposed a method based on Thompson Sampling, using the method's natural randomness to ensure variety in experiments. Ginsbourger et al. [2011] introduce a variant of Expected Improvement, which seeks to integrate out ongoing queries, and uses Monte Carlo methods to estimate the acquisition function. Snoek et al. [2012] introduces a more general version of this idea that 'fantasises' delayed observations. González et al. [2016a] and Alvi et al. [2019] propose mimicking sequential Bayesian optimisation by *penalising* the acquisition function at points where we are asynchronously running experiments.

An example where multi-fidelity approximations are being applied is on the area of molecule

design and synthesis [Alshehri et al., 2020], where we can use cheap computer approximations to search the space of molecules, and then choose which real experiments to carry out based on the best performing simulations [Coley, 2021]. Coley et al. [2019] use computer simulations to determine feasibility in synthesising organic compounds, and then use a robotic arm that carries out experiments in batch. This is could be seen as having a multi-fidelity step followed by a batching step, however, the methods are carried out separately. Petsagkourakis et al. [2021] used multi-fidelity in the context of safe Bayesian optimisation.

Recently, Li et al. [2021] combined batch and multi-fidelity methods using Bayesian Neural Networks (BNNs). Our work focuses on simpler, more robust and computationally feasible methods that work directly on the asynchronous setting. Takeno et al. [2020] and Moss et al. [2021b] propose a multi-fidelity Bayesian optimisation algorithm based on max-entropy search [Wang and Jegelka, 2017], specialising in asynchronous batching, we take inspiration from this work and generalise it to help avoid the computational complexities of the method, and possible pitfalls from being constrained to a single acquisition function.

A significant proportion of the work in this chapter was carried out in late 2020, and it was later revisited and completed in early 2022. In the interval, GIBBON [Moss et al., 2021a] was proposed as a general-purpose extension of max-entropy search, by using cheap approximations to the information gain. Due to the timings, GIBBON was not included as baseline, however, it tackles the same problem we are studying in this chapter and is highly relevant and applicable, in particular since it tackles our main criticism of entropy-based method i.e. that they are difficult to optimise due to the need to approximate the objective using Monte Carlo.

3.3 Multi-fidelity Modelling

Optimal experiment design relies on surrogate models to convey information about our knowledge of the search space. The surrogate model represents our *belief* of how the underlying black-box function looks like. Selecting new experiments near our surrogate's optimum is known as *exploiting*, which is when we choose new experiments close to the best experiments

we have observed so far. It is also important for the model to provide some measure of *uncertainty* [Hüllen et al., 2020], which we can leverage to *explore* the search space. Uncertainty should be high in areas where we do not have any data, and low where we have a lot of data.

In particular, in this section we focus on multi-fidelity modelling, where we assume our data-set is formed by a combination of smaller data-sets of varying quality. A few concrete examples where this happens include:

(a) In battery research, we may conduct experiments on pouch cell batteries, or cheaper experiments using coin cell batteries.

(b) Kennedy and O'Hagan [2000] consider the case of combining an expensive computational code which can also be run much faster at a lower level of sophistication, resulting in two data-sets.

(c) In machine learning, an expensive model, such as a neural network, can be quickly trained on a subset of the training data to approximate the behaviour of the fully-trained algorithm.

We focus on the case where we have a discrete number of fixed fidelities, which in practice will be two or three. We note that while we may be able to choose which fidelity to experiment on, we cannot choose the number of fidelities. For interested readers, Kandasamy et al. [2017], Moss et al. [2021b] provide a treatment Bayesian optimisation with a continuous fidelity space. An example of such a fidelity space is early termination of training in a machine learning model where we can choose how long to train for e.g. choose any time between 2 hours and 3 days (in this case the longer we train, the better the approximation).

Formally, we assume we are trying to model a function f. However, we have access to M auxiliary functions, $f^{(m)}$, which approximate f for $m = 1, ..., M$. Each function will have a data-set, $\mathcal{D}^{(m)} = \{(x_i^{(m)}, y_i^{(m)})\}_{i=1}^{N_m}$ consisting of noisy observations of each function:

$$y_i^{(m)} = f^{(m)}(x_i^{(m)}) + \sigma^{(m)}\epsilon_i, \tag{3.1}$$

where $\epsilon_i \sim \mathcal{N}(0,1)$. We want a model that will use information from all data-sets to have a more accurate model of the objective $f =: f^{(M)}$. For example, in the battery example, we have access to $M = 2$ types of data – the cheaper coin cell battery data, and the target pouch cell battery data. We want to use both types of data together, to obtain a more accurate model for pouch cell batteries.

As introduced in Section 2.2 Gaussian Processes will be used as surrogate models, under the Gaussian likelihood assumption which allows us to explicitly compute the posterior of f.

3.3.1 Independent Gaussian Processes

The first model we consider is the simplest. We fit an independent GP to each fidelity. That is, we assume the prior:

$$f^{(m)} \sim \mathcal{GP}(\mu_0^{(m)}, k_0^{(m)}), \tag{3.2}$$

and obtain the posterior independently, that is, for each fidelity $m = 1, ..., M$, the posterior is obtained by only conditioning on each fidelities data-set: $f^{(m)}|\cup_{m'=1}^{M} D^{(m')} = f^{(m)}|D^{(m)}$. There are two main benefits to this approach: (a) This is the simplest and computationally cheapest out of all exact inference GP methods. (b) It is the easiest on which to intuitively impose prior knowledge and information. However, the main drawback is significant: there is no transfer of information between the fidelities, so we will be forced to make simplifying assumptions relating to hierarchy and bias at the time of optimisation.

3.3.2 Multi-task Gaussian Processes

A more complete approach seeks to *jointly* model all fidelities. One effective and popular approach is to use a multi-task Gaussian Process [Alvarez et al., 2012]. This approach models each fidelity as an output of a Multi-output Gaussian Processes (MOGP). Instead of having a scalar-valued mean function, we have a M-dimensional vector valued function, $\mu : \mathcal{X} \to \mathbb{R}^M$ and a $M \times M$-dimensional matrix valued co-variance function $K : \mathcal{X} \times \mathcal{X} :\to \mathbb{R}^{M \times M}$, where

the (m, m')th entry of $K(x, x')$ corresponds to the covariance of $f^{(m)}(x)$ and $f^{(m')}(x')$.

We focus on separable kernels and on the Linear Model of Coregionalisation [Journel and Huijbregts, 1976, Goovaerts et al., 1997]. A *separable kernel* is where the input-dependence and the fidelity dependence can be separated into a product:

$$K(x, x')_{m,m'} = k^{(I)}(x, x')k^{(F)}(m, m') \tag{3.3}$$

where $k^{(I)}$ and $k^{(F)}$ are scalar kernels corresponding to the inputs and fidelities correspondingly. In notation, the matrix-valued kernel can be written as:

$$K(x, x') = k_I(x, x')\mathcal{B} \tag{3.4}$$

where k_I is now a scalar valued input kernel, and $\mathcal{B}$ is a matrix weighting the task-dependencies. The sum of kernels is also a valid kernel, therefore we can create more flexible models by considering the *sum* of W separable kernels:

$$K(x, x') = \sum_{w=1}^{W} k_w^{(I)}(x, x')\mathcal{B}_w. \tag{3.5}$$

This allows us to have different types of kernels in the sum, or kernels with different hyper-parameters within the same model. The class of MOGP models with kernels given as (3.5) is known as the Linear Model of Coregionalisation (LMC). Special cases of the model include the Intrinsic Coregionalisation Model [Goovaerts et al., 1997] (where all kernels k_w are the same) and the Semi-parametric Latent Factor Model [Teh et al., 2005].

The main benefit of using these models is that the task-dependency is learnt from the data! We learn to transfer information from the lower-fidelities to the target fidelity. The drawbacks against independent models are (i) higher computational expense when doing exact inference and (ii) the kernel hyper-parameters become difficult to set manually.

3.3.3 Further Extensions

This thesis limits itself to the methods mentioned above. However, we mention a few more complicated models that could be used when needed. The LMC is limited because it assumes that the co-variance function is a linear combination of separable kernels. A more general model allows for non-separable kernels by considering processes convolutions [Ver Hoef and Barry, 1998, Higdon, 2002, van der Wilk et al., 2017]. In addition, we can use multi-output Gaussian processes that allow for non-linear combinations of Gaussian processes, for example latent variable models [Lawrence, 2003] which Sedgwick et al. [2024] successfully applied in transfer learning.

Deep Gaussian Processes [Damianou and Lawrence, 2013] have also been used to model multi-fidelity systems, where each layer represents a fidelity [Cutajar et al., 2019]. Recently, Savage et al. [2022] used a variant of this method to efficiently run multi-fidelity Bayesian optimisation in a simulated chemical reactor, and later expanded the work to continuous fidelities [Savage et al., 2023, 2024].

Li et al. [2021] use Bayesian Neural Networks [MacKay, 1992, Neal, 2012] rather than Gaussian Processes, thereby allowing for learning of complex non-linearities while still providing uncertainty estimates. However, such models are notoriously difficult to train [Izmailov et al., 2021]. We were interested in applying Bayesian Neural Networks models, but, we were unable to consistently obtain good inference so we discarded them from our analysis.

3.4 Multi-fidelity Bayesian Optimisation

One of the main ideas in Bayesian optimisation is managing the trade-off between exploration and exploitation. The algorithm has to decide between experimenting in promising areas where it has observed high values of f, against choosing to experiment in areas where there is little information. If there is too much exploration, the algorithm will take too long to find the optimum. However, if there is too much exploitation, the algorithm might settle in a sub-

optimal area.

More concretely, at time t, we assume that we can *choose* an input-fidelity pair, (x_t, m), from which we obtain a possibly noise observation, $y_t^{(m)}$ of the mth fidelity function, $f^{(m)}$, as in Equation (3.1).

The target of the optimisation is to maximise the function at the highest fidelity $f^{(M)}$. We further assume that each fidelity will have a known cost $C^{(m)}$, which is lower than the cost at the highest fidelity. For the purposes of this work, we will assume the cost refers to a measure of *time*. However, it could represent financial costs, man-power, or other similar costs. We clarify that we assume the costs are scalar-values known a priori. While it would be useful to consider a trade-off between different types of costs, this is left to future work.

Multi-fidelity methods allow us to use cheap approximations to explore while saving the expensive evaluations for a separate exploitation phase of the algorithm. All multi-fidelity methods follow the same pattern: at first they will query the lower fidelities and as the optimisation loop progresses we begin to query the target fidelity.

3.4.1 Multi-fidelity Upper Confidence Bound

Kandasamy et al. [2016a] propose one of the most intuitive and simplest methods. They show that by making the right assumptions, you are able to use independent Gaussian Processes for each fidelity and still transfer information effectively by using Upper Confidence Bounds (UCB). The UCB acquisition function, for a Gaussian Process posterior with mean function, $\mu_t(\cdot)$, and variance function $\sigma_t^2(x)$ is given by:

$$\alpha_{UCB}(x; \beta_t, \mathcal{D}_t) = \mu_t(x) + \beta_t^{1/2}\ \sigma_t(x). \tag{3.6}$$

In this equation, $\mu_t(x)$ represents our *belief* of how the function looks like. We then add $\beta_t^{1/2}\sigma_t(x)$, which represents the uncertainty in our model, to generate an upper confidence bound of the black-box function. Here, $\beta_t > 0$ is a hyper-parameter that balances how much

we value the uncertainty when deciding which point to choose next. A large value represents being optimistic about areas where we are uncertain and will naturally lead to more exploration and reduced exploitation. In practice, β_t is usually chosen as an increasing logarithmic function which allows us to have theoretical guarantees [Srinivas et al., 2010].

Assume that there are M different fidelities, and we are interested in optimising the highest fidelity, $f^{(M)} := f$. The key assumption is that there is a decreasing *known* maximum bias among the fidelities, such that:

$$||f^{(m)} - f\ ||_\infty := \max_{x \in \mathcal{X}} |f^{(m)}(x) - f(x)| \leq \zeta^{(m)}_{MAX} \tag{3.7}$$

for $\zeta^{(1)}_{MAX} \geq \zeta^{(2)}_{MAX} \geq ... \geq \zeta^{(M)}_{MAX} = 0$. That is, we assume that we know the *worst possible* bias that each fidelity might have, and this worst possible bias is strictly non-increasing on fidelity space.

To combine information from multiple fidelities, we begin by fitting an independent GP to the data for each fidelity. That is, for each fidelity m, there is a data-set $\mathcal{D}^{(m)}_t$ of experiments, so that $f^{(m)} \mid \mathcal{D}^{(m)}_t \sim \mathcal{GP}(\mu^{(m)}_t, k^{(m)}_t)$. Then the upper confidence bound is defined as:

$$\alpha^{(m)}(x; D^{(m)}_t) = \mu^{(m)}_t(x) + \beta^{1/2}_t\ \sigma^{(m)}_t(x) + \zeta^{(m)}_{MAX}$$

where $(\sigma^{(m)}_t)^2(x) = (k^{(m)}_t)^2(x)$. That is, for each fidelity, we sum the mean prediction, the uncertainty, *and the bias*, to obtain an upper bound on the *objective* fidelity. Therefore we can build a narrow bound by considering the minimum of all M upper confidence bounds at each possible input x. With this idea in mind, the acquisition function is then defined as:

$$\alpha(x; \mathcal{D}_t) = \min_{m \in \{1,...,M\}} \alpha^{(m)}(x; D^{(m)}_t). \tag{3.8}$$

That is, each $\alpha^{(m)}$ represents the mth UCB, and we simply take the minimum to obtain the tightest bound. Figure 3.2 gives a graphical example of building this acquisition function.

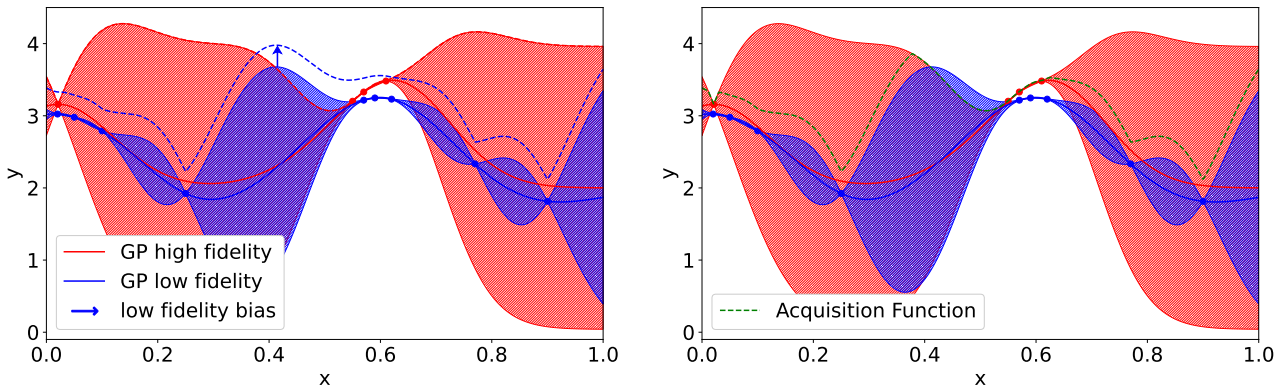
The next experiment is chosen by maximising the acquisition function given by (3.8). However,

the choice of a fidelity level still remains. Consider the standard deviation of the surrogate model for fidelity m at input x_{t+1}, $\sigma_t^{(m)}(x_{t+1})$. If this quantity is high, it means there remains a lot of information to learn about fidelity m at our specific input. If this quantity is low, it means we cannot learn much from experimenting at this fidelity.

Therefore we define a set of threshold levels $\{\gamma^{(m)}\}_{m=1}^{M}$. If the posterior variance is smaller than the corresponding threshold, we cannot learn much about the objective function by sampling at this lower fidelity so it is better to choose a higher fidelity. To try to minimise the cost of evaluation, we choose the lowest fidelity that contains enough information according to the thresholds. More precisely, the choice is given by:

$$m_{t+1} = \underset{m\in\{0,\dots,M\}}{\arg\min} \{m : \beta_t^{1/2}\sigma_t^{(m)} > \gamma^{(m)} \text{ or } m = M\}. \tag{3.9}$$

We note that this idea for choosing the fidelity is not restricted to upper-confidence bound acquisition functions. In fact, we can use *any* acquisition function and then choose the fidelity with rule defined by equation (3.9). We shall see that combining this rule with LMCs means we can drop the bias assumption and obtain better results than using independent GPs.



(a) Independent UCBs for each fidelity.

(b) Combination of UCBs to build acquisition function.

Figure 3.2: Building a multi-fidelity UCB, in the case where $M = 2$. The dotted lines represent upper confidence bounds based on bias and variances. In the first figure we can see the bounds for each independent fidelity, in the second figure both bounds are combined to obtain the final acquisition function.

3.4.2 Multi-fidelity Max Entropy Search

Information-based methods have become increasingly popular in Bayesian optimisation due to their strong theoretical backing. Entropy search [Villemonteix et al., 2009, Hennig and Schuler, 2012, Hernández-Lobato et al., 2014] tries to maximise the information gain about the optimum input, x_*. Unfortunately, this means estimating an expensive d-dimensional integral using Monte Carlo approximations. Max-value Entropy Search (MES) [Wang and Jegelka, 2017] instead focuses on maximising the information gain about the maximum value of the function, $f_* = f(x_*)$:

$$\alpha_t(x) = I((x, f(x); f_*|\mathcal{D}_t) = H(p(y|\mathcal{D}_t, x)) - \mathbb{E}_{f_*|\mathcal{D}_t}\left[H(p(y|\mathcal{D}_t, x, f_*))\right]. \tag{3.10}$$

The left hand side, $I((x, f(x); f_*|\mathcal{D}_t)$ represents the mutual information between the next (possibly) evaluated point, and the maximum value of the function f_*. It can be calculated by the difference between the current entropy of the predictive distribution of y at input x, and the expected entropy over the distribution of the maximum.

The big advantage of MES is that the expectation is reduced from a d-dimensional to 1-dimensional integral. The expectation is estimated using Monte Carlo methods since we are able to sample from the distribution $p(f_*|\mathcal{D}_t)$ via Thompson Sampling or an approximation via the Gumbel distribution. More recently, Tu et al. [2022] and Hvarfner et al. [2022] jointly consider the joint information gain of both the maximum value, and the optimum input.

A multi-fidelity extension of Max-value Entropy Search (MF-MES) was introduced by Takeno et al. [2020] and Moss et al. [2021b]. We easily extend to the multi-fidelity setting by considering a joint model, such as the LMC, and then maximising the cost-weighted information gain:

$$\alpha_t(x, m|\mathcal{D}_t) = \frac{I(x, f^{(m)}(x); f_*|\mathcal{D}_t)}{C^{(m)}}, \tag{3.11}$$

here f_* represents the maximum at the *highest* fidelity. So we calculate how much more information we gain about the black-box function's maximum, by querying at input x and fidelity

m. We divide this by the cost of querying fidelity m.

The biggest strength of this method, when compared to multi-fidelity upper confidence bound, is that we are not longer using independent GPs as a model. As such, we can *learn* the relationship between the functions and there is no need to make assumptions about the maximum bias. There are also no hyper-parameters in the acquisition function.

3.5 Asynchronous Batch Bayesian Optimisation

So far we have focused on sequential Bayesian optimisation. However, in some battery experiments, we cannot wait for the experiment results to return before choosing the next point, we must do this in batch. It is also important to note that we expect different fidelity levels to have a considerably different evaluation time. This can be a problem as most batch methods focus on the synchronous setting. That is, you select a batch, wait for all evaluations to return, and then select another. Large time discrepancies in the evaluation times pose a problem, as there are a lot of resources that will be left idle while longer evaluations finish. As such, we are more interested in the less-studied asynchronous setting.

Throughout this section, assume we are in the normal setting of Bayesian optimisation, however, further assume that at time t we are evaluating a batch of points $\mathcal{Q}_t = \{x_{t,q}\}_{q=1}^{Q}$ whose observations, $f_{\mathcal{Q}}$, we still do not know. We are then interested in choosing the next experiment while taking $\mathcal{Q}$ into account, so that the acquisition function depends on the pending evaluations:

$$x_{t+1} = \arg\max_{x \in \mathcal{X}} \alpha_t(x|\mathcal{D}_t, \mathcal{Q}_t). \tag{3.12}$$

3.5.1 Thompson Sampling

As described in Section 2.3, Thompson Sampling is a common technique in Bayesian decision making that uses samples from the posterior distribution to guide actions. Kandasamy et al. [2018] show it can be used effectively for asynchronous batch Bayesian optimisation. A sample

is selected by maximising a sample from the posterior GP:

$$f_{sample} \sim \mathcal{GP}(\mu_t, \kappa_t|\mathcal{D}_t)$$
$$x_{t+1} = \arg\max_{x\in\mathcal{X}} f_{sample}(x). \tag{3.13}$$

Note that the new point selected is *independent* of $\mathcal{Q}$, however, the randomness in the sampling is enough to ensure diversity in the batch. Due to the independence, it works well in the asynchronous batch setting, and large batches can be created cheaply.

3.5.2 Parallel Querying Through Fantasies

Snoek et al. [2012] allow for the asynchronous batching of any acquisition function. This can be achieved by creating multiple 'fantasies' at every point in $\mathcal{Q}$, and then using them to marginalise out the pending observations, $f_{\mathcal{Q}}$:

$$\begin{aligned}\alpha_t(x|\mathcal{D}_t, \mathcal{Q}_t) &= \int \alpha(x|\mathcal{D}_t, f_{\mathcal{Q}}, \mathcal{Q}_t)p(f_{\mathcal{Q}}|\mathcal{D}_t)\mathrm{d}f_{\mathcal{Q}} \\ &\approx \frac{1}{S}\sum_{s=1}^{S} \alpha_t(x|\mathcal{D}_t \cup \{\tilde{f}_{\mathcal{Q}}^{(s)}, \mathcal{Q}_t\})\end{aligned} \tag{3.14}$$

where $\tilde{f}_{\mathcal{Q}}^{(s)}$ are sample drawn from the normal distribution $f_{\mathcal{Q}}|\mathcal{D}_t$, and S is the number of samples. Note that we are trying to estimate a Q-dimensional integral, so the larger the number of fantasies, the better the approximation but the computational expense increases. One consideration though, is that we *do not* need to fully re-train the Gaussian Process for every fantasy. This is because the variance of the posterior only depends on the input locations, $\mathcal{Q}$, which are the same for every fantasy. Instead, we only need to re-train the mean of the process which is much cheaper.

In the case of Max-value Entropy Search, Takeno et al. [2020] show that using fantasies, the Q-dimensional integral can be written as a 2-dimensional integral which means we do not need as many fantasies to get good approximations.

Gaussian Process Bandit UCB [Desautels et al., 2014] provides a computationally cheap version of fantasising, where a single sample, equal to the predictive mean of the GP at all points in $\mathcal{Q}$ is used, i.e., you assume that all pending observations will simply equal the predictive mean (and so the predictive mean remains unchanged, while the predictive variance decreases around pending points). Recently, Zhang et al. [2022] showed its practicality by using to design bacterial ribosome binding sites. We do note that the empirical success of this method is restricted to the Upper Confidence Bound, and may not work as well with other acquisition functions.

3.5.3 Local Penalisation

Local penalisation is a batch Bayesian optimisation algorithm introduced by González et al. [2016a], and which was then extended to the asynchronous setting by Alvi et al. [2019].

The method relies on trying to mimic the sequential setting, where after receiving experimental information about a particular input we usually expect the acquisition function to decrease in the neighbourhood surrounding this point. More precisely, assume we wish to select a new point x_t, then we can mimic the sequential setting by defining the new acquisition function, $\tilde{\alpha}$, as:

$$\tilde{\alpha}(x; \mathcal{D}_t, \mathcal{Q}) = g(\alpha(x; \mathcal{D}_t)) \prod_{x_{t,q} \in \mathcal{Q}} \psi(x_{t,q}) \tag{3.15}$$

where $\psi(x)$ is a function that penalises the region around x, and $g(\cdot)$ is an increasing differentiable transformation that makes sure the acquisition function is positive. And so, in Equation 3.15 we have a positive transformation of the original acquisition function, multiplied by penalisers centered at every pending experiment. We use $g(z) = z$ for positive acquisition functions, and the soft-max transformation otherwise, $g(z) = \log(1 + e^z)$. The question then becomes, how do we choose such penalisers? To do this, we begin by assuming that the function satisfies the Lipschitz condition. That is, if $\mathcal{X}$ is compact, then:

$$|f(x_1) - f(x_2)| \leq L||x_1 - x_2|| \quad \forall x_1, x_2 \in \mathcal{X}$$

where $L \in \mathbb{R}^+$, and $|| \cdot ||$ is the L^2 norm in $\mathbb{R}^d$. We now consider the ball, around arbitrary

$x_j \in \mathcal{X}$:

$$B_{r_j}(x_j) = \{x \in \mathcal{X} : ||x_j - x|| \leq r_j\}$$

where the radius r_j is given by:

$$r_j = \frac{f(x_*) - f(x_j)}{L} =: r(x_j).$$

The radius is chosen because, if the Lipschitz assumption holds, then x_* cannot lie inside this ball. Note that due to f being modeled by a GP, then this radius is a random quantity. González et al. [2016a] then define the local penaliser as:

$$\psi(x; x_j) = 1 - p(x \in B_{r_j}(x_j)),$$

that is, we choose the local penaliser to be the probability that x does not lie inside $B_{r_j}(x_j)$. However, Alvi et al. [2019] argue that this penaliser may lead to redundantly sampling the same points. Instead they propose the penaliser:

$$\psi(x; x_j) = \min \left\{ \frac{||x - x_j||}{\mathbb{E}(r_j) + \sigma_t(x_j)/L},\ 1 \right\}. \tag{3.16}$$

Figure 3.3 shows an example of how the penalisation can be used to alter the acquisition function to choose a new experiment.

3.6 Relationship between multi-fidelity and asynchronous batch methods

In Tables 3.1 and 3.2 we can find a summary of all the introduced methods for multi-fidelity and batching, including information about the weaknesses and strengths of each method.

Ultimately the goal of using multi-fidelity and batch methods is to speed up the optimisation by using extra tools available to us. Multi-fidelity methods leverage cheap approximations to

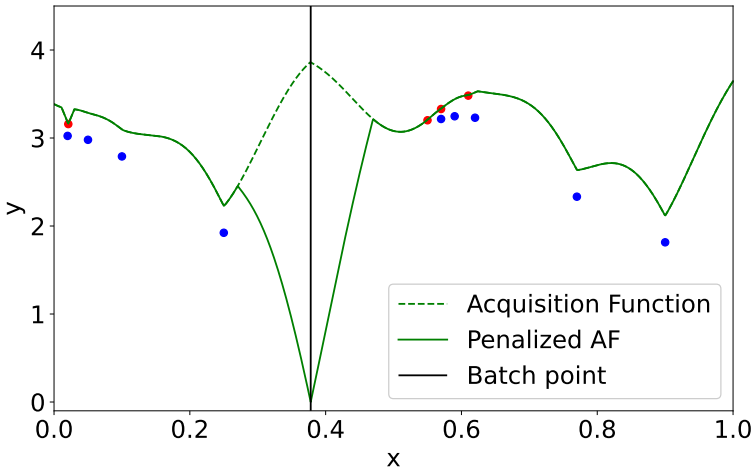


Figure 3.3: An example of the local penalisation process. We selected $x = 0.378$ as a point to evaluate. The acquisition function is then penalised as shown above until we obtain a new observation. This allows us to select new queries without receiving an observation, as the penalised acquisition function is guaranteed to have a different maximum. The blue and red dots represent low and high fidelity observations, respectively.

the objective, while batching takes advantage of running experiments in parallel. Therefore it is natural to want to use both methods at the same time to obtain even more speed gains in optimisation procedures.

However, we emphasise that we cannot fully use both tools simultaneously without asynchronicity arising. Indeed, by definition, lower fidelity approximations are meant to be cheaper and faster to obtain. This means if we select a batch of evaluations, those of lower fidelities will finish earlier than high fidelity evaluations. Without any asynchronicity we must wait for all experiments to finish before deciding the next batch, making the method inefficient.

There can also be strong relationships between fidelity and *batch size*. Indeed, it might be the case that many low fidelity experiments can easily be done in parallel, but due to having to use more resources, for high fidelities we can only parallelise a few. An example of this could be as simple as trying out cake recipes. A full fidelity observation would be baking a cake in the oven; while low fidelity observations could be making muffins instead of a full cake. In such an example, since we are limited by oven size, we can parallelise low-fidelity experiments much easier than high fidelity ones.

We focus on multi-fidelity and batch methods that allow mimicking of sequentially choosing

experiments and fidelities, this means we can select which experiments to run without worrying about selecting a batch size or limiting ourselves to single-fidelity batches.

Multi-fidelity			
Method	**Reference**	**Strengths**	**Weaknesses**
MF-GP-UCB	Kandasamy et. al [2016a].	(a) The acquisition function and fidelity criterion can be calculated cheaply and exactly (for data-sets of a few hundred experiments it should take a few seconds to optimise). (b) Using independent GPs allows for expert knowledge to be incorporated more easily.	(a) It requires the maximum bias to be known. By using independent GPs it does not learn relationships between fidelities. (b) We need to choose the value of the threshold hyper--parameter and for the UCB acquisition function.
MF-MES	Takeno et al. [2020] Moss et al. [2021b]	(a) Relationship for fidelities is learnt from data, allowing for more efficient experimental choices. (b) No hyper-parameter to choose outside of the approximation techniques. (c) Batch extension only requires calculation of 2-dimensional integral.	(a) The acquisition function and fidelity criterion cannot be calculated exactly and needs to be estimated. Computational cost may grow depending on the approximation quality (it could take up to a few minutes to optimise). (b) Suffers from normal BO drawbacks: e.g. over-exploration of borders. (c) Cannot incorporate expert knowledge easily.

Table 3.1: Comparison table for multi-fidelity methods, highlighting and summarising the strengths and weaknesses of each method.

Asynchronous Batch			
Method	**Reference**	**Strengths**	**Weaknesses**
Thompson Sampling	Kandasamy et al. [2018a]	(a) Parallel computation of batch points allows for cheap creation of large batches (but computational cost can be very variable depending on coarseness of sampling grid). (b) Hyper-parameter free.	(a) Experiment designs can be too exploitative (i.e., redundant in batch). (b) Calculation can be unfeasible for grids with a large number of points.
Fantasies	Snoek et al. [2012]	(a) Strong theoretical motivation. (b) No hyper-parameters to choose outside of those used in approximation techniques.	(a) Acquisition function is a large dimensional integral, approximation may be unfeasible for large batch sizes. (b) As such, acquisition function may take a long time to optimise (from a few minutes to a few hours).
Local Penalisation	González et al. [2016b] Alvi et al. [2019]	(a) Acquisition function can be calculated analytically. (b) Heuristic is well motivated, and has strong empirical performance.	(a) Requires approximation of various hyper--parameters. (b) Batch creation is done sequentially, so it can be expensive for large batch sizes (a few minutes to optimise).

Table 3.2: Comparison table for asynchronous batch methods, highlighting and summarising the strengths and weaknesses of each method.

3.7 Asynchronous Multi-fidelity Batch Optimisation

3.7.1 Problem Setting

We now seek to formalise the problem setting. The method presented seeks to find the best possible solution, as *quickly* as possible. We are interested in:

$$x_* = \arg\max_{x \in \mathcal{X}} f(x)$$

where $\mathcal{X}$ denotes a *continuous* input space. We assume f is expensive to evaluate, but, we have access to lower fidelity approximations, $f^{(m)}$, which tend to be cheaper (for $m = 1, ..., M$). These approximations may be biased or noisy.

Because we are trying to optimise with respect to time, we introduce a discrete time component. We denote the input of the ith experiment by $x_{i,t}^{(m)}$, where $t \in \{1, ..., T\}$ denotes the time at which the experiment began, and $m \in \{1, ..., M\}$ denotes the fidelity.

When we select an input point, $x_{i,t}^{(m)}$, we obtain a noisy observation of $f^{(m)}$. However, we may not observe it at time t. Instead, we observe it at time $t+\tau_i^{(m)}$, where $\tau_i^{(m)}$ can be a deterministic or random time. We denote the corresponding observation $y_{i,t+\tau_i}^{(m)}$. We assume the relationship between the observation and the experimental input is given by:

$$y_{i,t+\tau_i}^{(m)} = f^{(m)}\left(x_{i,t}^{(m)}\right) + \eta^{(m)}\epsilon_i, \tag{3.17}$$

where $\eta^{(m)}$ denotes the noise level. We assume ϵ_i is standard Gaussian, where the variance may depend on the fidelity level.

Finally we assume that each fidelity has an associated batch space $\lambda^{(m)}$ (which in practice typically increases with fidelity). We assume this batch space is not not freed up until we have an observation. We also have a maximum budget which we cannot exceed at any one time, Λ. If we let λ_i denote the batch space associated with the ith observation, and we let $\mathcal{I}$ denote

the index set of the unobserved queries, we can write this as the constraint:

$$\text{TotalBatchSpace}_t := \sum_{i \in \mathcal{I}} \lambda_i \leq \Lambda. \tag{3.18}$$

The idea behind this constraint is that we may be able to run a lot of low-fidelity observations or a few high-fidelity observations or a mix of both. We note that this batch space is independent of the time it takes to query each fidelity. We are introducing this as to allow variable batch sizes. An example of this in practice, because coin cell batteries are smaller and use less material, we can concurrently run more experiments than for pouch cell experiments.

The **aim** of the algorithm would be to obtain the best possible approximation of the optimum point at time T.

3.7.2 A general algorithm for any acquisition function

A general algorithm can be constructed from all the tools mentioned above. For this we require: (a) A Multi-task Gaussian Processes model, because we will require information transfer between the fidelities, (b) a method that allows for asynchronous batching, (c) a way of choosing which fidelity to query, and (d) an acquisition function. We propose iteratively choosing the points, we first optimise a batch-modified acquisition function *on the target fidelity*:

$$x_{i+1,\ t} = \underset{x \in \mathcal{X}}{\arg\max}\, \tilde{\alpha}_t(x, M | \mathcal{D}_t, \mathcal{Q}_t) \tag{3.19}$$

where $\mathcal{Q}_t$ is the batch of points being evaluated at time t, and $\tilde{\alpha}$ is a acquisition function modified to consider that we are currently evaluating $\mathcal{Q}_t$ by either local penalising or fantasising. Once we have chosen the point, we can choose the fidelity. We consider two alternatives, the first one is the heuristic used on MF-GP-UCB:

$$m_{i+1,\ t} = \underset{m \in \{0,\ldots,M\}}{\arg\min} \{m : \beta_t^{1/2} \sigma_t^{(m)}(x_{i+1,\ t}) > \gamma^{(m)} \text{ or } m = M\}. \tag{3.20}$$

Recall, $\beta_t^{1/2}\sigma_t^{(m)}(x_{i+1,\,t})$ measures the posterior uncertainty at fidelity m and input $x_{i+1,\,t}$, and therefore, we choose the smallest fidelity whose uncertainty is larger than $\gamma^{(m)}$. In other words, we query at the lower fidelities until the uncertainty at each fidelity is low. The second alternative is to consider the information-based criterion introduced by MF-MES:

$$m_{i+1} = \underset{m\in\{0,\ldots,M\}}{\arg\max} \frac{\mathbb{E}_{f_{\mathcal{Q}}|\mathcal{D}_t}\left[I(x, f^{(m)}(x_{i+1,\,t}); f_*|\mathcal{D}_t, f_{\mathcal{Q}}, \mathcal{Q}_t)\right]}{\mathbb{E}\left[\tau^{(m)}\right]}, \tag{3.21}$$

that is, we use the expected information gain of querying the experiment at the mth fidelity, where the expectation is taken over the pending observations (and we also weight by the expected cost). The main advantage of pre-selecting the experimental input using another acquisition function, instead of optimising MF-MES directly, is that we do not need to rely on approximating an information-based acquisition function during the optimisation, and instead we only evaluate the expensive acquisition function M-times. In addition, it allows us to use specialist acquisition functions, as we shall see in the experiment section. As any acquisition function can be selected (including multi-objective acquisition functions), future work can extend the proposed methodology to cases with multiple objectives, which arise in many engineering applications [Park et al., 2018, Schweidtmann et al., 2018, Thebelt et al., 2022b, 2020, Badejo and Ierapetritou, 2022, Tu et al., 2022, 2024].

We then repeat this procedure until the maximum budget is used up. Then, at each time point, we check if there are any new observations. If there are, we update the model, and carry out the previous procedure until the budget is full again. The full procedure is detailed in Algorithm 3.

3.7.3 Parameter Estimation

In this section, we provide brief and heuristic guidance for the estimation of parameters. In particular, all Gaussian Process models have learnable hyper-parameters. In addition, if we use equation (3.20) for fidelity-selection, then we introduce hyper-parameters relating to the fidelity thresholds, β_t and $\gamma^{(m)}$. For Local Penalisation, we are also required to estimate the

Algorithm 3 Asynchronous Multi-fidelity Batch Optimisation

INPUT: Maximum Budget: Λ. Batch Spaces: $\lambda^{(m)};\ m = 1, ..., M$. Acquisition Function: α. Choice of Asynchronous Bayesian Optimisation: Thompson Sampling, Fantasies, or Local Penalisation. Choice of fidelity-selector: UCB or Information-based. [Optional] Fidelity Thresholds Parameters: $\gamma^{(m)};\ m = 1, ..., M$ and β_t. Number of Fantasies: S.
Begin algorithm:
for $t = 1, 2, 3, ..., T$ **do**
  Obtain observations $\mathcal{Y}_t$ from queries $\mathcal{X}_t$, with index set $\mathcal{I}_t$
  $\mathcal{Q}_t = \mathcal{Q}_{t-1} \backslash \mathcal{X}_t$ *// remove observations from batch*
  $\mathcal{D}_t = D_{t-1} \cup (\mathcal{X}_t \times \mathcal{Y}_t)$ *// expand data set*
  $\text{TotalBatchSpace}_t = \text{TotalBatchSpace}_{t-1} - \sum_{j \in \mathcal{I}_t} \lambda^{m_j}$ *// free up cost of evaluations*
  Update Surrogate Model
  while $\text{TotalBatchSpace}_t < \Lambda$ **do**
    if BBO Method is Local Penalisation **then**
      Estimate Maximum Value, P, and Lipschitz Constant, L
      Build Initial Acquisition Function $\alpha_t(x, M|\mathcal{D}_t)$
      $\tilde{\alpha}_t(x, M|\mathcal{D}_t) = \alpha_t(x, M|\mathcal{D}_t) \prod_{x_q \in \mathcal{Q}} \psi(x_q; \hat{L}, \hat{P})$
    else if BBO Method is Fantasies **then**
      $\mathcal{S} \leftarrow$ Sample $f_{\mathcal{Q}}|\mathcal{D}_t$ (S times)
      $\tilde{\alpha}_t(x, M|\mathcal{D}_t) = \sum_{i=1}^{S} \alpha_t(x, M|\mathcal{D}_t, \mathcal{S}_i, \mathcal{Q})$
    else if BBO Method is Thompson Sampling **then**
      $f_{sample} \sim \mathcal{GP}(\mu_t, \kappa_t|\mathcal{D}_t)$
      $x_{t+1} = \arg\max_{x \in \mathcal{X}} f_{sample}(x)$
    $x_{i,t} = \arg\max \alpha_t(x, M|\mathcal{D}_t)$
    $m_{i,t} = \text{FidelitySelector}(x_{i,t+1},\ \mathcal{D}_t,\ \mathcal{Q})$
    $\mathcal{Q}_t = \mathcal{Q}_t \cup \{x_{i,t}\}$
    $\text{TotalBatchSpace}_t = \text{TotalBatchSpace}_t + \lambda^{(m_{i,t})}$
    $i = i + 1$

Lipschitz constant and the maximum value of the function. For Fantasising we need to choose the number of fantasies we want. MES-based fidelity selection is parameter-free, but more computationally costly. For ease of notation, we write x_i to refer to $x_{i,t}^{(m)}$.

The kernel and noise parameters

We estimate the parameters of the GP, θ, by maximising the marginal log likelihood [Rasmussen and Williams, 2005]:

$$\log p(y|x,\theta) = -\frac{1}{2}\log|K_\theta| - \frac{1}{2}(y-\mu)^T K_\theta^{-1}(y-\mu) - \frac{N}{2}\log(2\pi) \tag{3.22}$$

where K_θ and μ are the covariance and predictive mean of the training set. However, we must note that only the lowest fidelity will have observations across the whole search space. The higher fidelities will only get observations in promising areas, meaning that estimating the hyper-parameters directly may lead to inaccuracies when using independent GPs. To counter this, we only fully train the GP hyper-parameters on the lowest fidelity for any multi-fidelity methods. For the other fidelities, we set the prior mean constant and the length-scales equal to the lowest fidelity ones, and we only train the scaling constants and the likelihood noise. We note that the above heuristic is unnecessary for multi-task GPs, where we can train all the kernel hyper-parameters in the combined data-set of all observations.

Choosing the fidelity thresholds, $\gamma^{(m)}$

Recall that we use the fidelity thresholds to decide in which fidelity to experiment at, and therefore it is very important that they are tuned correctly. If they are too high, we will not use the lower fidelities effectively, but if they are too low we will be stuck in the lower fidelities for a long time. To counteract this, Kandasamy et al. [2016a] proposes the following heuristic: initialise the thresholds as small values, and if the process does not experiment above the mth fidelity for more than $\tau^{(m+1)}/\tau^{(m)}$ (the fidelity cost ratio) time points, double the value of $\gamma^{(m)}$.

Alternatively, we found good experimental results were achieved by setting $\gamma = 0.1$. This was chosen as all benchmarks were quasi-normalised beforehand so that the output values would be close to 1, therefore, setting $\gamma = 0.1$ was representative of the model knowing the low-fidelity value up to approximately 10% error. In practice, fixed thresholds should be carefully scaled with the outputs.

The penalisation parameters

The local penalisers depend on two parameters, the Lipschitz constant, L, and the maximum value of the function, $P = f(x_*)$. In practice, we will never know them, so it is important to have a good method of estimating them. González et al. [2016a] propose ways of estimating both.

The maximum value can be estimated cheaply with $\hat{P} = \max_i\{y_i\}$, or with the slightly less rough estimate $\hat{P} = \max_{x\in\mathcal{X}} \mu_t(x)$. We tend to prefer the first option, as it is slightly cheaper, and we expect the estimates to be very similar. To adapt it to the multi-fidelity case, we get an estimate independently for each fidelity, keeping in mind we are trying to mimic the sequential case.

Estimating the Lipschitz constant is slightly more elaborate. It can be shown that $L_\nabla = \max_{x\in\mathcal{X}} ||\nabla f(x)||$ is a valid Lipschitz constant. In addition, we know the distribution of the gradient:

$$\nabla f(x)|\mathcal{D}_t \sim \mathcal{N}(\mu_\nabla(x), \Sigma^2_\nabla(x))$$

where the mean vector is:

$$\mu_\nabla(x) = \partial K_{t,*}(x)\tilde{K}_t^{-1} y$$

for $\tilde{K}_t = K_t + \eta^2 I$ and $(\partial K_{t,*})_{i,l} = \partial k_t(x)/\partial x^{(i)}$, and y is the vector containing all observations so far. We can then estimate the Lipschitz constant as $\hat{L} = \max_{x\in\mathcal{X}} ||\mu_\nabla(x)||$.

This could further be improved, Alvi et al. [2019] propose using the local estimator, $L_{\nabla,i} = \max_{x\in\mathcal{X}_i} ||\mu_\nabla(x)||$, where X_i is a local region around the input of the ith experiment. This

allows the penalisation to adapt to the smoothness of the area it is being evaluated in.

3.8 Experimental Results

In this section we present our empirical results. As is common in the Bayesian optimisation literature, we use log-regret as a measure of performance, which is defined as:

$$\text{log-regret} = \log\left[f(x_*) - \max_{i=1,\ldots,N_t} f(x_i)\right]$$

where x_i represents the ith experiment, and N_t is the number of experiments finished at time t. It measures how close our best observation is to the optimum of the function. We take the logarithm to make comparisons easier, since all benchmarks were quasi-normalised so that the outputs are small.

3.8.1 Synthetic Benchmarks

We focus on implementing the same benchmarks as Kandasamy et al. [2016a] (some of which are taken from Xiong et al. [2013]). More details about how each fidelity is created can be found in Appendix A.1. We compare against a simple single-query, single-fidelity UCB [Srinivas et al., 2010], against single-query multi-fidelity Gaussian Processes Upper Confidence Bound (MF-GP-UCB) [Kandasamy et al., 2016a]. We also compare against a single-fidelity PLAyBOOK [Alvi et al., 2019], which is simply the UCB acquisition function, in conjunction with the hard local penalisers introduced in Equation (3.16), and with local estimation of the Lipschitz constant. Finally, we also include results for TuRBO [Eriksson et al., 2019], whose details are explained in Section 3.8.2. For standard Bayesian optimisation methods we use UCB as our acquisition function, for TuRBO methods we use Thompson Sampling, and we use MF-MES to represent information-based methods. A summary of all benchmarks and their properties can be found in Table 3.3.

In the interest of giving a fairer comparison, all single-query methods are turned into batch

Method	Reference	Multi-fidelity	Batching	Model
UCB	Srinivas et al. [2010]	✘	✘	GP
MF-GP-UCB	Kandasamy et al. [2016a]	V	✘	IGPs
MF-GP-UCB w LP	This work	V	LP	IGPs
PLAyBOOK (UCB)	Alvi et al. [2019]	✘	LP	GP
UCB-V-LP	This work	V	LP	MOGP
UCB-I-LP	This work	I	LP	MOGP
TuRBO-TS	Eriksson et al. [2019]	✘	TS	GP
TuRBO-V-TS	This work	V	TS	MOGP
TuRBO-I-TS	This work	I	TS	MOGP
MF-MES	Takeno et al. [2020] Moss et al. [2021b]	I	Fantasies	MOGP

Table 3.3: Summary of all the methods compared in this section. We split the table into three sections: the first one refers to UCB-based methods, the second to the our modifications of TuRBO, and for the third we include multi-fidelity Max Entropy Search (MF-MES) as a general baseline. Variance-based and Information-based multi-fidelity refers to using Equations (3.20) and (3.21) (respectively) to choose which fidelity to query. In the multi-fidelity column, V stands for "Variance-based" and I stands for "Information-based". In the batching column, LP stands for "Local Penalisation", TS stands for "Thompson Sampling. In the Modelling Column, GP stands for standard single GP, IGPs stands for "Independent GPs", and MOGP stands for "Multi-output GP". Local penalisation is implemented as described in [Alvi et al., 2019], using hard local penalisers and a local estimate of the Lipschitz constant. Thompson Sampling is carried out across a fixed grid in TuRBO trust regions. In all cases, for TuRBO, we use a single trust region. For all non-batching methods, we fill the batch through random sampling as to avoid idle resources. We used GPyTorch and BoTorch to implement the methods [Paszke et al., 2019, Gardner et al., 2018, Balandat et al., 2020].

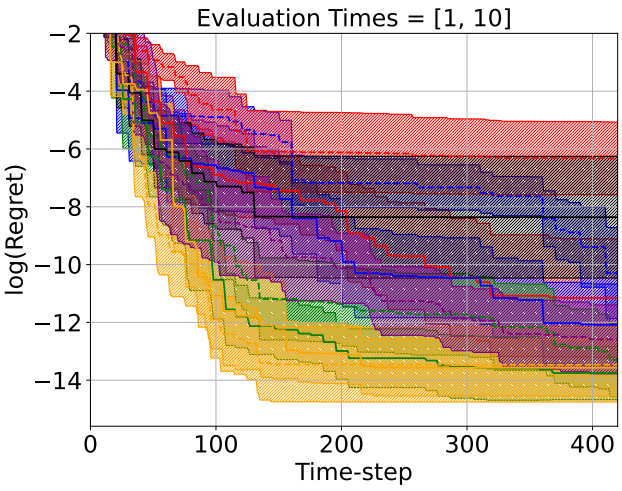
methods by selecting the first query using the acquisition function, and then filling the rest of the batch with random queries. We carry out the optimisation until there is enough budget to get 200 high fidelity observations. For all experiments we set the batch size to 4, and assume that the batch space of the low and high fidelities is the same. The results can be see in Figure 3.4, with the evaluation times for each fidelity shown in the title.

The Currin2D benchmark, is a simple function with a unique optima. Methods that use independent GPs for modelling perform badly. The strongest performers are the multi-fidelity versions of TuRBO (in yellow). In particular, we see that the best three performers (multi-fidelity TuRBO, and UCB-V-LP (in green)) achieve better regret (on average) after 150 time-steps than the rest of the methods in 400 – they seem to leverage the lower fidelity to explore the space quickly and obtain and they begin to exploit the optimum quickly.

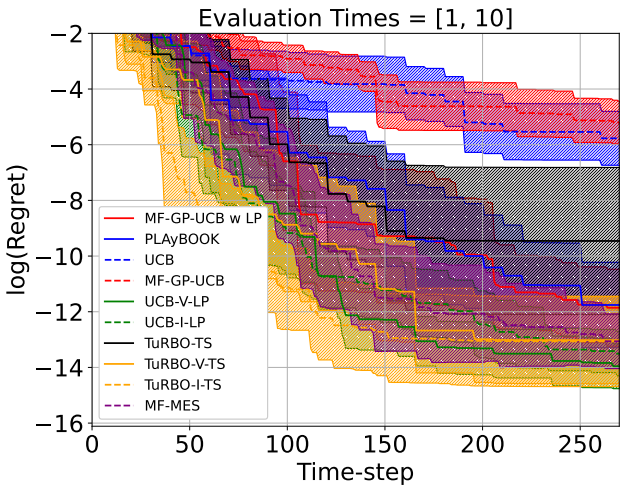
For the BadCurrin2D benchmark, the lower fidelity is simply the negative of the target fidelity, giving the worst possible approximation. However, methods using multi-output GPs (in yellow, green, purple) quickly learn the inverse correlation structure and leverage it to achieve basically identical results to the original benchmark. Methods using independent GPs (in red) on the other hand, have poor initial performance, but seem to recover well enough and have similar performance as in the original benchmark for larger budgets.

Hartmann3D is a smooth function with 4 local minima. TuRBO *without* multi-fidelity (in black) seems to get stuck in one of these minima, however once we introduce the ability to quickly explore using cheap approximations, the performance of the method increases significantly (in yellow) since sub-optimal areas are discarded without centering the trust region in a local optima. We still observe larger uncertainties than for most methods, suggesting that getting stuck in local optima is still a possibility. On the other hand, simple upper confidence bound methods with multi-output GPs (in green) seem to perform better and provide better reliability due to the ability to explore, even after a local optima has been found.

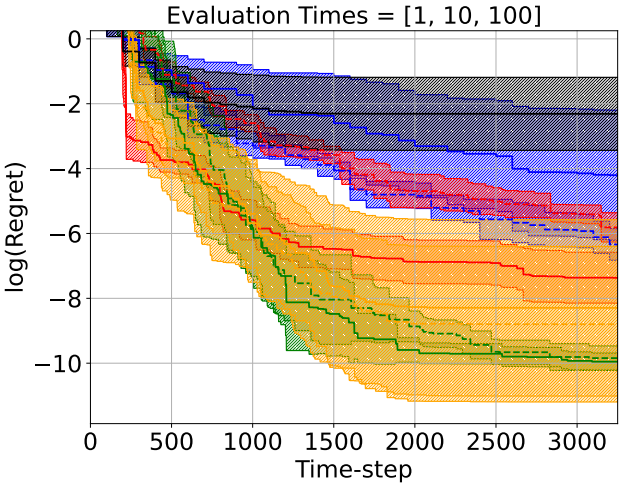
In Hartmann6D we have another smooth function but with 6 local minima. This time multi-fidelity TuRBO methods (in yellow) perform much stronger than every other method. They achieve significantly lower regret and with very small uncertainty bounds. We conjecture this



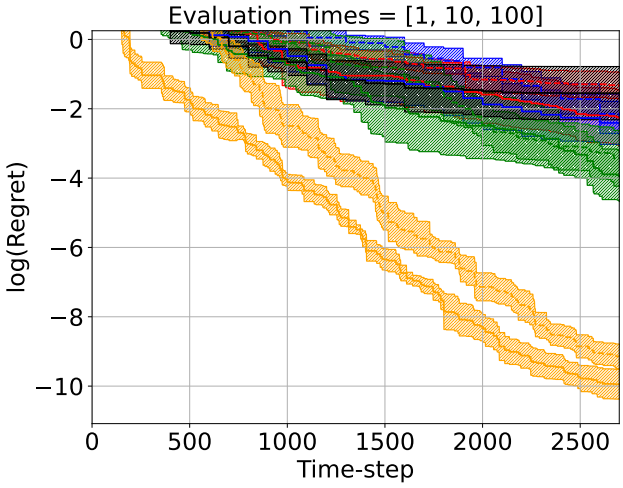
(a) Currin2D



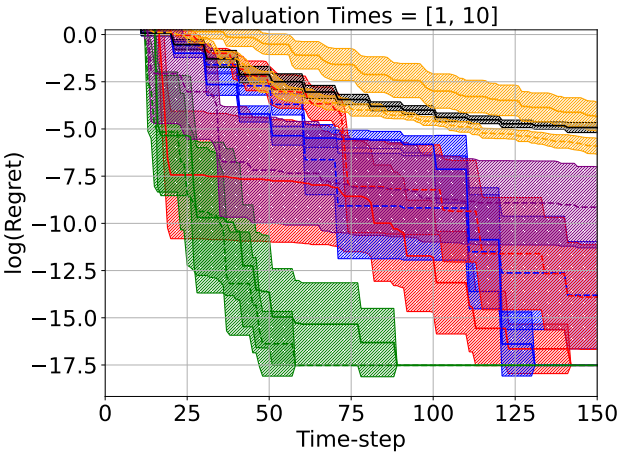
(b) BadCurrin2D



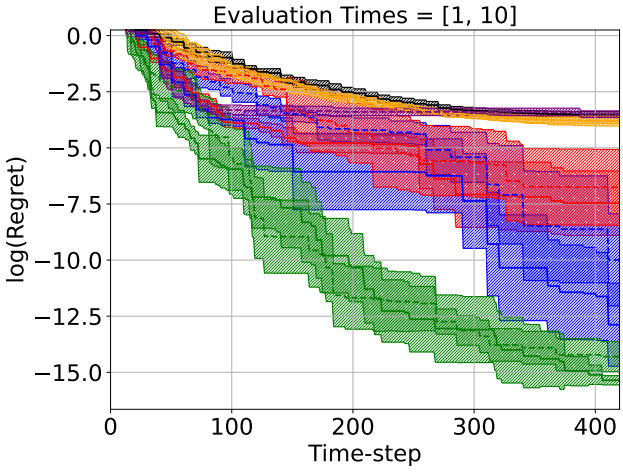
(c) Hartmann3D



(d) Hartmann6D



(e) Park4D



(f) Borehole8D

Figure 3.4: Results of synthetic experiments. We see that simply using UCB with our algorithm consistently achieves good results, always being one of the top performs. TuRBO also tends to perform fairly well, especially strongly in Hartmann6D, however, it also struggles badly in some benchmarks (Park4D and Borehole8D). MF-MES is not present in the Hartmann experiments due to the computational expense of testing the function when there are three fidelities.

is because the *initial* trust region is always large enough to contain the global optima (the size of the region depends on the initial estimated length-scales), and using the multi-fidelity approximations, the trust region can quickly center around the global optima. We further note that our modifications of UCB (in green), while far behind the multi-fidelity versions of TuRBO, they still outperform all the other benchmarks.

For Park4D and Borehole8D we obtain very similar results. In both benchmarks it seems that all versions of TuRBO (in yellow and black) perform very poorly. We conjecture this happens because the trust regions are initialised too small, leading to model misspecification (as all the samples lie in a small region), and therefore the algorithm getting stuck in local optima every time. On the other hand, in both cases we see very strong performance from our modified version of the UCB function (in green). The functions are simple enough, and the dimensionality low enough, so that exploration can be done quickly and effectively using the lower fidelity. The advantage of using multi-output GPs to transfer fidelity information is clear, as the independent GPs (in red) have significantly worse performance in both benchmarks.

Overall, we see very strong performance from at least one of the methods introduced in this work in each benchmark. Particularly strong are the methods that use multi-output GPs. We see little to no difference in performance when using the variance based or when using the information based criteria for choosing which fidelity to query. However, we recall that the information based method has fewer hyper-parameters, and note particularly strong performance in Hartmann6D.

3.8.2 High-dimensional optimisation

To highlight the usefulness of our proposed method, we show how we can use a specialist function to do high-dimensional multi-fidelity Bayesian optimisation. Bayesian optimisation traditionally struggles in high-dimensions where the size of the search spaces means the exploratory phase of classical algorithms is given too much importance. Eriksson et al. [2019] proposed TuRBO which focuses on local optimisation and tends to perform very well in high-dimensional settings. TuRBO uses Thompson Sampling to select the next experimental design,

however, the experimental region is restricted to a trust region. The trust region is defined as a hyper-rectangle centered around the best experiment we have observed so far. If there are a lot of consecutive improvements (i.e. we consecutively obtain observations better than anything we had observed before), then we double the size of the region (i.e. we increase exploration), if there are too many consecutive failures, we shrink the size of the region (i.e. we increase exploitation). We use BoTorch's [Balandat et al., 2020] implementation of TuRBO with the default hyper-parameters – for changing the size of the region we only consider evaluation at the highest fidelity.

Trust region approaches often work well on high-dimensional chemical engineering applications Bajaj et al. [2018], Kazi et al. [2021, 2022], since they avoid the problem of over-exploration in high-dimensional spaces. We show, using the Ackley 40D benchmark, that we can use this acquisition function in our algorithm to obtain much better performance than other benchmarks. For this case we set the batch size to 20, and again assume that the batch space of low and high fidelities is the same. We set the budget as to allow 500 high-fidelity observations. The results are shown in Figure 3.5.

After less than 50 time-steps multi-fidelity TuRBO (in yellow) achieves better regret than PLAyBOOK (non-dashed blue), normal UCB (dashed blue) and MF-GP-UCB (both in red) achieve in the whole optimisation, and similar performance to UCB-V-LP (green) and normal TuRBO (black). MF-MES also achieves very strong performance, however, it is not able to improve at the same rate that TuRBO does. All these results are expected, as the methods will over-explore in the high-dimensional space and never be able to exploit. TuRBO on the other hand, after a brief exploration phase, will shrink the trust region to something manageable and exploit locally, achieving much better regret. In addition we observe relatively small uncertainty regions, this due to the Ackley function being symmetric, and the high dimensionality of the problem making the starting points less likely to be relevant. The difference between the black and yellow lines gives strong evidence of the benefits of using multi-fidelity methods. We note, again, that there is little difference between the variance and information based multi-fidelity criteria (see both yellow lines).

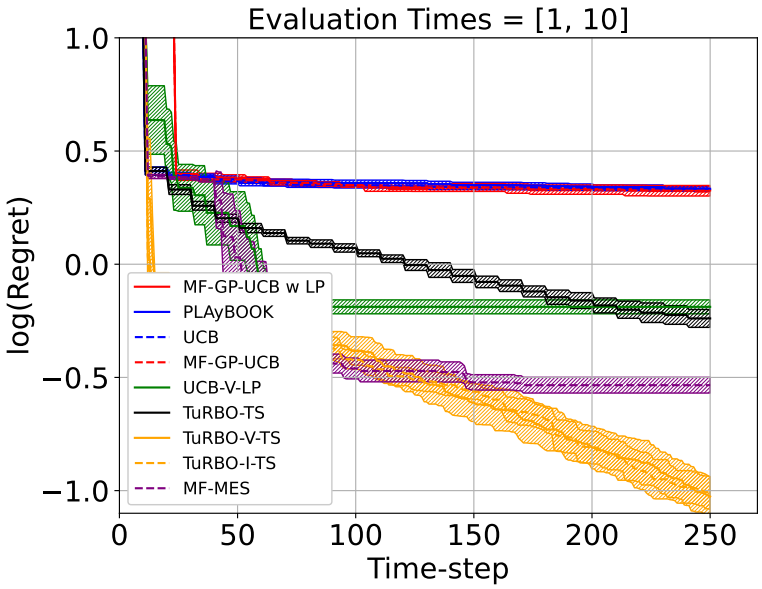


Figure 3.5: Results on Ackley 40D function. We can see that TuRBO is constantly improving, and outperforms other functions considerably. MF-MES and UCB-V-LP have good initial performance but begin to struggle after due to the over-exploration of classical Bayesian optimisation.

3.8.3 Battery Material Design

We explore the application to the automatic design of battery materials. In this particular example, we aim to learn the best ratios of dopants in high-nickel cathode materials such that the specific capacity (electrochemically accessible charge per unit mass) is maximised. Our target is to find materials that have a high capacity over many charge-discharge cycles in pouch cell batteries. However, designing, building and testing such batteries requires many resources and long experimental cycles that take multiple months to process. As such, it is common for experiments to be carried out in parallel, where the batch size ranges between 10 and 20 concurrent queries, or experiments.

It is possible to obtain an approximation of pouch cell performance by considering coin cell batteries, where the materials are put into much smaller cylindrical containers requiring less material. In addition, we can reduce the duration of the experimental tests, perhaps by increasing the cycling rates or performing fewer charge-discharge cycles in total. Combining coin cell tests with accelerated test procedures allows us to get data in only a few weeks, and to carry out more experiments due to the lower material costs – however some of the real problem's

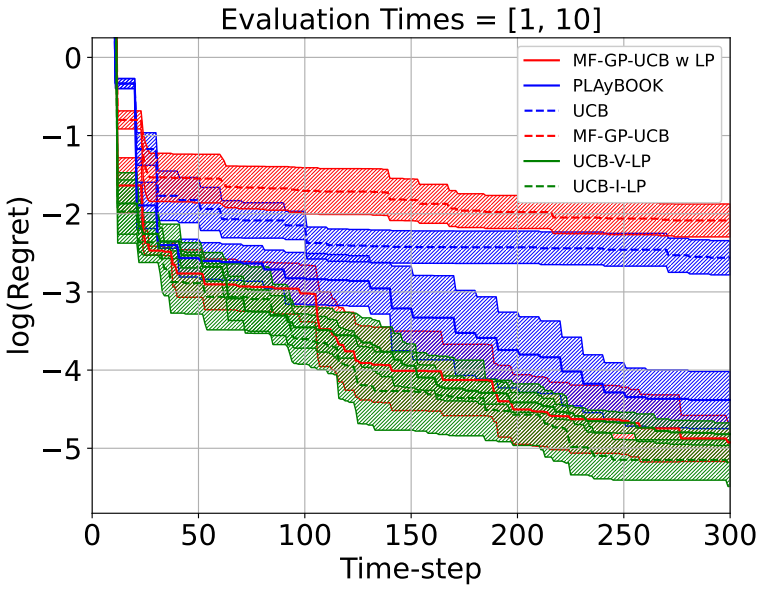


Figure 3.6: Results on battery design experiment. We can see that the asynchronous batching and multi-fidelity methods are significantly outperforming simple baselines (UCB), multi-fidelity without batching (MF-GP-UCB), and single-fidelity methods (PLAyBOOK). The performance of all batching multi-fidelity methods is similarly strong.

mechanisms may be suppressed. This represents the lower fidelity that we are interested in using.

For this example, we will be using an anonymised real-world data-set provided by BASF SE (the dataset, alongside the code used in the experiment section, can be found at `https://github.com/jpfolch/MFBoom`[1]). We fit a Gaussian Process model to the data and using the method described in Wilson et al. [2020], we take a single sample to create our objective, $\tilde{f}^{(0)}$. We also took a second sample, $\tilde{f}^{(1)}$, to create the lower fidelity, meaning the function that we query is defined as:

$$f^{(m)}(x) = \begin{cases} \tilde{f}^{(0)}(x) & \text{if } m = 0 \\ \tilde{f}^{(1)}(x) & \text{if } m = 1. \end{cases} \tag{3.23}$$

We also face a combinatorial constraint and an equality constraint. The benchmark is 6-dimensional, but we assume only three dimensions are allowed to be active at every moment, i.e.:

$$x_i = x_j = x_k = 0 \quad \text{for } i \neq j \neq k \tag{3.24}$$

[1] the acronym was chosen in homage of MF DOOM, the British-American rapper, who inspired many hip-hop classics for example Dogg et al. [2015]

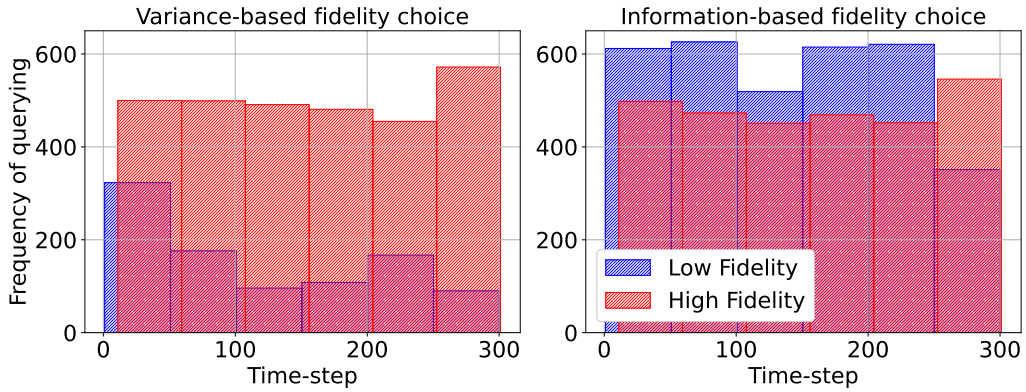


Figure 3.7: Experimental querying frequency comparison. The histogram above shows how often we queried the low and high fidelities for different time-intervals. We compare the querying frequency for the methods using UCB and Local Penalisation, with different ways of choosing the fidelity – variance-based (left) against information-based (right). The information-based fidelity criteria chooses to query the lower-fidelity more frequently, which suggests it could be the reason why it slightly outperforms the variance-based criteria.

where $i, j, k \in \{1, 2, ..., 6\}$. We further have the equality constraint:

$$x_{i_a} + x_{j_a} + x_{k_a} = 1 \tag{3.25}$$

where i_a, j_a, k_a represent the *active* problem dimensions. To optimise with these constraints, we create a 2-dimensional Sobol grid, and filter out all points such that $x_0 + x_1 > 1$. We then calculate $x_2 = 1 - x_1 - x_0$. Finally, we expand the grid to all 20 possible combinations of active variables.

Due to these constraints, we do not consider TuRBO (trust region and intersection of feasible region might be too small) or MF-MES (grid size is too large to optimise the acquisition function comfortably) on the Battery benchmark.

Real battery materials laboratories vary in the degree of automation and the achievable experimental throughput. Here we assume we have a maximum batch budget of 20 and consider differing batch spaces. We assume that for each coin cell battery, or low fidelity, querying costs 1 unit, while the pouch cell battery, or high fidelity, costs 2 units. This means we can concurrently do 20 coin cell experiments, or 10 pouch cell experiments, or a mix in between. We further assume that pouch cell experiments take 10 times longer than coin cell experiments, and set the time-budget to 300 (equivalent to being able to carry out 300 high-fidelity experiments).

This is due to the combination of an accelerated test procedure with coin cell testing rather than an inherent speed-up associated with coin cells. The results are shown on Figure 3.6.

We can see that the benefit to using the lower-fidelity with batching are substantial, as both methods that use both batching and multi-fidelity (green and non-dashed red) are significantly outperforming simple baselines, single-fidelity methods, and non-batching counterparts. For this specific example, we notice that using independent Gaussian Processes works surprisingly well, suggesting that the bias assumption is well suited to this benchmark.

We further include a histogram, that shows how often we queried the low fidelity against the high fidelity. We compare UCB-V-LP (variance-based fidelity choice) against UCB-I-LP (information-based fidelity choice). The comparison can be see in Figure 3.7. In both cases, we can see that towards the end of the optimisation, there is decrease in low-fidelity querying, and an increase in high fidelity querying. However, information-based fidelity choice queries far more at the lowest fidelity – and seems to get marginally better results. This highlights the advantage of the information-based method being parameter-free, as the variance-based requires better tuning of the UCB thresholds to use the lower-fidelity at its full capacity.

3.9 Discussion

In this chapter we have explored methods in multi-fidelity and batch Bayesian optimisation, explaining why there is a close link between them and providing a comprehensive overview of the current state-of-the-art methods. We further show how we can leverage recent advances and propose an algorithm that allows practitioners to use acquisition functions well suited for their particular problem. We then empirically show the benefits of using such methods, illustrating their usefulness in both synthetic benchmarks, and real-life inspired problems.

Being able to choose the right acquisition function is of great importance in Bayesian optimisation. How to choose the correct acquisition function remains an open problem, and is far too broad a subject to tackle in this chapter. However, it is clear that having flexibility in the

choice of acquisition function is important. As an example, as shown in Figure 3.4d, the choice of TuRBO as an acquisition function leads to much better results than the others.

In some cases there can be clear advantages to using an acquisition function that is known to work well. As an example, consider the high dimensional optimisation problem we showcase in section 3.8.2, where TuRBO is known to perform well. However, no single acquisition function will perform optimally in every situation. Indeed, TuRBO performs very poorly in Park4D and Borehole8D where the single trust region gets stuck in a local optima. This highlights a second important benefit of being able to use any acquisition function: sometimes we should prefer simple solutions.

Complex methods such as TuRBO can provide strong state-of-the-art performance in many benchmarks, but generalise poorly unless their parameters are carefully tuned. The size and location of the initial trust region, how frequently to shrink or enlarge it, or the number of trust regions are examples of unintuitive choices that we must make when using the method. In contrast, once we have selected methods that fit the problem parameters (in this case multi-fidelity and batching), many practitioners and experimentalists will prefer to use a simple acquisition function as it is more interpretable, familiar, and choosing the hyper-parameters tends to be easier. Experimentally, we showed that using the UCB and local penalisation (green lines in Figures 3.4, 3.5, and 3.6) always achieves a top four performance, regardless of the benchmark – following Occam's Razor, we may choose the simplest model that fits the problem parameters.

With this work we hope to give readers a solid understanding of how to improve optimisation procedures when you can parallelise experiments and obtain cheap approximations. We hope this will allow for more widespread use of such methods, because they exist and we show they can be incredibly useful. Indeed, it may be the case that many optimisation procedures can benefit from multi-fidelity approximations but they have never been considered, or they have been thought of as something to do instead of parallelisation, instead of doing them concurrently. We also hope that by making the algorithm flexible, it will appeal to a wide range of practitioners and problems.

3.10 Looking back...

This work was carried out in two separate time periods, as my first PhD project in late 2020, and it was then revisited and finished in 2022. This meant that the ideas do feel a little disjointed, as I originally focused on independent Gaussian Processes, before discovering multi-output GPs were a much better tool for the task. Looking back, I would have liked to focus on the latter class of models more intently, for example, by including latent variable GPs in the comparison.

In addition, the disjointness meant that I did not read about GIBBON [Moss et al., 2021a] until after the manuscript was finished. I would have liked to have included it in the baseline comparisons and tested more thoroughly as to when each method is preferable. I think more thorough testing and including more real-life benchmarks would have made the work even better.

Also important to mention, that we have recently added an implementation of our method in BoFire [Dürholt et al., 2024] [`https://github.com/experimental-design/bofire`] which can be used for more complete benchmarking.

Chapter 4

SnAKe: Bayesian Optimisation with Pathwise Exploration

This chapter is associated with the publications Folch et al. [2022] and Folch et al. [2023b].

4.1 Preliminaries

Classically, black-box optimisation finds an optimum by sequentially querying the function. In this chapter we will study a variation of this problem, with two important differences. First, we introduce the idea that large changes in inputs, between iterations, cause the function to become more expensive to evaluate. Second, we do not assume observations are available immediately: delay between querying the function and getting a result leads to asynchronous decision making.

As a motivating example, consider a droplet microfluidic reactor [Teh et al., 2008] (see Figure 4.1). In such a reactor, we can quickly pump in chemicals, expose them to certain conditions, and collect the results of our experiments as they exit. However, large changes in temperature mean that the reactor is no longer in steady-state, and this makes the evaluations unreliable until the system stabilises. Smaller changes can maintain a "quasi-steady state", as the system

is easier to stabilise. Further, we have to wait for droplets to exit the reactor before obtaining observations.

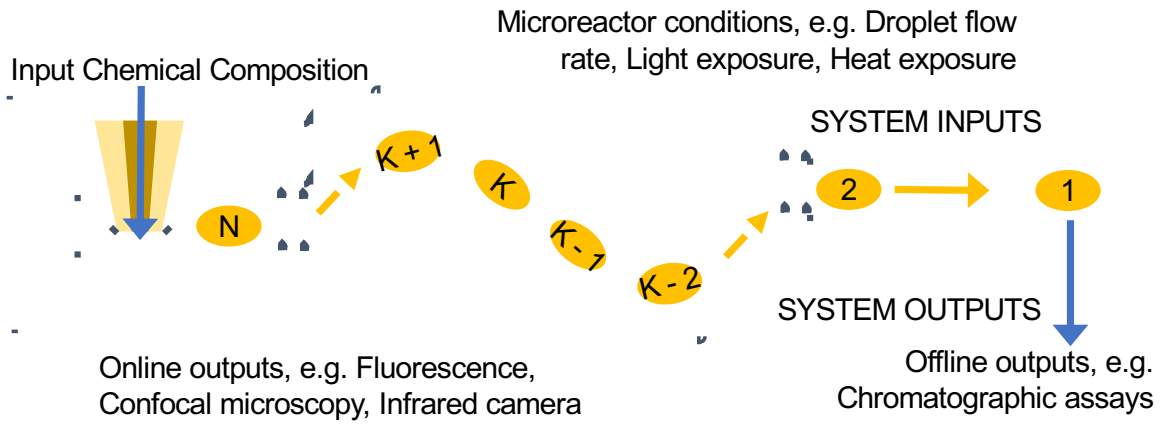


Figure 4.1: Motivating example. Droplets flow into the micro-reactor where we control conditions such as temperature and flow rate. The cost to change function inputs arises from how adjacent droplets are coupled, e.g. rapidly changing the temperature after droplet K means waiting for system equilibration before taking new measurements. Asynchronicity arises from choosing drops $2, 3, ..., N$ before getting the results of droplet 1.

More broadly, consider optimisation requiring spatially continuous exploration. Gaussian processes have shown promise in the context of modelling air pollution [Pinder et al., 2022, Stoddart et al., 2023]. Machine learning methods for detecting the sources of pollution have been developed [Leu and Ho, 2019] and more recently using black-box optimisation for the optimal placement of pollution sensors [Hellan et al., 2021]. Mobile or adaptive pollution readers, such as those used by AirQo [`https://www.airqo.net/about`], would face similar challenges if black-box optimisers were used to track the pollution sources [Grant et al., 2019]. Samaniego et al. [2021] show a more concrete example where autonomous vehicles use Bayesian optimisation to monitor and track water contamination sources in the Ypacarai lake in Paraguay.

Classical Bayesian optimisation [Jones et al., 1998, Shahriari et al., 2016] provides effective solutions to black-box optimisation; however, physics-based limitations are usually not taken into account [Thebelt et al., 2022c]. Typically Bayesian optimisation follows a myopic approach, in that the algorithm chooses the next query based only on the current state of the surrogate model. Classical Bayesian optimisation reduces uncertainty in unexplored areas and then returns to promising areas with no regard for the distance between consecutive query points. This means that the algorithmic procedure will incur very high *input costs*.

However, having zero changes in input space is obviously not a good solution. After all we want to explore the search space to find the optimum point. We seek an algorithm that preserves the essence of Bayesian optimisation. Consider a scenario where we know a large number of inputs we want to query. In this case, we could simply order the queries to attain the smallest input cost. A good solution would simply require selecting a large number of queries, creating an ordering, and then following the *path* defined by the ordering. Once we start obtaining new information, we could update our beliefs and update our optimisation path.

In this chapter, we propose ***S**eque**n**tial Bayesian optimisation via **A**daptive (**K**)Connecting Sampl**e**s*. We dub our algorithm SnAKe, joining other reptile named algorithms such as VIPER [Komodromos et al., 2024]. Just as the snake grows by carefully eating items in the classic arcade game, a SnAKe optimisation path grows from carefully adding queries to the evaluation path.

4.1.1 Problem Set-up

We consider finding the maximum, $x^* = \arg\max_{x \in \mathcal{X}} f(x)$, of a black-box function, f, where $\mathcal{X}$ is a compact subset of $\mathbb{R}^d$. We evaluate the function sequentially, over a discrete and finite number of samples, $t = 1, ..., T$. For every query, x_t, we follow the common theme of the thesis of obtaining observations of the objective with Gaussian noise, $y_t = f(x_t) + \epsilon_t$.

Focusing on the asynchronicity introduced in Chapter 3, we assume there is delay, t_{delay}, between choosing a query and getting an observation. So our data-set at iteration t is given by $\mathcal{D}_t = \{(x_i, y_i) : i = 1, ..., t - t_{delay} - 1\}$. If we set $t_{delay} = 0$, we revert to classical sequential Bayesian optimisation, otherwise we are in an *asynchronous* setting.

Finally, we assume there is a *known* cost to changing the inputs to our evaluation, $\mathcal{C}(x_t, x_{t+1})$. We use simple regret, $SR_t = f(x^*) - \max_{i=1,...,t} f(x_i)$ as the performance metric. We want to minimise regret, or equivalently, maximise f, for the smallest possible cumulative cost, $\sum_{t=1}^{T-1} \mathcal{C}(x_t, x_{t+1})$.

We note that the problem definition is invariant to cost scaling relative to the objective, because

an optimal trajectory will be defined solely by the shape of the function. Current cost-aware methods are not scale invariant, and as we shall see, SnAKe first chooses which points to query and then finds the most cost-efficient way of selecting them, making the method invariant to scaling of the cost.

4.2 Related Work

'Process-constrained Batch Bayesian optimisation' [Vellanki et al., 2017], is the closest analog to our setting of Bayesian optimisation with input costs. Vellanki et al. [2017] navigate the physical limitations in changing the input space by *fixing* the complicated inputs for every batch. Rather than fixing the difficult-to-change inputs, we *penalise* large input variations in line with the costs (time and resources) of changing conditions. SnAKe decides when to make expensive input changes. Recently, Ramesh et al. [2022] consider a similar cost setting to ours, inspired by applications to wind energy systems. Waldron et al. [2019] compare the use of transient variable ramps (small input changes in a pre-determined manner) against a full steady-state design of experiments approach for learning parameters of chemical kinetic models. They show that the transient approaches give less precise estimates, but much faster. SnAKe combines the best of both worlds, automatically designing experiments while keeping input changes small.

'Cost-aware Bayesian optimisation' [Snoek et al., 2012, Lee et al., 2020, Luong et al., 2021] optimises regret with respect to cost per iteration. These costs, which are fixed throughout the Bayesian optimisation process, arise when some regions of the input domain are more expensive to evaluate than others. Similar ideas appear in Multi-fidelity Bayesian optimisation [Kandasamy et al., 2016a, Poloczek et al., 2017] where cheap approximations for the objective function are used. Bogunovic et al. [2016] combines cost-aware and level-set estimation approaches to solve environmental monitoring problems. Diverging from prior works, our setting does not assume that different regions incur different costs. Instead, SnAKe addresses the setting where the *difference* between adjacent inputs defines costs. The query order changes the

cost, therefore we focus on optimisation paths instead of optimisation sets.

The baseline of Cost-aware Bayesian optimisation, Expected Improvement per unit Cost (EIpu), defined as $\mathrm{EI}(x)/\mathrm{Cost}(x)$, is not directly applicable to our setting because the cost of not changing inputs, i.e., choosing $x_t = x_{t-1}$, is zero. We can instead use:

$$\gamma\mathrm{EIpu}(x) = \mathrm{EI}(x)/(\gamma + \mathrm{Cost}(x, x_{t-1})) \tag{4.1}$$

which gives us three main issues: (a) Eq. (4.1) introduces new hyper-parameter γ with no obvious way to choose it, (b) Eq. (4.1) effectively penalises the acquisition function far away from the data, so near local optima we only penalise exploration and may over-exploit, (c) asynchronous extensions have a tendency to encourage batch diversity, so we introduce a new trade-off between penalising locally and far away. As we shall see, SnAKe shows robust results with just one hyper-parameter.

Look ahead Bayesian optimisation [Ginsbourger and Riche, 2010, Lam et al., 2016, González et al., 2016b] considers possible future queries to make the current choice less myopic. We could use it to select the inputs to query, as we could simply look-ahead at what any classical Bayesian optimisation would choose and then order the queries accordingly. However, one is only able to look ahead for a few iterations, due to the computational complexity of looking far ahead into the future [Bertsekas, 2012, Yue and Kontar, 2020]. For example, Lee et al. [2021] combines Markov Decision Processes [Puterman, 2014], look-ahead and cost-aware ideas, but is limited by the short time horizons rollout can handle. Reinforcement Learning [Sutton and Barto, 2018] can be used, but it requires access to a dedicated training environment. Mutný et al. [2023] show how similar ideas can be used for trajectory optimisation in environmental monitoring.

A computationally cheaper alternative that allows us to select many inputs to query is Batch Bayesian optimisation (BBO) [González et al., 2016a, Azimi et al., 2010]. BBO is the setting where we are able to parallelise function evaluations, and as such we want to select multiple queries simultaneously. González et al. [2016b] and Jiang et al. [2020a] link BBO with look-

ahead Bayesian optimisation, using a Local Penalization method [González et al., 2016a] and Expected Improvement (q-EI) [Ginsbourger et al., 2010], respectively. Unfortunately, they restrict themselves to smaller batch sizes ($q \leq 15$) due to computational expense. Beyond the use of planning, Savage and del Rio Chanona [2024] applied batch Bayesian optimisation to offer humans alternative querying options, incorporating expert knowledge in the optimisation loop. *Asynchronous* Bayesian optimisation [Kandasamy et al., 2018, Alvi et al., 2019] addresses the problem of choosing queries while waiting for delayed observations.

4.3 An ordering-based algorithm

4.3.1 General Approach

For our general approach, we will seek to create a large *batch* of queries that we want to evaluate, and then *plan-ahead* a whole optimisation path. This is useful for three reasons: (1) it allows us to order the queries in a way that reduces input cost. (2) It allows us to deal with any delay in getting observations, because we can pre-select future queries. (3) It remains computationally feasible to plan-ahead even for very large time horizons. We can follow this ordering or path until new information is available, after which we will update our path.

While we will provide a specific way of creating, planning and updating the paths, we note that the general ideas can be extended to suit different settings. We detail three different examples in Section 4.7. We detail the general procedure in Algorithm 4.

4.3.2 Creating a Batch Through Thompson Sampling

We need to produce batches that are representative of the current state of the surrogate model. In addition, the method should allow for big batch sizes, since we want to produce batches as big as our budget (which is usually much larger than the batch size most methods consider). For example, in a micro-reactor, we might be interested in batches that contain hundreds of

Algorithm 4 General Ordering-Based Optimisation

input: Optimisation budget of T samples. Method for creating batch of queries. Method for creating an ordering from a batch of queries. Method for updating paths.
begin: Create initial batch of size T and ordering, S
for $t = 1, 2, 3, ..., T$ **do**
  if there is new information **then**
    Update surrogate model
    Choose a batch of new points to query
    Create a new path, $\tilde{S}$
    $S \leftarrow \tilde{S}$
  Choose next query point from ordering: $x_t \leftarrow S_t$
  Evaluate $f(x_t)$

points [Teh et al., 2008].

Kandasamy et al. [2018] offers a promising solution where every point in the batch is independent. The method is based on Thompson Sampling, which uses the GP's inherent randomness to create a batch. Each batch point is chosen by drawing a realization of the GP, and optimising it, as per Eq. (2.12). The queries will fill out the space, and they are more likely to be on promising, and unexplored areas. It should work very well in our context given we expect our initial batch sizes to be very large, so the sample should be representative of the current state of our surrogate model.

4.3.3 Creating a Path via the Travelling Salesman Problem

After selecting a batch of queries, $\mathcal{P}_t = \{x_t^{(i)}\}_{i=1}^{\tilde{T}}$, we order them. We do this by embedding a graph into the batch, where the edge weights are the cost for changing one input to another. We then find the shortest path that visits every point, i.e., we solve the Travelling Salesman Problem (TSP) [Bellman, 1962, Dorigo and Gambardella, 1997]. Section 4.6 discusses the computational cost.

Mathematically, we define the graph $G = (V,\ E,\ W)$, with $V = \{i \in 1, ..., \tilde{T} : x_t^{(i)} \in \mathcal{P}_t\}$, $E = \{(i, j) : i, j \in 1, ..., \tilde{T}\}$, and $W = \{w_{ij} = \mathcal{C}(x_t^{(i)}, x_t^{(j)}) : (i, j) \in E\}$, where $\tilde{T}$ is the number of batch samples. We solve the TSP in G to obtain our latest optimisation path. A simple example would be to try to minimise the total distance travelled in input space, by selecting

the Euclidean norm as cost $\mathcal{C}(x_t, x_{t+1}) = ||x_t - x_{t+1}||$.

4.3.4 Naively Updating the optimisation path

After updating the GP with new observations, we want to use this information to update our path. We propose updating our strategy by creating a new batch of points. At iteration t, the remaining budget has size $T - t$. We first propose sampling $T - t$ queries through Thompson Sampling, and then solving the Travelling Salesman Problem. However, we show this leads to the algorithm getting 'stuck' in local optima. This is because every time we re-sample, we naturally include *some exploitation* in the batch, and this exploitation will always be the next point chosen by the TSP–we will never reach the *exploration* algorithm steps. Figure 4.2a shows an example.

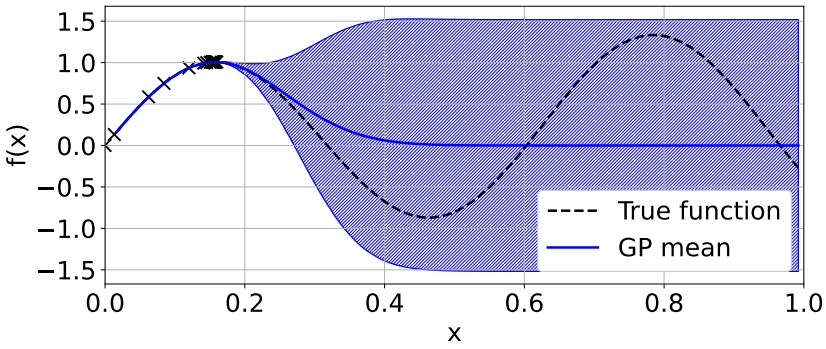
4.4 Escape Analysis

To try and solve the convergence problem introduced by naive resampling, we briefly analyse it. This analysis assumes we receive noise-less observations. However, all sampling can be done in the presence of noise by calculating the corresponding posterior. We note that this section will provide intuition for the next step of the algorithm, however, we are not providing rigorous theoretical justification for the step or proving any type of regret bounds. We focus on Thompson Sampling.

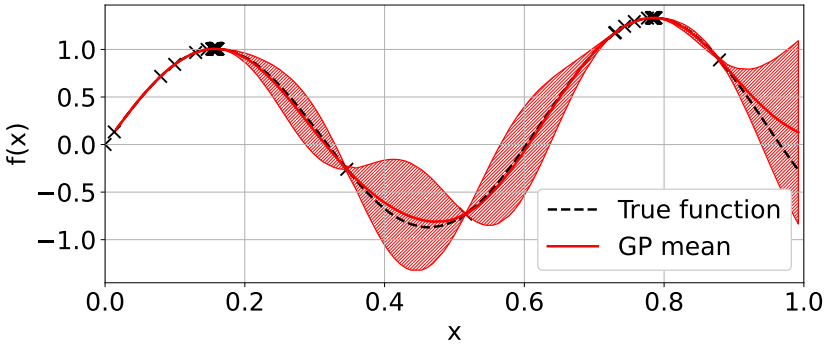
Definition 4.1. *(Thompson Sample) We say $x_t^{(i)}$ is Thompson Sampled, and we write $x_t^{(i)} \sim \tau_t$, if:*

$$x_t^{(i)} = \arg\max_{x \in \mathcal{X}} f_t^{(i)}(x), \quad \textit{where} \quad f_t^{(i)}(\cdot) \sim \mathcal{GP}(\mu_t,\ \kappa_t | \mathcal{D}_t).$$

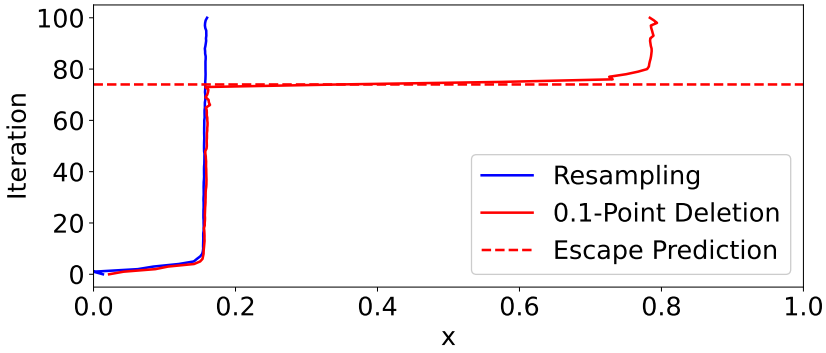
Note that the sampling distribution $\mathcal{GP}(\mu_t,\ \kappa_t|\mathcal{D}_t)$ changes at each iteration.



(a) Naively Resampling



(b) With Point Deletion



(c) optimisation paths.

Figure 4.2: Effect of Point Deletion for ordering-based Bayesian optimisation. (a) and (b) show the maximization objective and the underlying surrogate model, with black crosses representing queries and observations. (c) shows the optimisation paths. Naive resampling gets stuck in the local optimum ($x = 0.16$) whereas Point Deletion escapes the local solution. We also show the predicted escape iteration, Tp (after estimating $p \approx 0.74$ in Figure 4.4), which accurately predicts the algorithm behavior in this simple example.

One particular concern, diagrammed in Figure 4.2a, is the possibility of the method being stuck in a certain area. Let $B_\delta(a)$ be a Euclidean δ-ball centered at a, and assume that $x_{t-1} \in B_\delta(a)$. We further assume two more things: (a) we resample the batch at every iteration, and (b) the TSP solution, or any approximation, will always choose a point in $B_\delta(a)$ to be the next step, if

there are any. These assumptions represent the worst-case scenario for trying to escape a local optima.

Definition 4.2. *(Non-escape Probability) We define the non-escape probability, p_t, at iteration t, as the probability of a Thompson Sample falling into $B_\delta(a)$. That is, for $x_t^{(i)} \sim \tau_t$:*

$$p_t = \mathbb{P}(x_t^{(i)} \in B_\delta(a)) = \mathbb{P}(||x_t^{(i)} - a|| \leq \delta). \tag{4.2}$$

Let $\mathcal{P}_t = (x_t^{(1)}, ..., x_t^{(T-t)}) \sim \tau_t$ *i.i.d.*. Of particular interest to us, is the number of 'non-escapes' in the sample, $N_t = |\mathcal{P}_t \cap B_\delta(a)|$. We are guaranteed to escape if no sample falls in $B_\delta(a)$, so we say we have *fully* escaped if $x_t^{(i)} \notin B_\delta(a) \ \ \forall i$.

Remark 4.1. *Note that $N_t \sim$ Binomial$(T-t,\ p_t)$ as all the samples are mutually independent, therefore the probability of fully escaping is $\mathbb{P}(N_t = 0) = (1 - p_t)^{T-t}$. This means we can only expect to fully escape if p_t is very small.*

Therefore we are interested in the behavior of p_t as we gain more information about f in $B_\delta(a)$. We now consider the circumstances under which p_t becomes very small:

4.4.1 Areas without stationary points

Consider the case when our objective f does not contain a stationary point in $B_\delta(a)$. Then the maximum of f on the closure of the ball, $\bar{B}_\delta(a)$, must lie on the boundary of the ball. In particular, if we assume that our GP model has no error in $B_\delta(a)$, and assuming continuity of sample paths, then the non-escape probability must be zero, $p_t = 0$.

Intuitively, the area itself contains enough information to ensure, with complete certainty, that the global optimum does not lie in the area. We hope that, as we collect information inside areas without stationary points, $p_t \to 0$, and we will eventually leave them with small probability of returning. In Figure 4.3 we estimate the non-escape probability and show an example of quick convergence to zero in a case with no stationary points.

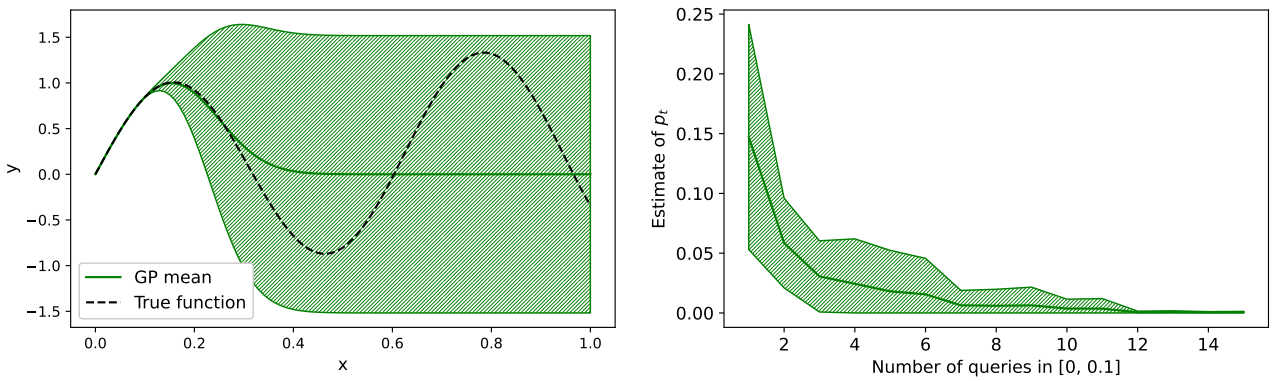


Figure 4.3: We estimate the probability of non-escape by taking 5000 independent Thompson Samples and counting the number of samples inside A (i.e. the MLE estimator of the Bernoulli distribution). We do this for increasing number of training points in A (which are chosen randomly with a uniform distribution in A). We repeat the experiment 10 times. The left plot shows the underlying function and the Gaussian Process for 15 training points. The right plot shows the evolution of our estimate as we increase training points inside $A = [0, 0.1]$ (we plot the mean of each run $\pm$ the standard deviation). We can see that p_t quickly converges to (almost) zero. We almost guaranteed to fully escape the area after 15 time-steps, even for very large budgets.

4.4.2 Areas with stationary points

Areas with stationary points pose a much bigger problem. We will restrict our arguments to local maxima, since this is where we have observed the problem. Assume that f has a local optimum in $B_\delta(a)$, higher than any other we have observed before. In this case, any sample taken from a Gaussian Process with no error in $B_\delta(a)$ will have a local maximum inside $B_\delta(a)$, and therefore it is possible that this local maximum is the global solution.

As we increase the information inside the area, p_t is not guaranteed go down to zero. This makes intuitive sense; the only way of knowing if a local optimum is not a global optimum is by sampling away from it–therefore with limited information we will allocate a certain probability to the global optimum being inside $B_\delta(a)$. We include a clear example where $p_t \to p > 0$ in Figure 4.4.

Sampling consistently in a promising area is not necessarily a bad thing, indeed we want to exploit near possible global optimum candidates. However, the question then becomes, how long will it take us to leave a local optimum? Recall the probability of fully escaping is $(1 - p_t)^{T-t}$, and therefore it will be increasing as t increases, even if p_t is (almost) constant.

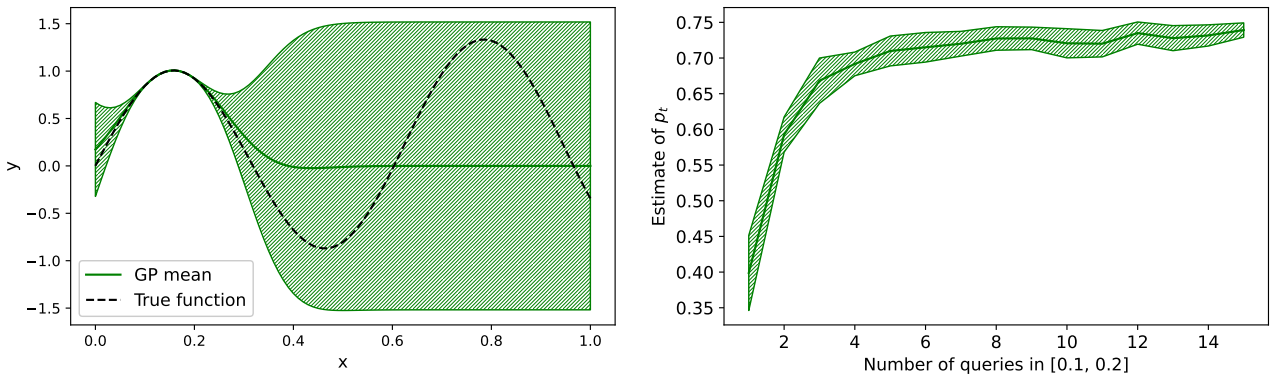


Figure 4.4: We estimate the probability of non-escape by taking 5000 independent Thompson Samples and counting the number of samples inside A (i.e. the MLE estimator of the Bernoulli distribution). We do this for increasing number of training points in A (which are chosen randomly with a uniform distribution in A). We repeat the experiment 10 times. The left plot shows the underlying function and the Gaussian Process for 15 training points. The right plot shows the evolution of our estimate as we increase training points inside $A = [0.1, 0.2]$ (we plot the mean of each run $\pm$ the standard deviation). This example makes it clear that p_t does not converge to zero. Furthermore, it seems to converge to just over 0.7 which a very large probability. This will make fully escaping the local minimum very difficult without Point Deletion.

Remark 4.2. *Assume that $p_t \to p > 0$, and that we have a high escape probability after t_e iterations, i.e., $(1-p)^{T-t_e}$ is large, leaving us $T - t_e$ iterations to explore the rest of the space. If we increase our budget from T to T', we will not have a high probability of escape until $(1-p)^{T'-t} = (1-p)^{T-t_e}$, i.e., $t = T' - T + t_e$. This leaves us with $T' - (T' - T + t_e) = T - t_e$ iterations to explore the rest of the space. Note that this is independent of T', meaning that increasing our budget does not increase our budget after leaving $B_\delta(a)$! Rather, it only means we will be stuck in $B_\delta(a)$ for a longer time. This is very concerning as the method will be very myopic; if it finds a local optimum, it is likely that it will spend a very large amount of the budget exploiting it.*

4.4.3 Escaping with ϵ-Point Deletion

Intuitively, the convergence problem stems from the constant resampling, because in every iteration resampling is reintroducing exploitative points. We seek a way of making sure that the samples used by the algorithm to create paths take into account the previously evaluated designs. Penalising the samples via distances to previous queries seems like a simple solution,

e.g., similar to local penalization [González et al., 2016a]. However, penalization introduces the need to carefully tune the strength of the penalization so that we can still exploit optima and remain cost-effective – the parameterization is nontrivial.

Around a local maximum, if we assume that $p_t \to p > 0$, then we can interpret p as the probability that the global maximum lies on the ball $B_\delta(a)$, as such, we focus on a method that has the property of exploiting the promising area for Tp iterations (that is, the total budget weighted by the probability). To achieve this, we propose creating a total of T samples, and then deleting batch points that are similar to previously-explored points (i.e., we remove excess exploitation introduced by resampling). Algorithm 5, which we term ϵ-Point Deletion, still allows us to exploit local optima if we sample many points near them, however, it should eventually move on.

Two points to note: (a) Point Deletion uses the Euclidean norm, and it is independent of the cost function. This is because we are trying to escape local minima of simple regret. (b) After the deterministic removals, it is likely the batch size is still larger than our remaining budget, so we balance it by randomly removing points from the batch.

Algorithm 5 ϵ-Point Deletion

input: New proposed batch $\mathcal{P}_t$ (size T), set of already queried points Q_t (size t), and deletion distance ϵ
for $x \in Q_t$ **do**
  $\tilde{d} \leftarrow \min_{x' \in \mathcal{P}_t} ||x - x'||$
  if $\tilde{d} < \epsilon$ **then**
    `# find the closest point to the query` x `in the new batch`
    $\tilde{x} \leftarrow \arg\min_{x' \in \mathcal{P}_t} ||x - x'||$
  else
    `# else pick a random sample`
    $\tilde{x} \leftarrow \text{Random}(\mathcal{P}_t)$
  `# remove said point from the batch`
  $\mathcal{P}_t \leftarrow \mathcal{P}_t \setminus \{\tilde{x}\}$
output: A batch $\mathcal{P}_t$ (size $T - t$)

ϵ-Point Deletion allows us to escape local optima by directly increasing the probability of fully escaping without changing p_t. Let q_t be the number of previously queried points inside the ball, and set $\epsilon \geq 2\delta$. Indeed, it follows that we will escape if $N_t \leq q_t$, (as q_t samples are guaranteed to be deleted due to previously queried points) which is much better than requiring $N_t = 0$.

Due to over-sampling, the expected number of 'non-escapes' is given by $\mathbb{E}[N_t] = p_t T$. Around a local maximum, if we assume that $p_t \to p > 0$, and hence, $p_t T \approx pT$, then we can reasonably expect an escape when $q_t = pT$, that is, we will exploit the local maximum for approximately pT iterations. This time we leave $T - q_t = T(1-p)$ extra iterations to explore the remaining space! Increasing the budget will benefit *both* the exploitation and the exploration instead of only the former (in contrast with Remark 4.2).

Figure 4.2 gives an empirical example where we use Point Deletion, with $\epsilon = 0.1$, to escape a local optimum. We observe the expected behavior from our brief analysis. For Point Deletion, we calculate the escape prediction as $pT \approx 74$, using $\hat{p} \approx 0.74$, which we estimated in Figure 4.4. We can see that without Point Deletion, we remain stuck in the first local optimum.

4.5 The SnAKe algorithm

Algorithm 6 which we dub 'Sequential Bayesian optimisation via Adaptive Connecting Samples' (SnAKe), combines the ideas of previous sections. Figure 4.5 diagrams the most important steps of SnAKe. Section 4.8 develops an effective, parameter-free alternative to the choice of ϵ.

Note there is no requirement for data to be available immediately following querying. If $t_{delay} > 0$, we can simply stick to the latest path. It works without modification on the asynchronous setting. This is vital since we were inspired by chemical experiment design which can exhibit asynchronicity. We also note that the problem is invariant to cost scaling, as the batch creation and point deletion steps are independent of the cost, and the solution to the Travelling Salesman Problem is also scale-invariant.

4.6 Computational Considerations

Unfortunately, Algorithm 6 (SnAKe) is computationally expensive in two aspects. First, for large budgets, we may struggle to train and sample the GPs. For our experiments, training

Algorithm 6 SnAKe

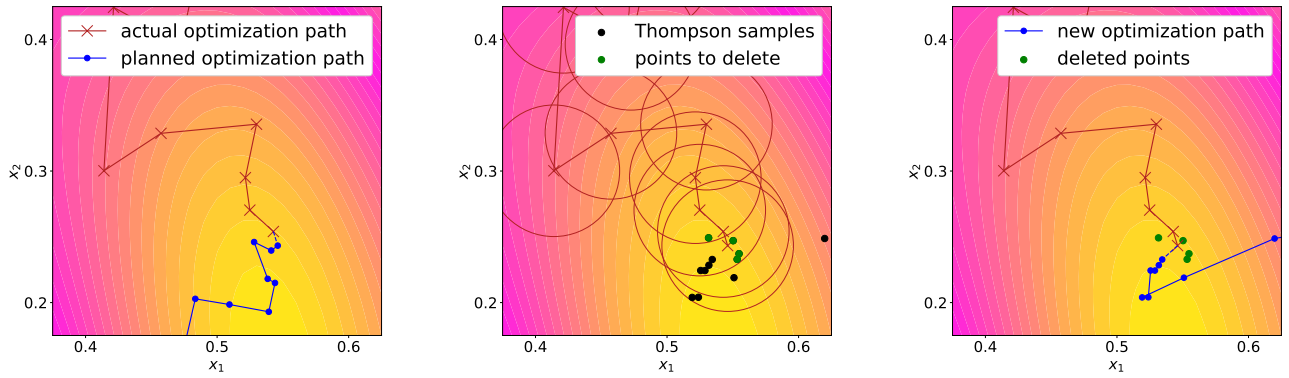
input: optimisation Budget, T. Deletion constant, ϵ.
begin: Create initial batch, $\mathcal{P}_0$, uniformly. Choose starting point x_0. $Q_0 \leftarrow \{x_0\}$. Create initial path, S_0, by solving TSP on $\mathcal{P}_0$.
for $t = 1, 2, 3, ..., T$ **do**
  Check if any running evaluations are finished
  if there are new observations **then**
    Update surrogate model
    Create batch of size T using Thompson Sampling, $\mathcal{P}_{t-1}$
    $\mathcal{P}_{t-1} \leftarrow \epsilon$-Point Deletion($\mathcal{P}_{t-1}$, Q_{t-1})
    $\tilde{S} \leftarrow$ TSP($\mathcal{P}_t$, source $= x_{t-1}$) $\setminus \{x_{t-1}\}$
    $S \leftarrow Q_{t-1} \cup \tilde{S}$
  Choose next query point from schedule: $x_t \leftarrow S_t$
  $Q_t \leftarrow Q_{t-1} \cup \{x_t\}$
  Evaluate $f(x_t)$

was not an issue and we were able to use full model GPs. However, we could use Sparse GPs [Snelson and Ghahramani, 2005] if needed. The Wilson et al. [2020] approach allows us to create GP samples efficiently, and possibly optimise them using gradient methods. The sampling can be done in linear time (*after* the GP has been trained). We use the Wilson et al. [2020] method to create our samples, and then optimise the samples using Adam [Kingma and Ba, 2014].

The second bottleneck is solving the TSP, which is NP-hard, and we may need to solve it almost at every iteration (for small values of t_{delay}). There are heuristic solutions that give approximate solutions quickly. We use Simulated Annealing [Kirkpatrick et al., 1983] which grows linearly with the budget size, T. For small budgets, simulated annealing should find good solutions, but it could struggle as the budget size increases. To solve this, note that we do not actually require a super-specific solution to the problem: we are only expecting to query the first few points on a path before replacing it by an entirely new path.

We build an adaptive grid (at each iteration) consisting of two separate parts. A very coarse grid, ξ_{global}, covers most of the search space, and a very fine grid, $\xi_{local,t}$, consisting of the N_l samples closest to x_t. This allows us to define the grid $\xi_t = \xi_{global} \cup \xi_{local,t}$. The remaining $T - N_l$ samples will be assigned to the closest point in ξ_t (using the Euclidean distance).

The adaptive grid means we expect to have multiple samples assigned to the same point, specifically in the coarse areas of our grid. But this is not important, because we expect our



(a) optimisation path at $t = 10$ (b) Sampling & Deletion Step (c) optimisation path at $t = 11$

Figure 4.5: Graphical example of SnAKe behavior in a full iteration if new information is available. The underlying function is Branin 2D. The feasible set is $[0, 1]^2$, so we do not see all the samples or the complete path with this zoomed-in view. (a) The red line shows the already-queried path. The blue path shows our future plans. (b) For each query, we can see the ϵ-ball under which the deletion step is deterministic. We plot the Thompson Samples as dots, the accepted ones in black and the deleted ones in green. (c) The new path is in blue and the points ignored (due to Point Deletion) are in green. Note the higher concentration of samples in what the model considers a promising area.

immediate attention will be in the area around our current input where there should be little to no repetition. This will allow the algorithm to focus on testing solutions which are relevant to our problem.

The adaptive grid introduces two hyper-parameters: the size of the global and local grids, respectively N_g and N_l. Using this method, we run the TSP heuristics on a graph with at most $\min(N_g + N_l, T)$ nodes. For the experiments, we create the global grid using a simple Sobol grid [Sobol', 1967].

With these modifications, we were able to comfortably run all experiments in a CPU (2.5 GHz Quad-Core Intel Core i7), where SnAKe shared a wall-time similar to Local Penalization methods.

4.7 Practical extensions of SnAKe

SnAKe is limited in some practical aspects. In this section we present three practical extensions:

I. SnAKe requires the total number of experiments to be selected *a priori*, but, in practice

we might not know how many experiments are needed, or we might want to minimize the number of experiments, e.g., to save on materials.

II. Even though SnAKe minimizes cumulative input changes, individual input changes between iterations are unbounded. This could pose a practical problem in some applications, where large changes may be unsafe or infeasible. For example, there may be limitations on maximum temperature or the rate of temperature change.

III. The algorithm assumes a single objective function. In many applications, experimentalists are interested in problems with many objectives. In multi-objective settings, we seek to find a Pareto front of solutions as opposed to optimizing a single function.

4.7.1 Self-stopping SnAKe

Algorithm 7 Expected Improvement Point Deletion

INPUT: Proposed batch, $\mathcal{P}_t$, of size $(T-t)$. Model-misspecification safety parameter ν, termination parameter δ.
for $x_p \in \mathcal{P}_t$ **do**
  if ($EI(x_p)/C^{(0)}(x_p) < \delta$) and ($\text{Var}[\, x_p \mid \mathcal{D}_t \,] < \nu$) **then**
    $\mathcal{P}_t \leftarrow \mathcal{P}_t \backslash \{x_p\}$
if $|\mathcal{P}_t| = 0$ **then**
  Terminate optimisation
OUTPUT: Reduced Batch, $\tilde{\mathcal{P}}$, of unknown size. Optimisation termination criteria.

We begin by investigating the first problem. We assume that we want to optimise an expensive black-box function f and that the cost of experiment $t+1$ can be described by:

$$C(x_t, x_{t+1}) = C^{(0)}(x_{t+1}) + C^{(\Delta)}(x_t, x_{t+1}) \tag{4.3}$$

where x_{t+1} is the next experimental input we want to query, x_t is our current input, $C^{\Delta}(x_t, x_{t+1})$ is an input-change cost function equal to zero when $x_{t+1} = x_t$, and $C^{(0)}(x_{t+1})$ represents the fixed cost of each experiment. The goal is to optimise the function for the smallest possible total cost. Finding an exact optimum of a black-box function is practically impossible; therefore the Bayesian optimisation loop should terminate once we have a good approximation.

To combine Bayesian optimisation stopping conditions and SnAKe, we modify the Point Deletion criteria [Folch et al., 2022, Section 3.7] using EIpu [Snoek et al., 2012] as a stopping condition. For a certain point x, if the EIpu, $EI(x)/C^{(0)}(x)$, is less than a given threshold $\delta > 0$, then we delete it. We also add an extra criterion based on the prediction variance of the point to avoid early termination in the case where the model is misspecified: we will not delete points that still have significant uncertainty. Using this same criteria (to avoid model misspecification), we were able to significantly boost EIpu's performance, achieving comparable results to SnAKe. We detail the method in Algorithm 7.

4.7.2 Truncated SnAKe

We now investigate the problem of maximum allowable input changes, motivated by, e.g., safety or physical limits. In particular, we assume the optimisation now involves the *rate-of-change* constraint:

$$|x_{t+1} - x_t| < \delta_{MAX}, \quad \forall\, t \in \{1, ..., T\}. \tag{4.4}$$

We modify the criteria that SnAKe uses when selecting which experiment to query next: truncate SnAKe's input change whenever it exceeds δ_{MAX}. The modification is shown in Algorithm 8, where $[v]^{(u)}$ denotes the unit-length vector in direction v.

Algorithm 8 Truncated SnAKe (TrSnAKe)

Run SnAKe as normal with the following extra criterion for each new experiment, x_{t+1}:
if $|x_{t+1} - x_t| > \delta_{MAX}$ **then**
$x_{t+1} \leftarrow x_t + [x_{t+1} - x_t]^{(u)} \delta_{MAX}$

4.7.3 Multi-objective SnAKe

We consider a simple multi-objective extension of SnAKe by using random scalarizations of the objective functions [Paria et al., 2020]. We define the problem as trying to find the Pareto front of a set of objective functions $\{f^{(k)}(x)\}_{k=1}^{K}$. The Pareto front (in the maximization setting) is defined as the set of points where we cannot increase the value of an objective without decreasing the value of another.

SnAKe creates a large batch of points using Thompson Sampling, which is then considered for evaluation and ordered in such a way that input costs are minimised. Paria et al. [2020] give a multi-objective extension of Thompson Sampling, where we consider a random parametric scalarization of the objectives as a single objective function.

Assume we have K objectives, each modelled by an independent GP. We choose a scalarization function $\mathcal{S}(\{f^{(k)}\}_{k=1}^K, \lambda)$ where λ is a random parameter with distribution $p(\lambda)$. Specifically, we test a linear scalarization function $\mathcal{S}(\{f^{(k)}\}_{k=1}^K, \lambda)(x) = \sum_{k=1}^K \lambda_k f^{(k)}(x)$ and Tchebyshev scalarization $\mathcal{S}(\{f^{(k)}\}_{k=1}^K, \lambda)(x) = \min_{k=1,...,K} \lambda_k (f^{(k)}(x) - z^{(k)})$, where $z^{(k)}$ are some reference points [Nakayama et al., 2009]. The latter can handle non-convex Pareto fronts. To create the batch, we simply sample both the independent GPs and $\lambda \sim p(\lambda)$ and then optimise the single objective (one for each GP sample). We choose this particular multi-objective method for its computational simplicity, which is required when creating large batches. We outline the procedure in Algorithm 9.

Algorithm 9 Multi-objective SnAKe (MO-SnAKe)

Run SnAKe as normal, but replace the Thompson Sampling in the batch creation step with Random Scalarization Thompson Sampling. That is, create the batch with the following algorithm:
for $i = 1, ..., T$ **do**
  $f^{(1)}, ..., f^{(K)} \sim GP(\mu_t, \sigma_t | \mathcal{D}_t)$
  $\lambda \sim p(\lambda)$
  $x_{t+1}^{(i)} = \arg\max_{x \in \mathcal{X}} \mathcal{S}(\{f^{(k)}\}_{k=1}^K, \lambda)(x)$

4.8 Experimental Results

For all experimental results we report the mean and the standard deviation over 25 experimental runs. We give the full implementation details in Appendix A.2. Classical Bayesian optimisation methods are implemented using BoTorch [Balandat et al., 2020] and GPyTorch [Gardner et al., 2018]. The code to replicate all results is available online at `https://github.com/cog-imperial/SnAKe`.

In all experiments, we examine SnAKe for $\epsilon = 0, 0.1$, and 1. We further introduce a parameter-

free alternative by adaptively selecting ϵ to be the smallest length scale from the GP's kernel, and denote it ℓ-SnAKe. SnAKe proves to be robust to non-zero choices of ϵ. We conjecture this happens because of Thompson Sampling's exploitative nature [Nava et al., 2022], so all excess exploitation is concentrated on a very small area. For $\epsilon = 0$ we observe very low cost, at the expense of some regret.

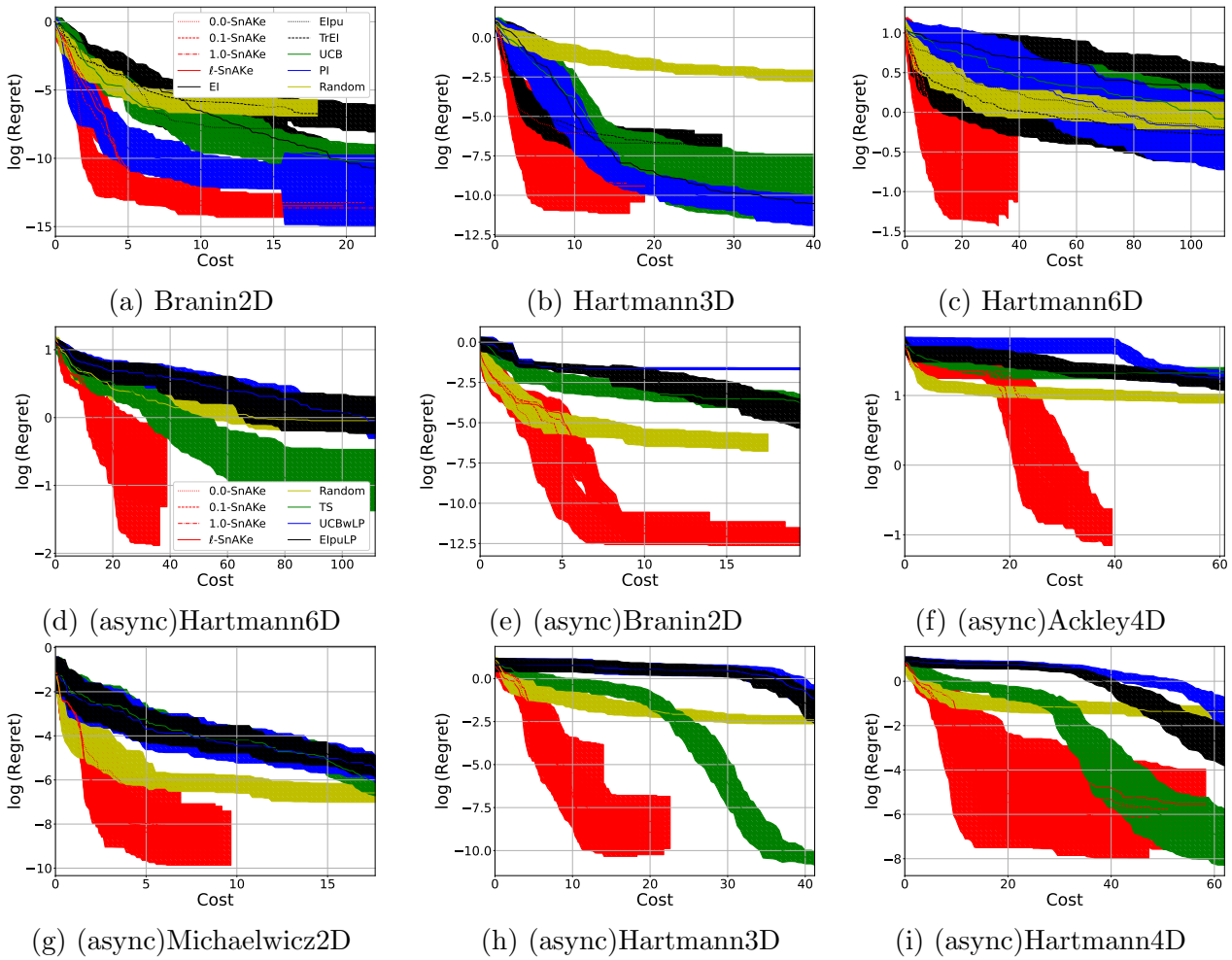
Across all experiments we can see how SnAKe can traverse the search space effectively, constantly achieving good regret at low cost. While other cost-aware methods may also achieve good performance, they are very inconsistent, suggesting they require careful tuning. SnAKe achieves good, robust results with a single hyper-parameter. We do highlight that SnAKe performs best when there are enough iterations for the samples to adequately fill the search space: this is highlighted by poor performance in Perm10D and by ordinary performance in low-budget experiments (see Figure 4.8).

4.8.1 Synthetic Functions

Sequential Bayesian optimisation This section examines the performance of SnAKe against Classical Bayesian optimisation algorithms. We compare Expected Improvement (EI) [Mockus et al., 2014], Upper Confidence Bound (UCB) [Srinivas et al., 2010], and Probability of Improvement (PI) [Kushner, 1964]. We compare against Expected Improvement per Unit Cost Snoek et al. [2012] (EIpu) as introduced in Eq. (4.1), with $\gamma = 1$, and with Truncated Expected Improvement Samaniego et al. [2021] (TrEI). We also introduce a simple baseline, where we create a random Sobol sample, and then arrange an ordering by solving the TSP, and never update the path again. We do this for three classical benchmark functions. We set the cost function to be the 2-norm distance between the inputs. Figure 4.6 shows the results in the top row. For completeness, Figure 4.8 showcases a failure of SnAKe in high-dimensional optimisation.

Asynchronous Bayesian optimisation We explore the asynchronous setting, comparing Local Penalisation with UCB (UCBwLP) [González et al., 2016a, Alvi et al., 2019], Thompson Sampling (TS) [Kandasamy et al., 2018], and the same Random baseline from the sequential

setting. We also test on EIpu with Local Penalisation (EIpuLP), with $\gamma = 1$. The results are shown in last two rows of Figure 4.6.



(a) Branin2D (b) Hartmann3D (c) Hartmann6D

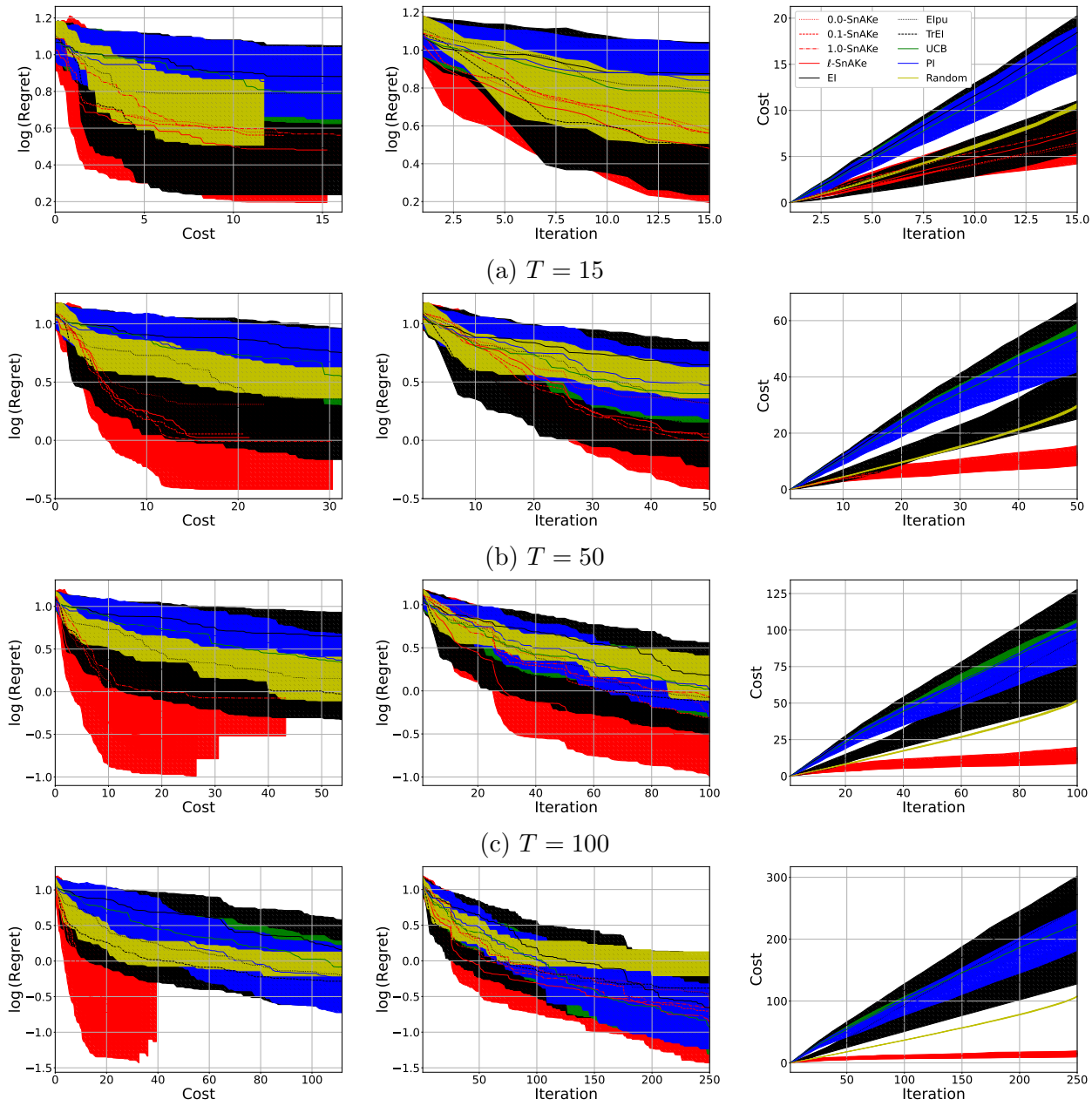
(d) (async)Hartmann6D (e) (async)Branin2D (f) (async)Ackley4D

(g) (async)Michaelwicz2D (h) (async)Hartmann3D (i) (async)Hartmann4D

Figure 4.6: Synthetic experimental results. The first row represents three of the synchronous benchmarks. The last two rows correspond to asynchronous experiments with $t_{delay} = 25$. We plot the average log(regret) achieved against the cost spent. For every experiment, $T = 250$, and we limit the x-axis to the maximum cost achieved by SnAKe or Random. ℓ-SnAKe is the parameter-free version. SnAKe consistently achieves good regret against at cost. The bounds are $\pm$ half the standard deviation of 25 different runs.

4.8.2 Reaction Control on SnAr Benchmark

We test our method on a real-world, SnAr chemistry benchmark [Hone et al., 2017]. We control four variables; equivalents of pyrrolidine, concentration of 2,4 dinitrofluorobenenze at reactor inlet, reactor temperature, and residence time. We assume changing temperature, concentration and residence time incur an input cost, owing to the response time required for the reactor to



(a) $T = 15$

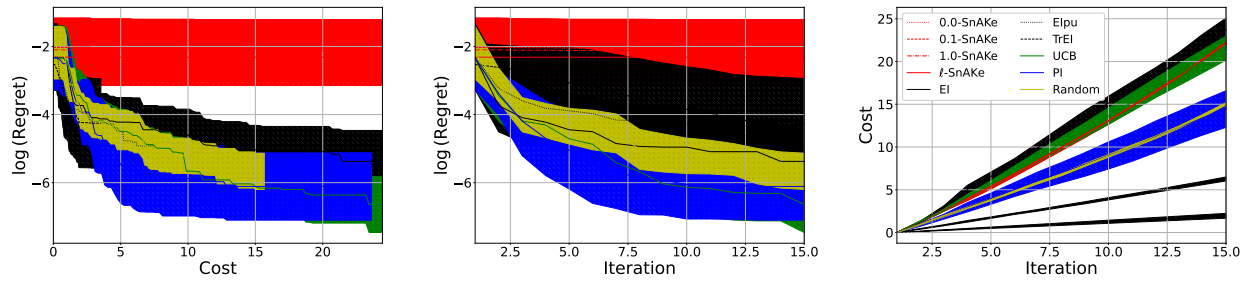
(b) $T = 50$

(c) $T = 100$

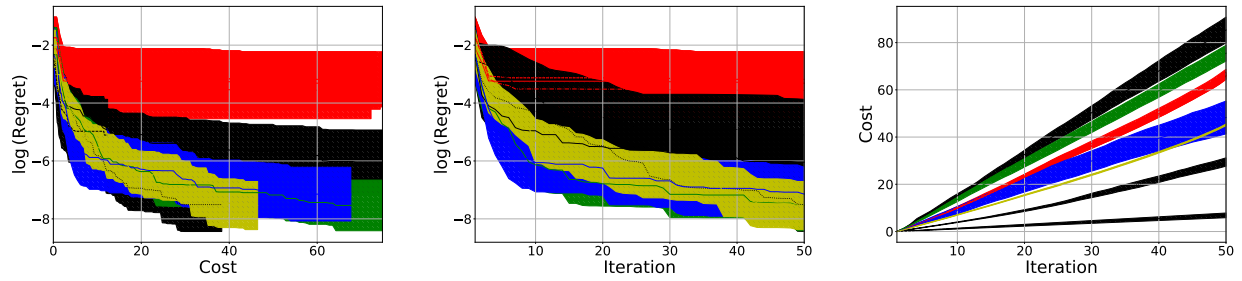
(d) $T = 250$

Figure 4.7: Hartmann6D. Here we showcase the effect of increasing the budget size, showing SnAKe performs increasingly better as the budget grows. Each row represents a different budget. The left column shows the evolution of regret against the cost used. The middle column shows the evolution of regret with iterations, and the right columns show the evolution of the 2-norm cost. A high-dimensional example where SnAKe performs exceedingly well, giving the best regret at low costs for all budgets except $T = 15$. The final cost is considerably lower for SnAKe than any other method.

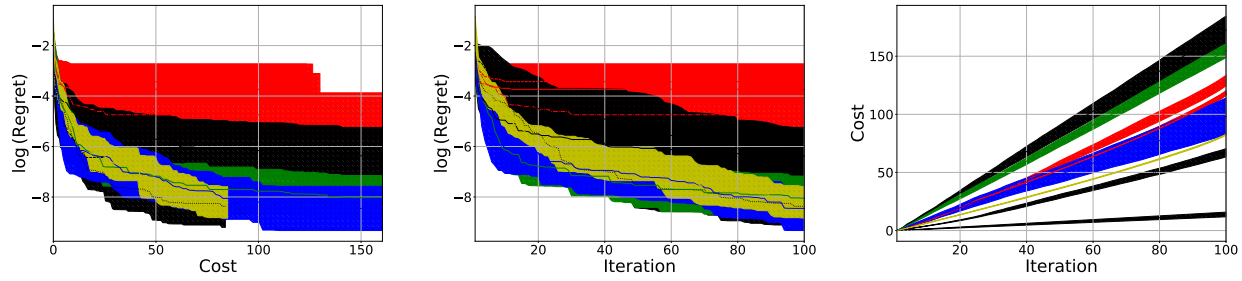
reach a new steady state. We assume the reactor as a first-order dynamic system, where the



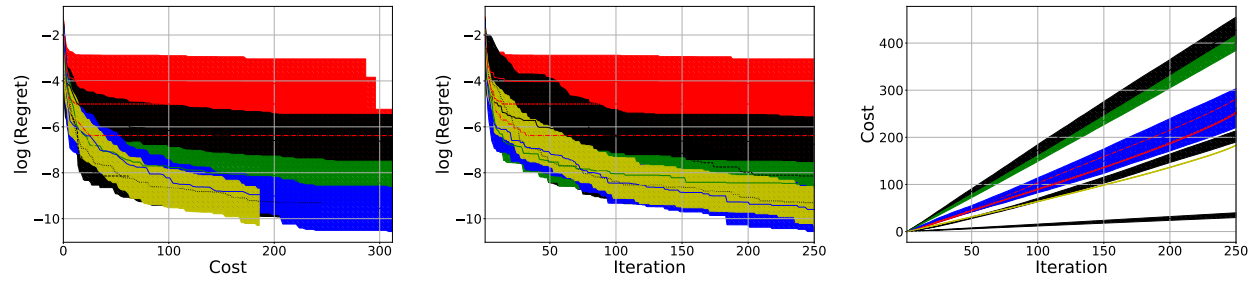
(a) $T = 15$



(b) $T = 50$



(c) $T = 100$



(d) $T = 250$

Figure 4.8: Perm10D. Here we showcase how SnAKe can fail to optimise well in higher dimensions. Each row represents a different budget. The left column shows the evolution of regret against the cost used. The middle column shows the evolution of regret with iterations, and the right columns show the evolution of the 2-norm cost. SnAKe struggles in this benchmark, however, EI also struggles. As an interesting observation, if we *did not* update the model, we would achieve a much better performance (as it would be equivalent to Random). EIpu and TrEI perform well in this example, we conjecture this is because they are doing far more localised searches while SnAKe is trying to cover all the space available (which is very difficult in higher dimensions).

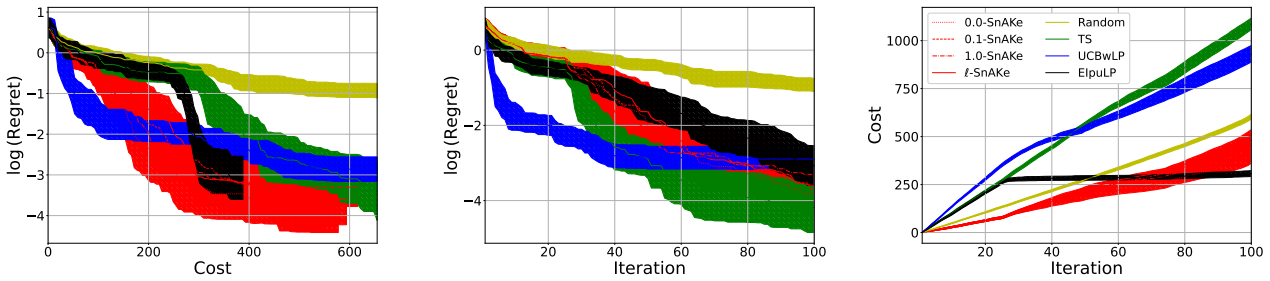
response to changes in input is given by:

$$(x_s)_i = (x_t)_i + (1 - e^{-s/\alpha_i})(\Delta x_t)_i \tag{4.5}$$

where s denotes the time after experiment x_t is finished, $(x_t)_i$ denotes the ith variable of the tth experiment, $(\Delta x_t)_i = (x_{t+1})_i - (x_t)_i$, and α_i is the system time constant. We define quasi-steady state using an absolute residual β_i, i.e., $|(x_s)_i - (x_{t+1})_i| \leq \beta_i$. For input changes smaller than β_i, we assume a linear cost, defined by a parameter γ_i. Combining with the response time from (4.5):

$$\mathcal{C}_i(x_t, x_{t+1}) = \gamma_i \min\{\beta_i, |\Delta^{(i)} x_t|\} + \max\left\{0, \alpha_i \log\left(|\Delta^{(i)} x_t|/\beta_i\right)\right\}. \tag{4.6}$$

We assume that we can change variables simultaneously, so the total input cost is simply the longest response time within a given set of input changes. That is, $C(x_t, x_{t+1}) = \max_{i \in I_c} C_i(x_t, x_{t+1})$, where I_c is the index set of the control variables. We implement the simulation using the Summit package [Felton et al., 2021]. More details can be found in Appendix A.2.6. We set a delay of $t_{delay} = 25$, and optimise for $T = 100$ iterations. The results of the experiment can be seen in Figure 4.9.



(a) Evolution of regret against input cost. We can see that SnAKe is able to achieve the best regret for low cost.

(b) Evolution of regret with iteration number. SnAKe's final regret is comparable with classic Bayesian optimisation methods, and better than EIpuLP.

(c) Evolution of input cost with iteration number. Low cost is achieved by EIpuLP, SnAKe and the TSP-ordered Random optimisation.

Figure 4.9: Results of experiments on the SnAr chemistry Benchmark. SnAKe achieves the best regret out of all low-cost methods. The bounds are created from ± half the standard deviation of all runs. The best performer, ℓ-SnAKe, is parameter-free.

4.8.3 Finding Contamination Sources in Ypacarai Lake

As a second real-world example that exhibits costly input changes, we investigate an alternative to the case study introduced by Samaniego et al. [2021]. Autonomous boats are used to monitor water quality in Ypacarai Lake in Paraguay. We consider three different objectives, illustrated in Figure 4.10, created using the Schekel benchmark function (as in Samaniego et al. [2021]).

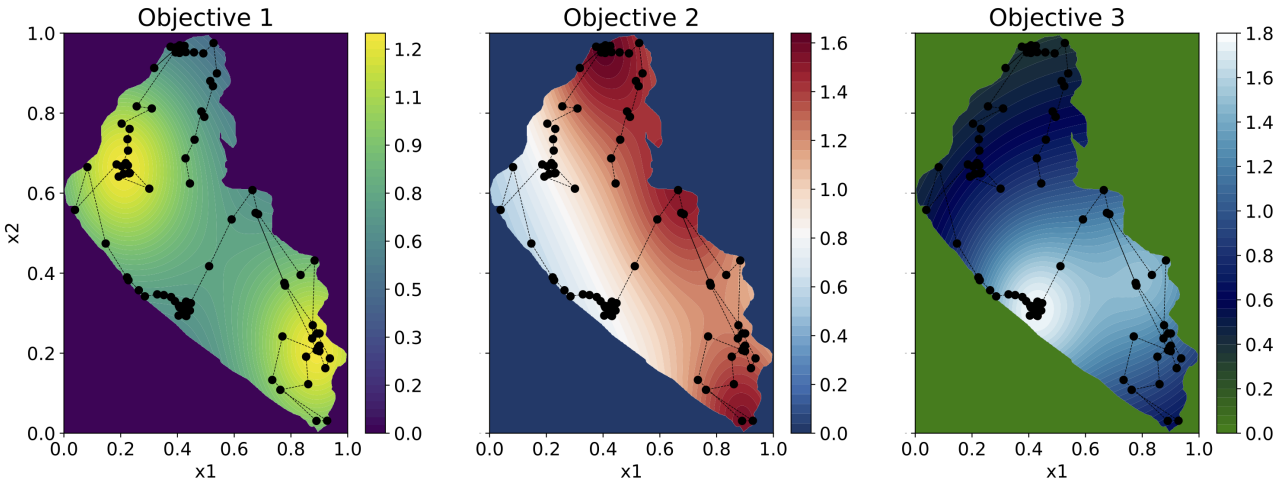


Figure 4.10: Visualization of Ypacarai Lake, and the objectives we optimise over. We show an example of a SnAKe optimisation path on the simultaneous optimisation problem (see Section 4.8.4).

Table 4.1: Ypacarai results. We present the average regret from 10 runs ± the standard deviation, multiplied by 10^3. O1, O2, O3 correspond to the different objectives. All runs are terminated after the cost exceeds 10 units (approximately 100 km). SnAKe is the best performer in 2 out of 3 objectives.

	Regret O1	Regret O2	Regret O3
TrEI	0.475 ± 0.596	0.118 ± 0.104	0.076 ± 0.050
EIpu	0.249 ± 0.225	$\mathbf{0.068 \pm 0.073}$	0.071 ± 0.049
SnAKe	$\mathbf{0.074 \pm 0.063}$	0.109 ± 0.186	$\mathbf{0.036 \pm 0.048}$

Each objective could correspond to a measure of water quality, such as pH, turbidity, CO2 levels, or many more. We consider the problem of finding the *largest* source of contamination in the lake by maximising these objectives. The cost between queries relates to the boat physically moving from one place to another. We assume that the boats can travel 10 units, which corresponds to approximately 100km (fuel limitations), and no observation time delay. We run optimisation runs on each objective and shows the results in Table 4.1.

4.8.4 Simultaneous optimisation of multiple objectives

In this section, we show that SnAKe can easily be extended to optimise multiple black-box functions *simultaneously* in the same search space.

Table 4.2: Ypacarai experimental results. We present the average regret from 10 runs ± the standard deviation multiplied by 10^3. MO1, MO2, MO3 represent the different maximums (one for each function). All runs are terminated after the cost exceeds 10 units (approximately 100 km). SnAKe is the only method with good regret in all metrics, as TrEI performs poorly in the first objective, and EIpu performs poorly in the second.

	Regret MO1	Regret MO2	Regret MO3
TrEI	13.636 ± 35.842	0.802 ± 0.828	0.162 ± 0.108
EIpu	0.995 ± 0.661	19.308 ± 55.901	0.212 ± 0.137
SnAKe	0.996 ± 2.743	1.740 ± 4.813	0.064 ± 0.053

We extend the Ypacarai benchmark introduced in Section 4.8.2 under the assumption we have differing non-competitive objectives. The application is to use autonomous boats to find the *largest* source of contamination in Lake Ypacarai in Paraguay. Figure 4.10 shows a visualization of the Lake, and three objectives we want to optimise over. Each objective corresponds to a different measure of water contamination, examples might include pH, turbidity, CO2 levels, and more.

In this section, we extend the problem to optimise over all of these *simultaneously*. Note that this is different from traditional multi-objective optimisation since our objectives are non-competing, they just share the same search space. This is important because it is inefficient to run multiple objective runs when we are exploring the search area. At every iteration we choose one query x_t, and obtain three outputs $f_1(x_t)$, $f_2(x_t)$, and $f_3(x_t)$ one for each objective.

We model each objective using an independent GP, meaning we can *create Thompson Samples from any objective*. Therefore we extend SnAKe by creating the batch using a mixture of samples from each objective. For the experiment we use the ratio 1:1:1. For TrEI and EIpu, we adapt to the multi-objective case by optimising the first objective until we have travelled 3.3 units (approximately 33km), after which we change to the subsequent objective. Table 4.2 shows the results.

4.8.5 Results on practical extensions

All implementations use GPyTorch [Gardner et al., 2018] and BoTorch [Balandat et al., 2020]. To implement SnAKe, we use the non-parametric version of point-deletion when applicable (denoted by an ℓ prefix). In all plots, we zoom into the area of interest (low-cost region) and plot ± 0.5 the standard deviations of 25 runs with random initial conditions. Dashed vertical lines in plots show the average cost used. As in Folch et al. [2022], the hyper-parameters of the GP are optimised by maximising the marginal-likelihood every 25 iterations. Random refers to uniform samples ordered by solving the Travelling Salesman Problem. We highlight a few benchmarks here and include further studies in the Appendix.

We tested self-stopping SnAKe (ssSnAKe) in three synthetic benchmarks: Branin2D, Hartmann3D and Hartmann6D. We compare against SnAKe, vanilla EIpu, and a new implementation of EIpu that also considers model certainty as termination criteria.

SnAKe achieves the best regret; however, it expends the entire budget and therefore incurs high costs for very small gains. ssSnAKe achieves similar performance to SnAKe up to the point where it automatically terminates. Our modification to EIpu significantly improves the performance, achieving the best results in Hartmann3D and comparable results in Branin2D benchmarks. It is good to note that the ssSnAKe seems robust to the maximum budget, using approximately the same number of experiments, even when the maximum budget is increased. We showcase the Hartmann6D benchmark in Figure 4.11.

For rate-of-change constraint problems, we test on two synthetic benchmarks: Branin2D and Hartmann3D. We compare Truncated SnAKe (TrSnAKe), Truncated Expected Improvement (TrEI) [Samaniego et al., 2021], SnAKe, and EIpu. We emulate movement constraints using a cost function with a jump when the movement is larger than δ_{MAX}:

$$\mathcal{C}(x_t, x_{t+1}) = 0.2||x_{t+1} - x_t||_2 + \mathbb{I}(||x_{t+1} - x_t||_2 > \delta_{MAX}).$$

For each benchmark, we test with $\delta_{MAX} \in \{0.1, 0.025\}$. TrSnAKe performs best in three out

of four cases. Methods that cannot consider movement constraints are heavily penalised and achieve poor performance. Figure 4.12 shows the results for the Branin2D benchmark.

Finally, we test multi-objective SnAKe (MO-SnAKe) on two benchmarks: the Schekel function, which was used to model pollution in Ypacarai Lake [Samaniego et al., 2021], and the SnAr chemistry benchmark, implemented in Summit [Felton et al., 2021]. We compare MO-SnAKe and Scalarised Thompson Sampling [Paria et al., 2020]. We note that Folch et al. [2022] used SnAr as a real-world application; however, they restricted the problem to single objective. This work is able to consider the original multi-objective problem.

Table 4.3: Results of multi-objective optimisation experiments on four dimensional benchmarks. Red and blue indicate linear and Tchebyshev scalarazations respectively.

	Method	GD	IGD	MPFE	Cost
Schekel	ℓ-MO-SnAKe	**0.05** ± 0.06	**0.09** ± 0.07	**0.23** ± 0.08	**9.60** ± 6.12
	ℓ-MO-SnAKe	0.07 ± 0.05	0.15 ± 0.05	0.29 ± 0.07	11.24 ± 9.94
	TS	**0.05** ± 0.04	0.11 ± 0.07	0.24 ± 0.08	57.21 ± 12.97
	TS	0.06 ± 0.04	0.16 ± 0.05	0.30 ± 0.07	63.18 ± 12.30
	Random	0.22 ± 0.05	0.22 ± 0.04	0.31 ± 0.05	59.62 ± 1.08
SnAr	ℓ-MO-SnAKe	0.00040 ± 0.00015	0.19 ± 0.06	0.67 ± 0.22	614.94 ± 229.98
	ℓ-MO-SnAKe	0.00044 ± 0.00020	0.18 ± 0.07	0.65 ± 0.26	**478.43** ± 219.16
	TS	0.00035 ± 0.00008	0.18 ± 0.06	0.64 ± 0.22	1967.56 ± 447.31
	TS	**0.00031** ± 0.00008	0.18 ± 0.07	0.63 ± 0.25	1602.68 ± 401.3
	Random	0.02498 ± 0.01157	**0.16** ± 0.04	**0.35** ± 0.15	1195.68 ± 42.07

For performance metrics, we use Euclidean generational distance (GD) which measures the

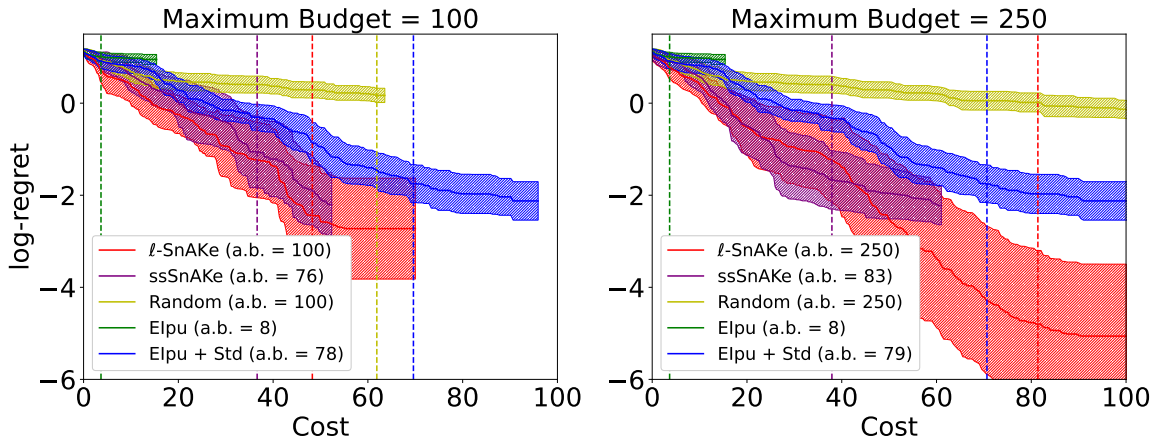


Figure 4.11: Self-stopping optimisation of Hartmann 6D with different maximum budgets. Vanilla EIpu stops too early due to model misspecification; our modification that includes uncertainty criteria greatly improves performance. The average budget used (a.b.) is robust to maximum budget choice.

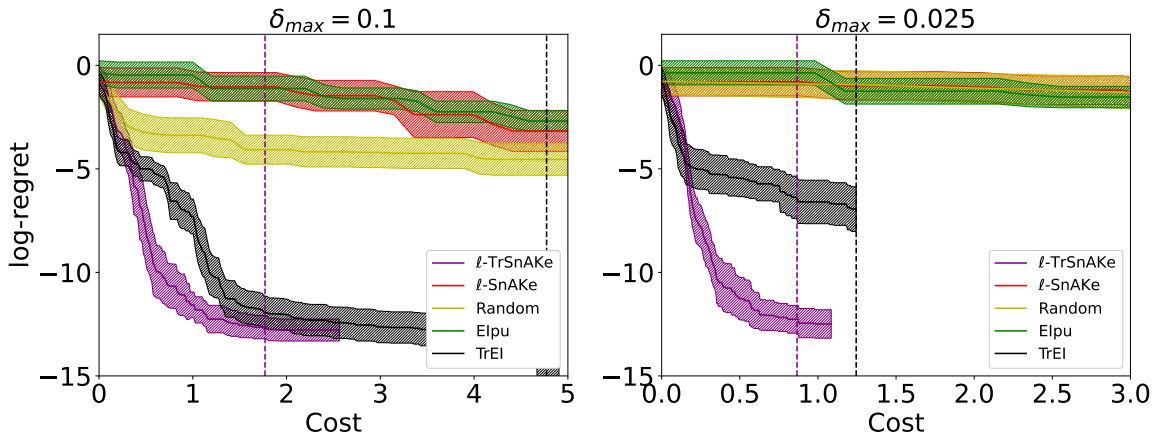


Figure 4.12: Results for truncated Branin2D with a budget of 250 queries. The cost of non-truncated methods is very high due to penalization for breaking movement constraints. TrSnAKe achieves the best performance for low cost.

average distance from the the approximated Pareto front to the true one, inverted generational distance (IGD) which measure the average distance from the true Pareto front to the approximated one, and the maximum Pareto frontier error (MPFE) which measures the largest error from the approximated Pareto frontier to the true one. We obtain a 'true' set of Pareto points by running `NSGA-II` [Deb et al., 2002]. Table 4.3 shows MO-SnAKe performs comparable to Scalarised Thompson Sampling [Paria et al., 2020] at much lower cost.

4.9 Discussion

In this chapter, we introduced and proposed a solution to the problem of optimising black-box functions under costs to changes in inputs. We empirically show that the regret achieved by our method is comparable to those of classical Bayesian optimisation methods and we succeed at achieving considerably lower input costs, while being particularly well suited to asynchronous problems. We further showcased the flexibility of the method by exhibiting a few practical extensions of the algorithms.

This setting, with input costs penalising experimental changes, makes a major step towards automating new reaction chemistry discovery, e.g., in line with the vision of Lazzari et al. [2020]. We substantially decrease experimental cost with respect to classical black-box optimisation, e.g., as used by Fath et al. [2021] and McMullen and Jensen [2011]. In the real-life SnAR

benchmark, SnAKe spends 45-55% of the cost while making similarly strong predictions to classical Bayesian optimisation. The synthetic benchmarks and Ypacarai Lake example offer similar advantages.

4.10 Looking back...

Looking back at this project, there is not so much I would change as I think it ended up being very complete and I had the opportunity to extended it in the follow-up workshop paper. However, there is always room for improvement. In particular, I think I could have provided a stronger codebase for the project. Indeed since publication SnAKe has been compared against in various publications and the performance they show has been disappointing. I think this is partly due to the code being finicky when it comes to the optimisation procedure, and best practices with hyper-parameters not being well researched and implemented.

This is, of course, something I can always work on in the future if I find the time. Particularly, I have since contributed to open-source Bayesian optimisation libraries BoFire [Dürholt et al., 2024] [`https://github.com/experimental-design/bofire`] and DoExPy [`https://github.com/Mojusko/experiment-design-mdp`] (which actually contains a more accurate version albeit much slower), and could perhaps incorporate efficient implementations of SnAKe in these libraries.

Chapter 5

Transition Constrained Bayesian Optimisation via Markov Decision Processes

This chapter is associated with the publication Folch et al. [2024].

5.1 Preliminaries

State-of-the-art algorithms are often ill-suited when physical sciences interact with potentially dynamic systems. In such circumstances, real-life constraints limit our future decisions while depending on the prior state of our interaction with the system. This chapter focuses on transition constraints influencing future choices depending on the current state of the experiment. Contrary to the last chapter, where our aim was to minimise movement, we are now dealing with strict constraints on the movement. In other words, reaching certain parts of the decision space (search space) requires long-term planning in our optimisation campaign. This effectively means we address a general sequential-decision problem akin to those studied in reinforcement learning (RL) or optimal control for the task of optimisation. We assume the transition constraints are known *a priori* to the optimiser.

Applications with transition constraints include chemical reaction optimisation [Vellanki et al., 2017, Folch et al., 2022], environmental monitoring [Bogunovic et al., 2016, Mutný et al., 2023], lake surveillance with drones [Hitz et al., 2014, Gotovos et al., 2013, Samaniego et al., 2021], energy systems [Ramesh et al., 2022], vapor compression systems [Paulson et al., 2023], electron-laser tuning [Kirschner et al., 2022] and seabed identification [Sullivan et al., 2023]. For example, Figure 5.1 depicts an application in environmental monitoring where autonomous sensing vehicles must avoid obstacles (similar to Hitz et al. [2014]). Our main focus application are transient flow reactors [Mozharov et al., 2011, Schrecker et al., 2023]. Such reactors allow efficient data collection by obtaining semi-continuous time-series data rather than a single measurement after reaching the steady state of the reactor. As we can only change the inputs of the reactor continuously and slowly to maintain quasi-steady-state operation, allowing arbitrary changes, as in conventional Bayesian optimisation, would result in measurement sequences which are not possible due to physical limitations. Note that this is different from the previous Chapter, since SnAKe minimizes distance, however, it could still allow for constraints to be broken in the process.

The problem of maximising an unknown function is typically considered in Bayesian optimisation by choosing to query $f(x)$ by sequentially maximising an *acquisition function*, u:

$$x_{t+1} = \arg\max_{x \in \mathcal{X}} u(x|\mathbf{X}_t), \tag{5.1}$$

depending on all the past data at iteration t, $\mathbf{X}_t$. Eq. (5.1) arises as a greedy one-step approximation whose overall goal is to minimise e.g. cumulative regret, and assumes that any choice of point in the search space $\mathcal{X}$ is available. However, given transition constraints, we must traverse the search space according to the system dynamics. This chapter extends the Bayesian optimisation framework and provides a method that constructs a potentially non-Markovian policy by myopically optimising a utility as,

$$\pi_{t+1} = \arg\max_{\pi \in \Pi} \mathcal{U}(\pi|\mathbf{X}_t), \tag{5.2}$$

where $\mathcal{U}$ is the greedy utility of the policy π and $\mathbf{X}_t$ encodes past trajectories through the

search space. In the following sections, we will show how to tractably formulate the overall utility, how to greedily maximise it, and how to adapt it to admit policies depending on the full optimisation history.

Contributions. We present a Bayesian optimisation framework that tractably plans over the complete experimentation horizon and respects Markov transition constraints, building on active exploration in Markov chains [Mutný et al., 2023]. Our key contributions include:

- We identify a novel utility function for maximum identification as a function of policies, and greedily optimise it. The optimisation is tractable, and does not scale exponentially in the policy horizon. In many cases, the problem is convex in the natural representation.
- We provide exact solutions to the optimisation problems using convex optimisation for discrete-state Markov chains. For continuous-state Markov chains, we propose a reparameterisation by viewing our problem as an instance of model predictive control (MPC) with a non-convex objective. Interestingly, in both cases, the resulting policies are history-dependent (non-Markovian).
- We analyse the scheme theoretically and empirically demonstrate its practicality on problems in physical systems, such as electron laser calibration and chemical reactor optimisation.

5.1.1 Related works

The most relevant prior work to ours is exploration in reinforcement learning through the use of Markov decision processes as in Mutný et al. [2023] and convex reinforcement learning of Hazan et al. [2019], Zahavy et al. [2021] which we will use to optimise the objective. Other related works are:

Optimising over sequences. Previous work, including Chapter 4 on this work, has focused on planning experimental sequences for minimising switching costs [Ramesh et al., 2022, Yang et al., 2023] however they are only able adhere to strict constraints under truncation heuristics

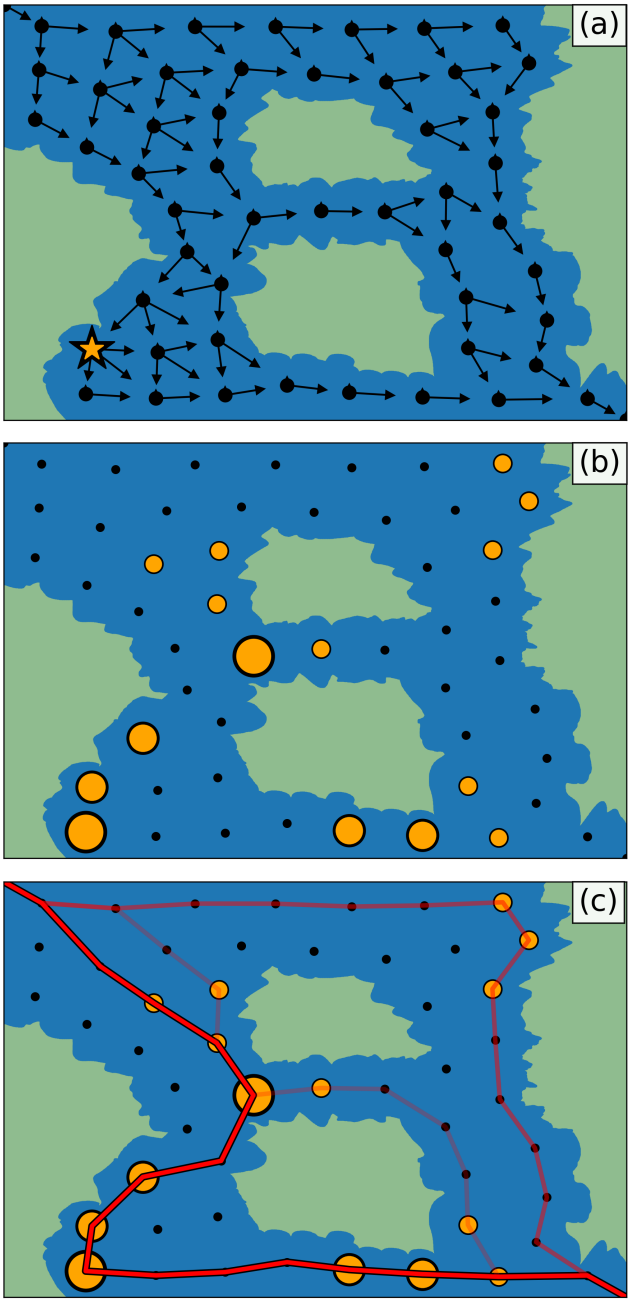


Figure 5.1: Representative task: finding pollution in a river while following the current. (a) Problem formulation: The star represents the maximiser and the arrows the Markov dynamics. (b) Objective formulation: Orange balls represent potential maximisers, with size corresponding to model uncertainty. (c) Optimisation: Deploy a potentially stochastic policy that minimises our objective.

[Samaniego et al., 2021, Paulson et al., 2023]. Concurrent work of Che et al. [2023] tackles a constrained variant of a similar problem using model predictive control with a different goal.

Regret vs Best-arm identification. Most algorithms in Bayesian optimisation focus on regret minimisation. This work focuses on maximiser identification directly, i.e., to identify

the maximum after a certain number of iterations with the highest confidence. This branch of Bayesian optimisation is mostly addressed in the bandit literature [Audibert and Bubeck, 2010]. Our work builds upon prior works of Soare et al. [2014], Yu et al. [2006], and specifically upon the seminal approach of Fiez et al. [2019] to design an optimal objective via hypothesis testing. Novel to our setting is the added difficulty of transition constraints necessitating planning.

Non-myopic Bayesian Optimisation. Look-ahead Bayesian Optimisation as described in Section 2.4 seeks to improve the greedy aspect of Bayesian optimisation. Such works also use an MDP problem formulation, however, they define the state space to include all past observations (e.g. [Jiang et al., 2020b, Astudillo et al., 2021]). This comes at the cost of simulating expensive integrals, and the complexity grows exponentially with the number of look-ahead steps (usually less than three steps). Our work follows a different path, we maintain the myopic approach to control computational efficiency (i.e. by optimising over the space of Markovian policies), and maintain provable and state-of-art performance. Even though the optimal policy through non-myopic analysis is non-Markovian, in Sec. 5.7, we show that *adaptive resampling* iteratively approximates this non-myoptic optimal policies in a numerically tractable way via receeding horizon planning. In our experiments we comfortably plan for over a hundred steps.

Reinforcement Learning for Optimisation. Our work lies in the intersection of reinforcement learning and optimisation, an area that has been extensively explored. Volpp et al. [2020] use reinforcement learning in the context of meta-learning acquisition functions. Similar but more direct approaches are explored by Chen et al. [2022] in the context of re-using Bayesian optimisation data, and more specifically by Yang et al. [2023] in the context of optimising over sequences. Pan et al. [2021], Savage et al. [2021], Petsagkourakis et al. [2022] explored the use of reinforcement learning for process optimization, also intersecting with Bayesian optimisation and model predictive control. The biggest difference between this branch of research and our work is that we focus on planning without approximate pre-training of a reinforcement learning agent, instead focusing on a formulation where we can calculate optimal policies without interacting with a simulated environment.

5.2 Visual abstract of MDP-BO

In this chapter we pull from the Bayesian optimisation, reinforcement learning, and multi-armed bandit literature. In order to clarify all connections, we include a visual abstract of the chapter in Figure 5.2.

5.3 Deriving a utility for maximiser identification

As discussed in Section 2.4, classical Bayesian optimisation is naturally myopic in its definition as a greedy one-step update (see (5.1)), but has the overall goal to minimise, e.g., the cumulative regret. Therefore acquisition functions are chosen to balance the exploration-exploitation trade-off so that overall non-myopic goals can be achieved.

In this chapter we follow similar ideas; however, our approach has strong connections to the bandits literature. Therefore we do not focus on regret but instead on gathering information to maximise our chances to identify $x^\star$, the maximiser of f. In order for this problem to be solvable, we will assume to be working on a finite search space $\mathcal{X}$, however, we will show the overall objective is also compatible with continuous state spaces.

For exposition purposes, we will be approximating the Gaussian processes via the finite dimensional approximations which we introduced in Section 2.2.3, so we can assume that:

$$f(x) = \theta^T \Phi(x)$$

where $\Phi(x)$ is the basis that defines the kernel, and $\theta \sim \text{Normal}(0, \mathbf{I})$.

5.3.1 Optimum identification via hypothesis testing

Maximum identification can naturally be expressed as a multiple hypothesis testing problem, where we need to determine which of the elements in $\mathcal{X}$ is the maximiser. To do so, we require

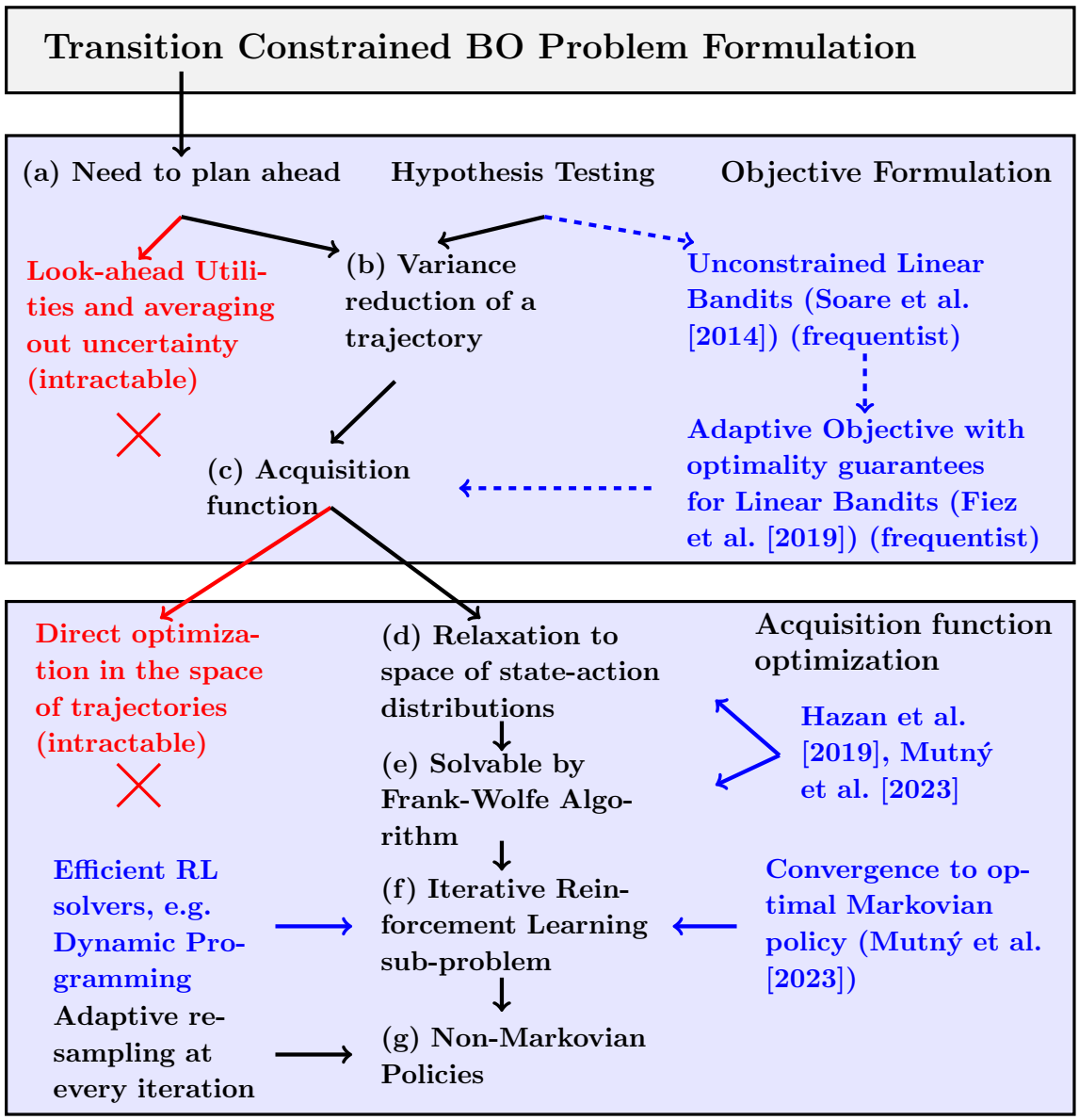


Figure 5.2: Visual abstract of the work. In black we show the method presented in this paper, with literature connections shown in blue. In red we show solutions which we did not pursue due to intractability. The problem creates the **(a) need to plan ahead**. To do this, we take inspiration from hypothesis testing and focus on **(b) the variance reduction in a set of maximisers**, which leads to our **(c) acquisition function**. The objective is the same as Fiez et al. [2019] introduced in the linear bandits literature from a frequentist perspective. To optimise it, we follow developments in Hazan et al. [2019], Mutný et al. [2023] by **(d) relaxing the acquisition function to the space of state-action distributions** and **(e) solving the planning problem using the Frank-Wolfe algorithm**. This consists of iteratively solving tractable **(f) reinforcement learning sub-problems** which give us optimal Markov policies. We then apply adaptive resampling to obtain **(g) non-Markovian policies**.

good estimates of the differences (or at least their signs) between individual queries $f(x_i) - f(x_j)$; $x_i, x_j \in \mathcal{X}$. For example, if $f(x_i) - f(x_j) \leq 0$, then x_i cannot be a maximiser. Given the current evidence, the set of arms which we cannot rule out are all *potential maximisers*, $\mathcal{Z} \subset \mathcal{X}$. At

termination we report our best guess for the maximiser as:

$$x_T = \arg\max_{x \in \mathcal{Z}} \mu_T(x), \qquad \text{where } \mu_T \text{ is the predictive mean at termination time } T.$$

Suppose we are in step t out of T, then let $\mathbf{X}_t$ be the set of previous queries, we seek to identify new $\mathbf{X}_{\text{new}}$ that when evaluated minimise the probability of returning a sub-optimal arm at the end. For a given function draw f, the probability of returning a wrong maximiser $z \neq x_f^\star$ is $P(\mu_T(z) - \mu_T(x_f^\star) \geq 0 | f)$. Note that since we are conditioning on the function f, then the only resulting randomness comes from the observation noise $y = f(x) + \epsilon$; under the finite dimensional approximation of the GP, our estimator $\hat{\theta}$ is simply a regularised least squares estimator and therefore is distributed as:

$$\hat{\theta}_T \sim \mathcal{N}(\theta^T(\mathbf{V}_T + \sigma^2 \mathbf{I}_{\mathcal{H}})^{-1}\mathbf{V}_T, \sigma^2(\mathbf{V}_T + \mathbf{I}_{\mathcal{H}})^{-1}\mathbf{V}_T(\mathbf{V}_T + \mathbf{I}_{\mathcal{H}})^{-1})$$

where $\mathbf{V}_T = \sum_{i=1}^T \Phi(x_i)\Phi(x_i)^T$, and we write $\mathbf{X}$ for $\Phi(\mathbf{X}_t \cup \mathbf{X}_{\text{new}})$ to avoid cluttering the equations and therefore we can calculate the distribution of $\mu_T(z) - \mu_T(x^*\!*)$:

$$\mathcal{N}(\underbrace{\theta^\top(\mathbf{V}_T + \mathbf{I}_{\mathcal{H}})^{-1}\mathbf{V}_T(\Phi(z) - \Phi(x_f^\star))}_{a}, \underbrace{\sigma^2(\Phi(z) - \Phi(x_f^\star)^\top(\mathbf{V}_T + \mathbf{I}_{\mathcal{H}})^{-1}\mathbf{V}_T(\mathbf{V}_T + \mathbf{I}_{\mathcal{H}})^{-1}(\Phi(z) - \Phi(x_f^\star)))}_{b^2}).$$

We can now bound the probability of making an error using a Gaussian tail bound inequality:

$$\mathbb{P}(\mu_T - \mu_T(x^*) \geq 0) = \mathbb{P}(\mu_T - \mu_T(x^*) \geq a_z + (-a_z)) \leq e^{-\frac{a^2}{b^2}}$$

with the caveat that the inequality only holds when the mean is negative. However note that $a_z \to f(z) - f(x^*) < 0$ as $T \to \infty$ therefore it will hold once T is large enough. From this we can take logarithms and then the expectation across the randomness in the GP:

$$\mathbb{E}_{f \sim GP}\left[\log \mathbb{P}(\mu_T(z) - \mu_T(x^*) | f)\right] \leq -\frac{1}{2}\mathbb{E}_{f \sim GP}\left[\frac{a_z^2}{b_z^2}\right]$$

Note that in the large T limit we have that $a_z \approx f(z) - f(x^*)$ and furthermore:

$$\begin{aligned} b_z^2 &= \sigma^2(\Phi(z) - \Phi(x_f^\star)^\top(\mathbf{V}_T + \mathbf{I}_\mathcal{H})^{-1}\mathbf{V}_T(\mathbf{V}_T + \mathbf{I}_\mathcal{H})^{-1}(\Phi(z) - \Phi(x_f^\star)) \\ &\leq \sigma^2(\Phi(z) - \Phi(x^*))^T(\mathbf{V}_T + \sigma^2\mathbf{I})^{-1}(\Phi(z) - \Phi(x^*)) = k_\mathbf{X}(z, x^*) \end{aligned}$$

where the inequality holds since $\mathbf{V}_T = \mathbf{X}^T\mathbf{X}$, and the equality holds from the definition of the posterior kernel. This finally leads to the inequality:

$$\mathbb{E}_{f\sim GP}\left[\log \mathbb{P}(\mu_T(z) - \mu_T(x^*)|f)\right] \leq -\frac{1}{2}\mathbb{E}_{f\sim GP}\left[\frac{(f(z) - f(x^*))^2}{k_\mathbf{X}(z, x^*)}\right].$$

Note the intuitive interpretation of the bound, the probability of making an error is controlled by how close the underlying function values are to the optimum, and we can further control it by reducing the uncertainty. We now want to minimise the worst-case probability of making a mistake and we do this by considering a supremum before constructing the exact same argument above, obtaining:

$$\mathbb{E}_{f\sim GP}\left[\sup_{z\in\mathcal{Z}\setminus\{x^*\}} \log \mathbb{P}(\mu_T(z) - \mu_T(x^*)|f)\right] \leq -\frac{1}{2}\mathbb{E}_{f\sim GP}\left[\sup_{z\in\mathcal{Z}\setminus\{x^*\}} \frac{(f(z) - f(x^*))^2}{k_\mathbf{X}(z, x^*)}\right]. \tag{5.3}$$

Finally to reach our objective of interest we must get rid of the expectation. We do this by considering the smallest possible gap between two function values (since we are dealing with a discrete number of values), $\text{gap}(f)$:

$$-\frac{1}{2}\mathbb{E}_{f\sim GP}\left[\sup_{z\in\mathcal{Z}\setminus\{x^*\}} \frac{(f(z) - f(x^*))^2}{k_\mathbf{X}(z, x^*)}\right] \leq -\frac{\mathbb{E}_{f\sim GP}[\text{gap}(f)^2]}{2} \sup_{z\in\mathcal{Z}\setminus\{x^*\}} \frac{1}{\text{Var}[f(z) - f(x_*)|\mathbf{X}]}.$$

In particular, to minimise the probability of an error we can just minimise the variance in the denominator instead (since $\arg\min_x -g(x)$ is equivalent to $\arg\min_x \frac{1}{g(x)}$ when $g(x) > 0$). However, the non-trivial distribution of $f(x^\star)$ [Hennig and Schuler, 2012] renders the utility intractable; therefore we employ a simple and tractable upper bound on the objective by min-

imising the uncertainty among *all* pairs in $\mathcal{Z}$:

$$U(\mathbf{X}_{\text{new}}) = \max_{z', z \in \mathcal{Z}, z \neq z'} \text{Var}[f(z) - f(z') | \mathbf{X}_t \cup \mathbf{X}_{\text{new}}]. \tag{5.4}$$

This objectives can be solved greedily in a similar way as acquisition functions in Eq. (5.1) by minimising U over $\mathbf{X}_{\text{new}}$. Note that Fiez et al. [2019] derive this objective for the same problem with linear bandits, albeit they consider the frequentist setting and (surprisingly) a different optimality criterion: minimising T for a given failure rate. For their setting, the authors prove that it is an asymptotically optimal objective to follow. They do not consider any Markov chain structure.

5.3.2 Utility with Embeddings

For illustrative purposes, consider a special case where the kernel k has a low rank due to existence of embeddings $\Phi(x) \in \mathbb{R}^m$, i.e., $k(x, y) = \Phi(x)^\top \Phi(y)$. Such embeddings can be, e.g., Nyström features [Williams and Seeger, 2000] or Fourier features [Mutný and Krause, 2018, Rahimi and Recht, 2007] as introduced in Section 2.2.3. While not necessary, these formulations make the objectives considered in this work more tractable and easier to deal with in the following sections. We can then rewrite the objective Eq. (5.4) as:

$$U(\mathbf{X}_{\text{new}}) = \max_{z, z' \in \mathcal{Z}} ||\Phi(z) - \Phi(z')||^2_{\left(\sum_{x \in \mathbf{X}_t \cup \mathbf{X}_{\text{new}}} \frac{\Phi(x)\Phi(x)^\top}{\sigma^2} + \mathbf{I}\right)^{-1}}. \tag{5.5}$$

This reveals an essential observation that the utility depends only on the visited states; not their order. This suggests a vast simplification, where we do not need to model whole trajectories, and Markov decision processes sufficiently describe our problem. Additionally, numerically, the objective involves the inversion of an $m \times m$ matrix instead of $|\mathcal{X}| \times |\mathcal{X}|$ (see Sec. 5.7).

5.4 The State-Action Visitation Polytope

To model the transition constraints, we use the versatile model of Markov Decision processes (MDPs). We assume an environment with state space $\mathcal{X}$ and action space $\mathcal{A}$, where we interact with an unknown function $f : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ by rolling out a policy for H time-steps (horizon) and obtain a trajectory, $\tau = (x_0, a_0, x_1, a_1, ..., x_{H-1}, a_{H-1})$. From the trajectory, we obtain a sequence of noisy observations $y(\tau) := \{y(x_0, a_0), ..., y(x_{H-1}, a_{H-1})\}$ s.t. $y(x_h) = f(x_h, a_h) + \epsilon(x_h, a_h)$, where $\epsilon(x_h, a_h)$ is zero-mean Gaussian with known variance which is potentially state and action dependent. The trajectory is generated using a *known* transition operator $P(x_{h+1}|x_h, a_h)$. A Markov policy $\pi(a_h|x_h)$ is a mapping that dictates the probability of action a_h in state h. Hence, the state-to-state transitions are $P(x_{h+1}, x_h) = \sum_{a \in \mathcal{A}} \pi_h(a|x_h) P(x_{h+1}|x_h, a)$. In fact, an equivalent description of any Markov policy π is in how often it is expected to visit corresponding states and actions, this follows a distribution which we denote $d \in \mathcal{D}$, where

$$\mathcal{D} := \Big\{ \forall h \in [H] \; d_h \mid d_h(x, a) \geq 0, \; \sum_{a,x} d_h(x, a) = 1, \; \sum_{a} d_h(x', a) = \sum_{x,a} d_{h-1}(x, a) p(x'|x, a) \Big\}.$$

We will use this state-action polytope to reformulate our optimisation problem from policies to state-action distributions. Any $d \in \mathcal{D}$ can be realised by a Markov policy π and vice-versa. We work with non-stationary policies, meaning the policies depend on horizon count h. The execution of deterministic trajectories is only possible for deterministic transitions. Otherwise, the resulting trajectories are random. In our setup, we repeat interactions T times (episodes) to obtain the final dataset of the form $\mathbf{X}_T = \{\tau_i\}_{i=1}^T$.

In the simplest, and most common case, we can think of the action space $\mathcal{A}$ as equal to the state-space, $\mathcal{X}$, where the action at time t simply represents the next state from which we obtain an observation $a_t = x_{t+1}$. Because of this, we will frequently simplify the state-action distribution and only talk of state distributions. However, the framework we develop works in more general scenarios and the action space need not be so restrictive.

Examples of more complicated actions spaces include scenarios where we might be able to reach the same state in different ways, for example, in a flow reactor we are able to generate

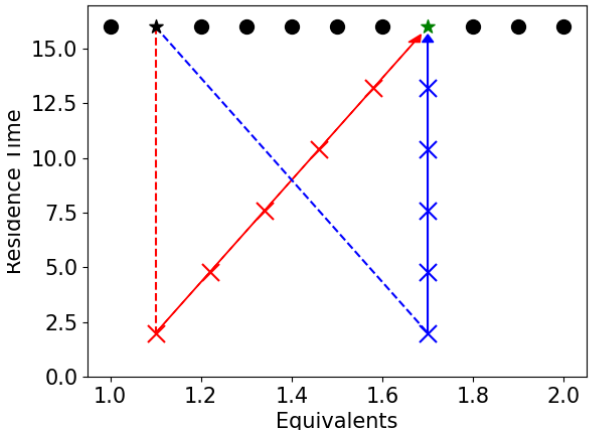


Figure 5.3: We can see the state space as black dots, the current state as a black star and the next state as a green star. In a transient flow reactor we can preload the reactor with different equivalents, resulting in differing sets of measurements when doing residence time ramps. The blue and red paths represent two different ways of moving from the black star to the green star, and the resulting measurements are displayed as crosses. For this action space we choose, not only the next state (green star) but which path to take there (blue or red path).

multivariate ramps in residence time and equivalents which end up at the same state, but generate a different set of measurements. Figure 5.3 showcases this example in more detail.

Other examples can include directions of movement, for example, if the state-space is a grid, the action space could be to move right, left, up, or down, resulting in transitions of the form:

$$x_{t+1} = x_t + a_t$$

As we will see in Section 5.8, we take advantage of this formulation to tractably generalize our framework to continuos MDPs.

5.5 Experiment Design in Markov Chains

Notice that the utility U in Eq.(5.5) depends on the states visited and hence states of the trajectory. In our notation, $\mathbf{X}_t$ will now form a set of executed trajectories. With deterministic dynamics, we could optimise over trajectories, but this would lead to an exponential blowup (i.e. $|S|^H$). In fact, for stochastic transitions, we cannot pick the trajectories directly and the

utility would be random $U(\mathbf{X}_t)$. Hence, we focus on expected utility over the randomness of the policy and the environment, namely,

$$\mathcal{U}(d_\pi) := U(\mathbb{E}_{\tau_1 \sim \pi_1, \ldots \tau_t \sim \pi_t}[\hat{d}_{\mathbf{X}_t}]) \tag{5.6}$$

where $\hat{d}_{\mathbf{X}_t}$, is the empirical state-distribution of the trajectory X. That is, a distribution over states with equal mass in each state visited during trajectory X. This formulation stems from Mutný et al. [2023] who try to tractably solve such objectives that arise in experiment design by performing planning in MDPs. They focus on learning linear operators of an unknown function, unlike identifying a maximum, as we do here. The key observation they make is that any policy π induces a distribution over the state-action visitations, d_π. Therefore we can reformulate the problem of finding the optimal policy, into finding the optimal distribution over state-action visitations as: $\min_{d_\pi \in \mathcal{D}} \mathcal{U}(d_\pi)$, and then construct policy π via the marginalisation:

$$\pi(a|x) = \frac{d(x,a)}{\sum_{a' \in \mathcal{A}} d(x,a')} \tag{5.7}$$

We refer to the optimisation of finding the optimal state-action distribution as the *planning problem*. The constraint $\mathcal{D}$ encodes the dynamics of the MDP.

5.6 Transition Constrained Bayesian Optimisation

This section introduces Bayesian optimisation with transition constraints. We use MDPs to encode constraints. Namely, the Markov dynamics dictates which inputs we are allowed to query at time-step $h+1$ given we previously queried state x_h. This mean that the transition operator is $P(x_{h+1}|x_h, a) = 0$ for any transition $x_h \to x_{h+1}$ not allowed by the physical constraints.

Motivated by our practical experiments with chemical reactors, we distinguish two different types of *feedback*.

- **Episodic feedback** occurs when we can be split the optimisation into episodes. At the end of each episode of length H, we obtain the whole set of noisy observations.

- **Instant feedback** is the setting where we obtain a noisy observation immediately after querying the function.

Asynchronous feedback describes a mix of previous two, where we obtain observations within the episodes but under a time-delay.

5.6.1 Expected Utility for Maximiser Identification

In section 5.3 we introduced the utility for maximum identification. Using the same simplifying assumption (finite rank approximation of GPs in Sec. 5.3.2, Eq. (5.4)), we can show that the expected utility $\mathcal{U}$ can be rewritten in terms of the state-action distribution induced by $\mathbf{X}_{\text{new}}$:

$$\mathcal{U}(d_\pi) = \max_{z,z' \in \mathcal{Z}} ||\Phi(z) - \Phi(z')||^2_{\mathbf{V}(d_\pi)^{-1}} \tag{5.8}$$

where $\mathbf{V}(d_\pi) = \left(\sum_{x,a \in \mathcal{X}\times\mathcal{A}} \frac{d_\pi(x,a)\Phi(x,a)\Phi(x,a)^\top}{\sigma^2(x,a)} + \frac{1}{TH}\mathbf{I}\right)$. The variable $d_\pi(x,a)$ is a state-action visitation, $\Phi(x)$ are e.g. Nyström features of the GP. The proof that the function is additive in terms of state-action pairs in Lemma B.1 is included in Appendix B.1.1 (not included in the main body of the thesis as it was developed by Mojmír Mutný), a condition required for the expression as a function of state-action visitations [Mutný et al., 2023]. Additionally, by rewriting the objective in this form, the dependence and convexity with respect to the state-action density d_π becomes clear as it is only composition of a linear function with an inverse operator. Also, notice that the constraint set is a convex polytope. Therefore we are able to use convex optimisation to solve the planning problem (see Sec. 5.7).

5.6.2 Set of potential maximisers $\mathcal{Z}$.

The definition of the objective requires the use of a set of maximisers. In the ideal case, we can say a particular input x, is not the optimum if there exists x' such that $f(x') > f(x)$ with high

confidence. We formalise this using the GP credible sets (Bayesian confidence sets) and define:

$$\mathcal{Z}_t = \{x \in \mathcal{X} : \text{UCB}(f(x)|\mathbf{X}_t) \geq \sup_{x' \in \mathcal{X}} \text{LCB}(f(x')|\mathbf{X}_t)\} \tag{5.9}$$

where UCB and LCB correspond to the upper and lower confidence bounds of the GP surrogate with a user specified confidence level defined via the posterior GP with data up to $\mathbf{X}_t$.

5.6.3 Discrete vs Continuous MDPs.

Until this point, our formulation focused on discrete $\mathcal{S}$ and $\mathcal{A}$ for ease of exposition. However, the framework is compatible with continuous state-action spaces. The probabilistic reformulation of the objective in Eq. (5.6) is possible irrespective of whether $\mathcal{X}$ (or $\mathcal{A}$) is a discrete or continuous subset of $\mathbb{R}^d$. In fact, the convexity of the objective in the space of distributions is still maintained. The difference is that the visitations d are no longer probability mass functions but have to be expressed as probability density functions $d_c(x, a)$. To recover probabilities in the definition of $\mathbf{V}$, we need to replace sums with integrals i.e.:

$$\sum_{x \in \mathcal{X}, a \in \mathcal{A}} d(x,a) \frac{\Phi(x,a)\Phi(x,a)^\top}{\sigma(x,a)^2} \rightarrow \int_{x \in \mathcal{X}, a \in \mathcal{A}} d_c(x,a) \frac{\Phi(x,a)\Phi(x,a)^\top}{\sigma(x,a)^2}.$$

In the Eq. (5.8) we need to approximate a maximum over all input pairs in $\mathcal{Z}$. While this can be enumerated in the discrete case without issues, it poses a non-trivial constrained optimisation problem when $\mathcal{X}$ is continuous. As an alternative, we propose approximating the set $\mathcal{Z}$ using a finite approximation of size K which can be built using Thompson Sampling [Kandasamy et al., 2018]:

$$\mathcal{Z}_{cont}^{(TS)} = \left\{ \arg\max_{x \in \mathcal{X}_c} f_i(x) : f_i \sim \mathcal{GP}(\mu_t, \sigma_t) \right\}_{i=1}^{K}$$

where K is a new hyper-parameter influencing the accuracy of the approximation of $\mathcal{Z}$. We found that the algorithm could be too exploratory in certain scenarios. Therefore, we also propose an alternative that encourages exploitation by guiding the maximisation set using Bayesian

Algorithm 10 Transition Constrained Bayesian Optimisation via MDPs

Input: Procedure for estimating sets of maximisers, initial point x_0, initial set of maximisers $\mathcal{Z}_0$
Initialise the empirical state-action distribution $\hat{d}_0 = 0$
for $t = 0$ **to** $T - 1$ **do**
  for $h = 0$ **to** $H - 1$ **do**
    $\mathcal{U}_{t,h}(d_\pi) \leftarrow \mathcal{U}(d_\pi \oplus \hat{d}_{t,h} | \mathcal{Z}_{t,h}, x_{t,h})$ *// define the objective, see eq. (5.8)*
    $\pi_{t,h} = \arg\min_{\pi : d_\pi \in \mathcal{D}_{t,h}} \mathcal{U}_{t,h}(d_\pi)$ *// solve MDP planning problem*
    $x_{t,h+1} = \pi_{t,h}(x_{t,h})$ *// deploy policy*
    if feedback is immediate **then**
      $y_{t,h+1} = f(x_{t,h+1}) + \epsilon_{t,h};\ \mathcal{GP}_{t,h},\ \mathcal{Z}_{t,h} \leftarrow \mathrm{Update}(\mathbf{X}_{t,h}, \mathbf{Y}_{t,h})$ *// update model*
    $\hat{d}_{t,h+1}(x) \leftarrow \hat{d}_{t,h} \oplus \delta(x_{t,h+1}, x)$ *// update empirical state-action distribution, see eq. (5.11)*
  if feedback is episodic **then**
    $\mathbf{Y}_{t,H} = f(\mathbf{X}_{t,H}) + \vec{\epsilon}_{t,:};\ \mathcal{GP}_{t+1,:},\ \mathcal{Z}_{t+1,:} \leftarrow \mathrm{Update}(\mathbf{X}_{t,H}, \mathbf{Y}_{t,H})$ *// update model*
Return: Estimate of the maximum using the GP posterior's mean $\hat{x}_* = \arg\max_{x \in \mathcal{X}} \mu_T(x)$

optimisation through the UCB acquisition function [Srinivas et al., 2010]:

$$\mathcal{Z}_{cont}^{(UCB)} = \left\{ \arg\max_{x \in \mathcal{X}_c} \mu_t(x) + \beta_i \sigma_t(x) : \beta_i \in \mathcal{B} \right\}_{i=1}^{K}$$

where $\mathcal{B} = \texttt{linspace(0, 2.5, K)}$ which serves as scaling for the size of set $\mathcal{Z}_{cont}^{(UCB)}$. Both cases reduce optimisation over $\mathcal{Z}$ to enumeration as with discrete cases.

In Section 5.9.5, we numerically benchmark reasonable choices of K, and show that the performance is not significantly affected by them.

5.6.4 General algorithm and Theory

The general algorithm combines the ideas introduced so far. We present it in Algorithm 10. Notice that apart from constructing the current utility, keeping track of the visited states and updating our GP model, an essential step is *planning*, where we need to find a policy that maximises the utility. As this forms the core challenge of the algorithm, we devote Sec. 5.7 to it. In short, it solves a sequence of dynamic programming problems defined by the steps of the Frank-Wolfe algorithm. From a theoretical point of view, under the assumption of episodic feedback, the algorithm provably minimises the utility as we show in Proposition B.3 in Appendix B.1.2.

5.7 Solving the planning problem

The planning problem, defined as $\min_{d_\pi \in \mathcal{D}} \mathcal{U}(d_\pi)$, can be thought of as analogous to optimising an acquisition function in traditional Bayesian optimisation, with the added difficulty of doing it in the space of policies. To solve it, following developments in Hazan et al. [2019] and Mutný et al. [2023], we use the classical Frank-Wolfe algorithm [Jaggi, 2013]. The Frank-Wolfe algorithm is a first order optimisation algorithm for constrained convex optimisation. We state the algorithm in the simplest optimization case over $x \in \mathbb{R}^d$ in Algorithm 11.

Algorithm 11 Frank-Wolfe Algorithm

$x^{(0)} \in \mathcal{D}$
for k = 0, ..., K **do**
  $s = \arg\min_{s \in \mathcal{D}} s^T \nabla f(x^{(k)})$ *// Solve a constrained sub-problem with linear objective*
  $x^{(k+1)} = (1 - \alpha)x^{(k)} + \alpha s$ *// Update the state*

In the simplest case we solve a sequence of sub-problems by considering linear objectives in the constrained space, and then update our approximation of the optimum. Hazan et al. [2019] show that this procedure can also be done to optimise over the space of state-action distributions.

In our case, each linearisation results in a policy, and we build a mixture policy consisting of optimal policies for each linearisation $\pi_{\text{mix},n} = \{(\alpha_i, \pi_i)\}_{i=1}^n$, and α_i step-sizes of Frank-Wolfe. Conveniently, after the linearisation of $\mathcal{U}$ the sub-problem on the polytope $\mathcal{D}$ corresponds to an RL problem with reward $\nabla\mathcal{U}$ for which many efficient solvers exist. Namely, for a single mixture component we have,

$$d_{\pi_{n+1}} = \arg\min_{d \in \mathcal{D}} \sum_{x,a,h} \nabla\mathcal{U}(d_{\pi_{\text{mix},n}})(x,a) d_h(x,a). \tag{5.10}$$

Due to convexity, the state-action distribution follows the convex combination, $d_{\pi_{\text{mix},n}} = \sum_{i=1}^n \alpha_i d_{\pi_i}$. The optimisation produces a Markovian policy due to the subproblem in Eq. (5.10) being optimised by one. We now detail how to construct a non-Markovian policies by adaptive resampling.

5.7.1 Adaptive Resampling: Non-Markovian policies.

A core contribution of our paper is receding horizon re-planning. This means that we keep track of the past states visited in the *current* and *past* trajectories and adjust the policy at every step h of the horizon H in each trajectory indexed by t. At h, we construct a Markov policy for a reward that depends on all past visited states. This makes the resulting policy history dependent. While in episode t and time-point h we follow a Markov policy for a single step, the overall policy is a history-dependent non-Markov policy.

We define the empirical state-action visitation distribution,

$$\hat{d}_{t,h} = \frac{1}{tH+h} \underbrace{(\sum_{j=1}^{t} \sum_{x,a \in \tau_j} \delta_{x,a}}_{\text{visited states in past trajectories}} + \underbrace{\sum_{x,a \in \tau_t|_h} \delta_{x,a})}_{\text{states at ep. } t \text{ up to } h} \tag{5.11}$$

where $\delta_{x,a}$ denotes a delta mass at state-action (x, a). Instead of solving the objective $\mathcal{U}(d)$ as in Eq. (5.10), we seek to find a correction to the empirical distribution by minimising,

$$\mathcal{U}_{t,h}(d) = \mathcal{U}\left(\frac{1}{H}\left(\frac{H-h}{1+t}d + \frac{tH+h}{1+t}\hat{d}_{t,h}\right)\right). \tag{5.12}$$

We use the same Frank-Wolfe machinery to optimise this objective: $d_{\pi_{t,h}} = \arg\min_{d_\pi \in \tilde{\mathcal{D}}} \mathcal{U}_{t,h}(d_\pi)$. The distribution $d_{\pi_{t,h}}$ represents the density of the policy to be deployed at trajectory t and horizon counter h. We now need to solve multiple (n due to FW) RL problems at each horizon counter h. Despite this, for discrete MDPs, the sub-problem can be solved extremely efficiently to exactness using dynamic programming. As can be seen in Section 5.9.6, our solving times are just a few seconds, even if planning for very long horizons. The resulting policy π can be found by marginalisation $\pi_h(a|x) = d_{\pi,h}(x, a)/\sum_a d_{\pi,h}(x, a)$, a basic property of MDPs [Puterman, 2014].

5.8 Continuous MDPs: Model Predictive Control

With continuous search space, the sub-problem can be solved using continuous RL solvers. However, this can be difficult. The intractable part of the problem is that the distribution d_π needs to be represented in a computable fashion. We represent the distribution by the sequence of actions taken $\{a_h\}_{h=1}^T$ with the linear state-space model, $x_{h+1} = Ax_h + Ba_h$. While this formalism is not as general as it could be, it gives us a tractable sub-problem formulation common to control science scenario [Rawlings et al., 2017] that is practical for our experiment and captures a vast array of problems. The optimal set of actions is solved with the following problem, where we state it for the full horizon H:

$$\arg\min_{a_0,\dots,a_H} \sum_{h=0}^{H} \nabla \mathcal{U}_{t,0}(d_{\pi_{\text{mix},t}})\,(x_h, a_h)\,, \tag{5.13}$$

such that $||a_h|| \leq a_{\max}, ||x_h|| \leq 0.5,$ and $x_{h+1} = Ax_h + Ba_h$, where the *known* dynamics serves as constraints. Notice that instead of optimising over the policy d_π, we directly optimise over the parameterisations of the policy $\{a_h\}_{h=1}^H$. In fact, this formulation is reminiscent of the model predictive control (MPC) optimisation problem. Conceptually, these are the same. The only caveat in our case is that unlike in MPC [García et al., 1989], our objective is non-convex and tends to focus on gathering information rather than stability. Due to the non-convexity in this parameterisation, we need to solve it heuristically.

5.8.1 Linearising the objective

To apply our method, we find ourselves having to frequently solve RL sub-problems where we try to maximise $\sum_{x,a} d(x,a)\nabla U(x,a)$. To approximately solve this problem in the continuous setting, it becomes very important to understand what the linearised functional looks like.

Remark 5.1. *Assuming that $f \sim \mathcal{GP}(0,k)$ where k has Mercer decomposition as $k(x,y) = \lambda_k \phi_k(x)\phi_k(y)$, so that*

$$f(x) = \sum_k \phi_k(x)\theta_k \quad \theta_k \sim \mathcal{N}(0, \lambda_k),$$

and further assume that we have a mixture of policies π_{mix} *with density* $d_{\pi_{\text{mix}}}$*, such that there exists a set* $\mathbf{X}_{\text{mix}}$ *satisfying* $d_{\pi_{\text{mix}}} = \frac{1}{N}\sum_{x\in\mathbf{X}_{\text{mix}}}\delta_x$ *for some integer* N*. Then:*

$$\nabla U(d_{\pi_{\text{mix}}})(x,a) \propto -\left(Cov[f(z_*), f(x)|\mathbf{X}_{\text{mix}}] - Cov[f(z'_*), f(x)|\mathbf{X}_{\text{mix}}]\right)^2$$

where $z_*, z'_* = \arg\max_{z,z'\in\mathcal{Z}} Var[f(z) - f(z')]$.

In particular, by understanding the linarised objective we are able to identify where in the space it is maximised. Therefore we will use this to build a heuristic that allows us to approximately solve the planning problem for continuous domains.

Proof. To show this, we begin by defining:

$$\begin{aligned}
\Sigma_{\theta,d} &= \left(\sum_{x\in\mathcal{X}} \Phi(x)\Phi(x)^T d(x) + \frac{\sigma^2}{TH} I\right)^{-1} \\
z_*, z'_* &= \underset{z,z'\in\mathcal{Z}}{\arg\max} ||\Phi(z) - \Phi(z')||^2_{\Sigma_{\theta,d}} \\
\tilde{z}_* &= \Phi(z_*) - \Phi(z'_*)
\end{aligned}$$

It then follows, by applying Danskin's Theorem that:

$$\begin{aligned}
\nabla U(d)(x) &= \nabla \tilde{z}_*^T \Sigma_{\theta,d} \tilde{z}_* \\
&= \nabla \text{Tr}\left\{\tilde{z}_* \tilde{z}_*^T \Sigma_{\theta,d}\right\} \\
&= \text{Tr}\left\{\tilde{z}_* \tilde{z}_*^T \nabla \Sigma_{\theta,d}\right\} \\
&= -\text{Tr}\left\{\tilde{z}_* \tilde{z}_*^T \Sigma_{\theta,d} \Phi(x)\Phi(x)^T \Sigma_{\theta,d}\right\} \qquad (\text{as } \partial K^{-1} = -K^{-1}(\partial K)K^{-1}) \\
&= -\text{Tr}\left\{\tilde{z}_*^T \Sigma_{\theta,d} \Phi(x)\Phi(x)^T \Sigma_{\theta,d} \tilde{z}_*\right\} \\
&= -\left(\tilde{z}_*^T \Sigma_{\theta,d} \Phi(x)\right)\left(\Phi(x)^T \Sigma_{\theta,d} \tilde{z}_*\right) \\
&\propto -\left(\text{Cov}[f(z_*), f(x)] - \text{Cov}[f(z'_*), f(x)]\right)\left(\text{Cov}[f(x), f(z_*)] - \text{Cov}[f(x), f(z'_*)]\right) \\
&= -\left(\text{Cov}[f(z_*), f(x)] - \text{Cov}[f(z'_*), f(x)]\right)^2
\end{aligned}$$

□

5.8.2 Practical Planning for Continuous MDPs

From remark 5.1 it becomes clear that for decreasing covariance functions, such as the squared exponential, ∇U will consist of two modes around z_* and z'_* The sub-problem seems to find a sequence that maximises the *sum* of gradients, therefore the optimal solution will try to reach one of the two modes as quickly as possible. For shorter time horizons, the path will reach whichever mode is closest, and for large enough horizons, the sum will be maximised by reaching the larger of the two modes.

Therefore we can approximately solve the problem by checking the value of the sub-problem objective in (5.13) for the shortest paths from $x_{t-1} \to z_*$ and $x_{t-1} \to z'_*$, which are trivial to find under the constraints in (5.13). Note that the paths might not necessarily be optimal, as they may be improved by small perturbations, e.g., there might be a small deviation that allows us to visit the smaller mode on the way to the larger mode increasing the overall value of the sum of gradients, however, they give us a good and quick approximation.

5.9 Experiments

Sections 5.9.1 – 5.9.3 showcase real-world applications under physical transitions constraints, using the discrete version of the algorithm. Section 5.9.4 benchmarks against other algorithms in the continuous setting, where we consider the additive transition model of Section 5.8 with $A = B = \mathbf{I}$. For each benchmark, we selected reasonable GP hyper-parameters and fixed them during the optimisation. These are summarised in Appendix A.3.1. As we are interested maximiser identification, in discrete problems, we report the proportion of reruns that succeed at identifying the true maximum. For continuous benchmarks, we report inference regret at each iteration: $\text{Regret}_t = f(x_*) - f(x_{\mu,t})$, where $x_{\mu,t} = \arg\max_{x \in \mathcal{X}} \mu_t(x)$.

Baselines. We include a naive baseline that greedily optimises the immediate reward to show-

case a method with no planning (Greedy-UCB). Likewise, we create a baseline that replaces the gradient in Eq. (5.10) with Expected Improvement (MDP-EI) a weak version of planning. In the continuous settings, we compare against truncated SnAKe (TrSnaKe) [Folch et al., 2023b], which minimises movement distance, and against local search region-constrained Bayesian optimisation or LSR [Paulson et al., 2023] for the same task. We compare two variants for approximating the set of maximisers, one using Thompson Sampling (MDP-BO-TS) and one using Upper Confidence Bound (MDP-BO-UCB).

5.9.1 Knorr pyrazole synthesis

Our chemical reactor benchmark synthetises Knorr pyrzole in a transient flow reactor. In this experiment, we can control the flow-rate (residence time) τ and ratio of reactants B in the reactor. We observe product concentration at discrete time intervals and we can also change inputs at these intervals. Our goal is to find the best parameters of the reaction subject to natural movement constraints on B, and τ. In addition, we assume *decreasing* the flow rate of a reactor can be easily achieved. However, *increasing* the flow rate can lead to inaccurate readings [Schrecker et al., 2024]. A lower flow rate leads to higher residence time, so we impose that τ must be non-decreasing.

The kernel. Schrecker et al. [2023] indicates the reaction can be approximately represented by simple kinetics via a differential equation model. We use this information along with techniques for representing linear ODE as constraints in GP fitting [Mutný and Krause, 2022, Besginow and Lange-Hegermann, 2022] to create an approximate ODE kernel k_{ode}, of which details are left to Appendix B.2 and may be of independent interest.

The ODEs in the reaction kinetics are nonlinear, but we can linearise them and build an embedding whose inner product defines a kernel:

$$\Phi_{ode}(\tau, B) = (1 - \mathcal{S}(B))y^{(1)}(\tau, B) + \mathcal{S}(B)y^{(2)}(\tau, B)$$

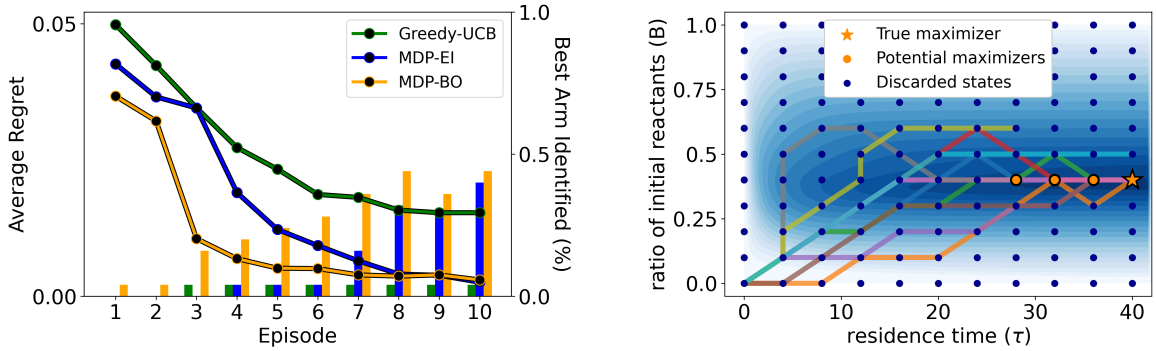


Figure 5.4: The Knorr pyrazole synthesis experiment. On the left, we show the quantitative results. The line plots denote the best prediction regret, while the bar charts denote the percentage of runs that correctly identify the best arm at the end of each episode. On the right, we show ten paths in different colours chosen by the algorithm. The underlying black-box function is shown as the contours, and we can see the discretisation as dots. We can see four remaining potential maximisers (in orange), which includes the true one (star). *Notice all paths are non-decreasing in residence time, following the transition constraints.*

where $y^{(i)}(\tau, B)$ are equal to:

$$\gamma_i(B)\left(\frac{\lambda_2^{(i)}}{\lambda_1^{(i)}-\lambda_2^{(i)}}e^{\lambda_1^{(i)}\tau}-\frac{\lambda_1^{(i)}}{\lambda_1^{(i)}-\lambda_2^{(i)}}e^{\lambda_2^{(i)}\tau}+1\right)$$

for $i = 1, 2$, where $\lambda_1^{(i)}$ and $\lambda_2^{(i)}$ are eigenvalues of the linearised ODE at different stationary points, $\gamma_1(B) = B$, $\gamma_2(B) = 1 - B$, and $\mathcal{S}(x) := (1 + e^{-\alpha_{sig}(x-0.5)})^{-1}$ is a sigmoid function. Appendix B.2 holds the details and derivations. Using Φ_{ode} it is then trivial to obtain a kernel which gives us GP samples approximately satisfying the ODE, k_{ode}.

As the above kernel is only an approximation of the true ODE kernel, which itself is imperfect, we must account for the model mismatch. Therefore, we add a squared exponential term to the kernel to ensure a non-parametric correction, i.e.: $k(\tau_R, B) = \alpha_{ode}k_{ode}(\tau_R, B) + \alpha_{rbf}k_{rbf}(\tau_R, B)$.

We report the examples of the trajectories in the search space in Figure 5.4. Notice that all satisfy the transition constraints. The paths are not space-filling and avoid sub-optimal areas because of our choice of non-isotropic kernel based on the ODE considerations. We run the experiment with episodic feedback, for 10 episodes of length 10 each, starting each episode with $(\tau_R, B) = (0, 0)$. Figure 5.4 reports quantitative results and shows that the best-performing algorithm is MDP-BO.

5.9.2 Monitoring Lake Ypacarai

Samaniego et al. [2021] investigated automatic monitoring of Lake Ypacarai, and Folch et al. [2022] and Yang et al. [2023] benchmarked different Bayesian optimisation algorithms for the task of finding the largest contamination source in the lake. We introduce local transition constraints to this benchmark by creating the lake containing obstacles that limit movement (see Figure 5.5 in the Appendix). Such obstacles in environmental monitoring may include islands or protected areas for animals. We add initial and final state constraints with the goal to model that the boat has to finish at the maintenance port.

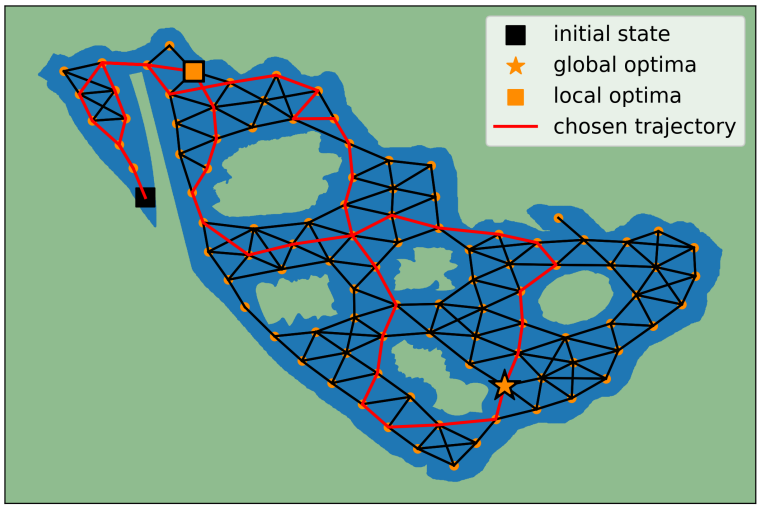
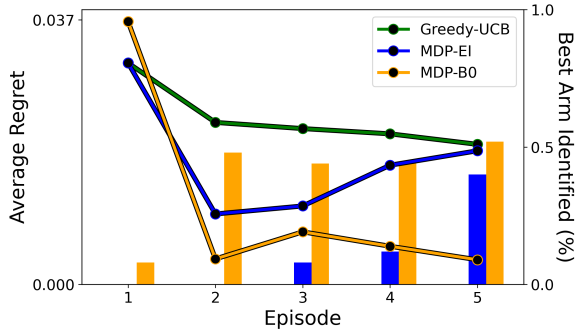


Figure 5.5: Lake Ypacari with the added movement constraints. We show one local optimum and one global one. The constraints of the problem requiring beginning and ending the optimisation in the dark square.

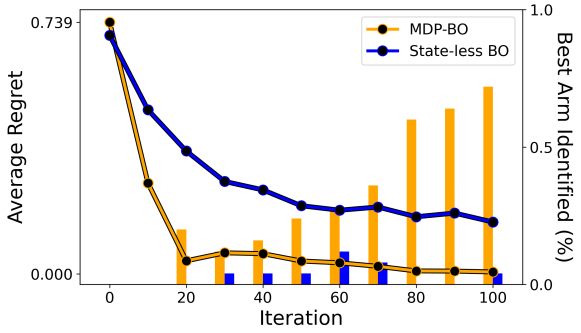
We focus on *episodic* feedback, where each episode consists of 50 iterations. Results can be seen in Figure 5.6a. MDP-EI struggles to identify the maximum contamination for the first few episodes. On the other hand, our method correctly identifies the maximum in approximately 50% of the runs by episode two and achieves better regret.

5.9.3 Free-electron laser: Transition-driven corruption

Apart from hard constraints, we can apply our framework to state-dependent Bayesian optimisation problems involving transitions. For example, the magnitude of noise ϵ may depend on



(a) Monitoring Lake Ypacarai.



(b) Free-electron laser tuning.

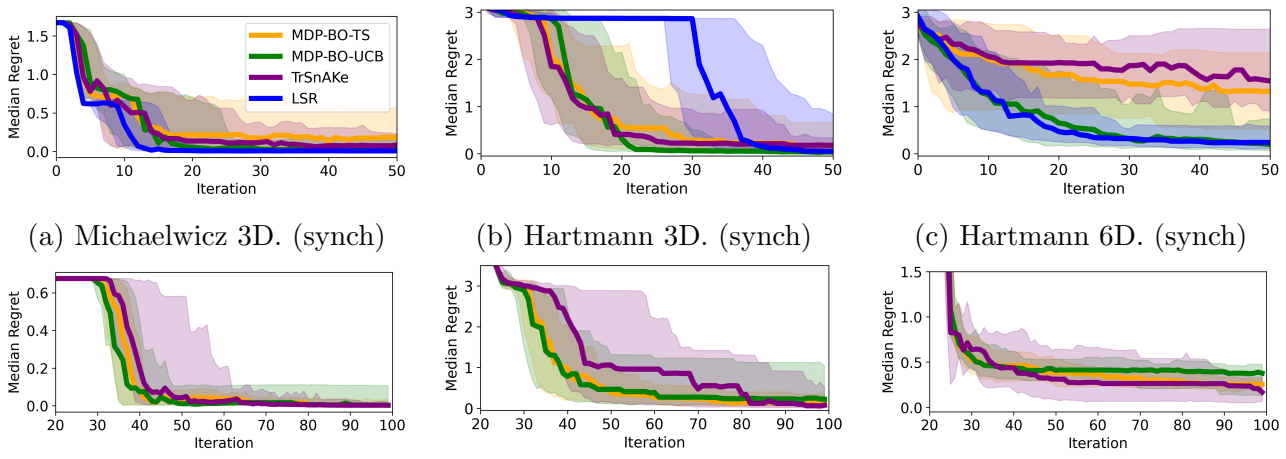
Figure 5.6: Results for Ypacarai and free electron-laser tuning experiments. On the left, the line plots denote the best prediction regret, while the bar charts denote the percentage of runs that correctly identify the best arm at the end of each episode. On the right, We plot the regret and compare against standard BO without accounting for movement-dependent noise.

the transition. This occurs in systems observing equilibration constraints such as a free-electron laser [Kirschner et al., 2022]. Using the simplified simulator of this laser [Mutný et al., 2020], we use our framework to model heteroscedastic noise depending on the difference between the current and next state, $\sigma^2(x, x') = s(1 + w||x - x'||_2)$. By choosing $\mathcal{A} = \mathcal{X}$, we rewrite the problem as $\sigma(s, a) = s(1 + w||x - a||_2)$. The larger the move, the more noisy the observation. This creates a problem, where the Bayesian optimisation needs to balance between informative actions and movement, which can be directly implemented in the objective (5.8) via the matrix $\mathbf{V}(d_\pi) = \sum_{x,a \in \mathcal{X}} d_\pi(x, a) \frac{1}{\sigma^2(x,a)} \Phi(x)\Phi(x)^\top + \mathbf{I}$. Figure 5.6b reports the comparison between worst-case stateless BO and our algorithm. Our approach substantially improves performance.

5.9.4 Synthetic Benchmarks

We benchmark on a variety of classical Bayesian optimisation problems while imposing local movement constraints and considering both immediate and asynchronous feedback (by introducing an observation delay of 25 iterations). We also include the chemistry SnAr benchmark, from Summit [Felton et al., 2021], which we treat as asynchronous as per Folch et al. [2022]. Results are in Figure 5.7. In the synchronous setting, we found using the UCB maximiser criteria for MDP-BO yields the best results (c.f. Appendix for details of this variant). We also found that LSR performs very competitively on many benchmarks, frequently matching the performance of MDP-BO. In the asynchronous settings we achieved better results using MDP-

BO with Thompson sampling. TrSnAKe baseline appears to be competitive in all synthetic benchmarks as well. However, MDP-BO is more robust having less variance in the chosen paths as seen in the quantiles.



(a) Michaelwicz 3D. (synch) (b) Hartmann 3D. (synch) (c) Hartmann 6D. (synch)

(d) Michalewicz 2D. (asynch) (e) Hartmann 3D. (asynch) (f) SnAr 4D. (asynch)

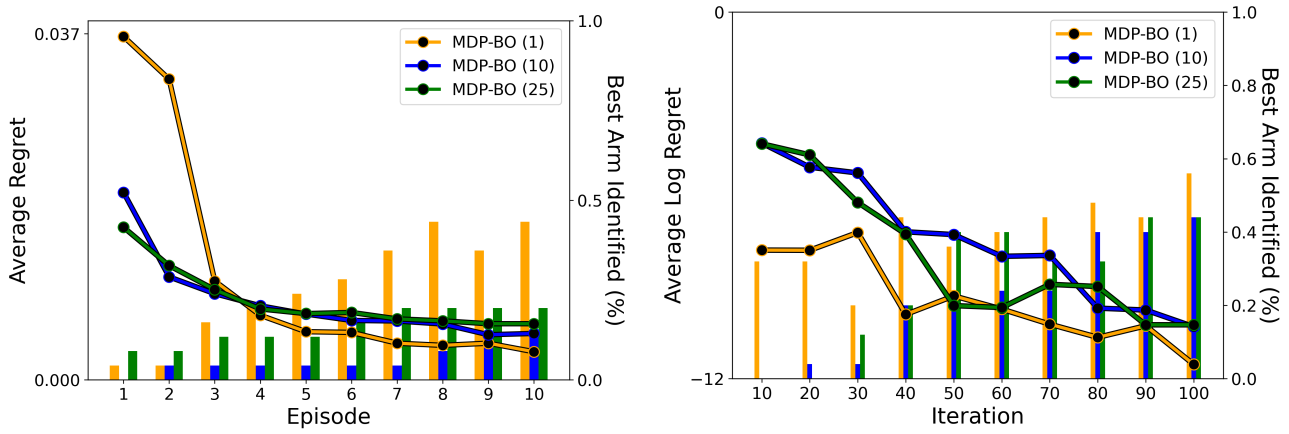
Figure 5.7: Results of experiments on the asynchronous and synchronous benchmarks. We plot the median predictive regret and the 10% and 90% quantiles. For the asynchronous experiments, we can see that the paths taken by MDP-BO-TS are more consistent, and the final performance is comparable to TrSnAKe. While in the asynchronous setting, we found creating the maximisation set using Thompson Sampling gave a stronger performance, in the synchronous setting, UCB is preferred. LSR gives a very strong performance, comparable to MDP-BO-UCB in almost all benchmarks.

5.9.5 Ablation Study

Number of mixture components

We investigate the effect number of components used in the mixture policy when optimising the Frank-Wolfe algorithm. We tested on the four real-world problem using $N = 1, 10$ and 25. In the Ypacari example (see Figure 5.9) we see very little difference in the results, while in the Knorr pyrazole synthesis (see Figure 5.8) we observe a much bigger difference. A single component gives a much stronger performance than multiple ones – we conjecture this is because the optimum is on the edge of the search space, and adding more components makes the policy stochastic and less likely to reach the boarder (given episodes are of length ten and ten right-steps are required to reach the border).

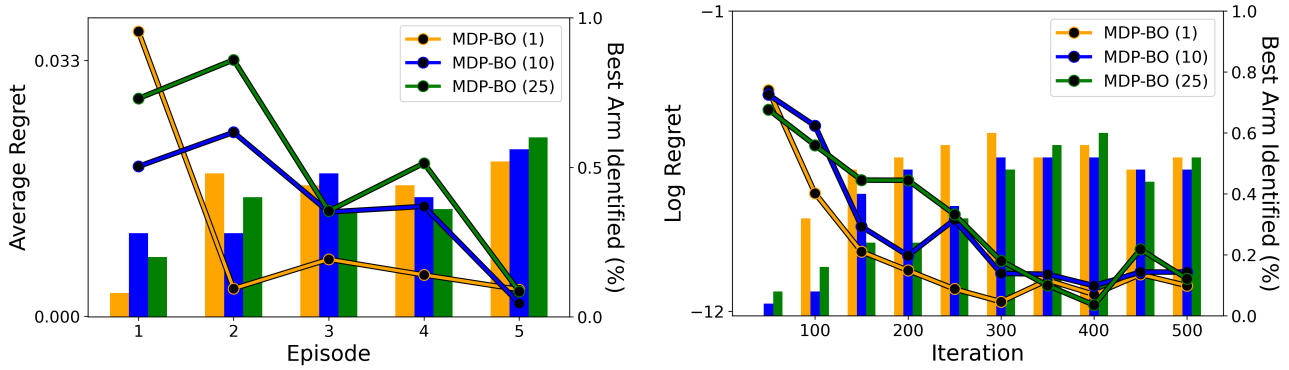
Overall, it seems the performance of a single component is better or at worst comparable as using multiple components. This is most likely due to the fact that we only follow the Markovian policies for a single time-step before recalculating, making the overall impact of mixture policies smaller. Based on this, we only present the single-component variant in the main paper.



(a) Episodic feedback results

(b) Immediate feedback results

Figure 5.8: Ablation study on the number of mixture components on the Knorr pyrazole synthesis benchmark



(a) Episodic feedback results

(b) Immediate feedback results

Figure 5.9: Ablation study on the number of mixture components on the Ypacarai benchmark

Size of batch for approximating the set of maximisers

We explore the effect of the number of maximisers, K, in the maximisation sets $\mathcal{Z}_{cont}^{(TS)}$ and $\mathcal{Z}_{cont}^{(UCB)}$. Overall we found the performance of the algorithm to be fairly robust to the size of the set in all benchmarks, with a higher K generally leading to a little less spread in the performance.

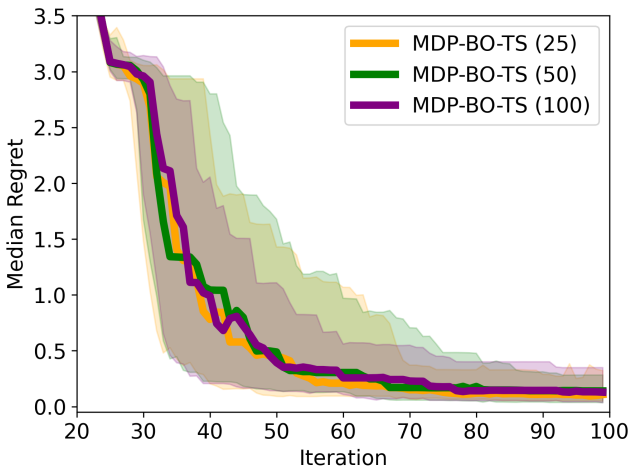
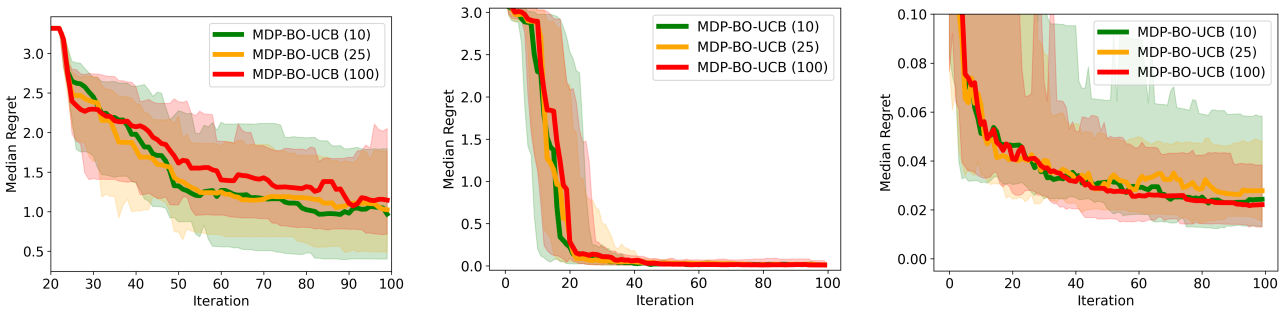


Figure 5.10: Ablation study into the size of the Thompson Sampling maximisation set in the asynchronous Hartmann3D function. We can see that the performance of the algorithm is very similar for all values of $K = 25, 50, 100$.



(a) Hartmann6D (asynch). (b) Hartmann3D. (c) Levy4D.

Figure 5.11: Ablation study into the size of the UCB maximisation set in a variety of benchmarks. We can see that the performance of the algorithm is very similar for all values of $K = 10, 25, 100$.

5.9.6 Computational study

We include the average acquisition function solving time for each of the discrete problems in Table 5.1. For the continuous case the running time was comparable to Truncated SnAKe [Folch et al., 2023b] since most of the computational load was to create the set of maximisers using Thompson Sampling. The times were obtained in a simple 2015 MacBook Pro 2.5 GHz Quad-Core Intel Core i7. The bulk of the experiments was ran in parallel on a High Performance Computing cluser, equipped with AMD EPYC 7742 processors and 16GB of RAM.

Table 5.1: Average acquisition function solving times for each practical benchmark. We give the solving times to the nearest second, and provide the size of the state-space, $|\mathcal{S}|$, the maximum number of actions one can take from a specific state, $|\mathcal{A}(S)|$, and the planning horizon. In all benchmarks we are able to solve the problem in a few seconds.

Benchmark	**Solve Time**	$\lvert\mathcal{S}\rvert$	**Maximum** $\lvert\mathcal{A}(S)\rvert$	**Planning horizon**
Knorr pyrazole	1s	100	6	10
Ypacarai	3s	100	8	50
Electron laser	15s	100	100	100

5.10 Discussion

We considered transition-constrained Bayesian optimisation problems arising in physical sciences, such as chemical reactor optimisation, that require careful planning to reach any system configuration. Focusing on maximiser identification, we formulated the problem with transition constraints using the framework of Markov decision processes and constructed a tractable algorithm for provably and efficiently solving these problems using dynamic programming or model predictive control sub-routines. This work takes an important step towards the larger application of Bayesian optimisation to real-world problems, as we have shown by strong performance in a large variety of problems with physical transitions. Further work could address the continuous variant of the framework to deal with more general transition dynamics.

5.11 Looking back...

This was the last project in my PhD and was written almost in conjunction with this thesis, therefore, there is not so much of a 'looking back' but a 'looking forward' instead. Indeed, since writing up the project I feel like I understand the chemistry application much better.

I have since become the co-founder of a start-up called SOLVE, where transient flow reactors are used and the method presented here is being applied for data collection. Indeed, Figure

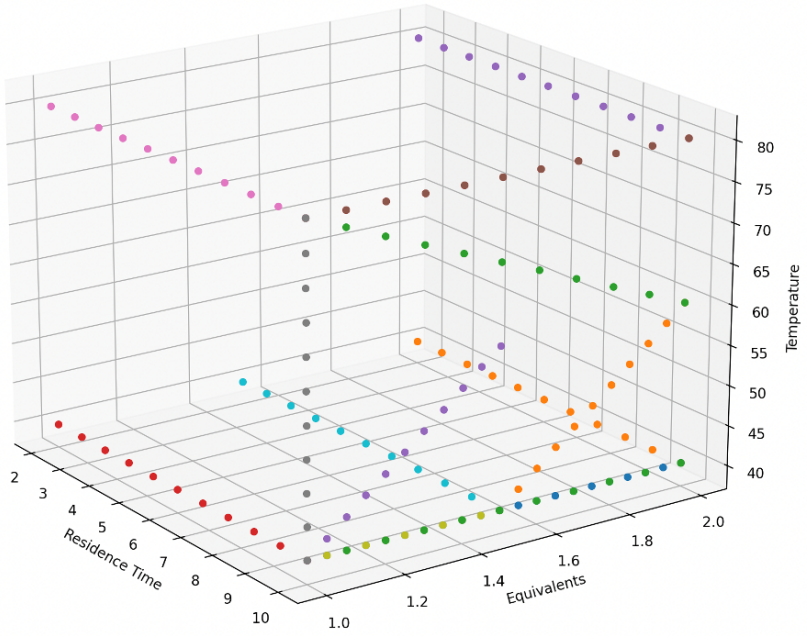


Figure 5.12: Example paths planned for data-collection in a transient flow reactor, the experiments were then carried out in a laboratory at SOLVE Chemistry.

5.12 shows the algorithm applied in an active learning setting, under realistic constraints, for data that was collected in a laboratory.

There is still much work to be done, in particular, I am interested in two scenarios: (a) objective functions that take costs of querying certain areas into account, and (b) scaling up the method as much as possible for planning in high dimensional spaces, either through the continuos version, or through a smart selection of a discrete number of states and actions.

Chapter 6

Conclusion

6.1 Summary of Thesis Achievements

In this thesis we have explored the challenges that many physical systems impose on Bayesian optimisation. In particular we have analysed three specific settings (i) under varying levels of measurement approximation and time-delay, (ii) under movement costs between subsequent experiments and (iii) under transition dynamics imposed by the system.

In Chapter 3, we introduce an algorithm that can use any acquisition function to choose future queries and multiple ways in which we can model the fidelities. We give an empirical study that shows that using multi-output Gaussian processes appears to be the most beneficial modelling method, while using an uncertainty-based criteria or an information based criteria for choosing the fidelity to query. The contributions of this chapter are engineering in nature, and this was recognised to be impactful by the journal Computers and Chemical Engineering who awarded "Best Paper of 2023" to the work.

In Chapter 4, we introduce SnAKe, an algorithm that minimises the movement costs while maintaining the strong empirical performance of Bayesian optimisation methods. We showcase the strength of the method in a variety of empirical settings, as well as demonstrate the flexibility of the method by making minor modifications and allowing the algorithm to be applied

to the multi-objective, self-stopping, and movement constrained setting.

Finally, in Chapter 5 we tackle a much more general problem of transition dynamics within Bayesian optimisation. Previously an unexplored area, we propose a theoretically grounded algorithm that allows long-term planning by leveraging state-of-the-art optimisation techniques in Markov Decision Processes. We focus on maximiser identification and give a new Bayesian motivation to an objective previously known in the bandit (frequentist) literature. Our algorithm is shown to give strong performance in a variety of empirical examples.

All three major contributions and achievements make an important step for the application of Bayesian optimisation to the physical sciences, and most importantly, chemistry, as we describe in the next section.

6.2 Applications

The potential applications of the work form a long list. However, the main thread across all of them has been the application of Bayesian optimisation to chemistry. We begin with transient flow reactors:

- **Transient method in flow reactors:** these allow for the extremely efficient collection of chemical data via the ramping of chemical conditions in a flow reactor, allowing the system to stay in "pseudo-steady-state" as opposed to having to wait long times for the system to stabilise. These reactors are an exceptional tool which can be made even more efficient through the automation of experiments. All three chapters are applicable here: (i) flow reactors can have inline measurement systems that give information at varying levels of confidence, and (ii) the conditions of the system must change slowly, making SnAKe and MDP-BO well suited for the system.

- **Battery design:** as showcased in Chapter 3, the design of batteries is expensive and time-consuming, which is why the use of coin-cell batteries to approximate the performance of

pouch-cell batteries is a promising avenue to explore, making multi-fidelity algorithms an important component of truly speeding up the discovery of more efficient batteries.

- **Environmental monitoring:** the use of automated vehicles for complete environmental monitoring is a promising future avenue to make sure we protect the health of ecosystems. As showcased in Chapter 4 and 5, these are some of the main applications that can be tackled through our algorithms. For example, it has been shown that Gaussian processes are a good model for the prediction of air quality in places with few and sparse sensor locations [Stoddart et al., 2023], Bayesian optimisation could under transition constraints could guide search for sources of contaminants. It is also important to note that many different types of measurements could be used to measure environmental threats – making multi-output Gaussian processes and multi-fidelity algorithms also applicable.

- **Other applications:** already explored in the literature, include the fine tuning of electron-lased microscopes [Kirschner et al., 2022], the optimisation of vapor compression systems [Paulson et al., 2023], seabed identification [Sullivan et al., 2023], and also the optimisation of energy systems [Ramesh et al., 2022]. However, given the generality of the methods, and the increasing popularity of Bayesian optimisation, it would not be surprising to see many more applications surface in the upcoming years.

6.3 Future Work

Perhaps the single most important future work is to deploy the methods in practice, beyond simulation studies. This will allow us to truly measure the impact of the methods as well as showcase new challenges that need to be addressed.

Other interesting future work is to move beyond Gaussian processes and investigate whether probabilistic deep learning models would allow us to use larger amounts of prior data to better guide the optimisation. In other words, explore how meta-learning or transfer learning route, which has shown promise recently [Volpp et al., 2020, Chen et al., 2022, Yang et al., 2023] and see how it would improve our algorithmic performance. Indeed, in Chapter 5 we showcased, in

the Knorr pyrazole experiment, how previous knowledge can make the trajectories through the space massively more informative of the location of the optimum – however this was an arduous manual task that required the careful analysis of the underlying ODE system. By using deep learning models trained on large quantities of data we might be able to get the same benefits for a much larger class of reactions. Another promising avenue includes looking at integrating our models with the automatic discovery of kinetic models [de Carvalho Servia et al., 2024].

Bibliography

Abdulelah S Alshehri, Rafiqul Gani, and Fengqi You. Deep learning and knowledge-based methods for computer-aided molecular design—toward a unified approach: State-of-the-art and future directions. *Computers & Chemical Engineering*, 141:107005, 2020.

Mauricio A Alvarez, Lorenzo Rosasco, Neil D Lawrence, et al. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012.

Ahsan Alvi, Binxin Ru, Jan-Peter Calliess, Stephen Roberts, and Michael A. Osborne. Asynchronous Batch Bayesian Optimisation with Improved Local Penalisation. In *International Conference on Machine Learning*, pages 253–262, 09–15 Jun 2019.

Satyajith Amaran, Nikolaos V Sahinidis, Bikram Sharda, and Scott J Bury. Simulation optimization: a review of algorithms and applications. *Annals of Operations Research*, 240(1): 351–380, 2016.

Raul Astudillo, Daniel Jiang, Maximilian Balandat, Eytan Bakshy, and Peter Frazier. Multi-step budgeted Bayesian optimization with unknown evaluation costs. *Advances in Neural Information Processing Systems*, 34:20197–20209, 2021.

Peter M Attia, Aditya Grover, Norman Jin, Kristen A Severson, Todor M Markov, Yang-Hung Liao, Michael H Chen, Bryan Cheong, Nicholas Perkins, Zi Yang, et al. Closed-loop optimization of fast-charging protocols for batteries with machine learning. *Nature*, 578 (7795):397–402, 2020.

Jean-Yves Audibert and Sébastien Bubeck. Best arm identification in multi-armed bandits. In *Conference on Learning Theory*, pages 13–p, 2010.

Javad Azimi, Alan Fern, and Xiaoli Z Fern. Batch Bayesian Optimization via simulation matching. In *Advances in Neural Information Processing Systems*, pages 109–117. Citeseer, 2010.

Oluwadare Badejo and Marianthi Ierapetritou. Integrating tactical planning, operational planning and scheduling using data-driven feasibility analysis. *Computers & Chemical Engineering*, 161:107759, 2022.

Ishan Bajaj, Shachit S Iyer, and MM Faruque Hasan. A trust region-based two phase algorithm for constrained black-box and grey-box optimization with infeasible initial point. *Computers & Chemical Engineering*, 116:306–321, 2018.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020.

Richard Bellman. Dynamic Programming Treatment of the Travelling Salesman Problem. *Journal of the ACM (JACM)*, 9(1):61–63, 1962.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems*, 24, 2011.

Dimitri Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, 2012.

Andreas Besginow and Markus Lange-Hegermann. Constraining Gaussian processes to systems of linear ordinary differential equations. *Advances in Neural Information Processing Systems*, 35:29386–29399, 2022.

Ilija Bogunovic, Jonathan Scarlett, Andreas Krause, and Volkan Cevher. Truncated variance reduction: A unified approach to Bayesian optimization and level-set estimation. *Advances in Neural Information Processing Systems*, 29, 2016.

Fani Boukouvala and Christodoulos A Floudas. ARGONAUT: AlgoRithms for Global Optimization of coNstrAined grey-box compUTational problems. *Optimization Letters*, 11(5): 895–913, 2017.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Kathryn Chaloner and Isabella Verdinelli. Bayesian Experimental Design: A Review. *Statist. Sci.*, 10(3):273–304, 08 1995.

Ethan Che, Jimmy Wang, and Hongseok Namkoong. Planning contextual adaptive experiments with model predictive control. *NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World*, 2023.

Shuru Chen, Chaojiang Niu, Hongkyung Lee, Qiuyan Li, Lu Yu, Wu Xu, Ji-Guang Zhang, Eric J. Dufek, M. Stanley Whittingham, Shirley Meng, Jie Xiao, and Jun Liu. Critical Parameters for Evaluating Coin Cells and Pouch Cells of Rechargeable Li-Metal Batteries. *Joule*, 3(4):1094–1105, 2019.

Yutian Chen, Xingyou Song, Chansoo Lee, Zi Wang, Richard Zhang, David Dohan, Kazuya Kawakami, Greg Kochanski, Arnaud Doucet, Marc'aurelio Ranzato, et al. Towards learning universal hyperparameter optimizers with transformers. *Advances in Neural Information Processing Systems*, 35:32053–32068, 2022.

Connor W Coley. Defining and exploring chemical spaces. *Trends in Chemistry*, 3(2):133–145, 2021.

Connor W. Coley, Dale A. Thomas, Justin A. M. Lummiss, Jonathan N. Jaworski, Christopher P. Breen, Victor Schultz, Travis Hart, Joshua S. Fishman, Luke Rogers, Hanyu Gao, Robert W. Hicklin, Pieter P. Plehiers, Joshua Byington, John S. Piotti, William H. Green, A. John Hart, Timothy F. Jamison, and Klavs F. Jensen. A robotic platform for flow synthesis of organic compounds informed by AI planning. *Science*, 365(6453), 2019.

Alison Cozad, Nikolaos V Sahinidis, and David C Miller. Learning surrogate models for simulation-based optimization. *AIChE Journal*, 60(6):2211–2227, 2014.

Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *Bulletin of the American mathematical society*, 39(1):1–49, 2002.

Kurt Cutajar, Mark Pullin, Andreas Damianou, Neil Lawrence, and Javier Gonzalez. Deep Gaussian Processes for Multi-fidelity Modeling, 2019.

Andreas Damianou and Neil D Lawrence. Deep Gaussian processes. In *Artificial intelligence and statistics*, pages 207–215. PMLR, 2013.

Miguel Ángel de Carvalho Servia, Ilya Orson Sandoval, King Kuok Mimi Hii, Klaus Hellgardt, Dongda Zhang, and Ehecatl Antonio del Rio Chanona. The automated discovery of kinetic rate models–methodological frameworks. *Digital Discovery*, 3(5):954–968, 2024.

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

Thomas Desautels, Andreas Krause, and Joel W Burdick. Parallelizing exploration-exploitation tradeoffs in Gaussian Process bandit optimization. *Journal of Machine Learning Research*, 15:3873–3923, 2014.

Snoop Dogg, Anna Wise Bilal, and Kendrick Lamar. Institutionalized. *To Pimp a Butterfly*, 2015.

Marco Dorigo and Luca Maria Gambardella. Ant colonies for the travelling salesman problem. *Biosystems*, 43(2):73–81, 1997.

Johannes P Dürholt, Thomas S Asche, Johanna Kleinekorte, Gabriel Mancino-Ball, Benjamin Schiller, Simon Sung, Julian Keupp, Aaron Osburg, Toby Boyne, Ruth Misener, et al. Bofire: Bayesian optimization framework intended for real experiments. *arXiv preprint arXiv:2408.05040*, 2024.

Susanne Dörfler, Holger Althues, Paul Härtel, Thomas Abendroth, Benjamin Schumm, and Stefan Kaskel. Challenges and Key Parameters of Lithium-Sulfur Batteries on Pouch Cell Level. *Joule*, 4(3):539–554, 2020.

David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local Bayesian optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Verena Fath, Philipp Lau, Christoph Greve, Philipp Weller, Norbert Kockmann, and Thorsten Röder. Simultaneous self-optimisation of yield and purity through successive combination of inline FT-IR spectroscopy and online mass spectrometry in flow reactions. *Journal of Flow Chemistry*, 11(3):285–302, 2021.

Kobi Felton, Jan Rittig, and Alexei Lapkin. Summit: Benchmarking Machine Learning Methods for Reaction Optimisation. *Chemistry Methods*, February 2021.

Tanner Fiez, Lalit Jain, Kevin G Jamieson, and Lillian Ratliff. Sequential experimental design for transductive linear bandits. *Advances in neural information processing systems*, 32, 2019.

Jose Pablo Folch, Shiqiang Zhang, Robert M Lee, Behrang Shafei, David Walz, Calvin Tsay, Mark van der Wilk, and Ruth Misener. SnAKe: Bayesian Optimization with Pathwise Exploration. *Advances in Neural Information Processing Systems*, 35, 2022.

Jose Pablo Folch, Robert M Lee, Behrang Shafei, David Walz, Calvin Tsay, Mark van der Wilk, and Ruth Misener. Combining multi-fidelity modelling and asynchronous batch Bayesian optimization. *Computers & Chemical Engineering*, 172:108194, 2023a.

Jose Pablo Folch, James Odgers, Shiqiang Zhang, Robert M Lee, Behrang Shafei, David Walz, Calvin Tsay, Mark van der Wilk, and Ruth Misener. Practical path-based Bayesian optimization. *NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World*, 2023b.

Jose Pablo Folch, Calvin Tsay, Robert M Lee, Behrang Shafei, Weronika Ormaniec, Andreas Krause, Mark van der Wilk, Ruth Misener, and Mojmír Mutný. Transition constrained Bayesian optimization via Markov decision processes. *Advances in Neural Information Processing Systems*, 37, 2024.

Edwin Fong and Chris C Holmes. On the marginal likelihood and cross-validation. *Biometrika*, 107(2):489–496, 2020.

Carlos E. García, David M. Prett, and Manfred Morari. Model predictive control: Theory and practice—A survey. *Automatica*, 25(3):335–348, 1989.

Jacob R. Gardner, Geoff Pleiss, David Bindel, Kilian Q. Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix Gaussian Process inference with GPU acceleration. *Advances in Neural Information Processing Systems*, pages 7576–7586, 2018.

David Ginsbourger and Rodolphe Le Riche. Towards Gaussian process-based optimization with finite time horizon. In *Advances in Model-Oriented Design and Analysis*, pages 89–96. Springer, 2010.

David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. A Multi-points Criterion for Deterministic Parallel Global Optimization based on Gaussian Processes. *Hal preprint 00260579*, March 2008.

David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. Kriging is well-suited to Parallelize Optimization. In *Computational Intelligence in Expensive Optimization Problems*, Springer series in Evolutionary Learning and Optimization, pages 131–162. 2010.

David Ginsbourger, Janis Janusevskis, and Rodolphe Le Riche. *Dealing with Asynchronicity in Parallel Gaussian Process based Global Optimization.* PhD thesis, Mines Saint-Etienne, 2011.

Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch Bayesian Optimization via Local Penalization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 648–657, 09–11 May 2016a.

Javier González, Michael Osborne, and Neil Lawrence. GLASSES: Relieving The Myopia Of Bayesian Optimisation. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 790–799, 09–11 May 2016b.

Pierre Goovaerts et al. *Geostatistics for natural resources evaluation.* Oxford University Press on Demand, 1997.

Alkis Gotovos, Nathalie Casati, Gregory Hitz, and Andreas Krause. Active learning for level set estimation. In *IJCAI 2013*, 2013.

James Grant and David Leslie. On thompson sampling for smoother-than-lipschitz bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 2612–2622. PMLR, 2020.

James Grant, Alexis Boukouvalas, Ryan-Rhys Griffiths, David Leslie, Sattar Vakili, and Enrique Munoz De Cote. Adaptive sensor placement for continuous spaces. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2385–2393, 2019.

Loic Le Gratiet. Recursive Co-Kriging Model for Design of Computer Experiments with Multiple Levels of Fidelity with an Application to Hydrodynamic, 2013.

Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using NetworkX. Technical report, Los Alamos National Lab. (LANL), 2008.

Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pages 2681–2691. PMLR, 2019.

Sigrid Passano Hellan, Christopher G Lucas, and Nigel H Goddard. Bayesian Optimisation for Active Monitoring of Air Pollution. In *36th AAAI Conference on Artificial Intelligence*, 2021.

Philipp Hennig and Christian J Schuler. Entropy Search for Information-Efficient Global Optimization. *Journal of Machine Learning Research*, 13(6), 2012.

José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. *Advances in Neural Information Processing Systems*, 27, 2014.

Dave Higdon. Space and space-time modeling using process convolutions. In *Quantitative methods for current environmental issues*, pages 37–56. Springer, 2002.

Gregory Hitz, Alkis Gotovos, Marie-Éve Garneau, Cédric Pradalier, Andreas Krause, Roland Y Siegwart, et al. Fully autonomous focused exploration for robotic environmental monitoring. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2658–2664. IEEE, 2014.

Christopher A. Hone, Nicholas Holmes, Geoffrey R. Akien, Richard A. Bourne, and Frans L. Muller. Rapid multistep kinetic model generation from transient flow data. *Reaction Chemistry & Engineering*, 2:103–108, 2017.

Gordon Hüllen, Jianyuan Zhai, Sun Hye Kim, Anshuman Sinha, Matthew J Realff, and Fani Boukouvala. Managing uncertainty in data-driven simulation-based optimization. *Computers & Chemical Engineering*, 136:106519, 2020.

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

Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Gordon Wilson. What are Bayesian neural network posteriors really like? In *International Conference on Machine Learning*, pages 4629–4640, 2021.

Martin Jaggi. Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 427–435. PMLR, 17–19 Jun 2013.

Runda Jia, Shulei Zhang, and Fengqi You. Transfer learning for end-product quality prediction of batch processes using domain-adaption joint-Y PLS. *Computers & Chemical Engineering*, 140:106943, 2020.

Shali Jiang, Henry Chai, Javier Gonzalez, and Roman Garnett. Binoculars for efficient, nonmyopic sequential experimental design. In *International Conference on Machine Learning*, pages 4794–4803. PMLR, 2020a.

Shali Jiang, Daniel Jiang, Maximilian Balandat, Brian Karrer, Jacob Gardner, and Roman Garnett. Efficient nonmyopic Bayesian optimization via one-shot multi-step trees. *Advances in Neural Information Processing Systems*, 33:18039–18049, 2020b.

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

Andre G Journel and Charles J Huijbregts. Mining geostatistics. 1976.

Kirthevasan Kandasamy, Gautam Dasarathy, Junier B Oliva, Jeff Schneider, and Barnabás Póczos. Gaussian Process Bandit Optimisation with Multi-fidelity Evaluations. *Advances in Neural Information Processing Systems*, 29, 2016a.

Kirthevasan Kandasamy, Gautam Dasarathy, Barnabas Poczos, and Jeff Schneider. The multi-fidelity multi-armed bandit. *Advances in neural information processing systems*, 29, 2016b.

Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. Multi-fidelity Bayesian optimisation with continuous approximations. In *International Conference on Machine Learning*, pages 1799–1808. PMLR, 2017.

Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabas Poczos. Parallelised Bayesian Optimisation via Thompson Sampling. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, pages 133–142, 2018.

Saif R Kazi, Michael Short, and Lorenz T Biegler. A trust region framework for heat exchanger network synthesis with detailed individual heat exchanger designs. *Computers & chemical engineering*, 153:107447, 2021.

Saif R Kazi, Ishanki A De Mel, and Michael Short. A new trust-region approach for optimization of multi-period heat exchanger networks with detailed shell-and-tube heat exchanger designs. In *Computer Aided Chemical Engineering*, volume 49, pages 241–246. Elsevier, 2022.

M. C. Kennedy and A. O'Hagan. Predicting the Output from a Complex Computer Code When Fast Approximations Are Available. *Biometrika*, 87(1):1–13, 2000.

Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, 12 2014.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

Johannes Kirschner, Mojmír Mutný, Andreas Krause, Jaime Coello de Portugal, Nicole Hiller, and Jochem Snuverink. Tuning particle accelerators with safety constraints using Bayesian optimization. *Phys. Rev. Accel. Beams*, 25:062802, 2022.

Michael Komodromos, Marina Evangelou, and Sarah Filippi. Logistic Variational Bayes Revisited. *International Conference on Machine Learning*, 2024.

Hermann König. *Eigenvalue distribution of compact operators*, volume 16. Birkhäuser, 2013.

Akshay Kudva, Farshud Sorourifar, and Joel A Paulson. Constrained robust Bayesian optimization of expensive noisy black-box functions with guaranteed regret bounds. *AIChE Journal*, page e17857, 2022.

Harold J. Kushner. A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. *Journal of Basic Engineering*, 86:97–106, 1964.

Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

Remi Lam, Karen Willcox, and David H Wolpert. Bayesian optimization with a finite budget: An approximate dynamic programming approach. *Advances in Neural Information Processing Systems*, 29:883–891, 2016.

Neil Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. *Advances in neural information processing systems*, 16, 2003.

Stefano Lazzari, Andree Lischewski, Yury Orlov, Peter Deglmann, Andreas Daiss, Eduard Schreiner, and Hugo Vale. *Toward a digital polymer reaction engineering*, volume 56, pages 187–227. 01 2020.

Eric Hans Lee, Valerio Perrone, Cedric Archambeau, and Matthias Seeger. Cost-aware Bayesian optimization. *Proceedings of the 7th ICML Workshop on Automated Machine Learning*, 2020.

Eric Hans Lee, David Eriksson, Valerio Perrone, and Matthias Seeger. A Nonmyopic Approach to Cost-Constrained Bayesian Optimization. In *Uncertainty in Artificial Intelligence*, pages 568–577. PMLR, 2021.

Fang-Yie Leu and Jia-Sheng Ho. Air Pollution Source Identification by Using Neural Network with Bayesian Optimization. In *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 514–524. Springer, 2019.

Bowen Li and Srinivas Rangarajan. A conceptual study of transfer learning with linear models for data-driven property prediction. *Computers & Chemical Engineering*, 157:107599, 2022.

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

Shibo Li, Robert Kirby, and Shandian Zhe. Batch multi-fidelity Bayesian optimization with deep auto-regressive networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Weijun Li, Sai Gu, Xiangping Zhang, and Tao Chen. Transfer learning for process fault diagnosis: Knowledge transfer from simulation to physical processes. *Computers & Chemical Engineering*, 139:106904, 2020.

Ya-Tao Liu, Sheng Liu, Guo-Ran Li, and Xue-Ping Gao. Strategy of enhancing the volumetric energy density for lithium–sulfur batteries. *Advanced Materials*, 33(8):2003955, 2021.

Sanae Lotfi, Pavel Izmailov, Gregory Benton, Micah Goldblum, and Andrew Gordon Wilson. Bayesian model selection, the marginal likelihood, and generalization. In *International Conference on Machine Learning*, pages 14223–14247. PMLR, 2022.

Phuc Luong, Dang Nguyen, Sunil Gupta, Santu Rana, and Svetha Venkatesh. Adaptive cost-aware Bayesian optimization. *Knowledge-Based Systems*, 232:107481, 2021.

David JC MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992.

Benedict C May, Nathan Korda, Anthony Lee, David S Leslie, and Nicolo Cesa-Bianchi. Optimistic Bayesian sampling in contextual-bandit problems. *Journal of Machine Learning Research*, 13(6), 2012.

Jonathan P. McMullen and Klavs F. Jensen. Rapid Determination of Reaction Kinetics with an Automated Microfluidic System. *Organic Process Research & Development*, 15(2):398–407, 2011.

J. Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 09 2014.

Jonas Mockus. On Bayesian methods for seeking the extremum. In *Proceedings of the IFIP Technical Conference*, pages 400–404, 1974.

Henry B Moss, David S Leslie, Javier Gonzalez, and Paul Rayson. GIBBON: General-purpose information-based Bayesian optimisation. *Journal of Machine Learning Research*, 22(235): 1–49, 2021a.

Henry B Moss, David S Leslie, and Paul Rayson. Mumbo: Multi-task max-value Bayesian optimization. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part III*, pages 447–462. Springer, 2021b.

Max Mowbray, Thomas Savage, Chufan Wu, Ziqi Song, Bovinille Anye Cho, Ehecatl A Del Rio-Chanona, and Dongda Zhang. Machine learning for biochemical engineering: A review. *Biochemical Engineering Journal*, 172:108054, 2021.

Sergey Mozharov, Alison Nordon, David Littlejohn, Charlotte Wiles, Paul Watts, Paul Dallin, and John M Girkin. Improved method for kinetic studies in microreactors using flow manipulation and noninvasive Raman spectrometry. *Journal of the American Chemical Society*, 133(10):3601–3608, 2011.

Mojmír Mutný, Johannes Kirschner, and Andreas Krause. Experimental design for optimization of orthogonal projection pursuit models. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Mojmír Mutný and Andreas Krause. Efficient high dimensional Bayesian optimization with additivity and quadrature fourier features. *Advances in Neural Information Processing Systems*, 31, 2018.

Mojmír Mutný and Andreas Krause. Experimental design for linear functionals in reproducing kernel Hilbert spaces. *Advances in Neural Information Processing Systems*, 35:20175–20188, 2022.

Mojmír Mutný, Tadeusz Janik, and Andreas Krause. Active Exploration via Experiment Design in Markov Chains. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206, pages 7349–7374, 2023.

Hirotaka Nakayama, Yeboon Yun, and Min Yoon. *Sequential approximate multiobjective optimization using computational intelligence*. Springer Science & Business Media, 2009.

Elvis Nava, Mojmír Mutný, and Andreas Krause. Diversified Sampling for Batched Bayesian Optimization with Determinantal Point Processes. In *International Conference on Artificial Intelligence and Statistics*, pages 7031–7054, 2022.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Simon Olofsson, Mohammad Mehrian, Roberto Calandra, Liesbet Geris, Marc Peter Deisenroth, and Ruth Misener. Bayesian multiobjective optimisation with mixed analytical and black-box functions: Application to tissue engineering. *IEEE Transactions on Biomedical Engineering*, 66(3):727–739, 2018.

Elton Pan, Panagiotis Petsagkourakis, Max Mowbray, Dongda Zhang, and Ehecatl Antonio del Rio-Chanona. Constrained model-free reinforcement learning for process optimization. *Computers & Chemical Engineering*, 154:107462, 2021.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

Biswajit Paria, Kirthevasan Kandasamy, and Barnabás Póczos. A flexible framework for multi-objective Bayesian optimization using random scalarizations. In *Uncertainty in Artificial Intelligence*, pages 766–776. PMLR, 2020.

Seongeon Park, Jonggeol Na, Minjun Kim, and Jong Min Lee. Multi-objective Bayesian optimization of chemical reactor design using computational fluid dynamics. *Computers & Chemical Engineering*, 119:25–37, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, pages 8024–8035. Curran Associates, Inc., 2019.

Joel A Paulson and Congwen Lu. COBALT: COnstrained Bayesian optimizAtion of computationaLly expensive grey-box models exploiting derivaTive information. *Computers & Chemical Engineering*, 160:107700, 2022.

Joel A. Paulson, Farshud Sorouifar, and Ankush Chakrabarty. Efficient Multi-Step Lookahead Bayesian Optimization with Local Search Constraints. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 123–129, 2022.

Joel A Paulson, Farshud Sorouifar, Christopher R Laughman, and Ankush Chakrabarty. LSR-BO: Local search region constrained Bayesian optimization for performance optimization of vapor compression systems. In *2023 American Control Conference (ACC)*, pages 576–582. IEEE, 2023.

Panagiotis Petsagkourakis, Benoit Chachuat, and Ehecatl Antonio del Rio-Chanona. Safe real-time optimization using multi-fidelity Gaussian processes. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6734–6741. IEEE, 2021.

Panagiotis Petsagkourakis, Ilya Orson Sandoval, Eric Bradford, Federico Galvanin, Dongda Zhang, and Ehecatl Antonio del Rio-Chanona. Chance constrained policy optimization for process control and optimization. *Journal of Process Control*, 111:35–45, 2022.

Thomas Pinder, Kathryn Turnbull, Christopher Nemeth, and David Leslie. Street-level air

pollution modelling with graph gaussian processes. *ICLR: AI for Earth and Space Science*, 2022.

Matthias Poloczek, Jialei Wang, and Peter Frazier. Multi-information source optimization. *Advances In Neural Information Processing Systems*, 30, 2017.

Friedrich Pukelsheim. *Optimal Design of Experiments (Classics in Applied Mathematics) (Classics in Applied Mathematics, 50)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2006. ISBN 0898716047.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley Sons, 2014.

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

Shyam Sundhar Ramesh, Pier Giuseppe Sessa, Andreas Krause, and Ilija Bogunovic. Movement Penalized Bayesian Optimization with Application to Wind Energy Systems. *ICML Workshop on Adaptive Experimental Design and Active Learning in the Real World*, 2022.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

James Blake Rawlings, David Q Mayne, and Moritz Diehl. *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing, 2017.

Alexander W Rogers, Fernando Vega-Ramon, Jiangtao Yan, Ehecatl A del Río-Chanona, Keju Jing, and Dongda Zhang. A transfer learning approach for predictive modeling of bioprocesses using small data. *Biotechnology and Bioengineering*, 119(2):411–422, 2022.

Daniel Russo. Simple Bayesian algorithms for best arm identification. In *Conference on Learning Theory*, pages 1417–1418. PMLR, 2016.

Daniel Russo and Benjamin Van Roy. Learning to optimize via information-directed sampling. *Advances in neural information processing systems*, 27, 2014.

Federico Peralta Samaniego, Daniel Gutiérrez Reina, Sergio L Toral Marín, Mario Arzamendia, and Derlis O Gregor. A Bayesian Optimization Approach for Water Resources Monitoring Through An Autonomous Surface Vehicle: The Ypacarai Lake Case Study. *IEEE Access*, 9: 9163–9179, 2021.

Thomas Savage, Dongda Zhang, Max Mowbray, and Ehecatl Antonio Del Río Chanona. Model-free safe reinforcement learning for chemical processes using Gaussian processes. *IFAC-PapersOnLine*, 54(3):504–509, 2021.

Tom Savage and Ehecatl Antonio del Rio Chanona. Human-algorithm collaborative Bayesian optimization for engineering systems. *Computers & Chemical Engineering*, 189:108810, 2024.

Tom Savage, Nausheen Basha, Omar Matar Ehecatl, and Antonio Del-Rio Chanona. Deep Gaussian process-based multi-fidelity Bayesian optimization for simulated chemical reactors. In *NeurIPS Workshop on Gaussian Processes, Spatiotemporal Modeling, and Decision-making Systems*. Dec 2022.

Tom Savage, Nausheen Basha, Jonathan McDonough, Omar K Matar, and Ehecatl Antonio del Rio Chanona. Multi-fidelity data-driven design and analysis of reactor and tube simulations. *Computers & Chemical Engineering*, 179:108410, 2023.

Tom Savage, Nausheen Basha, Jonathan McDonough, James Krassowski, Omar Matar, and Ehecatl Antonio del Rio Chanona. Machine learning-assisted discovery of flow reactor designs. *Nature Chemical Engineering*, pages 1–10, 2024.

Linden Schrecker, Joachim Dickhaut, Christian Holtze, Philipp Staehle, Marcel Vranceanu, Klaus Hellgardt, and King Kuok Mimi Hii. Discovery of unexpectedly complex reaction pathways for the Knorr pyrazole synthesis via transient flow. *Reaction Chemistry & Engineering*, 8(1):41–46, 2023.

Linden Schrecker, Joachim Dickhaut, Christian Holtze, Philipp Staehle, Marcel Vranceanu, Andy Wieja, Klaus Hellgardt, and King Kuok Hii. A comparative study of transient flow rate steps and ramps for the efficient collection of kinetic data. *Reaction Chemistry & Engineering*, 2024.

Artur M Schweidtmann, Adam D Clayton, Nicholas Holmes, Eric Bradford, Richard A Bourne, and Alexei A Lapkin. Machine learning meets continuous flow chemistry: Automated optimization towards the Pareto front of multiple objectives. *Chemical Engineering Journal*, 352:277–282, 2018.

Steven L Scott. A modern Bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658, 2010.

Ruby Sedgwick, John P Goertz, Molly M Stevens, Ruth Misener, and Mark van der Wilk. Transfer learning Bayesian optimization to design competitor DNA molecules for use in diagnostic assays. *ArXiv*, 2024.

Rajat Sen, Kirthevasan Kandasamy, and Sanjay Shakkottai. Multi-fidelity black-box optimization with hierarchical partitions. In *International Conference on Machine Learning*, pages 4538–4547, 2018.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, 18, 2005.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 25, 2012.

Marta Soare, Alessandro Lazaric, and Rémi Munos. Best-arm identification in linear bandits. *Advances in Neural Information Processing Systems*, 27, 2014.

Il'ya Meerovich Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4):784–802, 1967.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *International Conference on Machine Learning*, pages 1015–1022, 2010.

Clara Stoddart, Lauren Shrack, Richard Sserunjogi, Usman Abdul-Ganiy, Engineer Bainomugisha, Deo Okure, Ruth Misener, Jose Pablo Folch, and Ruby Sedgwick. Gaussian processes for monitoring air-quality in Kampala. *NeurIPS 2023 Workshop: Tackling Climate Change with Machine Learning*, 2023.

Matthew Sullivan, John Gebbie, and John Lipor. Adaptive sampling for seabed identification from ambient acoustic noise. *IEEE CAMSAP*, 2023.

S. Surjanovic and D. Bingham. Virtual Library of Simulation Experiments: Test Functions and Datasets. Available at `http://www.sfu.ca/~ssurjano`, 2013.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Shion Takeno, Hitoshi Fukuoka, Yuhki Tsukada, Toshiyuki Koyama, Motoki Shiga, Ichiro Takeuchi, and Masayuki Karasuyama. Multi-fidelity Bayesian optimization with max-value entropy search and its parallelization. In *International Conference on Machine Learning*, pages 9334–9345. PMLR, 2020.

Shia-Yen Teh, Robert Lin, Lung-Hsin Hung, and Abraham P. Lee. Droplet microfluidics. *Lab Chip*, 8:198–220, 2008.

Yee Whye Teh, Matthias Seeger, and Michael I Jordan. Semiparametric latent factor models. In *International Workshop on Artificial Intelligence and Statistics*, pages 333–340. PMLR, 2005.

Alexander Thebelt, Jan Kronqvist, Robert M Lee, Nathan Sudermann-Merx, and Ruth Misener. Global optimization with ensemble machine learning models. In *Computer Aided Chemical Engineering*, volume 48, pages 1981–1986. Elsevier, 2020.

Alexander Thebelt, Jan Kronqvist, Miten Mistry, Robert M Lee, Nathan Sudermann-Merx, and Ruth Misener. ENTMOOT: a framework for optimization over ensemble tree models. *Computers & Chemical Engineering*, 151:107343, 2021.

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

Alexander Thebelt, Calvin Tsay, Robert M Lee, Nathan Sudermann-Merx, David Walz, Tom Tranter, and Ruth Misener. Multi-objective constrained optimization for energy applications via tree ensembles. *Applied Energy*, 306:118061, 2022b.

Alexander Thebelt, Johannes Wiebe, Jan Kronqvist, Calvin Tsay, and Ruth Misener. Maximizing information from chemical engineering data sets: Applications to machine learning. *Chemical Engineering Science*, 252:117469, 2022c.

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

Ben Tu, Axel Gandy, Nikolas Kantas, and Behrang Shafei. Joint entropy search for multi-objective Bayesian optimization. *Advances in Neural Information Processing Systems*, 35: 9922–9938, 2022.

Ben Tu, Nikolas Kantas, Robert M Lee, and Behrang Shafei. Random Pareto front surfaces. *arXiv preprint arXiv:2405.01404*, 2024.

Damien Van De Berg, Thomas Savage, Panagiotis Petsagkourakis, Dongda Zhang, Nilay Shah, and Ehecatl Antonio del Rio-Chanona. Data-driven optimization for process systems engineering applications. *Chemical Engineering Science*, 248:117135, 2022.

Mark Van der Wilk. *Sparse Gaussian process approximations and applications*. PhD thesis, 2019.

Mark van der Wilk, Carl Edward Rasmussen, and James Hensman. Convolutional Gaussian Processes. *Advances in Neural Information Processing Systems*, 30, 2017.

Pratibha Vellanki, Santu Rana, Sunil Gupta, David Rubin, Alessandra Sutti, Thomas Dorin, Murray Height, Paul Sanders, and Svetha Venkatesh. Process-constrained Batch Bayesian

Optimisation. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017.

Jay M Ver Hoef and Ronald Paul Barry. Constructing and fitting models for cokriging and multivariable spatial prediction. *Journal of Statistical Planning and Inference*, 69(2):275–294, 1998.

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

Michael Volpp, Lukas P. FrÃ¶hlich, Kirsten Fischer, Andreas Doerr, Stefan Falkner, Frank Hutter, and Christian Daniel. Meta-Learning Acquisition Functions for Transfer Learning in Bayesian Optimization. In *International Conference on Learning Representations*, 2020.

Conor Waldron, Arun Pankajakshan, Marco Quaglio, Enhong Cao, Federico Galvanin, and Asterios Gavriilidis. An autonomous microreactor platform for the rapid identification of kinetic models. *Reaction Chemistry & Engineering*, 4:1623–1636, 2019.

Ke Wang and Alexander W Dowling. Bayesian optimization for chemical products and functional materials. *Current Opinion in Chemical Engineering*, 36:100728, 2022.

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

Christopher Williams and Carl Rasmussen. Gaussian processes for regression. *Advances in neural information processing systems*, 8, 1995.

Christopher Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13, 2000.

James Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Deisenroth. Efficiently Sampling Functions from Gaussian Process Posteriors. In *International Conference on Machine Learning*, pages 10292–10302, 13-18 Jul 2020.

Shifeng Xiong, Peter Z. G. Qian, and C. F. Jeff Wu. Sequential Design and Analysis of High-Accuracy and Low-Accuracy Computer Codes. *Technometrics*, 55(1):37–46, 2013.

Adam X Yang, Laurence Aitchison, and Henry B Moss. MONGOOSE: Path-wise Smooth Bayesian Optimisation via Meta-learning. *arXiv preprint arXiv:2302.11533*, 2023.

Kai Yu, Jinbo Bi, and Volker Tresp. Active learning via transductive experimental design. In *Proceedings of the 23rd international conference on Machine learning*, pages 1081–1088, 2006.

Xubo Yue and Raed AL Kontar. Why Non-myopic Bayesian Optimization is Promising and How Far Should We Look-ahead? A Study via Rollout. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, pages 2808–2818, 26–28 Aug 2020.

Tom Zahavy, Brendan O'Donoghue, Guillaume Desjardins, and Satinder Singh. Reward is enough for convex MDPs. *Advances in Neural Information Processing Systems*, 34:25746–25759, 2021.

Mengyan Zhang, Maciej Bartosz Holowko, Huw Hayman Zumpe, and Cheng Soon Ong. Machine learning guided batched design of a bacterial ribosome binding site. *ACS Synthetic Biology*, 11(7):2314–2326, 2022.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109 (1):43–76, 2021.

Appendix A

Implementation Details:

A.1 Combining Multi-fidelity and Asynchronous Batch Bayesian Optimisation

We include implementation details about the methods used and the benchmarks. For full details, we refer any reader to the code, available at `https://github.com/jpfolch/MFBoom`.

A.1.1 Gaussian Processes

To implement the Gaussian Processes, we used the GPyTorch [Gardner et al., 2018] and PyTorch Paszke et al. [2019] libraries. When using independent Gaussian Processes, we used an RBF kernel. For Multi-Task Gaussian Processes, we used the kernel:

$$K(x, x') = \sum_{w=1}^{W} k_w^{(I)}(x, x')\mathcal{B}_w$$

where each $k_w^{(I)}(x, x')$ is given by an RBF kernel, each with *different hyper-parameters*. This choice was made to allow the model to vary the hyper-parameters at each fidelity. We further set the hyper-parameters $W = 2M$ and $\text{rank}(\mathcal{B}_w) = M$, where M is the number of

fidelities in the problem. This choice was arbitrary, but yielded good results. The kernel hyper-parameters whose value we don't explicitly mention were all learnt by maximizing the marginal log-likelihood [Rasmussen and Williams, 2005].

To initialize the hyper-parameters, we trained a Gaussian Process model on $80\log(d)$ randomly sampled data-points, and maximized the marginal log-likelihood. The data-points were then discarded, and the hyper-parameters from the initial GP were used to initialize the hyper-parameters of the GP used by Bayesian Optimization methods. We then retrained the hyper-parameters every time we obtained twenty new observations of either fidelity. To optimize the marginal log-likelihood, we used ADAM [Kingma and Ba, 2014] across 75 epochs, with a learning rate of 0.1.

To avoid any ill-behaviour, we also put a Smoothed Box Prior on each hyper-parameter. For length-scales the box went from 0.025 to 0.6, for the output-scale from 0.05 to 2, for the noise from 10^{-5} to 0.2, and for the prior mean constant from -1 to 1. All these values were chosen based on the scaling of the objective functions.

A.1.2 Methods

All Bayesian Optimization methods were implemented using GPyTorch [Gardner et al., 2018] and BoTorch [Balandat et al., 2020]. For the synthetic benchmarks, the acquisition functions were optimized using ADAM [Kingma and Ba, 2014] with a learning rate of 0.01 for 75 epochs. The optimization procedure was first evaluated on 7500d (independent GP methods), 7500 (multi-task based methods), or 3750 (MF-MES), random inputs, and the gradient optimization was multi-started on the best 10 outputs. The differences in number of points evaluated is based on the computational burden of each method.

TuRBO

In contrast, TuRBO [Eriksson et al., 2019] was optimized by grid search inside the trust region, using a Sobol grid of size $\min(5000, \max(2000, 200d))$.

To grow, shrink or move the trust region, we *only considered the high-fidelity observations.*

A.1.3 MF-MES

We created our own implementation of MF-MES (see Takeno et al. [2020] for notation that follows). We generated 100 samples of $f_* := \max f(x)$ using Thompson Sampling on a Sobol grid of size 7500. We also used 100 fantasies of $f_{\mathcal{Q}}$ to approximate the integral when doing asynchronous Bayesian Optimization.

The numerical integration was done using the Trapezium Rule, with the integration range broken into 500 intervals. For the integration range, we used $[-10^k, 10^k]$, where k was chosen the be the smallest element from $\{-6, ..., 2\}$, such that $\eta(10^k) < 10^{-30}$.

Whenever we would detected issues from singular matrices while calculating MF-MES, we added 10^{-30} to the diagonal to obtain numerically stable behaviour.

A.1.4 Benchmark Details

All benchmarks are considered in the hyper-cube $[0, 1]^d$, where d is the problem dimension.

Synthetic Examples

We took the synthetic examples to be the same as those analysed in Kandasamy et al. [2016b], with slightly different scaling. They are given by:

Currin Exponential 2D

$$f^{(2)}(x) = \frac{1}{10}\left(1 - \exp\left(\frac{-1}{2x_2}\right)\right)\left(\frac{2300x_1^3 + 1900x_1^2 + 2092x_1 + 60}{100x_1^3 + 500x_1^2 + 4x_1 + 20}\right)$$

$$f^{(1)}(x) = \frac{1}{4}f^{(2)}(x_1 + 0.05, x_2 + 0.05)\frac{1}{4}f^{(2)}(x_1 + 0.05, x_2 - 0.05) + \\ + \frac{1}{4}f^{(2)}(x_1 - 0.05, x_2 + 0.05) + \frac{1}{4}f^{(2)}(x_1 - 0.05, x_2 - 0.05)$$

Bad Currin Exponential 2D

For the highest fidelity, we use the same $f^{(2)}$ as in the normal Currin function. For the low fidelity, we use the negative of $f^{(2)}$, i.e. $f^{(1)}(x) = -f^{(2)}(x)$.

Park Function 4D

$$10f^{(2)} = \frac{x_1}{2}\left(\sqrt{1 + (x_2 + x_3^2)\frac{x_4}{x_1^2} - 1} - 1\right) + (x_1 + 3x_4)\exp\left(1 + \sin(x_3)\right)$$
$$10f^{(1)} = \left(1 + \frac{\sin(x_1)}{10}\right) f^{(2)}(x) - 2x_1^2 + x_2^2 + x_3^2 + 0.5$$

Borehole Function 8D

$$f^{(2)}(x) = \frac{2\pi x_3(x_4 - x_6)}{100\log(x_2/x_1)\left(1 + 2x_7x_3/(\log(x_2x_1)x_1^2x_8) + x_3/x_5\right)}$$
$$f^{(1)}(x) = \frac{5x_3(x_4 - x_6)}{100\log(x_2/x_1)(1.5 + 2x_7x_3/(\log(x_2/x_1)x_1^2x_8) + x_3/x_5)}$$

Hartmann 3D

The M^{th} fidelity is given by:

$$f^{(M)}(x) = \sum_{i=1}^{4} \alpha_i^{(m)} \exp\left(-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2\right)$$

where:

$$A = \begin{pmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix}, \qquad P = 10^{-4} \begin{pmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{pmatrix}$$

and $\alpha^{(m)} = \alpha + (M - m)\delta$ where $\alpha = (1, 1.2, 3, 3.2)$, and $\delta = (0.01, -0.01, -0.1, 0.1)$. Recall that we set $M = 3$.

Hartmann 6D

The 6D Hartmann takes the same form as the 3D Hartmann except we use the matrices:

$$A = \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}, \qquad P = 10^{-4} \begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{pmatrix}$$

Ackley 40D

For Ackley, we created the fidelities using the following equations:

$$6f^{(2)}(x) = 20 + \exp\left(-0.2\sqrt{(\frac{1}{40}\sum_{i=1}^{40}((9x_i - 4)^2)}\right) + \exp\left(\frac{1}{40}\sum_{i=1}^{40}\cos(2\pi(9x_i - 4))\right) - 20 - e^1$$

$$6f^{(1)}(x) = 20 + \exp\left(-0.2\sqrt{(\frac{1}{45}\sum_{i=1}^{40}((9x_i - 3.8)^2)}\right) + \exp\left(\frac{1}{43}\sum_{i=1}^{40}\cos(2\pi(9x_i - 3.8))\right) - 20 - e^1$$

A.2 SnAKe: Bayesian Optimization via Pathwise Exploration

This section outlines the implementation details and hyper-parameter choices for all the methods compared Chapter 4. The code used is available at the following link: `https://github.com/cog-imperial/SnAKe`.

A.2.1 Gaussian Processes

For every GP, we use the RBF Kernel with an output-scale. For the prior mean, we used a constant function with trainable value, μ_0. We implemented them all using the package GPyTorch [Gardner et al., 2018].

A.2.2 Training the hyper-parameters of the Gaussian Processes

Our method is well suited for physical systems. Hence, we assume that there is good prior knowledge of the hyper-parameters. In particular, we found it reasonable that each hyper-parameter would be given a lower and upper bound. Normally, we would simply have a large initialization sample. However, we believe this goes against the nature of the problem because we want to explore the space slowly to avoid large input costs. So any type of initialization would be costly.

We simulate this in the following way: we first randomly sample $\max(T/5, 10d)$ points from the d-dimensional input space, and train a GP on these data-points. The hyper-parameters of the GP are optimized by maximizing the marginal log-likelihood [Rasmussen and Williams, 2005] over 500 epochs using Adam [Kingma and Ba, 2014] with a learning rate of 0.01. The resulting hyper-parameters will correspond to the 'educated guess'. We then set the following bounds:

a) For the length-scale, the lower bound is half the educated guess, and the upper bound double the 'educated guess'.

b) For the output-scale, θ_0, the lower bound is half the educated guess, and the upper bound double the 'educated guess'.

c) For the initial mean, μ_0, the lower bound is the educated guess minus a third of the initial variance, the upper bound is the educated guess plus a third of the initial variance.

d) The noise parameter we simply set to be greater than 10^{-5}.

We (partially) enforce the constraints by setting a SmoothedBoxPrior on each parameter, with a variance of 0.001. Finally, under the constraints defined above, we re-estimate the hyper-parameters every time we obtain 25 new observations.

To make sure all models receive fair initializations, we set the same seed for each run and function pair.

A.2.3 SnAKe

We used Simulated Annealing [Kirkpatrick et al., 1983] to solve the Travelling Salesman Problem. We implemented it using the NetworkX package [Hagberg et al., 2008]. We initalized the cycle with the 'greedy' sub-algorithm and used all default options.

We generated the Thompson Samples using the method introduced in Wilson et al. [2020] which we implemented ourselves. To optimize the samples we used Adam [Kingma and Ba, 2014] and PyTorch [Paszke et al., 2019] over $10d$ epochs, with a learning rate of 0.01. We used $10d$ multi-starts for each sample. To create the samples, we used $\ell = 1024$ Fourier bases.

For ℓ-SnAKe, we define an adaptive deletion constant, $\epsilon_t = \min(\ell_{1,t}, ..., \ell_{d,t})$, where $\ell_{i,t}$ denotes the length scale of the ith variable at time t (recall we are re-training the hyper-parameters every new 25 observations, so the length scales change with time).

For the adaptive grid, we use $N_l = 25$ local samples, and a corse global Sobol grid [Sobol', 1967] of $N_g = 100$ points.

A.2.4 Classical Bayesian Optimization

We used BoTorch [Balandat et al., 2020] to implement all methods in this section. We optimized the acquisition functions across 150 epochs using Adam [Kingma and Ba, 2014] with a learning rate of 0.0001 using 7500 random multi-starts.

Cost-aware Expected Improvement

Expected Improvement per Unit Cost Snoek et al. [2012] optimizes the acquisition function:

$$EI(x, x_{t-1}) = \frac{EI(x))}{C(x, x_{t-1}) + \gamma}$$

We set $\gamma = 1$ in all experiments to address the fact that not moving incurs zero cost. Note that as $\gamma \to \infty$, the method will behave closer to normal Expected Improvement, and as $\gamma \to 0$ the method will tend to stay closer and closer to the current input.

Truncated Expected Improvement

Truncated Expected Improvement was developed by Samaniego et al. [2021] trying to solve the task of using an automated boat to monitor the water quality of Ypacari Lake in Paraguay. The method seeks to take distance travelled into account when doing Bayesian Optimization. It does by selecting a point using Expected Improvement, setting a direct path from our current input to the new point, and then *truncating* the path one length-scale away. That is:

$$p_t = \arg\max_{x \in \mathcal{X}} EI(x \mid D_t)$$
$$x_{t+1} = x_t + \frac{p_t - x_t}{||p_t - x_t||} \min(\ell, ||p_t - x_t||)$$

A.2.5 Asynchronous Bayesian Optimization

Local Penalization

We use the penalization method as is described in González et al. [2016a]. For the Lipschitz constant, we estimate it by calculating the gradient of μ_t (using auto-differentiation) in a Sobol grid [Sobol', 1967] of $50d$ points and selecting L to be the maximum gradient in the grid.

UCB with Local Penalization We set $\beta_t = 0.2d\log(2t)$ and $M = y_{best}$.

EIpu with Local Penalization We set $\gamma = 1$.

Thompson Sampling

We use the sample procedure as in A.2.3, except we only optimize a single sample at every iteration.

A.2.6 Description of Benchmark Functions

We chose the benchmark functions to observe the behavior of SnAKe in a variety of scenarios. More details of all the benchmark functions can be found in Surjanovic and Bingham [2013].

Branin2D

The two-dimensional Branin function is given by. The function has three global maximums:

$$f(x) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1-t)\cos(x_1) + s$$

where we optimize over $\mathcal{X} = [-5, 10] \times [0, 15]$. We set $a = -1$, $= 5.1/(4\pi^2)$, $c = 5/\pi$, $r = 6$, $s = -10$, and $t = 1/(8\pi)$.

Ackley4D

The four-dimensional Ackley function has a lot of local optimums, with the optima in the center of the search space. The function is given by:

$$f(x) = a \exp\left(-b\sqrt{\frac{1}{4}\sum_{i=1}^{4} x_i^2}\right) + \exp\left(\frac{1}{d}\sum_{i=1}^{4}\cos(cx_i)\right) - a - \exp(1)$$

where we slightly shift the search space and optimize over $\mathcal{X} = [-1.8, 2.2]^4$. This is to avoid having the optimum exactly at a point in the Sobol grid and giving SnAKe an unfair advantage. We set $a = 20$, $= 0.2$, and $c = 2\pi$.

Michaelwicz2D

The two-dimensional Michalewicz function is characterized by multiple local maxima and a lot of flat regions. The function is given by:

$$f(x) = \sum_{i=1}^{2} \sin(x_i) \sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$$

where we set $m = 10$ and we optimize on the region $\mathcal{X} = [0, \pi]^2$.

Hartmann

We select this function to see how the algorithms behave in *similar* functions as dimension increases. We do three versions of the Hartmann function, with dimensions $d = 3, 4$, and 6. The equation is given by:

$$f(x) = \sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{d} A_{ij}(x_j - P_{ij})\right)$$

where $\alpha = (1, 1.2, 3, 3.2)^T$. For $d = 3$ we use:

$$A = \begin{pmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix} \qquad P = 10^{-4} \begin{pmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{pmatrix}$$

for $d = 4$ and $d = 6$ we use:

$$A = \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix} \qquad P = 10^{-4} \begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{pmatrix}$$

They are all evaluated on the unit cube $[0, 1]^d$.

Perm10D

We select the 10-dimensional version of the Perm benchmark to test the capabilities of the algorithms in a very high-dimensional setting (by BO standards). The equation is given by:

$$f(x) = -10^{-21} \sum_{i=1}^{10} \left(\sum_{j=1}^{10} (j^i + \beta) \left(\left(\frac{x_j}{j} \right)^i - 1 \right) \right)^2$$

where we set $\beta = 10$. We evaluate it on $\mathcal{X} = [-10, 10]^d$.

SnAr4D

We implement the simulation using Summit Felton et al. [2021]. We control temperature between 40 and 120 degrees, concentration from 0.1 to 0.5 moles per liter, and residence time between 0.5 and 2 minutes. We set the cost parameters for temperature $\alpha_{temp} = 5, \beta_{temp} = 1$, $\gamma_{temp} = 1$, for concentration $\alpha_{conc} = 2, \beta_{conc} = 0.01$, $\gamma_{conc} = 1$, and for residence time $\alpha_{residence} =$

3, $\beta_{residence} = 0.05$ and $\gamma_{residence} = 1$. We further optimize over the equivalents of pyrrolidine between 1 and 5 units, but we assume changing it incurs no input costs.

Since the SnAr benchmark is a multi-objective problem, we optimize a weighted sum of the two objectives:

$$\text{SnAr}(x) = \omega_1 \times \text{yield} - \omega_2 \times \text{e-factor}$$

where we set $\omega_1 = 10^{-4}$ and $\omega_2 = 0.1$. We optimize over $\mathcal{X} = [40, 120] \times [0.1, 0.5] \times [0.5, 2] \times [1, 5]$.

A.2.7 Ypacarai Implementation Details

As per Samaniego et al. [2021] we model the Lake using the Shekel function which is given as:

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

For the objective we use $m = 2$, $m = 3$, and $m = 2$. The other parameters are:

$$C^{(1)} = \begin{pmatrix} 2 & 6.7 \\ 9 & 2 \end{pmatrix} \qquad C^{(2)} = \begin{pmatrix} 7 & 6 \\ 3.8 & 9.9 \\ 9 & 0.1 \end{pmatrix} \qquad C^{(3)} = \begin{pmatrix} 4 & 3 \\ 8.5 & 4 \end{pmatrix}$$

$$\beta^{(1)} = \begin{pmatrix} 9 & 9 \end{pmatrix} \qquad \beta^{(2)} = \begin{pmatrix} 10 & 8 & 8 \end{pmatrix} \qquad \beta^{(3)} = \begin{pmatrix} 7 & 9 \end{pmatrix}$$

Where $C^{(i)}$ and $\beta^{(i)}$ represent the parameters of the ith objective. All objectives are optimized across on a subset of $\mathcal{X} = [0, 1]^2$. The subset is defined by creating grid of points mapping a high-resolution black and white image of Lake Ypacarai onto $\mathcal{X}$. For simplicity, we assume the cost of moving from one point to another is simply the distance between the points, even though in practice we might need to take a longer route to avoid land. For Truncated Expected Improvement, we simply project the truncation into the closest point grid, to avoid sampling points outside of the Lake.

A.3 Transition Constrained Bayesian Optimisation via Markov Decision Processes

In this section we provide additional implementation details of Chapter 5.

A.3.1 Benchmark Details

For all benchmarks, aside from the knorr pyrazole synthesis example, we use a standard squared exponential kernel for the surrogate Gaussian Process:

$$k_{rbf}(x, x') = \sigma_{rbf}^2 \exp\left(-\frac{(x - x')^2}{2\ell_{rbf}}\right)$$

where σ_{rbf}^2 is the prior variance of the kernel, and ℓ_{rbf} the kernel. We fix the values of all the hyper-parameters a priori and use the same for all algorithms. The hyper-parameters for each benchmarks are included in Table A.1.

For the knorr pyrazole synthesis example, we further set $\alpha_{ode} = 0.6$, $\alpha_{rbf} = 0.001$, $k_1 = 10$, $k_2 = 874$, $k_3 = 19200$, $\alpha_{sig} = 5$. Recall we are using a finite dimensional estimate of a GP such that:

$$f(x) \approx \omega_{ode}\Phi_{ode}(x) + \sum_{i=1}^{M} \omega_{rbf,i}\Phi_{rbf}(x) \tag{A.1}$$

in this case we set a prior to the ODE weight such that $\omega_{ode} \sim \mathcal{N}(0.6, 0.0225)$. This is incorporating two key pieces of prior knowledge that (a) the product concentration should be positive, and (b) we expect a maximum product concentration between 0.15 and 0.45.

The number of features for each experiment, M, is set to be $M = |\mathcal{X}|$ in the discrete cases and $M = \min\left(2^{5+d}, 512\right)$ where d is the problem dimensionality.

In the case of Local Search Region BayesOpt (LSR) [Paulson et al., 2023] we set the exploration hyper-parameter to be $\gamma = 0.01$ in all benchmarks.

Table A.1: Benchmark and hyper-parameter information. Δ_{max} represents the size of the box constraints in the traditional benchmarks. For the synchronous benchmarks and for SnAr we used a noise level of $\sigma^2 = 0.001$. For the asynchronous benchmarks, and the knorr pyrazole example we used $\sigma^2 = 0.0001$. For the Ypacarai example we used $\sigma^2 = 0.001$ and $\sigma^2 = 0.01$ for the episodic and immediate feedback respectively.

Benchmark Name	Δ_{max}	**Variance** σ_{rbf}	**Lengthscale** ℓ_{rbf}
Knorr pyrazole	–	0.001	0.1
Constrained Ypacarai	–	1	0.2
Branin2D	0.05	0.6	0.15
Hartmann3D	0.1	2.0	0.13849
Hartmann6D	0.2	1.7	0.22
Michalewicz2D	0.05	0.35	0.179485
Michalewicz3D	0.1	0.85	0.179485
Levy4D	0.1	0.6	0.14175
SnAr	0.1	0.8	0.2

A.3.2 Free-electron Laser

We use the simulator from Mutný et al. [2020] that optimizes quadrupole magnet orientations for our experiment with varying noise levels. We use a 2-dimensional variant of the simulator. We discretize the system on 10×10 grid and assume that the planning horizon $H = 100$. The simulator itself is a GP fit with $\gamma = 0.4$, hence we use this value. Then we make a choice that the noise variance is proportional to the change made as $\sigma^2(x, a) = s(1 + w||x - a||^2)$, where $s = 0.01$ and $w = 20$. Note that $x \in [-0.5, 0.5]^2$ in this modeling setup. This means that local steps are indeed very desired. We showcase the difference to classical BayesOpt, which uses the worst-case variance $\sigma = \sup_{x,a} s(1 + w||x - a||^2)$ for modeling as it does not take into account the state in which the system is. We see that the absence of state modeling leads to a dramatic decrease in performance as indicated by much higher inference regret in Figure 5.6b.

Appendix B

Derivations

B.1 Transition Constrained Bayesian Optimisation

B.1.1 Objective reformulation and linearization

For the main objective we try to optimize over a subset of T trajectories $\mathbf{X} \subset \mathcal{X}^H$. Let be $\mathcal{X}^H$ be the set of sequences of inputs $\tau = (x_1, ..., x_H)$ where they consist of states in the search space $\mathcal{X}$. Furthermore, assume there exists, in the deterministic environment, a constraint such that $x_{h+1} \in \mathcal{C}(x_h)$ for all $h = 1, ..., H-1$. Then we seek to find the set $\mathbf{X}_*$, consisting of T trajectories (possibly repeated), such that we solve the constrained optimization problem:

$$\mathbf{X}_* = \underset{\mathbf{X} \in \mathcal{X}^H}{\arg\min} \max_{z, z' \in \mathcal{Z}} \text{Var}[f(z) - f(z')|\mathbf{X}] \quad \text{s.t.} \quad x_{h+1} \in \mathcal{C}(x_h) \quad \forall t = 1, ..., h-1 \tag{B.1}$$

We define the objective as:

$$U(\mathbf{X}) = \max_{z, z' \in \mathcal{Z}} \text{Var}\left[f(z) - f(z')|\mathbf{X}\right] \tag{B.2}$$

Our goal is to show that optimization over sequences can be simplified to state-action visitations as in Mutný et al. [2023]. For this, we require that the objective depends additively involving terms x, a separately. We formalize this in the next result. In order to prove the result, we

utilize the theory of reproducing kernel Hilbert spaces [Cucker and Smale, 2002] [1].

Lemma B.1 (Additivity of Best-arm Objective)**.** *Let* $\mathbf{X}$ *be a collection of* t *trajectories of length* H*. Assuming that* $f \sim \mathcal{GP}(0, k)$*. Assuming that* k *has Mercer decomposition as* $k(x, y) = \lambda_k \phi_k(x)\phi_k(y)$*.*

$$f(x) = \sum_k \phi_k(x)\theta_k \quad \theta_k \sim \mathcal{N}(0, \lambda_k).$$

Let $d_{\mathbf{X}}$ *be the visitation of the states-action in the trajectories in* $\mathbf{X}$*, as* $d_{\mathbf{X}} = \frac{1}{HT}\sum_{t=1}^{T}\sum_{x,a\in\tau_t}\delta_{x,a}$*, where the* δ *represent delta function supported on* x, a*. Then optimization of the objective Eq.* (B.1) *can be rewritten as:*

$$U(d_{\mathbf{X}}) = \max_{z,z'\in\mathcal{Z}} ||\Phi(z) - \Phi(z')||^2_{\mathbf{V}(d_{\mathbf{X}})^{-1}},$$

where $\mathbf{V}(d) = \sum_i \sum_{x,a\in\tau_i} d(x, a)\Phi(x)\Phi(x)^\top + \frac{\sigma^2}{TH}\mathbf{I}$ *is a operator* $\mathbf{V}(d) : \mathcal{H}_k \to \mathcal{H}_k$*, the norm is RKHS norm, and* $\Phi(z)_k = \phi_k(z)$*.*

Proof. Notice that the posterior GP of any two points z, z' is $(f(z), f(z')) = \mathcal{N}((\mu(z), \mu(z')), \mathbf{K}_{z,z'})$, where $\mathbf{K}_{z,z'}$ is posterior kernel (consult Rasmussen and Williams [2005] for details) defined via a posterior kernel $k_{\mathbf{X}}(z, z') = k(z, z') - k(z, \mathbf{X})(\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I})^{-1}k(\mathbf{X}, z')$.

Let $k_t(z) = \Phi(\mathbf{X})\phi(z)$. Utilizing $k(z, z') = \Phi(z)^\top\Phi(z)$ (RKHS inner product) with the Mercer decomposition and the matrix inversion lemma, the above can be written as using $\mathbf{V} = \sum_{t=1}^{T}\sum x \in \tau_t\Phi(x)\Phi(x)^\top + \sigma^2\mathbf{I}_{\mathcal{H}}$.

$$\begin{aligned}
k_{\mathbf{X}}(z, z') &= k(z, z') - k_t(z)^\top(\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma^2\mathbf{I})^{-1}k_t(z') \\
&\overset{\text{Mercer}}{=} \Phi(z)^\top\Phi(z') - \Phi(z)^\top\Phi(\mathbf{X})^\top(\Phi(\mathbf{X})\Phi(\mathbf{X})^\top + \sigma^2\mathbf{I})^{-1}\Phi(\mathbf{X})\Phi(z') \\
&\overset{\text{Lemma B.2}}{=} \Phi(z)^\top\Phi(z') - \Phi(z)^\top\mathbf{V}^{-1}(\mathbf{V} - \mathbf{I}\sigma^2)\Phi(z') \\
&= \Phi(z)^\top\mathbf{V}^{-1}\mathbf{V}\Phi(z') - \Phi(z)^\top\mathbf{V}^{-1}(\mathbf{V} - \mathbf{I}\sigma^2)\Phi(z') \\
&= \Phi(z)^\top\mathbf{V}^{-1}(\mathbf{V} - \mathbf{V} + \mathbf{I}\sigma^2)\Phi(z')
\end{aligned}$$

[1] Proof developed by Mojmír Mutný

Leading finally to:

$$k_{\mathbf{X}}(z, z') = \sigma^2 \Phi(z)^\top \left(\sum_{t=1}^{T} \sum_{x \in \tau_t} \Phi(x)\Phi(x)^\top + \sigma^2 \mathbf{I}_{\mathcal{H}_k} \right)^{-1} \Phi(z'). \tag{B.3}$$

Let us calculate $\mathrm{Var}[f(z) - f(z')|\mathbf{X}]$. The variance does not depend on the mean. Hence,

$$\begin{aligned}
& \mathrm{Var}\left[f(z) - f(z')|\mathbf{X}\right] \\
= \;& \mathrm{Var}(f(z)) - \mathrm{Var}(f(z')) - 2\mathrm{Cov}(f(z), f(z')) \\
= \;& k_{\mathbf{X}}(z, z) + k_{\mathbf{X}}(z', z') - 2k_{\mathbf{X}}(z, z') \\
\overset{(B.3)}{=} \;& (\Phi(z) - \Phi(z')) \left(\sum_{t=1}^{T} \sum_{x \in \tau_t} \Phi(x)\Phi(x)^\top + \mathbf{I}_{\mathcal{H}_k}\sigma^2 \right)^{-1} (\Phi(z) - \Phi(z')) \\
= \;& (\Phi(z) - \Phi(z')) \left(\frac{TH}{TH} \sum_{t=1}^{T} \sum_{x \in \mathcal{X}} \#(x \in \tau_t)\Phi(x)\Phi(x)^\top + \mathbf{I}_{\mathcal{H}_k}\sigma^2 \right)^{-1} (\Phi(z) - \Phi(z')) \\
= \;& TH(\Phi(z) - \Phi(z')) \left(\sum_{x \in \mathcal{X}} d(\mathbf{X})\Phi(x)\Phi(x)^\top + \frac{\sigma^2}{TH}\mathbf{I}_{\mathcal{H}_k} \right)^{-1} (\Phi(z) - \Phi(z'))
\end{aligned}$$

The symbol # counts the number of occurrences. Notice that we have been able to show that the objective decomposes over state-action visitations as $d_{\mathbf{X}}$ decomposes over their visitations □

Note that the objective equivalence does *not* imply that optimization problem in Eq. (B.1) is equivalent to finding,

$$d_* = \arg\min_{d_\pi \in \mathcal{D}} \max_{z, z' \in \mathcal{Z}} ||\Phi(z) - \Phi(z')||_{\mathbf{V}(d_\pi)^{-1}}. \tag{B.4}$$

In other words, optimization over trajectories and optimization over $d_\pi \in \mathcal{D}$ is not equivalent. The latter is merely a continuous relaxation of discrete optimization problems to the space of Markov policies. It is in line with the classical relaxation approach addressed in experiment design literature with a rich history, e.g., Chaloner and Verdinelli [1995]. For introductory texts on the topic, consider Pukelsheim [2006] for the statistical perspective and Boyd and Vandenberghe [2004] for the optimization perspective. However, as Mutný et al. [2023] points out, optimization of this objective does optimize objective as Eq. (B.1). In other words, by optimizing the relaxation with a larger budget of trajectories or horizons, we are able to decrease

Eq. (B.1) as well.

For completeness, we state the auxiliary lemma.

Lemma B.2 (Matrix Inversion Lemma)**.** *Let* $\mathbf{A} \in \mathbb{R}^{n \times q}$ *then*

$$\mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top + \rho^2 \mathbf{I})^{-1} = (\mathbf{A}^\top \mathbf{A} + \rho^2 \mathbf{I})^{-1} \mathbf{A}^\top. \tag{B.5}$$

Note that instead of inverting $n \times n$ *matrix, we can invert a* $q \times q$ *matrix.*

Proof.

$$\begin{aligned}
\mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top + \rho^2 \mathbf{I})^{-1} &\overset{\text{SMW}}{=} \mathbf{A}^\top (\rho^{-2}\mathbf{I} - \rho^{-2}\mathbf{A}(\rho^2 \mathbf{I} + \mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top) \\
&= (\rho^{-2}\mathbf{I} - \rho^{-2}\mathbf{A}^\top \mathbf{A}(\rho^2 \mathbf{I} + \mathbf{A}^\top \mathbf{A})^{-1}) \mathbf{A}^\top \\
&= (\rho^{-2}(\rho^2 \mathbf{I} + \mathbf{A}^\top \mathbf{A}) - \rho^{-2}\mathbf{A}^\top \mathbf{A})(\rho^2 \mathbf{I} + \mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \\
&= (\mathbf{A}^\top \mathbf{A} + \rho^2 \mathbf{I})^{-1} \mathbf{A}^\top
\end{aligned}$$

□

B.1.2 Theory: convergence to the optimal policy

The fact that our objective is derived using Bayesian decision theory makes it well-rooted in theory. We can prove that our scheme is able to converge in terms of the utility [2].

Notice that the set of potential maximizers is changing over time, and hence we add a time subscript to $\mathcal{Z}$ as $\mathcal{Z}_t$. Let us contemplate for a second what could the optimal policy. As the set of $\mathcal{Z}_t$ is changing, we follow the line of work of started by Russo [2016] and introduce an optimal algorithm that knows the true $x_f^\star$ for each possible realization of the prior f. In other words, its an algorithm that any time t, would follow:

$$d_t^\star = \min_{d \in \mathcal{D}} \mathbb{E}_f \left[\max_{z \in \mathcal{Z}_t \setminus \{x_f^\star\}} k_{\hat{d}_t \oplus d}(z, x_f^\star) \right],$$

[2] Proof developed by Mojmír Mutný

where in the above $\hat{d}_t \oplus d$ represents the weighted sum as in the main Algorithm 10 that scales them properly according to t and T. Notice that in contrast to our objective, it does not take the maximum over $z' \in \mathcal{Z}$, but fixes it to the value $x_f^\star$ that the hypothetical algorithm has privileged access to. To eliminate the cumbersome notation, we will refer to the objectives as $\mathcal{U}(d|\mathcal{Z}_t, \mathcal{Z}_t)$ as the objective used by our algorithm (real execution) and $\mathcal{U}(d|\mathcal{Z}_t, \{x_f^\star\})$, as the objective that the privileged algorithm is optimizing which serves as theoretical baseline.

The visitation of $d_t^\star$ represents the best possible investment of the resources (of the size $T - t$) to execute at time t had we known the $x_f^\star$ instead of only $\mathcal{Z}_t$. This is interpreted as if the modeller knows $x_f^\star$, and sets up an optimal curriculum that is being shown to an observer in order to convince him/her of that $x_f^\star$ is the optimal value. He or she is using statistical testing to elucidate it from execution of the policy. Like the algorithm, the optimal policy changes along the optimization procedure due to changes in $\mathcal{Z}_t$. Hence, our goal is to show that we are closely tracking the performance of these optimal policies in time t, and eventually there is little difference between our sequence of executed policies (visitations) $\hat{d}_t$ and the algorithm optimal $d_t^\star$.

In order to prove the theorem formally, we need to assume that $\mathcal{Z}_t$ is decreasing. The rate at which this set is decreasing determines the performance of the algorithm to a large extent. Namely, we assume that given two points in time, having the same empirical information $\hat{d}_t$. Given, f, suppose

$$\sup_{d \in \mathcal{D}} |d^\top (\nabla \mathcal{U}(\hat{d}_t|\mathcal{Z}_t, \mathcal{Z}_t) - \nabla \mathcal{U}(\hat{d}_t|\mathcal{Z}_t, \{x_f^\top\}))| \leq C_t. \qquad (\star)$$

As we gather information in our procedure the, $\{x_f^\star\} \subset \mathcal{Z}_t \subseteq \mathcal{Z}_{t-1}$, but the exact decrease depends on how $\mathcal{Z}_t$ is constructed. We leave the particular choice for C_t to make the above hold for future work. We conjecture that this is decreasing as $C_t \approx \frac{\gamma_t}{\sqrt{t}}$, where γ_t is the information gain due to Srinivas et al. [2010]. We are now ready to state the formal theorem along with its assumptions.

Proposition B.3. *Assuming episodic feedback, and suppose that for any $\mathcal{Z}$,*

1. *$\mathcal{U}$ is convex on $\mathcal{D}$*

2. *B-locally Lipschitz continuous under* $||\cdot||_\infty$ *norm*

3. *locally smooth with constant* L*, i.e,*

$$\mathcal{U}(\eta + \alpha h) \leq \mathcal{U}(\eta) + \nabla\mathcal{U}(\eta)^\top h + \frac{L_{\eta,\alpha}}{2}||h||_2^2. \tag{B.6}$$

for $\alpha \in (0,1)$ *and* $\eta, h \in \Delta_p$*,* $L := \max_{\eta,\alpha} L_{\eta,\alpha}$

4. *condition in* $(\star)$ *holds with Bayesian posterior inference,*

we can show that the Algorithm 10 satisfied for the sequences of iterates $\{\hat{d}_t\}_{t=1}^T$*:*

$$\frac{1}{T}\sum_{t=1}^{T-1}\mathcal{U}(d_t|\mathcal{Z}_t, \{x_f^\star\}) - \mathcal{U}(d_t^\star|\mathcal{Z}_t, \{x_f^\star\}) \leq \mathcal{O}\left(\frac{1}{T}\sum_{t=1}^{T-1} C_t + \frac{L\log T}{T} + \frac{B}{\sqrt{T}}\log\left(\frac{1}{\delta}\right)\right),$$

with probability $1-\delta$ *on the sampling from the Markov chain. The randomness on the confidence set is captured by Assumption in Eq.* $(\star)$*.*

The previous proposition shows that as the budget of the experimental campaign T is increasing, we are increasingly converging to the optimal allocation of the experimental resources on average also on the objective that is unknown to us. In other words, our algorithm is becoming approximately optimal also under the privileged information setting representing the best possible algorithm. Despite having a limited understanding of potential maximizers at the beginning by following our procedure, we show that we are competitive to the best possible allocation of the resources. Now, we prove the Proposition. The proof is an extension of the Theorem 3 in [Mutný et al., 2023]. Whether the objective satisfied the above conditions depends on the set $\mathcal{X}$. Should the objective not satisfy smoothness, it can be easily extended by using the Nesterov smoothing technique as explained in the same priorly cited work.

Proof of Proposition B.3. The proof is based on the proof of Frank-Wolfe convergence that appears Appendix B.4 in Thm. 3. in Mutný et al. [2023].

Let us start by notation. We will use the notation that $\mathcal{U}_t$ is the privileged objective $\mathcal{U}(d|\mathcal{Z}_t, \{x_t^f\})$, while the original objective will be specified as $\mathcal{U}(d|\mathcal{Z}_t, \mathcal{Z}_t)$.

First, what we follow in the algorithm:

$$q_t = \arg\min_{d \in \mathcal{D}} \nabla \mathcal{U}(\hat{d}_t | \mathcal{Z}_t, \mathcal{Z}_t)^\top d \tag{B.7}$$

The executed visitation is simply generated via sampling a trajectory from q_t. Let us denote the empirical visiation of the trajectory as δ_t,

$$\delta_t \sim q_t. \tag{B.8}$$

For the analysis, we also need the best greedy step for the unknown (privileged) objective $\mathcal{U}$ as

$$z_t = \arg\min_{d \in \mathcal{D}} \nabla \mathcal{U}_t \left(\hat{d}_t\right)^\top d. \tag{B.9}$$

Let us start by considering the one step update:

$$\begin{aligned}
\mathcal{U}_t(\hat{d}_{t+1}) &= \mathcal{U}_t\left(\hat{d}_t + \frac{1}{t+1}(\delta_t - \hat{d}_t)\right) \\
&\overset{L\text{-smooth}}{\leq} \mathcal{U}_t(\hat{d}_t) + \frac{1}{t+1}\nabla \mathcal{U}_t(\hat{d}_t)^\top(\delta_t - \hat{d}_t) + \frac{L}{2(1+t)^2}||\delta_t - \hat{d}_t||^2 \\
\mathcal{U}(\hat{d}_{t+1}) &\overset{\text{bounded}}{\leq} \mathcal{U}_t(\hat{d}_t) + \frac{1}{t+1}\nabla \mathcal{U}_t(\hat{d}_t)^\top(\delta_t - \hat{d}_t) + \frac{L}{(1+t)^2} \\
&= \mathcal{U}_t(\hat{d}_t) + \frac{1}{t+1}\nabla \mathcal{U}_t(\hat{d}_t)^\top(q_t - \hat{d}_t) + \frac{1}{t+1}\underbrace{\nabla \mathcal{U}_t(\hat{d}_t)^\top(-q_t + \delta_t)}_{\epsilon_t} + \frac{L}{(1+t)^2}
\end{aligned}$$

We will now carefully insert and subtract two set of terms depending on the real objective so that we can bound them usign ($\star$):

$$\begin{aligned}
&= \mathcal{U}_t(\hat{d}_t) + \frac{1}{t+1}\nabla(\mathcal{U}_t(\hat{d}_t)^\top - \nabla \mathcal{U}_t(\hat{d}_t | \mathcal{Z}_t, \mathcal{Z}_t)^\top)(q_t - z_t) + \frac{1}{1+t}\mathcal{U}_t(\hat{d}_t)^\top(z_t - \hat{d}_t) \\
&\quad + \frac{1}{1+t}\epsilon_t + \frac{L}{(1+t)^2} \\
&\overset{\text{Using } \star}{\leq} \mathcal{U}_t(\hat{d}_t) + 2\frac{1}{1+t}C_t + \frac{1}{t+1}\mathcal{U}_t(\hat{d}_t)^\top(z_t - \hat{d}_t) + \frac{1}{1+t}\epsilon_t + \frac{L}{(1+t)^2}
\end{aligned}$$

Carrying on,

$$\begin{aligned}
\mathcal{U}_t(\hat{d}_{t+1}) &\overset{(B.9)}{\leq} \mathcal{U}_t(\hat{d}_t) + \frac{1}{t+1}\nabla\mathcal{U}_t(\hat{d}_t)^\top(d_t^\star - \hat{d}_t) + \frac{1}{1+t}\epsilon_t + \frac{L}{(1+t)^2} \\
&\overset{\text{convexity}}{\leq} \mathcal{U}_t(\hat{d}_t) - \frac{1}{t+1}(\mathcal{U}_t(\hat{d}_t) - \mathcal{U}_t(\eta_t^\star)) + \frac{1}{1+t}\epsilon_t + \frac{L}{(1+t)^2} + \frac{1}{1+t}C_t \\
\mathcal{U}_t(\hat{d}_{t+1}) - \mathcal{U}_t(d_t^\star) &\leq \mathcal{U}_t(\hat{d}_t) - \mathcal{U}_t(d_t^\star) - \frac{1}{t+1}(\mathcal{U}_t(\hat{d}_t) - \mathcal{U}_t(d_t^\star)) + \frac{1}{1+t}\epsilon_t + \frac{L}{(1+t)^2} + \frac{1}{1+t}C_t \\
&\leq \frac{t}{1+t}\left(\mathcal{U}_t(\hat{d}_t) - \mathcal{U}_t(d_t^\star)\right) + \frac{1}{1+t}\epsilon_t + \frac{L}{(1+t)^2} + \frac{1}{1+t}C_t \\
&= \frac{t}{1+t}\left(\mathcal{U}_t(\hat{d}_t) - \mathcal{U}_t(d_t^\star)\right) + \frac{1}{1+t}\epsilon_t + \frac{L}{(1+t)^2} + \frac{1}{1+t}C_t
\end{aligned}$$

Now multiplying by $t+1$ both sides, and summing on $\frac{1}{T-1}\sum_{t=1}^{T-1}$. Using the shorthand $\rho_t(\hat{d}_t) = \mathcal{U}_t(\hat{d}_t) - \mathcal{U}_t(d_t^\star)$ we get:

$$\frac{1}{T}\sum_{t=1}^{T-1}(t+1)\rho_{t+1}(\hat{d}_{t+1}) \leq \frac{1}{T}\sum_{t=1}^{T-1} t\rho_t(\hat{d}_t) + \frac{1}{T}\sum_{t=1}^{T-1}(\epsilon_t + C_t + L/(1+t))$$

First notice that $\frac{1}{T-1}\sum_{t=1}^{T-1}\epsilon_t \leq \frac{B}{\sqrt{T}}\log(1/\delta)$ by Lemma in Mutný et al. [2023] due to ϵ_t being martingale difference sequence. The other term is the sum on $\frac{1}{T-1}\sum_{t=1} C_t$ which appears in the main result. The sum on $\sum_{t=1}^{T-1}\frac{1}{1+t} \leq L\frac{\log T}{T}$. The rest is eliminated by the reccurence of the terms, and using that $\mathcal{U}(d|\mathcal{Z}_t, \{x_f^\star\} \leq \mathcal{U}(d|\mathcal{Z}_{t-1}, \{x_f^\star\})$ for any d. This is due to set $\mathcal{Z}_t$ decreasing over time. We report the result in asymptotic notation as function of T and $\log(1/\delta)$. □

B.2 Kernel for ODE Knorr pyrazole synthesis

The kernel is based on the following ODE model, which is well known in the chemistry literature and given in Schrecker et al. [2023].

$$R_1 = k_1 y_2 y_3 - k_2 y_4 y_5 \tag{B.10}$$

$$R_2 = k_3 y_4 \tag{B.11}$$

and then:

$$\begin{aligned}
\frac{\mathrm{d}y_1}{\mathrm{d}t} &= R_2 \\
\frac{\mathrm{d}y_2}{\mathrm{d}t} &= -R_1 \\
\frac{\mathrm{d}y_3}{\mathrm{d}t} &= -R_1 \\
\frac{\mathrm{d}y_4}{\mathrm{d}t} &= R_1 - R_2 \\
\frac{\mathrm{d}y_5}{\mathrm{d}t} &= R_1 + R_2
\end{aligned}$$

Our main goal is to optimize the product concentration of the reaction, which is given by y_1. We do this by sequentially querying the reaction, where we select the residence time, and the initial conditions of the ODE, in the form $y_0 = [0, A, B, 0, 0]$, where $A = 1 - B$.

Due to the non-linearity in Eq. (B.10) we are unable to fit a GP to the process directly. Instead, we first linearize the ODE around two equilibrium points. The set of points of equilibrium are given by:

$$S_1 = \{y_1 = a_1, y_2 = b_1, y_3 = 0, y_4 = 0, y_5 = c_1 | a_1, b_1, c_1 \in \mathbf{R}\}$$

$$S_2 = \{y_1 = a_2, y_2 = 0, y_3 = b_2, y_4 = 0, y_5 = c_2 | a_1, b_1, c_1 \in \mathbf{R}\}$$

And the Jacobian of the system is:

$$\mathbf{J} = \begin{bmatrix} 0 & 0 & 0 & k_3 & 0 \\ 0 & -k_1 y_3 & -k_1 y_2 & k_2 y_5 & k_2 y_4 \\ 0 & -k_1 y_3 & -k_1 y_2 & k_2 y_5 & k_2 y_4 \\ 0 & k_1 y_3 & k_1 y_2 & -k_2 y_5 - k_3 & -k_2 y_4 \\ 0 & k_1 y_3 & k_1 y_2 & -k_2 y_5 + k_3 & -k_2 y_4 \end{bmatrix}$$

Giving:

$$\mathbf{J}_1 = \mathbf{J}|_{S_1} = \begin{bmatrix} 0 & 0 & 0 & k_3 & 0 \\ 0 & 0 & -k_1 b_1 & k_2 c_1 & 0 \\ 0 & 0 & -k_1 b_1 & k_2 c_1 & 0 \\ 0 & 0 & k_1 b_1 & -k_2 c_1 - k_3 & 0 \\ 0 & 0 & k_1 b_1 & -k_2 c_1 + k_3 & 0 \end{bmatrix}$$

$$\mathbf{J}_2 = \mathbf{J}|_{S_2} = \begin{bmatrix} 0 & 0 & 0 & k_3 & 0 \\ 0 & -k_1 b_2 & 0 & k_2 c_2 & 0 \\ 0 & -k_1 b_2 & 0 & k_2 c_2 & 0 \\ 0 & k_1 b_2 & 0 & -k_2 c_2 - k_3 & 0 \\ 0 & k_1 b_2 & 0 & -k_2 c_2 + k_3 & 0 \end{bmatrix}$$

Unfortunately, since the matrices are singular, we do not get theoretical results on the quality of the linearization. However, linearization is still possible, with the linear systems given by:

$$\frac{\mathrm{d}\vec{y}}{\mathrm{d}t} = \mathbf{J}_1 \vec{y} \qquad \frac{\mathrm{d}\vec{y}}{\mathrm{d}t} = \mathbf{J}_2 \vec{y}$$

We focus on the first system for now. The matrix has the following eigenvalues:

$$\lambda_{1,2} = -\frac{1}{2}\left(b_1 k_1 + c_1 k_2 + k_3 \pm \sqrt{b_1^2 k_1^2 + c_1^2 k_2^2 + k_3^2 + 2 b_1 c_1 k_1 k_2 - 2 k_3 (b_1 k_1 - c_1 k_2)}\right)$$

$$\lambda_{3,4,5} = 0$$

Note that the three eigenvalues give us the corresponding solution based on their (linearly separable) eigenvectors:

$$v_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad v_4 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \quad v_5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\vec{y}(t) = p_3 v_3 + p_4 v_4 + p_5 v_5$$

where p_i are constants. The behaviour of the ODE when this is not the case will depend on whether the remaining eigenvalues will be real or not. However, note:

$$b_1^2 k_1^2 + c_1^2 k_2^2 + k_3^2 + 2b_1 c_1 k_1 k_2 - 2k_3(b_1 k_1 - c_1 k_2) \geq b_1^2 k_1^2 + k_3^2 - 2b_1 k_1 k_3 = (b_1 k_1 - k_3)^2 \geq 0$$

and therefore all eigenvalues will always be real. Therefore we can write down the solution as:

$$\vec{y}(t) = p_1 v_1 e^{\lambda_1 t} + p_2 v_2 e^{\lambda_2 t} + p_3 v_3 + p_4 v_4 + p_5 v_5$$

where we ignore the case of repeated eigenvalues for simplicity (this is the case where $b_1^2 k_1^2 + 2b_1 c_1 k_1 k_2 + c_1^2 k_2^2 - 2k_3(b_1 k_1 - c_1 k_2) + k_3^2$ is exactly equal to zero). We further note that the eigenvalues will be non-negative as:

$$\begin{aligned} b_1 k_1 + c_1 k_2 + k_3 &= \sqrt{(bk_1 + c_1 k_2 + k_3)^2} \\ &= \sqrt{b_1^2 k_1^2 + c^2 k_2^2 + k_3^2 + 2b_1 c_1 k_1 k_2 + 2c_1 k_2 k_2 + 2b_1 k_1 k_3} \\ &\geq \sqrt{b_1^2 k_1^2 + c_1^2 k_2^2 + k_3^2 + 2b_1 c_1 k_1 k_2 + 2c_1 k_2 k_2 - 2b_1 k_1 k_3} \end{aligned}$$

therefore:

$$\lambda_1 \leq \lambda_2 \leq 0$$

which means the solutions will always be a linear combination of exponentially decaying functions of time plus constants.

The eigenvectors have the closed form:

$$v_1 = \begin{pmatrix} 1, \\ \frac{1}{2}\left(b_1k_1 + c_1k_2 - k_3 + \sqrt{b_1^2k_1^2 + 2b_1c_1k_1k_2 + c_1^2k_2^2 - 2(b_1k_1 - c_1k_2)k_3 + k_3^2}\right)/k_3, \\ \frac{1}{2}\left(b_1k_1 + c_1k_2 - k_3 + \sqrt{b_1^2k_1^2 + 2b_1c_1k_1k_2 + c_1^2k_2^2 - 2(b_1k_1 - c_1k_2)k_3 + k_3^2}\right)/k_3, \\ -\frac{1}{2}\left(b_1k_1 + c_1k_2 + k_3 + \sqrt{b_1^2k_1^2 + 2b_1c_1k_1k_2 + c_1^2k_2^2 - 2(b_1k_1 - c_1k_2)k_3 + k_3^2}\right)/k_3, \\ -\frac{1}{2}\left(b_1k_1 + c_1k_2 - 3k_3 + \sqrt{b_1^2k_1^2 + 2b_1c_1k_1k_2 + c_1^2k_2^2 - 2(b_1k_1 - c_1k_2)k_3 + k_3^2}\right)/k_3 \end{pmatrix}$$

$$= \begin{pmatrix} 1, \\ -\lambda_1/k_3 - 1, \\ -\lambda_1/k_3 - 1, \\ \lambda_1/k_3, \\ \lambda_1/k_3 + 2 \end{pmatrix}$$

$$v_2 = \begin{pmatrix} 1, \\ \frac{1}{2}\left(b_1k_1 + c_1k_2 - k_3 - \sqrt{b_1^2k_1^2 + 2b_1c_1k_1k_2 + c_1^2k_2^2 - 2(b_1k_1 - c_1k_2)k_3 + k_3^2}\right)/k_3, \\ \frac{1}{2}\left(b_1k_1 + c_1k_2 - k_3 - \sqrt{b_1^2k_1^2 + 2b_1c_1k_1k_2 + c_1^2k_2^2 - 2(b_1k_1 - c_1k_2)k_3 + k_3^2}\right)/k_3, \\ -\frac{1}{2}\left(b_1k_1 + c_1k_2 + k_3 - \sqrt{b_1^2k_1^2 + 2b_1c_1k_1k_2 + c_1^2k_2^2 - 2(b_1k_1 - c_1k_2)k_3 + k_3^2}\right)/k_3, \\ -\frac{1}{2}\left(b_1k_1 + c_1k_2 - 3k_3 - \sqrt{b_1^2k_1^2 + 2b_1c_1k_1k_2 + c_1^2k_2^2 - 2(b_1k_1 - c_1k_2)k_3 + k_3^2}\right)/k_3 \end{pmatrix}$$

$$= \begin{pmatrix} 1, \\ -\lambda_2/k_3 - 1, \\ -\lambda_2/k_3 - 1, \\ \lambda_2/k_3, \\ \lambda_2/k_3 + 2 \end{pmatrix}$$

We are optimizing over initial set of conditions $y_0 = [0, A, B, 0, 0]$, so solving for the specific

values of the constants gives:

$$\begin{aligned} p_1 &= \frac{\lambda_2}{\lambda_1 - \lambda_2} B \\ p_2 &= -\frac{\lambda_1}{\lambda_1 - \lambda_2} B \\ p_3 &= B \\ p_4 &= A - B \\ p_5 &= 2B \end{aligned}$$

Finally, since we are setting $A = 1 - B$ and we are only optimizing the first component of $\vec{y}$ we can obtain it in closed form:

$$\begin{aligned} y_1(t, B) &= \frac{\lambda_2}{\lambda_1 - \lambda_2} B e^{\lambda_1 t} - \frac{\lambda_1}{\lambda_1 - \lambda_2} B e^{\lambda_2 t} + B \\ &= B \left(\frac{\lambda_2}{\lambda_1 - \lambda_2} e^{\lambda_1 t} - \frac{\lambda_1}{\lambda_1 - \lambda_2} e^{\lambda_2 t} + 1 \right) \end{aligned} \tag{B.12}$$

The second ODE is very similar to the first, recall it depends has the following matrix:

$$\mathbf{J}_2 = \mathbf{J}|_{s_2} = \begin{bmatrix} 0 & 0 & 0 & k_3 & 0 \\ 0 & -k_1 b_2 & 0 & k_2 c_2 & 0 \\ 0 & -k_1 b_2 & 0 & k_2 c_2 & 0 \\ 0 & k_1 b_2 & 0 & -k_2 c_2 - k_3 & 0 \\ 0 & k_1 b_2 & 0 & -k_2 c_2 + k_3 & 0 \end{bmatrix}$$

The resulting ODE is symmetric to the alternate linearization giving the same solution:

$$y(t) = p_1 v_1 e^{\lambda_1 t} + p_2 v_2 e^{\lambda_2 t} + p_3 v_3 + p_4 v_4 + p_5 v_5$$

with the only difference being the eigenvectors now are:

$$v_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad v_4 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \quad v_5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

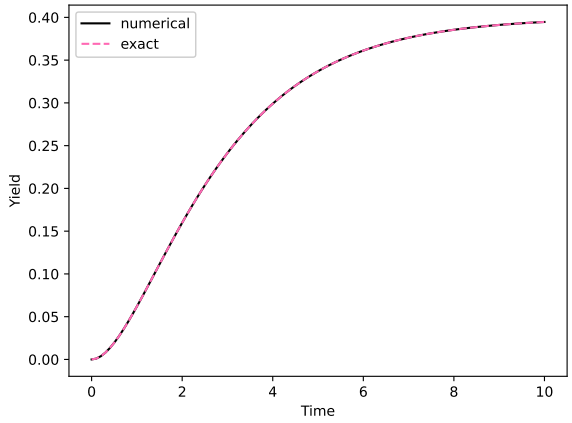


Figure B.1: Comparing the numerical solution against the solutions found in equation (B.12).

which in turn leads to solutions of the form:

$$\begin{aligned} y_1(t,B) &= \frac{\lambda_2}{\lambda_1-\lambda_2}Ae^{\lambda_1 t} - \frac{\lambda_1}{\lambda_1-\lambda_2}Ae^{\lambda_2 t} + A \\ &= A\left(\frac{\lambda_2}{\lambda_1-\lambda_2}e^{\lambda_1 t} - \frac{\lambda_1}{\lambda_1-\lambda_2}e^{\lambda_2 t} + 1\right) \end{aligned}$$

where $A = 1 - B$. Note that we now have four different eigenvalues, which depend on the linearization points:

$$\begin{aligned} \lambda_{1,2}^{(1)} &= -\frac{1}{2}\left(b_1k_1 + c_1k_2 + k_3 \pm \sqrt{b_1^2k_1^2 + c_1^2k_2^2 + k_3^2 + 2b_1c_1k_1k_2 - 2k_3(b_1k_1 - c_1k_2)}\right) \\ \lambda_{1,2}^{(2)} &= -\frac{1}{2}\left(b_2k_1 + c_2k_2 + k_3 \pm \sqrt{b_2^2k_1^2 + c_2^2k_2^2 + k_3^2 + 2b_2c_2k_1k_2 - 2k_3(b_2k_1 - c_2k_2)}\right) \end{aligned}$$

Giving solutions:

$$\begin{aligned} y_1^{(1)}(t,B) &= B\left(\frac{\lambda_2^{(1)}}{\lambda_1^{(1)}-\lambda_2^{(1)}}e^{\lambda_1^{(1)}t} - \frac{\lambda_1^{(1)}}{\lambda_1^{(1)}-\lambda_2^{(1)}}e^{\lambda_2^{(1)}t} + 1\right) \\ y_1^{(2)}(t,B) &= A\left(\frac{\lambda_2^{(2)}}{\lambda_1^{(2)}-\lambda_2^{(2)}}e^{\lambda_1^{(2)}t} - \frac{\lambda_1^{(2)}}{\lambda_1^{(2)}-\lambda_2^{(2)}}e^{\lambda_2^{(2)}t} + 1\right) \end{aligned}$$

Due to the length of the derivation, we confirm that our analysis is correct by comparing the numerical solution of the ODE to the exact solution we found in Figure B.1. Finally, we look at interpolating between the two solutions; so given the solutions $y^{(1)}(t,B)$ and $y^{(2)}(t,B)$

corresponding to the linearization with stationary point in S_1 and S_2 respectively, we consider a solution of the form:

$$y(t, B|k_1, k_2, k_3, \alpha) = (1 - \mathcal{S}(B))y^{(1)}(t, B) + \mathcal{S}(B)y^{(2)}(t, B) \tag{B.13}$$

where $\mathcal{S}(x) := (1 + e^{-\alpha_{sig}(x-0.5)})^{-1}$ is a sigmoid function centered at $B = 0.5$ and where we have introduced a new hyper-parameter α_{sig}. Finally, given Eq. (B.13) we can obtain the kernel. In particular, we want (B.13) to be a feature we are predicting on; therefore the kernel is simply the (dot) product of the features therefore:

$$k_{ode}((t, B), (t', B')) = y(t, B|k_1, k_2, k_3, \alpha) \times y(t', B'|k_1, k_2, k_3, \alpha)$$

And because we know we are simply approximating the data we can simply correct the model by adding an Gaussian Process correction; giving us the final kernel:

$$k_{joint}((t, B), (t', B')) = \alpha_{ode}k_{ode}((t, B), (t', B')) + \alpha_{rbf}k_{rbf}((t, B), (t', B'))$$

where α_{ode} and α_{rbf} are parameters we can learn, e.g. using the marginal likelihood.