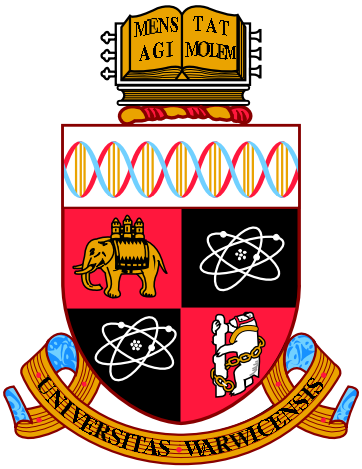


Bayesian Optimisation with Multi-Task Gaussian Processes

by

Michael Arthur Leopold Pearce

Thesis

Submitted to the University of Warwick

in partial fulfilment of the requirements

for admission to the degree of Doctor of Philosophy of

Mathematics and Complexity Science

Doctor of Philosophy

Centre for Complexity Science

September 2019

Contents

Acknowledgments

This thesis would not have been possible without the guidance, patience and kindness from my supervisor, the brilliant Professor Jürgen Branke whose endless stream of ideas and long meandering discussions that have made my PhD a pleasure, a journey with much excitement and stimulation. I am also truly grateful for much guidance and support from my external collaborator, Matthias Poloczek, whose cool calm advice and ideas led me to working on many fascinating projects.

I would also like to extend my gratitude to the students and members of staff, past and present, of the Centre for Complexity Science, particularly Jeremy Reizenstein, Janis Klaise, Rob Eyre, Ayman Boustati, Alex Bishop, Jim Skinner, Iliana Peneva, Jason Lewis, Matthew Groves, Jessie Liu and Juan Ungredda, all of whom I am always happy to sit in the common room and procrastinate, discussing random research ideas and just gossiping.

To my parents, I extend my gratitude for support in the form of free accommodation, loans, restaurant dinners. To my siblings I am grateful for the crazy awesome ski trips, intense computer games and bike rides.

Finally I owe unlimited thanks my wife, Ai Funaki, who has always provided support, tolerated my long working hours before big deadlines (including this one), organised so many things, our wedding, multiple holidays, flat hunting. Thank you, Ai!

Declarations

This thesis is submitted to the University of Warwick in support of my application for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree.
Parts of this thesis have been previously published by the author in the following:

- Michael Pearce and Jürgen Branke. Efficient information collection on portfolios. *Warwick Research Archive Portal*, 2017c
- Michael Pearce and Jürgen Branke. Efficient expected improvement estimation for continuous multiple ranking and selection. In *Proceedings of the 2017 Winter Simulation Conference*, pages 171–183. IEEE Press, 2017b
- Michael Pearce and Jürgen Branke. Continuous multi-task bayesian optimisation with correlation. *European Journal of Operational Research*, 270(3): 1074–1085, 2018
- Michael Pearce and Jürgen Branke. Bayesian simulation optimization with input uncertainty. In *2017 Winter Simulation Conference (WSC)*, pages 2268–2278. IEEE, 2017a
- Michael Pearce, Matthias Poloczek, and Branke Jürgen. Bayesian simulation optimization with common random numbers. In *2019 Winter Simulation Conference*, page to appear, 2019
- Michael Pearce, Matthias Poloczek, Juergen Branke, Bayesian Optimization Allowing For Common Random Numbers. *In Preparation*

Research was performed in collaboration during the development of this thesis, but does not form part of the thesis:

- Matthew Groves, Michael Pearce, and Jürgen Branke. On parallelizing multi-task bayesian optimization. In *2018 Winter Simulation Conference (WSC)*, pages 1993–2002. IEEE, 2018

Abstract

Gaussian processes are simple efficient regression models that allows a user to encode abstract prior beliefs such as smoothness or periodicity and provide predictions with uncertainty estimates. Multi-Task Gaussian processes extend these methods to model functions with multiple outputs or functions over joint continuous and categorical domains. Using a Gaussian process as a surrogate model of an expensive function to guide the search to find the peak is the field of Bayesian optimisation. Within this field, Knowledge Gradient is an effective family of methods based on a simple Value of Information derivation yet there are many problems to which it hasnt been applied. We consider a variety of problems and derive new algorithms using the same Value of Information framework yielding significant improvements over many previous methods. We first propose the Regional Expected Value of Improvement (REVI) method for learning the best of a set of candidate solutions for each point in a domain where the best solution varies across the domain. For example, the best from a set of treatments varies across the domain of patients. We next generalize this method to optimising a range of continuous global optimization problems, multi-task conditional global optimization, querying one objective/task can inform the optimisation of other tasks. We then follow with an natural extension of KG to the optimization of functions that are an average over tasks that the user aims to maximise. Finally, we cast simulation optimization with common random numbers as optimization of an infinite summation of tasks where each task is the objective with a single random number seed. We therefore propose the Knowledge Gradient for Common Random Numbers that sequentially determines a seed and a solution to optimise the unobservable infinite average over seeds.

Sponsorships and Grants

This work was performed whilst at the University of Warwick Centre for Complexity Science funded by the Engineering and Physical Sciences Research Council UK.

Chapter 1

Introduction

Black–box optimisation is the general name given to algorithms that do not incorporate any problem specific knowledge in their optimisation procedure and therefore such algorithms are mostly application agnostic. We focus on a range of algorithms for expensive black box optimisation where each call to the objective function (black–box) may take minutes or hours. In such a setting, an optimisation algorithm is afforded more time to decide at which points in the domain of the function to evaluate based on the history of already observed points. This extra time in decision making means that surrogate models may be used and we specifically use Gaussian processes. The statistical predictions provided by the model are combined with a value of information recipe for deriving principled efficient search methods.

In Chapter 4, we consider a problem inspired by algorithm selection and the no-free-lunch theorems. Given a large collection of problem instances and a small set of algorithms that can be applied to each instance, no single algorithm is better than the rest therefore a user aims to find the best algorithm for each instance. Similarly, given a population of patients and multiple treatments, a clinician aims to learn the optimal treatment for each patient. We approach the problem using Gaussian processes to model the relationship between problem instance features and algorithm performance. As a consequence, for a single instance, the search for the best algorithm may be treated as a Bayesian ranking and selection problem (R&S). The other added benefits of using a Gaussian process are first, information can be transferred across instance space, predicting algorithm performance on a new

instance. Second, and more importantly, the influence of a new observation can be aggregated across instances such that data is collected by running one algorithm on one instance to maximise information gain across all instances.

In Chapter 5, we extend the work of the previous chapter to the case of a continuous domain of problem instances, for example optimising a simulator for each environment over a range of simulator settings. First, we consider freezing certain randomly generated quantities to enable caching of computations. Second, further exploiting the covariance structure of the fitted Gaussian process model, the computational overhead of the sampling algorithm can be greatly reduced without affecting the convergence or sample efficiency thereby enabling the use of much larger Monte-Carlo sample sizes. We use the same freezing and caching of computations in the next chapter.

In Chapter 6, we tackle the most general version of the previous two chapters, both the space of problems instance and the space of solutions are continuous. For a given instance, we must solve a continuous global optimisation. As the previous methods generalised ranking and selection to multiple correlated R&S problems, this generalises global Bayesian optimisation to multiple correlated global optimisation problems. Again, by accounting for similarity across problems in both the model fitting and in the data acquisition, we propose a method that significantly improves upon two other algorithms from the literature.

In Chapter 7, we move away from the previous works and focus on the optimisation of a weighted sum of objective functions. Such problems arise in simulation optimisation with input distribution uncertainty to a simulator averaged over a range of possible input distributions, each function is an instance of the simulator with different input distribution and the user aims to optimise the average of simulators. We propose augmentations to both the popular EGO and Knowledge Gradient algorithms. On synthetic experiments, we demonstrate significant advantages over optimising simply the function with largest weight.

Finally in Chapter 8, we consider simulation optimisation with common random numbers. A common problem setting where a user must optimise the expectation of a stochastic objective, however the objective takes a random number

stream (or just its seed) as input. By using the same common random number stream for multiple calls to the objective, the stochasticity of differences in outputs for two inputs is reduced and can be exploited for optimisation. We cast the problem as simulation optimisation with "input uncertainty" where the seed is an unknown input and the target of optimisation is the infinite average over seeds. This perspective a natural adaptation of the Knowledge Gradient method. We highlight multiple nuances of the problem and show that modelling choices made by previous works lead to significant inefficiencies which we duly handle resulting in a significantly improved algorithm.

We next summarise previous works in a variety of fields upon which we build and provide a technical background in Chapter 3.

Chapter 2

Background and Related Work

Imagine a user is given an objective function with multiple inputs and a single output, optimization is the task of finding the input that maximizes the output. If the function is a differentiable mathematical expression of the inputs, the optimum may be found by either solving for roots of the derivative or by gradient ascent. However, in many applications, the function from inputs to outputs has no mathematical expression or derivatives, and there may be uncontrollable randomness in the function, i.e. the input does not fully determine the output. Simulators are such functions and are frequently used as a cheap fast surrogate of real world experiments. For example, evaluating the traffic throughput of a junction with given traffic light timings under many randomly generated traffic conditions. The throughput can be approximated with a computer simulation much easier than with real world experiments that may take weeks. Although, the relationship between timings and traffic throughput is a non-trivial and stochastic function. With the increasing power of computation in the last few decades, expensive simulation that used to be for prediction is increasingly used for optimization [Fu, 2002]. However, the optimization of such a function requires an algorithm capable of efficiently searching over the domain, e.g. efficently searching traffic light timings to maximise throughput.

There are many algorithms for such problems; genetic algorithms (GA) [Davis, 1991, Whitley, 1994] maintain a population of inputs and sequentially evaluate the output, or "fitness" of all population members, then evolve the population with crossover and mutation of the "fittest" members to produce a new population of

children. Simulated annealing (SA) [Kirkpatrick et al., 1983] is based on statistical mechanics and treats the input as a state and the objective function as an energy landscape. The state is randomly perturbed at each time step with a bias toward decreasing energy whose strength is controlled by a "temperature" parameter that varies over time trading off exploration and exploitation.

All optimization algorithms make assumptions about the objective function (with the exception of uniform random search). For example, genetic algorithms incorporate a crossover operator, that selects two high fitness input vectors (parents) and creates a new input (child) mixing the dimensions of the two parents. Effectively, this operator is useful if the objective function has "additive structure", the output is a function of some of the input dimensions added to another function of the remaining dimensions. In such a case, the objective may be optimized by holding one set of input dimensions fixed and optimizing over the remaining and vice versa, much like the crossover operator. Likewise, random perturbations such as in mutation and simulated annealing loosely assume that good inputs are near other good inputs, local similarity. These methods are robust to violations of such assumptions and still find good inputs for functions that are highly discontinuous, or with varying output magnitudes or roughness across the domain and other capricious features.

However for functions that are known a priori to be smooth and continuous, GAs and SA methods may be wasteful, e.g. not using a numerical gradient approximation for gradient ascent. Instead, building an algorithm that aggressively exploits more features of the function than weakly exploiting additive structure and local similarity of the of outputs can be far more efficient.

The input-output evaluation history from an optimization algorithm may be viewed as a dataset. In order to predict output for a new input, data must be combined with user assumptions about the interaction between input and output using a function approximator. Examples include linear regression (assumes the output is a weighted average of basis functions), k-nearest neighbors (local similarity), neural networks assume (highly non-linear and complex), Gaussian processes (smoothness). In order to construct an optimization algorithm that can more aggressively exploit continuity and smoothness or linearity between input and output, we may start by

incorporating a function approximator that includes such prior beliefs. These kind of algorithms come from the class of "model-based" methods, and data collection may be viewed as iteratively constructing an "optimal training set" for the prediction model. The model needs to be accurate at peaks and less accurate at troughs. In contrast, GAs and SA are "model-free".

Examples of model-based optimisation algorithms include the Efficient Global Optimization (EGO) [Jones et al., 1998b] that combines a Gaussian process model with the expected improvement a new output will have over the best point so far to determine a new input to evaluate. The Stochastic trust-region response-surface method (STRONG) [Chang et al., 2013], combines a quadratic approximation to the objective function with a trust region to guide the search for the optimal input where the model is accurate. Sequential model-based optimization for general algorithm configuration (SMAC) Hutter et al. [2011], combines a random forest regression model with the same expected improvement of EGO.

When an algorithm with built–in assumptions is applied to an objective function that has the assumed properties, model-based methods are generally much more evaluation efficient than model-free, meaning they find good inputs in much fewer function calls. However they can be less robust to violation of assumptions, applying a model based method that assumes smoothness to a discontinuous objective function may cause unpredictable behaviour and may fail to find good inputs. However, simulated annealing or a genetic algorithm will carry on happily.

Fundamentally, the idea that no one algorithm can beat all other algorithms on all *uniformly distributed* problems is formalised in the no-free-lunch theorems [Wolpert, 2013, Wolpert et al., 1997]. The uniform distribution over functions is equivalent to making no assumptions regarding the relation between visited and unseen search points, intuitively no algorithm can optimise a realisation of white-noise better than another algorithm. Although practical relevance of such theorems is debated [Igel, 2014, Igel and Toussaint, 2003], the theorems remind us that one can always adversarially search for problems where even random search is the best algorithm [Oltean, 2004].

Therefore, to soften the assumptions of a model, one may use Bayesian

statistics. Philosophically, this branch of mathematics emphasises making prior beliefs explicit and accounting for uncertainty in such assumptions. Model based optimisation algorithms make a user's prior beliefs about an unknown objective function explicit in order to be exploited by an optimization algorithm. Therefore these fields are complementary and their intersection is Bayesian optimisation. This is a powerful class of data efficient methods that has been receiving increasing attention in recent years, sparked by the now famous EGO algorithm in the engineering literature and more recently by Snoek et al. [2012] in the machine learning literature.

There exist many Bayesian function approximators, Bayesian linear regression [Bishop, 2006], Bayesian neural networks [Hernández-Lobato and Adams, 2015], or Gaussian processes [Rasmussen and Williams, 2004]. Such function approximators provide uncertainty estimates with their predictions and therefore are models that "know what they don't know", returning high uncertainty for a prediction that is believed to be inaccurate. These uncertainty estimates are ideal for data efficient optimisation, there is less value in function evaluations for inputs with low uncertainty. Likewise, where predictions are high and uncertainty is moderate, there may be much value in further evaluation for such an input to learn about the peak. Contrast this with a standard GA or SA algorithm that both ignore their evaluation history, the price to pay for robustness.

We next summarise literature in a range of optimisation domains.

2.1 Ranking and Selection

A user is faced with a problem (objective function) and there exist a small set of possible solutions (inputs), however the best solution is not known and checking the quality of a solution is stochastic, returning a noisy observation. This is an optimization problem over a categorical search space and we assume there is no information about the solution therefore only model-free methods apply. Myopic sampling policies sequentially sample from alternatives in such a way that at each step, the sample has the highest expectation of improving the objective, which typically is either probability of correct selection (learn the identity of the best alternative), or

expected opportunity cost (learn about alternatives so as to maximize performance). The Knowledge Gradient (KG) method investigated in Frazier et al. [2008] and Chick et al. [2010] but going back to Gupta and Miescke [1996] is a myopic policy that maintains an *independent* Bayesian posterior of the output of each solution. The method allocates budget to myopically maximize the improvement in the peak of the posterior mean, it is guaranteed to sample all solutions infinitely often so the true best will become known when outputs are real valued. Frazier et al. [2009a] extend the method to the case of correlated solutions, i.e., when sampling from one solution may teach us something about the performance of similar solutions, however this is only applicable when extra features of each alternative are available and may be used in a model. The one-step look ahead nature of Knowledge Gradient means it can fail to find good solutions when outputs are binary as only a single observations cannot provide enough information to yield a one-step benefit. The optimal computing budget allocation class of methods (OCBA) [Chen, 1996] sequentially allocate samples to alternatives that maximize a lower bound on the probability of correct selection under a posterior distribution of average output for each alternative. OCBA is applicable even when outputs are binary and is also asymptotically consistent. Branke et al. [2007] provide a review and comparison of several ranking and selection methods, a more recent survey focusing on algorithms derived under the Bayesian framework can be found in Chen et al. [2015]. Both KG and OCBA Ranking and selection methods have been extended to multi-objective [Frazier and Kazachkov, 2011, Lee et al., 2004] and other optimisation scenarios which we cover in more detail next under the Bayesian optimisation class of methods. Görder and Kolonko [2019] provide a comprehensive review and summary of statistical methods. In contrast with the above methods that utilise a statistical distribution over outputs, Racing algorithms compare all solutions and sequentially eliminate solutions that become significantly worse, or "fall-behind" [Birattari et al., 2002, Branke and Elomari, 2013], which are easily implemented however good solutions may be eliminated due to bad luck and hence the true optimal solution may not become known.

2.2 Gaussian Process Model Based Optimization

Contrasting with independent ranking and selection is the case when the search space of the optimisation includes features that can be used with a model. For this, Gaussian process regression (GP) gives a statistical distribution over possible objective function values based on the data collected and a prior belief about similarity of outputs across the domain. When function evaluations are costly, datasets of collected points may be small and the ability to incorporate prior knowledge with uncertainty is particularly beneficial compensating for the lack of data. For this reason GPs have gained much attention in Bayesian optimization and simulation optimisation. Along with a Bayesian surrogate model, the second component of a Bayesian optimisation algorithm is an *acquisition function*, a function over the space of inputs that quantifies the expected benefit of hypothetically augmenting the dataset with the current input and corresponding (unknown) output. Given a hypothetical new input, the distribution over the hypothetical new output is informed by the Bayesian model. The acquisition function is optimised over inputs and the best input is evaluated to yield output and the pair is added to the dataset. The model is updated and the acquisition search repeats. The exploration-exploitation trade-off is all incorporated into the acquisition function. We next discuss a variety of algorithms that mainly differ by their acquisition function.

For deterministic functions, the popular Efficient Global Optimization (EGO) algorithm [Jones et al., 1998a] uses a Gaussian process model combined with an acquisition function that returns the Expected Improvement (EI) of a new output over the current best output. The Sequential Kriging Optimization (SKO) algorithm of [Huang et al., 2006a] extends the EGO algorithm to the case when function evaluations are noisy. GPs are easily adapted to model noisy outputs and the acquisition function is a heuristically designed extension of EI that incorporates uncertainty yet lacks a principled derivation. On the other hand, Scott et al. [2011a] extend the Knowledge Gradient for finite correlated alternatives to the continuous input case, a GP over a discretisation of a continuous space is a multivariate Gaussian and therefore equivalent to KG for correlated alternatives. Both SKO

and KG methods naturally recover the EI function when applied to deterministic functions however KG is typically more efficient and we show in Chapter 3.3 that both KG and EI can be derived from the same principle unlike SKO. The main advantage of KG is that when evaluating the quality of a new input, the method takes into account how the model at other inputs will change, the correlation across the input space and predictions at other points are required in computing the acquisition function, unlike SKO and EI which only use the mean and variance at a single point to compute the acquisition function. The disadvantage is the added computational cost. We refer to these more sophisticated acquisition functions as "correlation aware". Other popular Bayesian optimisation variants that take into account covariance across inputs include Stepwise Uncertainty Reduction (SUR) [Chevalier et al., 2014, Picheny, 2014] that computes the expected volume of the surrogate model that will improve upon the current best point, an excursion set. And the Integrated Expected Conditional Improvement (ICEI) [Gramacy and Lee, 2011] that is very similar to step wise uncertainty reduction and was made to handle non-linear constraints on the inputs. Entropy Search (ES) [Hernandez-Lobato et al., 2014, Villemonteix et al., 2009a] uses a GP over outputs to build a distribution over the input domain that models the probability of any input being the maximiser. The method aims to minimise entropy of this induced distribution over the input domain, and can require much more computation than other methods. An extensive overview and empirical comparison of various acquisition functions (referred to as infill criteria) on a range of noisy problems has been provided by Jalali et al. [2017], Picheny et al. [2013b]. In other fields such as bandits or control, a user typically aims to maximise cumulative sum of outputs (instead of just the best final output). For this, Upper Confidence Bound (UCB) algorithms are more popular [Srinivas et al., 2010] that use a varying confidence bound to explicitly trade off exploitation, where prediction is high, and exploration, where uncertainty is high. These methods are more amenable to deriving bounds on the cumulative reward yet they are a function of only the mean and variance at a given input and hence are not correlation aware. Cumulative bounds have been derived for Knowledge Gradient methods however they are often not informative.

2.3 Multi–Task Gaussian process Based Optimization

Multi–task Gaussian processes [Bonilla et al., 2008], also known as co–Kriging, are Gaussian process models that are trained to predict vector valued functions, or, equivalently, scalar valued functions that contain a categorical variable as an input dimension, further described in Chapter 3.2.5. Hence they are useful for multi-objective optimization, multi information source optimization, task conditional optimisation, and average task optimisation. We next summarise work on each of these different optimisation scenarios.

2.3.1 Multi-Objective Optimization

Given an objective function over a continuous domain with multiple outputs, each output is an objective, or a criterion, and a user must find the set of inputs that optimise each output as well as the compromise between competing objectives. Each call to the objective function returns a vector of objective values. Pareto optimality is a property of a current point in the output (vector) space. If there are any other evaluated points in the output space for which *all* objectives are superior than the current point, then it is dominated, if for all other points, the current point has at least one objective that is better, then it is non-dominated. The set of points that are non-dominated are denoted the Pareto set and such points are Pareto optimal and form the Pareto front. The Pareto-Efficient Global Optimisation (ParEGO) algorithm of Knowles [2006] is a sequential method that at each time step uses a random projection to reduce each output vector to a scalar value (a scalarisation), this preprocessing step converts the problem into a standard single objective problem such that an iteration of standard EGO may be applied. The randomness of the projection ensures that all objectives and all possible "compromises" between objectives are considered and an approximation of the true Pareto set is discovered. It's simplicity and ease of implementation has led to it being widely used as a baseline for other multi–objective algorithms. There exists a wide range of pre-processing scalarisation functions and Chugh [2019] provides an overview. These methods do not model similarity of outputs across objectives (not multi-task) and the acquisition is only

informed by mean and variance of output at the new input and past observations (not correlation aware), hence there is much room for improvement.

When there are multiple outputs, fitting a multi-task model can be more data efficient, however a correlation aware acquisition function must account for multiple outputs across the same domain adding complexity. The Multi-Objective Knowledge Gradient with unknown utility [Astudillo and Frazier, 2017] fits a multi-task GP to model all objectives. It then uses a random linear weighted average, a linear scalarisation, of the GP modelled outputs to form a new single output GP for which standard KG is used to quantify the benefit of a new observation. Further, a Monte-Carlo average over scalarisations is used, and the KG for each one is computed and the average is the acquisition function. Linear combinations of GPs form another GP allowing the use of single objective acquisition functions. However, optimising linear scalarisations of objectives can only ever learn the convex hull of the Pareto front, not the entire front including concave regions. Secondly, the acquisition function is single task method (i.e. integrating over a univariate output) applied to an "average task", it is an approximation to a true value of information procedure (integrating over a multivariate output). However in our (extensive, unpublished) experiments, making a true multi-task acquisition function did not lead to any benefit. Picheny [2015] similarly adapts the single objective SUR to multiple-objectives, computing the volume of improvement over the Pareto front, a multi objective excursion set. Contrasting with all the previous methods, the Predictive Entropy Search for Multiple Objectives, PESMO, [Hernández-Lobato et al., 2016], follows entropy search and instead considers a distributions over the input space induced by the GP over outputs. The method builds a binary classifier mapping each input point to either dominated or a non-dominated in the output space. The method collects data to maximise integrated entropy of the binary classifier over the whole input domain. There have been many alternative extensions of BO methods to multiple objectives that also consider a simpler binary classifier based method [Zuluaga et al., 2013], a more recent entropy search based method, [Suzuki et al., 2019]. As with global optimisation, accounting for changes in the model at other inputs can provide performance improvements however this comes

with added compuation.

2.3.2 Multi–Information–Source Optimization

Multi information source, or multi-fidelity, optimization is where a user aims to optimise an expensive objective function, e.g. a big accurate simulator, but also has access to a cheaper but less accurate surrogate of the objective function, e.g. a smaller toy simulator. These methods use a multi–task Gaussian process model to share information across surrogate models and the acquisition search is over both inputs and information source. The acquisition function is also cost–adjusted, samples are allocated to the input with the highest acquisition benefit per unit cost. Forrester et al. [2007] provide an algorithm that combines co-kriging with EI to optimise wing design using two levels of simulation accuracy and Huang et al. [2006b] extend the SKO algorithm to the multi-fidelity setting. Again both methods do no use a correlation aware acquisition function and are heuristic modifications of EI. The Multi-Task Bayesian Optimization (MTBO) algorithm of Swersky et al. [2013] applies a multi–task Gaussian process to optimise hyperparameters of machine learning models. By cheaply training the models on small subsets of training data, one can learn about the models when trained using the expensive full training dataset. The method uses both entropy search and EI and ES for acquisition. This approach of using subsets of data was extended by Klein et al. [2016] to be able to pick an arbitrary dataset size. The acquisition again used a modification of EI.

The Multi-Information Source algorithm [Poloczek et al., 2016a] builds upon the MTBO algorithm with a more sophisticated Gaussian process model and a cost-adjusted Knowledge Gradient is used as the acquisition function. Picheny et al. [2013a] consider a similar case where the precision of an evaluation can be chosen, at the expense of higher computational cost for higher precision.

2.3.3 Task Conditional Optimization

A few papers extend the idea of Bayesian Optimisation to the case where several related optimisation problems have to be solved sequentially. Morales-Enciso and Branke [2015] consider the optimisation of a changing objective function where

the collected data and the GP model is over the joint space of inputs and time. The acquisition function uses expected improvement of a new point over other points already evaluated for the same objective function. Treating time as an extra input dimension is also used by Poloczek et al. [2016b] to "warm-start" Bayesian Optimisation, however with the Knowledge Gradient as the acquisition function for the current optimisation task. The field of Contextual multi-armed bandits is an extension of the multi-armed bandit problem where the best arm depends on a context that is randomly changing with time. In such a setting, sampling policies must aim to maximise cumulative reward, see Zhou [2015] for a survey. Krause and Ong [2011] use Gaussian processes over a domain of both inputs and context variables and propose an UCB acquisition function to find the best arm for each context. Related to contextual multi-armed bandits is the field of contextual policy search, which tries to identify the best parameters of a lower level policy depending on the context, see Deisenroth et al. [2013] for a survey. Bardenet et al. [2013], Ginsbourger et al. [2014] consider the same problem we do in Chapter 6, the simultaneous *conditional* optimisation of multiple functions with similarity across functions and a user aims to find the peak of each function. Which function to evaluate (or context features/task setting) as well as the input is free to be chosen by the algorithm. These methods fit a Gaussian process over the joint space of task settings and inputs, and both methods use an adaptation of the EI acquisition function and therefore acquisition does not consider how new data will affect other tasks and other peaks. Sambakhé et al. [2019] consider the same problem however with noisy function evaluations and adapt the SKO acquisition function. Again, acquisition is not correlation aware nor mathematically principled. In our work, we use a value of information procedure which equates to adapting KG and further accounting for how evaluating one task can inform optima of other tasks, yielding significant improvements.

As a special case of the above problem, if the task/context is continuous, however the input to be optimised for each context is categorical, this may be viewed as a task conditional generalisation of Ranking and Selection. The problem of learning which drug from a small set of treatments is most effective for any given

patient was considered by Xu et al. [2014]. Patients are characterized by continuous biomarkers and there are three treatments, the approach uses a hierarchical Bayesian model over partitions of the patient biomarker space. Hu and Ludkovski [2015] consider the same problem, however conveyed as ranking response surfaces— for each point across a domain, find the best surface. The method uses independent Gaussian process models and proposes an acquisition function based on an approximation to the maximum of all surfaces and how this maximum is affected by a new sample, although this method is not correlation aware. In Chapters 4 and 5, we again adapt KG and integrate the effect of acquisition on each task over the task domain. Shen et al. [2017] consider the same approach of integrating over contexts with a PCS ranking and selection algorithm and more recently, Zhang et al. [2019] used KG building upon our work by proving consistency in continuous domains; the true best alternative for each task will become known eventually.

2.3.4 Optimization of Sums of Functions and Common Random Numbers

Imagine a user is given multiple functions over a common domain and is required to find the input that maximises the average or sum of the functions but can only evaluate one at a time. This arises in simulation with distribution uncertainty [Pearce and Branke, 2017a] and tuning parameters of machine learning algorithms by k-fold cross validation [Swersky et al., 2013] and, as we show in Chapter 8, simulation optimisation with common random numbers. A simulator requires calibration from real world data in order to be a realistic surrogate of reality. When such calibration parameters are not known with accuracy, the best action a user can take is to optimize the average over a range of simulators each instantiated with a different parameter setting. In our work [Pearce and Branke, 2017a], we apply the KG acquisition function to the weighted sum of functions, effectively reproducing the multi-objective algorithm with linear scalarisations of Astudillo and Frazier [2017] discussed above. More recently, Toscano-Palmerin and Frazier [2018] independently proposed a similar method to our work however using a more recently proposed evaluation method for the KG acquisition function and perfroming a more extensive analysis.

By evaluating two solutions with the same random number seed, one can learn more about the *relative* ranking of solutions, the difference between their outputs may have reduced stochasticity. For example, the seed may influence the difficulty of a randomly generated scenario, and the performance of all solutions degrades for difficult scenarios and improves for easy scenarios. When modelling, this is accounted for by incorporating correlation in the stochastic noise of outputs from the same seed, in contrast to the common modelling assumption of independent noise for all outputs. Combining CRN with ranking and selection has been extensively studied with two-stage methods [Chick and Inoue, 2001, Nelson and Matejcik, 1995] that initially sample all solutions multiple times to learn noise covariance between solutions and a second stage exploits the learnt structure. Fu et al. [2004] further investigate the second stage of the two stage process. More recently, a sequential method has been proposed by Görder and Kolonko [2019] that keeps track of all sampled candidates and uses the same series of seeds for all candidates enabling reuse of seeds. The effects of CRN (or non-diagonal noise matrices) upon Gaussian processes was considered by Chen et al. [2012] that showed interesting results we discuss in Chapter 8. Xie et al. [2016] augmented correlated KG with a second acquisition function that allowed for measuring the difference of solutions using CRN and demonstrated performance improvements over non-CRN KG. However, despite being a KG based method, the algorithm was not based on Value of Information and assumed that a seed could not be reused from the history. As a result a workaround is proposed that we show leads to many sub-optimal effects we discuss in Chapter 8 and propose solutions.

Chapter 3

Technical Background

3.1 Gaussian Process Bayesian Optimisation

We assume we are given an objective function $F(x)$ defined over a domain $x \in X$ and X is typically a box constrained search space, $X \subset \mathbb{R}^d$ for $d < 10$. We further assume $F(x)$ is stochastic, repeated calls with the same x do not return the same value. We assume $F(x)$ is not differentiable and each call is expensive taking from minutes to hours for a single evaluation. The goal is to find the input that is best in expectation

$$\arg\max_{x \in X} \mathbb{E}[F(x)]$$

and we have an evaluation budget of N calls to the objective function. The standard Bayesian optimization algorithm is composed of two phases. The first phase initializes a dataset by randomly choosing $n_{init} << N$ input values $X^{n_{init}} = \{x^1, ..., x^{n_{init}}\}$ and evaluating the objective $Y^{n_{init}} = F(X^{n_{init}}) \in \mathbb{R}^{n_{init}}$. The input-output pairs form a dataset $D^{n_{init}}$ which is used to build a Gaussian process surrogate model, a function approximator, of $F(x)$. The second phase is to use the Gaussian process to inform an acquisition function that quantifies the benefit of a new evaluation $F(x^{n+1})$ at x^{n+1}. Optimizing the acquisition function determines the next input x^{n+1}, then the objective is evaluated $y^{n+1} = F(x^{n+1})$. Using the new dataset D^{n+1} the method repeats until the whole budget is consumed. We next introduce Gaussian process regression and then a recipe for deriving acquisition functions.

3.2 Gaussian Process Regression

Given a stochastic black-box function $F : X \to \mathbb{R}$ where $X \subset \mathbb{R}^d$, we may model F as a realisation of a stochastic process. A stochastic process is a set of indexed random variables, indices may be viewed as inputs and realizations of the random variables as outputs and effectively stochastic processes are random functions.

3.2.1 Generative Model

A Gaussian process is a random function generator whose sampled outputs are Gaussian distributed: given an input x, or a set of inputs ("indices") $X^n = \{x^1, ..., x^n\}$, the distribution of the n corresponding y values is an n-dimensional multivariate Gaussian,

$$Y^n \sim \mathcal{N}(\underline{\mu}, \Sigma) \tag{3.1}$$

where $\mu \in \mathbb{R}^n$ and $\Sigma \in \mathbb{R}^{n \times n}$ are the mean and covariance matrix. The probability density of any Y^n vector given parameters is

$$\mathbb{P}[Y^n | \underline{\mu}, \Sigma] = \frac{1}{(2\pi)^{n/2}|\Sigma|} \exp\left(-\frac{1}{2}(Y^n - \underline{\mu})^\intercal \Sigma^{-1}(Y^n - \underline{\mu})\right). \tag{3.2}$$

The mean vector may be given by evaluating a mean function $\underline{\mu} = \mu(X^n)$, and likewise the covariance matrix elements are given by evaluating a covariance function for all pairs of inputs $\Sigma_{ij} = k(x^i, x^j)$ for all $1 \leq i, j \leq n$. Sampling a single $Y^n \in \mathbb{R}^n$ vector and plotting Y^n versus X^n reveals a single random function evaluated at X^n.

Any vector may be viewed as a function from it's element indices to real values $\underline{\mu} : \{1, 2, ..., n\} \to \mathbb{R}$, and any matrix may be viewed as a function from pairs of element indices to real values $\Sigma : \{1, 2, ..., n\} \times \{1, 2, ..., n\} \to \mathbb{R}$. In this sense a function over a continuous domain may be viewed as a vector with a continuous "index", or an infinite dimensional vector. Similarly a function of two arguments is a infinite dimensional matrix. Therefore a Gaussian process may be viewed as an infinite dimensional multivariate normal distribution and Gaussian processes are often called "distributions over functions". Mathematically, they are a generalization of the Gaussian Measure to infinite dimensional spaces (see Eldredge [2016] for a

tutorial), however this distribution differs in that it cannot be written down as a density function. The sample space is infinite dimensional and computing volume under a probability density function over such a domain would be an integral over infinitely many variables! Instead, note that a Gaussian process does provide a traditional distribution over function values *on a finite discretization*, X^n, of larger space. The mean and covariance functions can be evaluated for arbitrary cardinality of $|X^n|$.

The covariance function, $k(x^i, x^j)$, dictates the covariance between sampled y^i and y^j. For example if $x^i \approx x^j$ then we would like $y^i \approx y^j$. This is captured by the popular squared exponential covariance function

$$k(x, x') = \sigma^2 \exp\left(\frac{(x - x')^2}{2l^2}\right) \tag{3.3}$$

where σ^2 and l are hyperparameters dictating the magnitude of vertical and horizontal fluctuations in the generated functions and for now we assume they are known and fixed.

3.2.2 Conditioning a Gaussian Distribution

We first temporarily ignore any X^n values and focus on vectors $Y^n \sim N(\underline{\mu}, \Sigma)$. Given a multivariate Gaussian distribution with mean $\underline{\mu} \in \mathbb{R}^n$ and covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$, let $Y^n \in \mathbb{R}^n$ be a single sample vector from the given distribution. Suppose we know $\underline{\mu}$, Σ and only the first $n_a < n$ dimensions of a sample vector Y^n denoted $Y^a \in \mathbb{R}^{n_a}$ and the remaining $n_b = n - n_a$ dimensions as $Y^b \in \mathbb{R}^{n_b}$. We can ask the question "what must the $Y^b \in \mathbb{R}^{n_b}$ look like?" For example given a population distribution of people's heights and shoe sizes, and only a person's shoe size, can we statistically predict their height? The distribution of the unobserved dimensions Y^b must be consistent with both Y^a as well as the generated distribution

of all dimensions, the joint distribution is as follows

$$\begin{aligned}\mathbb{P}[(Y^a, Y^b)] = \mathbb{P}[Y^n] &= N(Y^n|\underline{\mu}, \Sigma) \\ &= N\left(Y^n \middle| \begin{pmatrix}\underline{\mu}^a \\ \underline{\mu}^b\end{pmatrix}, \begin{pmatrix}\Sigma^{aa} & \Sigma^{ab} \\ \Sigma^{ba} & \Sigma^{bb}\end{pmatrix}\right)\end{aligned}$$

where the mean vector and covariance matrix may both be partitioned into elements corresponding to observed and unobserved parts. The probability density function is a mapping $\mathbb{P}[Y^n|\underline{\mu}, \Sigma] : \mathbb{R}^n \to \mathbb{R}^+$. The *unnormalised* distribution of Y^b *conditioned on* Y^a may be easily computed by taking the above density and setting Y^a arguments as fixed to the known values

$$\mathbb{P}[Y^b|Y^a, \underline{\mu}, \Sigma] : \mathbb{R}^{n_b} \to \mathbb{R}$$
$$\mathbb{P}[Y^b|Y^a, \underline{\mu}, \Sigma] \propto \mathbb{P}[(Y^a, Y^b)|\underline{\mu}, \Sigma]$$

where the constant of proportionality is from Bayes Rule $\mathbb{P}[Y^a|\underline{\mu}^a, \Sigma^{aa}]$. The normalised density function may be tediously computed using the matrix inversion lemma and is given by

$$\mathbb{P}[Y^b|Y^a, \underline{\mu}, \Sigma] \sim N(\underline{\mu}^{b|a}, \Sigma^{b|a}) \tag{3.4}$$
$$\underline{\mu}^{b|a} = \underline{\mu}^b + \Sigma^{ba}(\Sigma^{aa})^{-1}(\underline{\mu}^a - Y^a) \tag{3.5}$$
$$\Sigma^{b|a} = \Sigma^{bb} - \Sigma^{ba}(\Sigma^{aa})^{-1}\Sigma^{ab}, \tag{3.6}$$

another multivariate Gaussian. By observing only a subset of dimensions of a vector Y^n, we may "freeze" the known dimensions to their values Y^a and see how the rest are affected. Alternatively, we may sample many $Y_1^n, ..., Y_k^n$ vectors and reject all the sampled vectors for which the first n_a dimensions are significantly different from Y^a. The left over samples will all have similar values in the Y^a dimensions and the Y^b dimensions will be approximately distributed according to $N(\mu^{b|a}, \Sigma^{b|a})$. Thankfully this process can be done in closed form for multivariate Gaussian distributions.

Next, assume the multivariate Gaussian distribution mean and covariance come from a Gaussian process mean and covariance function evaluated on a discret-

isation of the input domain. The Y^a are associated with X^a, the Y^b with X^b and $\underline{\mu}^a = \mu(X^a)$, $\Sigma^{aa} = k(X^a, X^a)$ and $\Sigma^{ab} = k(X^a, X^b)$ and the same for $\underline{\mu}^b, \Sigma^{bb}$. Since X^a and Y^a are observed data, they are fixed. However, X^b is not fixed, we can vary the discretization elements X^b and any discretisation still yields a multivariate Gaussian for outputs Y^b. By using the mean and covariance functions and the above formula for conditioning, the posterior distribution over outputs Y^b at new inputs X^b is given by

$$\begin{aligned}
\mathbb{P}[Y^b|Y^a, X^a, X^b] &\sim N(\mu^a(X^b), k^a(X^b, X^b)) \\
\mu^a(x) &= \mu(x) + k(x, X^a)(k(X^a, X^a))^{-1}(\mu(X^a) - Y^a) \\
k^a(x, x') &= k(x, x') - k(x, X^a)(k(X^a, X^a))^{-1}k(X^a, x').
\end{aligned}$$

Note that when the posterior mean function is evaluated at a past point, $X^b = \{x^i\} \subset X^a$, the prediction is exactly the corresponding observed value y^i,

$$\begin{aligned}
\mu^a(x^i) &= \mu(x^i) + k(x^i, X^a)(k(X^a, X^a))^{-1}(Y^a - \mu(X^a)) \\
&= \mu(x^i) + \underbrace{\big(k(X^a, X^a)\big)_i}_{i^{th} \text{ row}} \big(k(X^a, X^a)\big)^{-1}(Y^a - \mu(X^a)) \\
&= \mu(x^i) + e_i^\intercal(Y^a - \mu(X^a)) \\
&= \mu(x^i) + y^i - \mu(x^i) \\
&= y^i
\end{aligned}$$

where e_i is the i^{th} column of an $n_a \times n_a$ identity matrix.

3.2.3 Stochastic Function Outputs

However we are interested in stochastic objective functions $F(x) = \theta(x) + \epsilon$ where $\epsilon \sim N(0, \sigma_\epsilon^2)$. We can dictate what the vectors Y^a and Y^b represent. All we need to do is state how the concatenated vector (Y^a, Y^b) is randomly generated then we may use the conditioning described above in Equation 3.4 to get a posterior over dimensions Y^b. We may set Y^a given X^a to be noisy outputs $F(X^a) = \theta(x) + \epsilon$ and we can set Y^b to represent the (unobservable) noise-free outputs at X^b, that

is $\theta(X^b)$ and therefore $(Y^a, Y^b) = (\theta(X^a) + \epsilon, \theta(X^b))$. So, to encode this we simply assume that (Y^a, Y^b) is jointly generated from a multivariate Gaussian whose mean and covariance come from a mean function, $\mu(x)$, and kernel, $k(x, x')$, evaluated at the given X^a and X^b, and that *only* the Y^a elements have additional white noise. The vector $(Y^a, Y^b) = (F(X^a), \theta(X^b))$ has generative Gaussian distribution

$$\mathbb{P}[(F(X^a), \theta(X^b))] = N\left(\begin{pmatrix}\mu(X^a)\\ \mu(X^b)\end{pmatrix}, \begin{pmatrix}k(X^a, X^a) + I\sigma_\epsilon^2 & k(X^a, X^b)\\ k(X^b, X^a) & k(X^b, X^b)\end{pmatrix}\right) \tag{3.7}$$

which when conditioned on the observed dimensions, $Y^a = F(X^a)$, we get

$$\begin{aligned}\mathbb{P}[\theta(X^b)|Y^a, X^b] &\sim N(\mu^a(X^b), k^a(X^b, X^b))\\ \mu^a(x) &= \mu(x) + k(x, X^a)\big(k(X^a, X^a) + I\sigma_\epsilon^2\big)^{-1}(Y^a - \mu^0(X^a))\\ k^a(x, x') &= k(x, x') - k(x, X^a)\big(k(X^a, X^a) + I\sigma_\epsilon^2\big)^{-1}k(X^a, x').\end{aligned} \tag{3.8}$$

In this case the Gaussian process mean function does not interpolate past evaluations $\mu^a(x^i) \neq y^i$ almost surely.

3.2.4 Hyperparameters

The mean function is either set to the mean of data $\mu(x) = \bar{Y}^a$ (equivalently Y^a is centered and mean 0 is used) or a constant that is learnt with the other hyperparameters. In this work we use the former approach. All of the GP parameters are then $\Theta = \{\sigma^2, l, \sigma_\epsilon^2\}$ which are learnt by maximising log marginal likelihood

$$\begin{aligned}\log \mathbb{P}[Y^a|X^a, \Theta] &= -\frac{1}{2}\Big(n_a \log(2\pi) + \log\left|k(X^a, X^a; \sigma^2, l) + I\sigma_\epsilon^2\right|\\ &\qquad + Y^{a\intercal}\left(k(X^a, X^a; \sigma^2, l) + I\sigma_\epsilon^2\right)^{-1} Y^a\Big)\end{aligned}$$

which may be maximised by multi start gradient ascent. An alternative approach is to sample hyper-parameters from the likelihood and fit multiple GP models to the data. For each model, an acquisition function is constructed the mean of acquisition functions is used [Murray and Adams, 2010]. We do not take this approach in

this work as it is more expensive to compute requiring many calls to the marginal likelihood which has $O(n_a^3)$ computational cost.

3.2.5 Multi-Task Gaussian Processes

Given a function with vector valued outputs,

$$F_{MT} : \mathbb{R}^n \to \mathbb{R}^k$$

this is often referred to as multi-task learning, particularly if each output of the vector is over a different domain, e.g. $[0, 1]$ for the first dimension and $\mathbb{R}$ for the second. For Gaussian process regression with real valued outputs and Gaussian noise, multi-task is identical to a standard scalar valued Gaussian process model with an augmented input

$$F_S : \mathbb{R}^n \times \{1, ..., k\} \to \mathbb{R}$$

where $F_{MT}(x) = (F_S(x, 1), .., F_S(x, k))$. To use Gaussian process regression as a function approximator of $F_S(x)$, we redefine the *augmented* domain as $\tilde{X} = \mathbb{R}^n \times \{1, ..., k\}$ with points $(x, s) \in \tilde{X}$ and require appropriate mean and kernel functions

$$\mu : \tilde{X} \to \mathbb{R}$$
$$k : \tilde{X} \times \tilde{X} \to \mathbb{R}.$$

In general, it is possible to perform Gaussian process regression over any domain given a mean function and a positive semi-definite kernel. The sum of any two kernels is also positive-semi-definite as is their product allowing to compose more sophisticated kernels. Below are four valid kernels over the augmented domain $\tilde{X} = [0, 50] \times \{1, 2, 3\}$. Randomly generated functions are shown in Figure 3.1.

A. Each task is a totally independent Gaussian process realisation with the unique magnitude and length scale,

$$k(x, s, x', s') = \delta_{s,s'} \sigma_s^2 \exp\left(-\tfrac{(x-x')^2}{2l_s^2}\right).$$

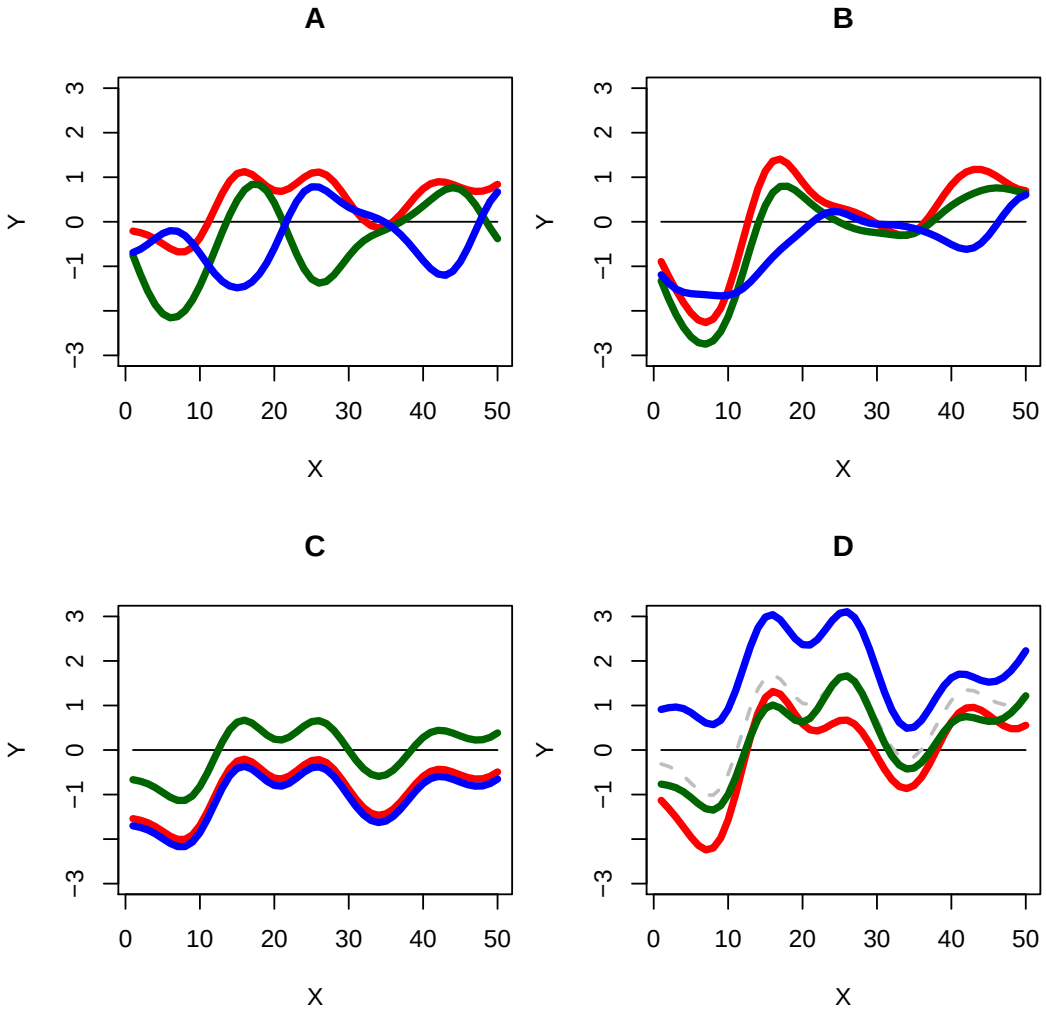


Figure 3.1: Randomly generated function. A, all functions are independent Gaussian processes. B, each function is a different weighted average of the functions from A. C, each function is a common GP realisation with a unique offset. D, each function is a common GP, with unique offset and GP realisation.

B. Each task is a weighted average of a common basis set of Gaussian process realisations, Σ is a positive semi definite matrix

$$k(x, s, x', s') = \Sigma_{s,s'} \exp\left(-\frac{(x-x')^2}{2l^2}\right).$$

C. Each task is composed of a common Gaussian process realisation and an independent task specific offset,

$$k(x, s, x', s') = \sigma_0^2 \exp\left(-\frac{(x-x')^2}{2l^2}\right) + \delta_{s,s'}.$$

D. Each task is composed of a common Gaussian process realisation and an independent task specific realisation and offset,

$$k(x, s, x', s') = \sigma_0^2 \exp\left(-\frac{(x-x')^2}{2l^2}\right) + \delta_{s,s'} \exp\left(-\frac{(x-x')^2}{2l^2} + \delta_{s,s'}\right).$$

When viewed as a scalar valued Gaussian process over an augmented domain, the conditioning Equations 3.8 may be used without modification.

3.3 Value of Information for Acquisition Functions

Given only partial data about an uncertain world, a decision maker is required to make a choice under such uncertainty. Firstly, a Bayesian model allows the decision maker to make predictions about the world accounting for such uncertainty, i.e. a posterior distribution over parameters of interest. Secondly, the decision maker has a loss function that dictates the (negated) quality of a decision for a particular state of the world, one parameter value. Thirdly by integrating the loss function for one particular parameter value over the posterior distribution of parameter values yields the expected loss. Finally, the decision maker can take the decision that minimizes the expected loss over states of the world. If more information about the world is collected, the posterior, expected loss and the optimal decision will all change. The *Value of Information* is the expected change in the minimum expected loss due to knowing more about the world. We next give two concrete examples of this derivation process.

3.3.1 Knowledge Gradient

In the case of Bayesian optimization, we aim to optimize a stochastic function

$$F : X \to \mathbb{R}$$

which we assume to be a latent function with white noise $F(x) = \theta(x) + \epsilon$. The decision to be made is to pick an input $x_r \in X$ that is predicted optimum. Let $X^N = (x^1, ..., x^N)$ and $Y^N = (y^1, ..., y^N)$ be a dataset of N input-output pairs $D^N = \{X^N, Y^N\}$, the Bayesian "model of the world" is a Gaussian process fitted to the dataset yielding the posterior distribution over any underlying latent function output

$$\mathbb{P}[\theta(x_r)|D^n].$$

In this thesis, without loss of generality, we focus on maximisation. Given one particular realization of the function $\theta(x)$, the loss function of a decision x_r is the (negative) output of the decision

$$\text{Loss}(x_r) = -\theta(x_r).$$

Therefore the expected loss is simply the posterior mean,

$$\mathbb{E}[\text{Loss}(x_r)|D^n] = \mathbb{E}[-\theta(x_r)|D^n] = -\mu^N(x_r).$$

And finally the optimal decision is the x_r value that minimises expected loss:

$$x_r = \arg\min -\mu^N(x) = \arg\max \mu^N(x)$$

and the associated minimum expected loss we define as performance, P^N, and is the peak posterior mean

$$P^N = \max \mu^N(x).$$

Next, if we have a budget of N function calls, at time n, the dataset X^n, Y^n contains n input-output pairs. The remaining $N - n$ function evaluations still need to be

allocated updating all of the above quantities. The value of information, VoI, is the update in the final performance to the decision maker,

$$\begin{aligned}\mathrm{VoI}^n(x^{n+1},...,x^N) &= \mathbb{E}[P^N - P^n | D^n, x^{n+1},...,x^N] \\ &= \mathbb{E}[\max \mu^N(x) - \max \mu^n(x) | D^n, x^{n+1},...,x^N]\end{aligned}$$

where the expectation is over the unobserved random $y^{n+1},...,y^N$ values whose multi-variate distribution is given by the Gaussian process

$$\mathbb{P}[y^{n+1},...,y^N | D^n, x^{n+1},...,x^N] =$$

$$N\left(\begin{pmatrix}\mu^n(x^{n+1})\\ \vdots \\ \mu^n(x^N)\end{pmatrix}, \begin{pmatrix}k^n(x^{n+1},x^{n+1})+\sigma_\epsilon^2 & \cdots & k^n(x^{n+1},x^N)\\ \vdots & \ddots & \vdots \\ k^n(x^N,x^{n+1}) & \cdots & k^n(x^N,x^N)+\sigma_\epsilon^2\end{pmatrix}\right)$$

In many applications, calls to $F(x)$ can only be made sequentially, then it is more efficient to allocate the x^{n+1} based on the most recent information $x^1,...,x^n$ and $y^1,...,y^n$, and allocate x^{n+2} given all data up to $n+1$ etc. Also, the value of information of multi-step look ahead can be very expensive, it requires optimisation over the whole batch $(x^{n+1},...,x^N) \in X \times ... \times X$ and each call requires integration over $y^{n+1},...,y^N$. Therefore, we may write the value of information of looking one-step ahead, this is known as the *Knowledge Gradient* acquisition function

$$\mathrm{KG}^n(x) = \mathbb{E}[\max_{x'} \mu^{n+1}(x') - \max_{x''} \mu^n(x'') | D^n, x^{n+1} = x]$$

and this is graphically depicted in Figure 3.2 (L). If we augment the dataset with a new input-output pair (x^{n+1}, y^{n+1}), the new posterior mean $\mu^{n+1}(x)$ is easily computed from $\mu^n(x)$ by taking Equation 3.8 above and performing a simple change from prior mean and kernel to posterior mean and kernel at time n, an instance of

Bayesian updating

$$\mu^{n+1}(x) = \mu^n(x) + \frac{k^n(x, x^{n+1})}{k^n(x^{n+1}, x^{n+1}) + \sigma_\epsilon^2}(y^{n+1} - \mu^n(x^{n+1})) \tag{3.9}$$

$$= \mu^n(x) + \frac{k^n(x, x^{n+1})}{\sqrt{k^n(x^{n+1}, x^{n+1}) + \sigma_\epsilon^2}} \frac{y^{n+1} - \mu^n(x^{n+1})}{\sqrt{k^n(x^{n+1}, x^{n+1}) + \sigma_\epsilon^2}}. \tag{3.10}$$

The predictive distribution of the new output given the next input x^{n+1}, and the data so far X^n, Y^n is given by

$$y^{n+1} \sim N(\mu^n(x^{n+1}), k^n(x^{n+1}, x^{n+1}) + \sigma_\epsilon^2)$$

therefore the second term of the above factorisation is the z-score of y^{n+1} and so we may write

$$\mu^{n+1}(x) = \mu^n(x) + \tilde{\sigma}^n(x; x^{n+1})Z \tag{3.11}$$

where, at time n, $Z \sim N(0, 1)$ is unknown and random, $\tilde{\sigma}^n(x; x^{n+1})$ is a deterministic function of x' parameterised by x^{n+1} and is an additive update to $\mu^n(x)$ scaled by the stochastic Z,

$$\tilde{\sigma}^n(x; x^{n+1}) = \frac{k^n(x, x^{n+1})}{\sqrt{k^n(x^{n+1}, x^{n+1}) + \sigma_\epsilon^2}}. \tag{3.12}$$

Therefore we may write Knowledge Gradient as

$$\mathrm{KG}^n(x) = \mathbb{E}\left[\max_{x'} \mu^n(x') + \tilde{\sigma}^n(x'; x)Z - \max_{x''} \mu^n(x'') \middle| D^n, x^{n+1} = x\right]$$

There are multiple ways to evaluate this expectation of maximisations and we will discuss them later in Chapter 5 when we use this form of the acquisition function for the first time.

3.3.2 Efficient Global Optimization

We follow the above Value of Information with two simple modifications, the function to be optimized is deterministic and the decision maker can only choose x_r from the history of observations X^n. With these changes, the same derivation procedure yields the Expected Improvement acquisition function popularised by Jones et al.

[1998a]. Given a dataset X^n, Y^n, we fit a Gaussian process model to the dataset to yield $\mathbb{P}[\theta(x)|D^n]$, the decision maker loss is again given by $-\theta(x)$ and therefore the expected loss $-\mu^n(x)$. As the decision maker is restricted to $x_r \in X$, the optimal decision is $x_r = \arg\max_{x \in X^n} \mu^n(x)$ and the one-step value of information is

$$\mathrm{VoI}^n(x) = \mathbb{E}\left[\max_{x' \in X^n \cup \{x'\}} \mu^{n+1}(x') - \max_{x'' \in X^n \cup \{x''\}} \mu^n(x'') \middle| D^n\right].$$

This may be simplified to the EI acquisition function as follows. The model assumes deterministic outputs therefore perfectly interpolates the data and $\mu^n(x^i) = y^i$, therefore the performance is

$$\begin{aligned} P^n &= \max_{x \in X^n} \mu^n(x) \\ &= \max\{\mu^n(x^1), .., \mu^n(x^n)\} \\ &= \max\{y^1, .., y^n\} \\ &= \max Y^n. \end{aligned}$$

The model also interpolates the new data point too, $\mu^{n+1}(x^{n+1}) = y^{n+1}$. Substituting both terms into the $\mathrm{VoI}^n(x)$ and using the shorthand $\bar{y}^n = \max Y^n$ yields the EI acquisition function

$$\begin{aligned} \mathrm{EI}^n(x) &= \mathbb{E}\left[\max_{x' \in X^n \cup \{x\}} \mu^{n+1}(x') - \max_{x'' \in X^n \cup \{x\}} \mu^n(x'') \middle| D^n\right] \\ &= \mathbb{E}[\max Y^{n+1} - \max Y^n | x^{n+1} = x] \\ &= \mathbb{E}[\max\{0, y^{n+1} - \bar{y}^n\} | x^{n+1} = x] \\ &= \mathbb{E}[\max\{0, \mu^n(x) + \sqrt{k^n(x,x)}Z - \bar{y}^n\}] \\ &= (\mu^n(x) - \bar{y}^n)\,\Phi\left(-\frac{\bar{y}^n - \mu^n(x)}{\sqrt{k^n(x,x)}}\right) + \sqrt{k^n(x,x)}\,\phi\left(\frac{\bar{y}^n - \mu^n(x)}{\sqrt{k^n(x,x)}}\right) \end{aligned}$$

where $Z \sim N(0,1)$ is a standard Gaussian random variable, $\Phi(z)$ and $\phi(z)$ are the cumulative and density functions of the Gaussian distribution. This is graphically depicted in Figure 3.2(R).

Note the following two properties of the expected improvement function. First, $\mathrm{EI}(x)$ is non-negative as it is the expectation of a non-negative quantity,

$\max\{0, y^{n+1} - \bar{y}^n\} \geq 0$. Second, if the expectation is zero, then $\max\{0, y^{n+1} - \bar{y}^n\}$ must be zero for all realizations of y^{n+1} implying $k^n(x, x) = \text{Var}[\theta(x)] = 0$ (given $\mu^n(x) \leq \bar{y}^n$). The second condition means that $\text{EI}(x) = 0$ implies $\theta(x)$ is known. These two properties also hold for the Knowledge Gradient acquisition function; $\text{KG}(x) > 0$ and $\text{KG}(x) = 0$ where $\theta(x)$ is known exactly. As a result, for both methods that collect data at locations that maximise their acquisition function, they will converge to learning the true value of $\theta(x)$ for all $x \in X$ and the true optimum becomes known $x_r^n = \underset{x \in X}{\text{argmax}}\, \theta(x)$. The proof is a simple argument by contradiction, if data is sequentially allocated to *maxima* of $\text{KG}(x)$ or $\text{EI}(x)$, then data will never be re-allocated to points for which $\theta(x)$ is known as such points are known *minima* of $\text{KG}(x)$ and $\text{EI}(x)$. Hence, at each time step, only unknown $\theta(x)$ points are sampled. Assuming an appropriate differentiable kernel, this conclusion applies to both continuous and discretised search spaces [Toscano-Palmerin and Frazier, 2018, Zhang et al., 2019].

There exist problem settings where deriving a myopic acquisition function using the Value of Information recipe leads to a acquisition function for which the second property does not hold, $\text{KG}(x) = 0$ or $\text{EI}(x) = 0$ whilst $\theta(x)$ is still unknown, $k^n(x, x) > 0$. This is discussed in Section 3.3.4.

3.3.3 General Derivation Procedure

In the following chapters, we follow the above value of information procedure to derive myopic one-step look ahead acquisition functions. We use Gaussian processes for every application and we always assume the expected loss is simply a variant of the posterior mean of the GP such as $\sum_x \mu^n(x, a)$ and the decision maker will always choose $\arg\max_a \sum_x \mu^n(x, a)$ and therefore the performance is $\max_a \sum_x \mu^n(x, a)$. Consequently, deriving a value of information procedure only requires specifying the current performance P^n. By incorporating all of the previous steps into one, the performance may be viewed as a set function from any given dataset to real values

$$\text{P} : \mathcal{D} \to \mathbb{R}$$

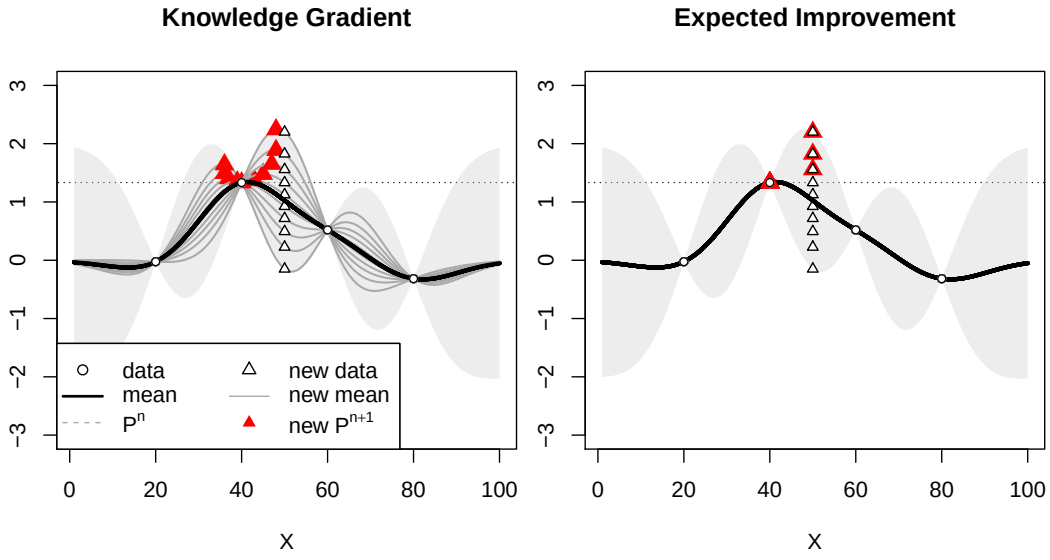


Figure 3.2: Consider evaluating F(50), $x^{n+1} = 50$. Left: Knowledge Gradient computes the average peak over realisations of the new *mean*. Right: EI computes the average peak over realisations of the new *dataset*.

where $D^n, D^{n+1} \in \mathcal{D}$ is the collection of all possible datasets. In the case of Knowledge Gradient, this performance function is

$$\mathrm{P}_{\mathrm{KG}}(D^n) = \max_x \mathbb{E}[\theta(x)|D^n]$$

and for Expected Improvement

$$\mathrm{P}_{\mathrm{EI}}(D^n) = \max Y^n.$$

After specifying the performance function, the acquisition function is given by

$$\mathrm{AF}(x; D^n) = \mathbb{E}_{y^{n+1}}[\mathrm{P}(D^{n+1})|x^{n+1} = x] - \mathrm{P}(D^n)$$

where $D^{n+1} = D^n \cup \{(x^{n+1}, y^{n+1})\}$ is the one-step look ahead dataset, x^{n+1} is the argument to the acquisition function, the candidate x to be added to the dataset and the expectation is over the unknown y^{n+1}. We use the predictive distribution of y^{n+1} from the GP model given D^n and x^{n+1}. Note that this does not account for any possible change in the hyperparameters. Therefore we are intrinsically

assuming that the model hyperparameters are known and therefore model mismatch can greatly affect algorithm performance. The performance set function is usually straightforward given the problem definition and a Gaussian process approximation of an underlying function. The difficulty comes in evaluating the expectation of the one-step difference in performance functions.

3.3.4 A Failure Case of the Myopic Value of Information Recipe

In Chapter 2.3.4, we discussed maximisation over $x \in X$ of a sum of functions indexed by $w \in W$,

$$F(x) = \sum_{w \in W} f(x, w),$$

and evaluations on $f(x, w)$ are collected to learn $\arg\max F(x)$. Consider the problem setting of worst-case optimisation where a user aims to maximise the minimum over a set of functions,

$$F(x) = \min_{w \in W} f(x, w).$$

Let us assume that both sets X and W are finite sets with no correlation across outputs, evaluating $f(x, w)$ does not inform $f(x', w')$ for $(x', w') \neq (x, w)$. Therefore, we have an independent prior over outputs $\mathbb{P}[f(x, w)]$ for each $(x, w) \in X \times W$ (although failure cases exist for the correlated case too). Consider the simplest case $X = \{a, b\}$ and $W = \{u, v\}$, after n observations we have a posterior with mean $\mu^n(x, w)$ and the performance is given by

$$\begin{aligned} \mathrm{P}^n &= \max_{x \in X} \min_{w \in W} \mu^n(x, w) \\ &= \max \left\{ \min\{\mu^n(a, u), \mu^n(a, v)\}, \min\{\mu^n(b, u), \mu^n(b, v)\} \right\} \end{aligned}$$

Given, $(x, w)^{n+1}$, the posterior mean $\mu^n(x, w)$ is only updated for the sampled input pair

$$\mu^{n+1}(x, w) = \begin{cases} \mu^{n+1}(x, w) & \text{if} (x, w) = (x, w)^{n+1} \\ \mu^n(x, w) & \text{else.} \end{cases}$$

Next, let choosing $x = b$ be the best option, that is

$$\min\{\mu^n(a,u), \mu^n(a,v)\} < \min\{\mu^n(b,u), \mu^n(b,v)\}$$

and $\mathrm{P}^n = \min\{\mu^n(b,u), \mu^n(b,v)\}$. For the $x = a$ decision, let $w = v$ be the worst case output $\min\{\mu^n(a,u), \mu^n(a,v)\} = \mu^n(a,v)$. In this case, sampling $y^{n+1} = f(a,u)$ can only reduce the worst case for choosing $x = a$,

$$\mathbb{E}[\min\{\mu^{n+1}(a,u), \mu^n(a,v)\}] \;\; \leq \;\; \mu^n(a,v).$$

and the value of information is zero as follows,

$$\begin{aligned}
&\mathbb{E}[\mathrm{P}^{n+1} - \mathrm{P}^n | D^n, (x,w)^{n+1} = (a,u)] \\
&\qquad = \mathbb{E}\left[\max\left\{\underbrace{\min\{\mu^{n+1}(a,u), \mu^n(a,v)\}}_{\leq \mu^n(a,v)}, \;\; \min\{\mu^n(b,u), \mu^n(b,v)\}\right\}\right] - \mathrm{P}^n \\
&\qquad \leq \max\left\{\mu^n(a,v), \;\; \underbrace{\min\{\mu^n(b,u), \mu^n(b,v)\}}_{\text{dominant term} = \mathrm{P}^n}\right\} - \mathrm{P}^n \\
&\qquad = \mathrm{P}^n - \mathrm{P}^n \\
&\qquad = 0.
\end{aligned}$$

The one-step value of information is non-negative for all (x,w), though we haven't shown it. However it is also zero for (a,u) while $\mathbb{P}[f(a,u)|D^n]$ is still not known. Regardless of the outcome of the new measurement $y^{n+1} = f(a,u)$, the decision maker's choice will not change, $x_r^n = x_r^{n+1} = b$. Hence sampling to maximise *myopic* value of information can get stuck and result in failure to converge. Multi-step look ahead can somewhat alleviate this issue. However in general alternative methods are required, such as OCBA which allocates according to the posterior distributions and not the one-step look ahead posterior mean.

Chapter 4

Discrete Task Conditional Ranking and Selection

4.1 Introduction

Given a finite set of tasks, each described by some continuous features, and a finite set of tools that can be applied to any task, we consider the problem of sampling to efficiently identify a mapping from the set of tasks to the best tool for each task. We approach the problem as an extension of ranking and selection problems, where an experimenter is typically required to find the single best overall tool from a set, and where best is defined as having the highest expected performance which can only be inferred via sampling. The main difference of our problem is that we aim to simultaneously identify the best tool for multiple tasks and there is some correlation of tool performance across tasks with similar features. This problem is very similar to contextual multi-armed bandits. However, we are interested in finding the best tool for each task instead of trying to maximise cumulative reward during sampling and we assume that the user is free to choose any tool-task pair (or state-action pair, context-arm pair), the "context" is not an exogenous variable. This problem has many practical applications, including the following three possible examples.

1. *Algorithm selection.* For most hard optimization problems, there exist multiple algorithms. Although some algorithms may work better than others overall,

usually different algorithms work best for different problem instances. Thus, there is the problem of deciding which algorithm to use for which problem instance, based on features of the problem instance. Smith-Miles [2008] provides a survey on the algorithm selection problem and Smith-Miles et al. [2014] perform a case study with graph coloring algorithms.

2. *Personalized Medicine.* The pharmaceutical industry is currently experiencing a shift from the one-drug-fits-all paradigm towards personalization, where therapies are targeted towards particular subgroups of patients. Clinical trials then not only have to determine whether a drug is effective or not, but also which drug works best for which type of patient, depending on patient characteristics. Xu et al. [2014] propose a method to sequentially test treatments on patients to find patient subgroups in a simulated breast cancer trial.

3. *Online marketing.* It is easy to deploy several different advertisements and advertisement formats (banner, video, etc.). A website designer aims to find the best advertisement for each user without showing too many adverts to many users and so instead must choose a series of users and adverts to learn the user-advert mapping that maximizes ad revenue for the website.

We tackle the problem of efficient information collection to find a mapping from the set of tasks to the set of tools by using Gaussian Process Regression as a metamodel to predict a tool's expected performance across task space. We then propose and empirically compare three such myopic sampling policies, Regional Expected Value of Improvement (REVI), and two simpler heuristics NEVI and EVI. The REVI policy maximizes the expected improvement in predicted mapping performance, accounting for how one sample on one task can influence all the other tasks. NEVI and EVI are approximations to REVI that are much cheaper to compute yet still perform well. For a given budget, the three policies sequentially create sample designs producing mappings that perform significantly better than Latin Hypercube designs reducing the necessary sampling budget to obtain a desired level of performance by up to 67% in our experiments.

The chapter is structured as follows. In Section 4.2 the problem and math-

ematical framework are laid out, followed in Section 4.3 by the derivation of our methods. Empirical results on both synthetic benchmarks and a heuristic selection problem are reported in Section 4.4. We conclude in Section 4.5 with a summary and some ideas for future work.

4.2 Problem Definition

There exists a finite set of M tasks indexed by $t \in \{1, ..., M\}$ and a set of A alternative tools indexed by $a \in \{1, ..., A\}$. Each task t is characterized by D continuous features, $x_t \in \mathbb{R}^D$. The set of feature vectors for all tasks is denoted $X = \{x_1, ..., x_M\}$. We can apply a tool a to a task t to obtain a stochastic performance measurement that is a realization of a random variable $Y_{t,a} = \zeta_{t,a} + \epsilon_a$ where $\zeta_a \in \mathbb{R}^M$ is a vector of M true expected performance values for all tasks for tool a. $\epsilon_a \sim N(0, \sigma^2_{\epsilon,a})$ is white noise distributed with known variance which in practice is estimated by preliminary sampling. In this work we consider noise that is heteroscedastic over tools $a \in A$ but homoscedastic over tasks $x \in X$. This serves as a simplifying assumption removing the need to learn a compolex for noise variance over X greatly reducing the number of points (often called replications) required. The values of $\zeta_1, ..., \zeta_A$ are assumed to be underlying latent functions of the task features $\tilde{\zeta}_1(x), ..., \tilde{\zeta}_A(x) : \mathbb{R}^D \to \mathbb{R}$ and $\zeta_{t,a} = \tilde{\zeta}_a(x_t)$, however, they are unknown to the user and must be inferred. We are given a finite budget of N performance measurements, or samples, to be allocated to the (task, tool) design space $\{1, ..., M\} \times \{1, ..., A\}$, and the goal is to learn a classifier, or a mapping, $S : X \to \{1, ..., A\}$ from features to the best tool. Given only limited information, the mapping $S(x_t)$ must approximate $\underset{a}{\text{argmax}}\ \zeta_{t,a}$ for each $x_t \in X$ and therefore maximize the expected portfolio performance summed over all considered tasks:

$$\text{Portfolio Performance} = \sum_{t=1}^{M} w_t \zeta_{t,S(x_t)} \tag{4.1}$$

where w_t are user defined positive weights representing the relative importance of each task, which we include to allow a user to account for differences between the distribution of tasks $x_1, ..., x_M \sim \mathbb{P}_X[x_t]$ and a desired target distribution $\mathbb{P}_T[x_t]$, examples are given below. The objective of sampling is the expected portfolio

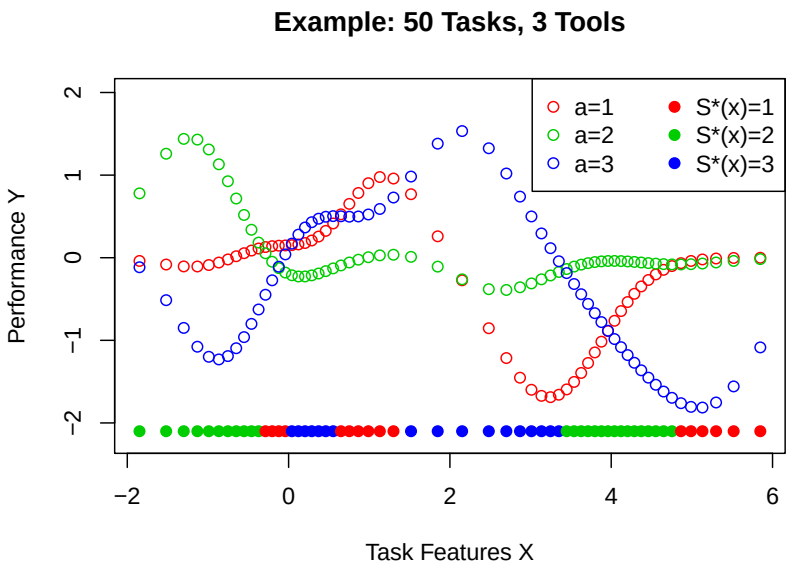


Figure 4.1: Empty points show tool performance on a given task ζ_a vs X, solid points represent the optimal mapping $S^*(x_t)$. Each task, x, is a ranking and selection problem over $a \in \{1, 2, 3\}$ and we desire an optimisation algorithm that can find the best tool for each task.

performance and not the classification error and so we refer to $S(x)$ as a mapping, as opposed to a classifier. See Figure 4.1 for an illustrative example.

In the remainder of this section, we explain in more detail how the applications mentioned in Section 4.1 are three instances of the same general problem class defined by the framework above.

In the algorithm case study of Smith-Miles et al. [2014], 8 graph coloring algorithms and a dataset of 6,948 graphs were considered. After measuring the number of colors for every algorithm on every instance once (8*6,948 measurements), the authors extract features of graphs and train a support vector machine to classify instances based on the density, algebraic connectivity and energy of each graph to the best algorithm. In the above formulation the number of tasks is $M = 6,948$, each task has features $x_1, ..., x_{6948} \in \mathbb{R}^3$, and there are $A = 8$ graph coloring algorithms. The performance, $Y_{t,a}$, is the number of colors when running one (possibly stochastic) algorithm a on a graph with features x_i and $\zeta_{t,a}$ is the expected number of colors if the same algorithm were applied many times to the same instance. Although $Y_{t,a}$ is

count data and not normally distributed, a log transform may be used, as in Poisson regression, approximating the count data as log-normal to fit the above assumptions. All graphs are considered equally important and so $w_t = 1/6948$ for all t. A user aims to find a mapping $S(x_t) : \mathbb{R}^3 \to \{1, ..., 8\}$ that recommends the algorithm that minimizes the expected number of colors. Given restricted computing resources a user must carefully choose a sequence of algorithms and problem instances for measuring performance in order to learn the optimal mapping over the problem instance distribution.

Xu et al. [2014] simulate a breast cancer trial in which there are $M = 300$ patients each with 4 biomarkers that are uniformly distributed and measured at the start of the trial $x_1, ..., x_{300} \in [-1, 1]^4$. In their synthetic scenario there are $A = 3$ treatments, performance is the presence or not of a response to treatment $Y_{t,a} \in \{0, 1\}$ therefore $\zeta_{t,a}$ is the expected frequency of positive responses to treatment for patient t when given treatment a, although the performance $Y_{t,a}$ may be replaced by the continuous size of the response itself to fit the above formulation. The trial organizers must sequentially choose a patient t and a treatment a and then observe the response $Y_{t,a}$ to learn the optimal partitions of biomarker space. The patients recruited in the trial may not be representative of the whole patient population and so weights $w_t = P_T[x_t]/P_X[x_t]$ may be used as importance sampling weights to account for the difference in the distribution of biomarkers of patients inside the trial $\mathbb{P}_X[x_t]$, approximated using kernel density estimates, and the target population $\mathbb{P}_T[x_t]$. A clinician aims to find a mapping $S : [-1, 1]^4 \to \{\text{drug 1, drug 2, drug 3}\}$ that returns the treatment maximizing the expected response for a patient with biomarkers x over the whole target population distribution $\mathbb{P}_T[x]$ whilst sampling form the trial population $x_1, ..., x_{300}$.

In online advertising, at any one point in time there is a population of M users logged onto a given website. Various features of each user can be measured, for example a user's fraction of time spent on consuming content types such as news, entertainment and educational content, in which case $x_1, ..., x_M \in [0, 1]^3$. The website may have a choice of $A = 3$ possible advertisements, $a \in \{\text{pop-up, banner, video}\}$, to show to any given user i in order to maximize the number of advert clicks (with

a log-transform for count data) or the amount of money spent, $Y_{t,a}$, in a fixed measurement time period. $\zeta_{t,a}$ is then the expected number of clicks or revenue if a user of type x_t is shown an advert of type a. Again the weights may be importance sampling weights accounting for the difference between the current user population distribution and a desired target population $w_t = \mathbb{P}_T[x_t]/\mathbb{P}_X[x_t]$ where the target population may be the overall long term stationary distribution of users. Given that the website aims to minimize interruptions of user experience, or because of bandwidth constraints, it is not desirable to show adverts to all users all the time, thus the website must carefully choose a sequence of users from the current M users and adverts to learn a mapping $S : [0, 1]^3 \to \{\text{pop-up, banner, video}\}$ over the entire target population $\mathbb{P}_T[x]$. It should be relatively straightforward to adapt the algorithms proposed in our paper to allow the actual pool of users change over time and to account for sampling multiple tools and tasks simultaneously.

Note that we assume that during learning, the task set is fixed and given. This is true in many applications. For example, it is not easy to generate additional problem instances with particular features, or recruit additional patients with particular features onto a drug testing trial. However, once the mapping has been constructed, it can be used to predict the best tool for any task in the feature space (not only the ones used for learning).

4.3 Sampling Methods

In this section, we first introduce a mathematical framework building on the work of Frazier et al. [2009a], followed by three novel myopic sampling policies.

Consider a state during sampling when n samples have been collected.. We denote the sequence of sampled tasks $(t^1, ..., t^n)$, tools $(a^1, ..., a^n)$ and the sequence of pairs $(t, a)^1, ..., (t, a)^n$ is written as $\{(t, a)\}_1^n$. The vector of n corresponding performance measurements is denoted $(y^1, ..., y^n) = Y^n$. For a given tool, let n_a be the number of samples allocated to tool a, denote the tasks that have been measured as $T_a^n = \{t^i | a^i = a, i = 1, ..., n\}$ and the subset of performance measurements as $Y_a^n \subseteq Y^n$ and feature values $X_a^n \subseteq X^n$. We next define the filtration, $\mathcal{F}^n$, to be the

sigma algebra generated by the tasks, tools and performance measurements sampled so far $\mathcal{F}^n = \sigma\{(t^1, a^1, y^1), ..., (t^n, a^n, y^n)\}$.

For each tool, we treat the unknown true performance vectors $\zeta_a \in \mathbb{R}^M$ as Bayesian random variables denoted by $\theta_a \in \mathbb{R}^M$. Starting with a multivariate normal prior

$$\mathbb{P}[\theta_a] \sim \mathcal{N}(\mu_a^0, \Sigma_a^0)$$

with parameters informed by the features $\mu_a^0 = \mu_a(X) \in \mathbb{R}^M$, $\Sigma_a^0 = k_a(X, X) \in \mathbb{R}^{M \times M}$. We have independent and identically distributed Gaussian observation noise with mean that is the t^i element of θ_{a^i} with a Gaussian likelihood

$$\mathbb{P}[y^i|\theta_1, ..., \theta_A, t^i, a^i] = N(y|\theta_{a^i,t^i}, \sigma^2_{\epsilon,a^i}),$$

and therefore the posterior for a given θ_a after n samples is also multivariate Gaussian

$$\mathbb{P}[\theta_a|\mathcal{F}^n] \sim \mathcal{N}(\mu_a^n, \Sigma_a^n)$$

where μ_a^n denotes the posterior mean after n samples and likewise for Σ_a^n.

As described in Chapter 3.2, by using the Matrix Inversion Lemma [Hager, 1989] to condition the prior on the data points collected so far, the posterior mean and covariance for a single tool a, $\mathbb{P}[\theta_a|\mathcal{F}^n] = \mathcal{N}(\mu_a^n, \Sigma_a^n)$, are then given by the following matrix equations:

$$\mu_a^n = \mu_a^0 + (k_a^n)^\intercal (K_a^n)^{-1}(Y_a^n - \mu_a(X_a^n)), \tag{4.2}$$

$$\Sigma_a^n = \Sigma_a^0 - (k_a^n)^\intercal (K_a^n)^{-1} k_a^n, \tag{4.3}$$

where $k_a^n = [\Sigma_a^0]_{T_a^n} = k_a(X, X_a^n)$ is the columns of the prior matrix with indices T_a^n which is also the kernel evaluated between all tasks and samples tasks for tool a. Similarly for the K_a^n matrix,

$$K_a^n = \left[\Sigma_a^0\right]_{T_a^n, T_a^n} + I\sigma^2_{\epsilon,a} = k_a(X_a^n, X_a^n) + I\sigma^2_{\epsilon,a}$$

is the prior matrix with both rows and columns with indices T_a^n (or the kernel

evaluated at observed features) with an added noise matrix.

True vectors of $\zeta_1, ..., \zeta_A$ are unknown to the experimenter, therefore the mapping is constructed by selecting the tool, a, for each task t with the highest predicted performance,

$$S^n(t) = \underset{a}{\text{argmax}}\ \mu^n_{t,a}.$$

To derive a Value of Information based algorithm, we first require a measure of current performance. Using the Bayesian mode, the predicted mapping performance at time n is given by

$$P^n = \sum_{t=1}^{M} w_t \mu^n_{t,S^n(t)} = \sum_{t=1}^{M} w_t \max_a \mu^n_{t,a}. \tag{4.4}$$

We aim to iteratively maximize improvement in the above predicted performance in as a measurable surrogate for maximising true performance Equation 4.1.

In order to maximise Equation 4.4, we require a measure of how much a new performance measurement from a given (task, tool) pair will affect the predicted performance, and we aim to derive $\mathbb{E}[P^{n+1}|\mathcal{F}^n, (t,a)^{n+1}]$. The updated predicted performance depends on the posterior mean vectors $\mu^{n+1}_1, ..., \mu^{n+1}_A$ and so we derive a one step look-ahead recursion formula for the posterior parameters. At a given stage during sampling, n, measuring the performance of tool a^{n+1} applied to task t^{n+1} generates the next performance value y^{n+1}. By concatenating the appropriate values to form the updated matrices $k^{n+1}_{a^{n+1}}, K^{n+1}_{a^{n+1}}$ and $Y^{n+1}_{a^{n+1}}$, one can use the Matrix Inversion Lemma again for the updated inverse $(K^{n+1}_{a^{n+1}})^{-1}$. The following recursion (which is also found in Frazier et al. [2009a]) is derived and it does not contain the $(K^n_a)^{-1}$ matrix inversion, t^{n+1} has temporarily been replaced by t for brevity:

$$\mu^{n+1}_a = \begin{cases} \mu^n_a + \frac{y^{n+1} - \mu^n_{t,a}}{\Sigma^n_{tt,a} + \sigma^2_{\epsilon,a}} \Sigma^n_{t,a} & a = a^{n+1} \\ \mu^n_a & \text{otherwise} \end{cases} \tag{4.5}$$

$$\Sigma^{n+1}_a = \begin{cases} \Sigma^n_a - \frac{\Sigma^n_{t,a} \Sigma^{n\intercal}_{t,a}}{\Sigma^n_{tt,a} + \sigma^2_{\epsilon,a}} & a = a^{n+1} \\ \Sigma^n_a & \text{otherwise} \end{cases} \tag{4.6}$$

where $\Sigma^n_{t,a}$ denotes the t^{th} column of the symmetric matrix Σ^n_a. Note that the Σ^n_a matrices only depend on the sampling decisions $\{(t,a)\}^n_1$. At time n, the scalar y^{n+1} and vector $\mu^{n+1}_{a^{n+1}}$ are not $\mathcal{F}^n$ measurable. However, given the posterior predictive distribution of the new observation

$$\mathbb{P}[y^{n+1}|\mathcal{F}^n,(t,a)^{n+1}=(t,a)] = N(\theta_{t,a},\sigma^2_{\epsilon,a}) = N(\mu^n_t,\Sigma^n_{tt,a}+\sigma^2_{a,\epsilon}),$$

the posterior predictive distribution of $\mu^{n+1}_{a^{n+1}}$ is calculated using the above recursion formula. For a given standard normal random variable $Z \sim N(0,1)$ we have the following (for clarity we have dropped the subscript a in the following two formulae):

$$\begin{aligned}\mu^{n+1} &= \mu^n + \frac{(\mu^n_t + \sqrt{\Sigma^n_{tt}+\sigma^2_\epsilon}Z) - \mu^n_t}{\Sigma^n_{tt}+\sigma^2_\epsilon}\Sigma^n_t \\ &= \mu^n + \frac{\Sigma^n_t}{\sqrt{\Sigma^n_{tt}+\sigma^2_\epsilon}}Z.\end{aligned}$$

And so we define the vector valued function $\tilde{\sigma}^n : \{1,..,M\}\times\{1,..,A\} \to \mathbb{R}^M$ with entries

$$\tilde{\sigma}^n(t,a) = \frac{\Sigma^n_{t,a}}{\sqrt{\Sigma^n_{tt,a}+\sigma^2_{\epsilon,a}}} \tag{4.7}$$

which gives the deterministic additive update to the posterior mean vector which is scaled by the stochastic Z. Therefore, when conditioned on $\mathcal{F}^n$ and the next (task, tool) pair,

$$\begin{aligned}\mu^{n+1}_a &\sim \mathcal{N}\left(\mu^n_a, \tilde{\sigma}^n(t,a)\tilde{\sigma}^n(t,a)^\intercal\right), \\ \Sigma^{n+1}_a &= \Sigma^n_a - \tilde{\sigma}^n(t,a)\tilde{\sigma}^n(t,a)^\intercal,\end{aligned}$$

where the distribution of the new posterior mean for the j^{th} task caused by a new sample for the t^{th} task is given by $\mu^{n+1}_{j,a} \sim N(\mu^n_{j,a},\ \tilde{\sigma}^n_j(t,a)^2)$. With a predictive distribution for the posterior mean after a new sample, we can calculate the expectation of predicted mapping performance after the next sample $\mathbb{E}[P^{n+1}|\mathcal{F}^n,(t,a)^{n+1}]$. We now use this to define three sampling policies.

4.3.1 Regional Expected Value of Improvement Policy

Ideally, the most useful data collect has large performance uncertainty, high predicted performance thus accounting for exploration and exploitation. Also, the data must be informative about as many other tasks as possible, we need to account for regional effects across the task space. We define the Regional Expected Value of Improvement (REVI) of a new sample at task t and tool a as the expectation of improvement in predicted mapping performance:

$$\text{REVI}^n(t,a) = \mathbb{E}\left[P^{n+1} - P^n \middle| \mathcal{F}^n, (t,a)^{n+1} = (t,a)\right], \tag{4.8}$$

and the formula may be computed analytically:

$$\begin{aligned}
\text{REVI}^n(t,a) &= \mathbb{E}\left[\sum_{j=1}^{M} w_j \max_b \mu_{j,b}^{n+1} \middle| \mathcal{F}^n, (t,a)^{n+1} = (t,a)\right] - \sum_{j=1}^{M} w_j \max_b \mu_{j,b}^n && (4.9)\\
&= \sum_j w_j \mathbb{E}\left[\max\{0, -|\mu_{j,a}^n - \max_{b\neq a} \mu_{j,b}^n| + \tilde{\sigma}_j^n(t,a)Z\}\right] && (4.10)\\
&= \sum_j w_j \text{h}(\Delta_{j,a}^n, \tilde{\sigma}_j^n(t,a)), && (4.11)
\end{aligned}$$

where the intermediate steps between Equations (4.9) and (4.10) are provided in the online appendix, $\Delta_{j,a}^n = |\mu_{j,a}^n - \max_{b\neq a} \mu_{j,b}^n|$ and the function $\text{h} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is the well-known EI function derived from the truncated normal distribution of Z:

$$\text{h}(\Delta, \sigma) = |\sigma|\phi(\Delta/|\sigma|) - \Delta\Phi(-\Delta/|\sigma|), \tag{4.12}$$

where ϕ and Φ are standard normal density and distribution functions, respectively. In this case $|\sigma|$ is necessary because $\tilde{\sigma}_j^n(t,a)$ may be negative and only the magnitude is necessary by the symmetry of the normal distribution. At a given stage during sampling, under the REVI policy, the next sample is allocated to the (task, tool) pair that satisfies:

$$(t,a)^{n+1} = \text{argmax } \text{REVI}^n(t,a) \tag{4.13}$$

which is simply maximized by full inspection. The REVI sampling policy allocates

each sample to maximize the expected improvement in the predicted mapping performance and thus is myopically optimal by construction. It is also asymptotically optimal, meaning that given an infinite sampling budget, the policy will always find the true best tool for each task and find the mapping that maximizes the true mapping performance. This is because the expected improvement of sampling a (task, tool) pair decreases, on average, towards zero as more samples are allocated and thus any unsampled (task, tool) pair eventually becomes the (task, tool) pair that maximizes regional expected improvement and is chosen for sampling. Therefore, as the budget approaches infinity, all (task,tool) pairs are sampled infinitely often and by the law of large numbers, posterior distributions $\theta_1, ..., \theta_A$ become point masses at the true values $\zeta_1, ..., \zeta_A$:

Theorem 4.3.1 *When sampling according to the REVI policy, as the budget goes to infinity, $N \to \infty$, the sequence of mappings converges almost surely to $S^N(x_t) = \underset{a}{\text{argmax}}\ \zeta_{t,a}$ for all t.*

A proof of Theorem 4.1 can be found in the online appendix where we also provide a Dynamic Programming formulation for this problem. A bound on the sub optimality gap between the surrogate objective of an optimal policy and the REVI policy for finite budgets can also be calculated and is simply the largest possible sum of the future additive updates to posterior means which is finite as N tends to infinity.

Theorem 4.3.2 *When sampling according to the REVI policy, the difference in the surrogate objective between optimal sampling and REVI sampling is bounded above by*

$$\max_{\pi} \mathbb{E}[P^N|\mathcal{F}^n, \pi] - \mathbb{E}[P^N|\mathcal{F}^n, REVI] \leqslant \max_{(t,a)_{n+1}^{N-1}} \sum_{k=n}^{N-2} \sqrt{2\pi} \sum_{j=1}^{M} w_j |\tilde{\sigma}_j^k(t^{k+1}, a^{k+1})| \tag{4.14}$$

where π is the set of all possible future sampling sequences $\{(t,a)\}_{n+1}^N$ and where $\tilde{\sigma}^k(t,a)$ for $k > n$ may be calculated deterministically given $\{(t,a)^l\}_{l=0}^k$ only as it does not depend on the stochastic observations $Y_{t,a}$.

The REVI policy allocates samples based on a trade-off between three considerations. Priority is given to (task, tool) pairs which have large posterior variance,

low difference in posterior means between the selected tool and best of the other tools, and tasks whose performance is highly correlated to many other tasks. When using the squared exponential kernel for the Gaussian process, highly correlated tasks have similar features hence the REVI policy gives sampling priority to tasks in task-dense regions of feature space. From an alternative view, the REVI policy de-prioritizes sampling of tasks with outlying features for which improving the mapping will not significantly contribute to total predicted performance. Figure 4.2 provides an example comprising two tools and 50 equally weighted tasks with features in $\mathbb{R}$ and Gaussian Processes with the squared exponential kernel. One can see that the REVI function is larger where tasks are dense and where posterior means are close and variance is large, i.e., where there is large uncertainty about which tool is the best across many tasks.

Complete computation of $\text{REVI}^n(t, a)$ for all (t, a) requires M^2A function calls to $\text{h}(\Delta, \sigma)$ and also requires the entire matrices $\Sigma_1^n, ..., \Sigma_A^n$ and all evaluations of $\tilde{\sigma}^n(t, a)$ which each have an M^2A memory requirement. This can be prohibitively expensive in scenarios with many tasks where M is large. The following two policies make simplifying assumptions that reduce this computational complexity.

4.3.2 Noisy Expected Value of Improvement Policy

The Noisy Expected Value Improvement (NEVI) policy assumes that the (task, tool) pair that maximizes the expected improvement in a tool's predicted performance on the selected task also maximizes the expected improvement in the predicted mapping performance. It would therefore be possible to approximate the sum in Equation (4.9) by taking only the t^{th} term such that the computational complexity is reduced to $O(MA)$. Intuitively, the NEVI policy neglects the impact the sample would have on the predicted performance of other correlated tasks, neglecting the task density.

We define the Noisy Expected Value of Improvement of a new sample at task t and tool a as the expected improvement in tool performance for the sampled task

alone:

$$\begin{aligned} \text{NEVI}^n(t,a) &= w_t \, \mathbb{E}\left[\max_b \mu_{t,b}^{n+1} - \max_b \mu_{i,b}^n \middle| \mathcal{F}^n, a^{n+1} = a\right] && (4.15) \\ &= w_t \, \text{h}(\Delta_{t,a}^n, \tilde{\sigma}_t^n(t,a)), && (4.16) \end{aligned}$$

and the next sample is determined by maximizing the above improvement:

$$(t,a)^{n+1} = \arg\max \ \text{NEVI}^n(t,a) \tag{4.17}$$

which is also optimized by inspection. The NEVI policy allocates samples to (task, tool) pairs based on a trade-off between only two considerations, where the posterior means of the sampled tool and the best of the other tools is close, and where the posterior variance is large for the sampled (task, tool) pair. This policy does not account for the effect a new measurement will have on covarying predictions but it does account for noisy measurements which is why it is called the Noisy Expected Value of Improvement policy.

The NEVI policy is not myopically optimal, but like the REVI policy, it is asymptotically optimal. We show in Section 4.4 that the NEVI and REVI policies perform comparably in our synthetic benchmarks when task features are uniformly distributed, whereas REVI outperforms NEVI when task features are clustered. Figure 4.2 shows how NEVI and REVI differ, for example the NEVI function gives more priority to tasks with outlying features than REVI. In the special case where there is no covariance between tasks, the NEVI and REVI policies allocate samples equally and therefore NEVI is also myopically optimal. In the special case where there is only one task, the problem reduces to standard ranking and selection, the REVI and NEVI policies become identical and both are equivalent to the Knowledge Gradient policy for sampling from A independent alternatives.

Each iteration of the NEVI policy only requires MA function calls to $\text{h}(\Delta, \sigma)$ and only the diagonal elements of the $\Sigma_1^n, ..., \Sigma_A^n$ and single values $\tilde{\sigma}_i(i,a)$ which in total have a memory requirement that scales as MA. Thus the NEVI policy is much more efficient to compute than REVI for large M. However, one cannot use the recursion formula given in Equations (4.5) and (4.6). Instead the typically smaller

$(K_a^n)^{-1}$ matrix inversion with computation $O(n_a^3)$ is required in Equations (4.2) and (4.3) and the computational complexity can be reduced by using formula for inverse of partitioned matrices given in Press et al. [1996] p.77.

4.3.3 Expected Value of Improvement Policy, EVI

In addition to the simplifying assumption of NEVI, the Expected Value of Improvement (EVI) policy also assumes that the noise in performance measurements is negligible. Therefore $\tilde{\sigma}_t^n(t,a) = \Sigma_{tt,a}^n / \sqrt{\Sigma_{tt,a}^n + \sigma_{\epsilon,a}} \approx \sqrt{\Sigma_{t,a}^n}$ and $\mu_{t,a}^{n+1}$ is equal in distribution to $\theta_{t,a}$. We define the EVI of a new sample at task t and tool a as the following:

$$\begin{aligned} \mathrm{EVI}^n(t,a) &= w_t \, \mathbb{E}\left[\max\{\theta_{t,a}, \max_{b \neq a} \mu_{t,b}^n\} - \max_b \mu_{t,b}^n \middle| \mathcal{F}^n \right] && (4.18) \\ &= w_t \, \mathrm{h}\left(\Delta_{t,a}^n, \sqrt{\Sigma_{tt,a}^n} \right), && (4.19) \end{aligned}$$

and the next sample is given by maximizing the above expected improvement:

$$(t,a)^{n+1} = \arg\max \ \mathrm{EVI}^n(t,a). \tag{4.20}$$

We include this policy for its simplicity and we demonstrate numerically that it performs similarly to the REVI and NEVI policies when tasks are uniformly distributed and sampling budgets are small. However, like the NEVI policy it loses some efficiency when tasks are clustered. However, when the variance of the noise for each tool, $\sigma_{\epsilon,1}^2, ..., \sigma_{\epsilon,A}^2$, are comparable to the posterior variances for each tool, the extra simplifying assumption of EVI becomes less applicable and the policy is less efficient. As sampling budget N increases, posterior variances for all the (task, tool) pairs tend to zero whilst the noise variance is constant, therefore the EVI policy will always perform worse than the REVI and NEVI policies as budget increases.

In the example in Figure 4.2, NEVI and EVI are relatively similar and both have peaks for the same (task, tool) pair.

When performance measurements are deterministic ($\sigma_{\epsilon,a} = 0$ for all a) the EVI and NEVI policies allocate samples identically. Despite the added simplification,

the EVI policy is also asymptotically optimal. This policy requires MA function calls to $\mathrm{h}(\Delta, \sigma)$ and the only diagonal elements of the posterior covariance matrices. It does not require computation or storage of $\tilde{\sigma}^n(t, a)$. We include it for its simplicity and similarity to the expected improvement of the EGO algorithm that measures how much a normally distributed new sample will improve upon a current best sample, in this case it is measuring how much the normally distributed prediction of one tool $\theta_{t,a}$ improves upon the other tools $\mu^n_{t,b}$ where $b \neq a$.

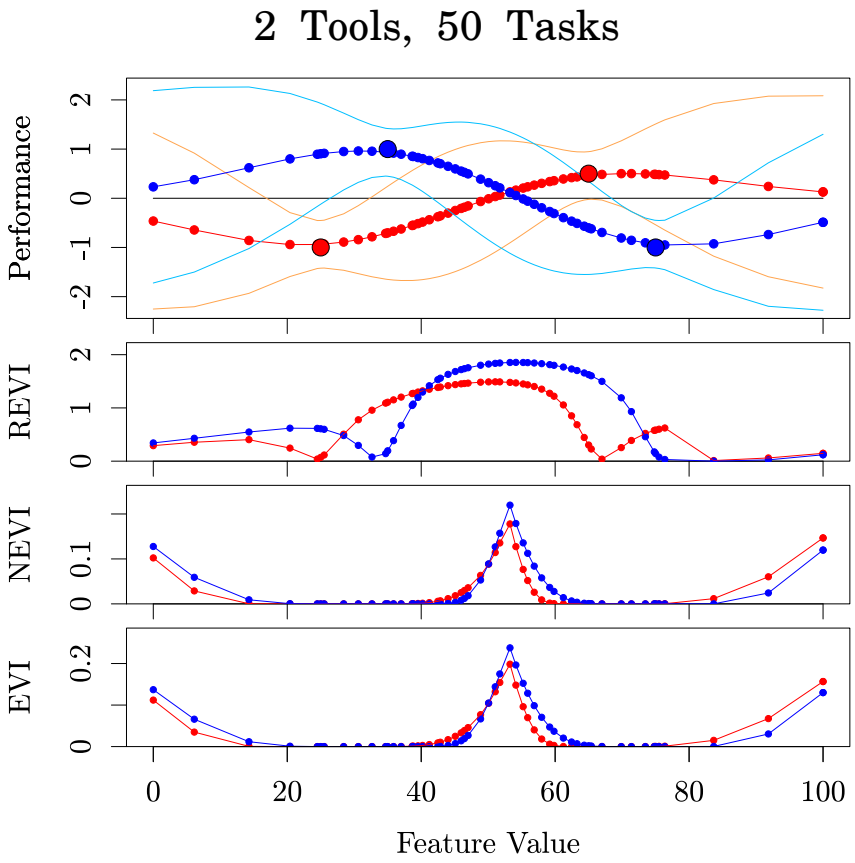


Figure 4.2: In all plots the x-axis is the single feature of the 50 tasks, $x_t \in \mathbb{R}$. Top: the Gaussian Processes for two tools after 4 performance measurements (large points) $(x_{t^1}, y^1), ..., (x_{t^4}, y^4)$ colored according to tool, and posterior mean performance for 50 tasks (small points) $(x_t, \mu^4_{t,a})$ with confidence intervals. Below: $\mathrm{REVI}^4(t, a)$, $\mathrm{NEVI}^4(t, a)$ and $\mathrm{EVI}^4(t, a)$ plotted against x_t for both tools where all tasks have equal weight. REVI is high where task density is high, posterior means are close and posterior variance is large. NEVI and EVI don't account for task density therefore give relatively larger value to the outlying tasks.

4.4 Numerical Experiments

In order to test the methods empirically, we perform two experiments. First we create a dataset based upon the case study performed by Smith-Miles et al. [2014] and second use a dataset of performance measurements of scheduling heuristics over a range of factory conditions.

4.4.1 Synthetic Experiments Setup

Given $M = 6,948$ graphs to be coloured and $A = 8$ different coloring algorithms, Smith-Miles et al. [2014] ran all algorithms on all instances and constructed a support vector machine (mapping) from graph features $X \subset \mathbb{R}^2$ to best algorithm $\{1, ..., 8\}$. The support vector used a Gaussian kernel with length scale of the kernel was approximately 1/5 of the spread of the instances. We synthesize data based on these observations. We create two artificial data sets of $M = 500$ tasks with all equal weights, $w_t = 1$ for all $i \in \{1, ..., 500\}$. The first set, the uniform case X_U, has feature values in the unit square $x_t \in (0, 1)^2$ sampled from the uniform distribution. The second set of tasks, the bimodal case X_B, is composed of points in $\mathbb{R}^2$ where 250 of the x_t values are distributed according to a bivariate normal distribution $\mathcal{N}((0, 0), \mathbf{I}0.125^2)$ and the remaining 250 points are distributed according to $\mathcal{N}((0.5, 0), \mathbf{I}0.125^2)$. The points form two circular clusters whose centers are 4 standard deviations apart, the task distributions are plotted in Figures 4.3 (a) and (d). We use these two distributions to emphasize the differences between the REVI policy that accounts for the task correlation and the NEVI and EVI policies that do not.

We perform experiments with $A = 3, 5$ and 8 tools. For each experiment in each set of tasks, we generate 8 vectors of true performance values, $\zeta_1, \ldots, \zeta_8 \in \mathbb{R}^{500}$, and use only the first 3 or first 5 when $A = 3, 5$. Each performance vector, ζ_a, was randomly generated by sampling from a Gaussian Process prior over the task features with a squared exponential kernel, $\zeta \sim \mathcal{N}(\underline{0}, \Sigma)$ where $\Sigma_{ij} = \sigma_0 \exp(-D(x_t - x_j, l_1, l_2)/2)$ and $D(x_t - x_j, l_1, l_2) = (x_{i,1} - x_{j,1})^2/l_1^2 + (x_{i,2} - x_{j,2})^2/l_2^2$. The parameters for the Gaussian Process generating the data were $\sigma_0 = 1$, and $l_1 = l_2 = 0.1$, the same

hyperparameters were used for all generated data and both task sets. The variance of the added noise was set to $\sigma^2_{\epsilon,a} = 1/10^2$ for all tools, and noise is independently and identically distributed for each sample. Figures 4.3 (b) and (e) show surface plots of the one of the sets of generated latent functions when $A = 3$.

We initialize each sampling procedure with $n_0 = 20$ measurements per tool, 10 per dimension as recommended by Jones et al. [1998a], allocated to tasks by a Latin Hypercube Design described in Section 4.4.2. After the initialization, a Gaussian Process computing $\mu^{20}_1, ..., \mu^{20}_A$ and $\Sigma^{20}_1, ..., \Sigma^{20}_A$. We separately apply REVI, NEVI and EVI sequential policies until the budget of $N = 300, 500, 800$ has been consumed for experiments with 3, 5 and 8 tools respectively. For comparison we also construct mappings using samples allocated by Latin Hypercube Designs with equivalent budgets $N = 20A$ to $N = 100A$.

All reported results are averaged over 400 replications, with 400 different sets of performance vectors, 400 unique initial LHC designs and random number streams for the noise. At each stage during sampling, the mapping is constructed by choosing the highest predicted tool for each task, $S(x_t) = \underset{a}{\text{argmax}}\ \mu^n_{t,a}$, and the true opportunity cost of the mapping is measured,

$$OC = \sum_t \max_a \zeta_{t,a} - \zeta_{t,S^n(t)}.$$

We repeat each experiment 400 times and report the empirical average opportunity cost, EOC, in Figure 4.4. Table 4.1 reports the final average Opportunity Cost with standard errors.

4.4.2 Mapping based on Latin Hypercube Design

We consider a custom rank based latin hypercube. Given a sampling budget that is a multiple of A, we allocate $N_{LHD} = N/A$ samples to each tool. N_{LHD} tasks are chosen from the set of 500 by a Latin Hypercube applied to the ranks of the sorted feature values $X \subset \mathbb{R}^2$. This makes no difference for X_U as the ranks and feature values are both uniformly distributed. However for X_B, an LHD applied to the feature values would undersample task dense regions and oversample sparse

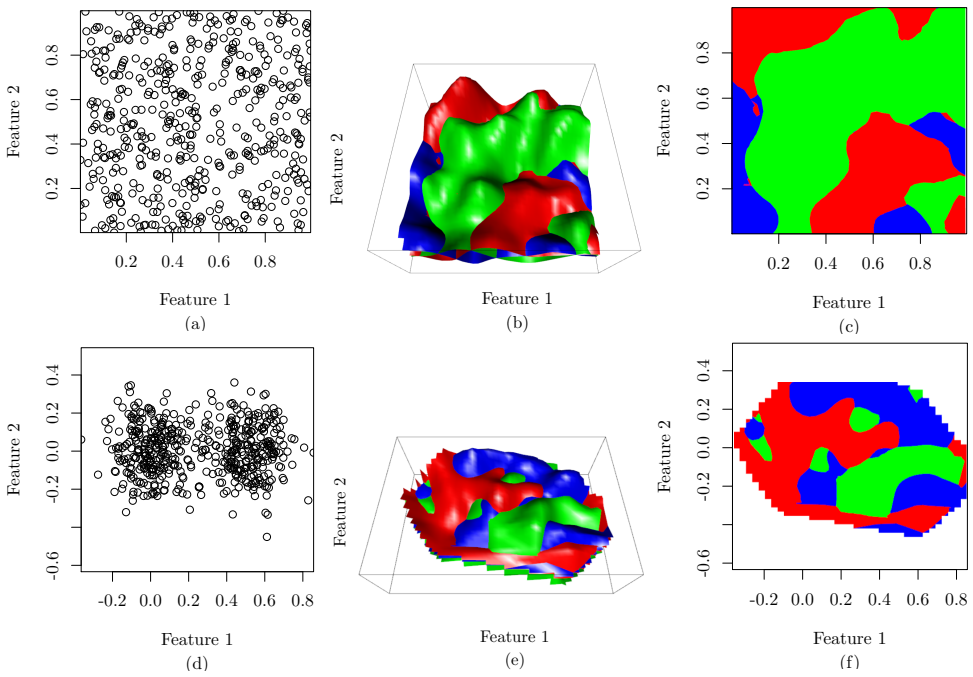


Figure 4.3: The upper row is for uniform problem instances and the lower is for bimodal. Figures (a),(d) show the sets of task features with randomly distributed values X_U (a) and X_B (d). (b),(e) show one example of the performance surfaces (generated by a bicubic spline interpolation) for three tools ζ_1, ζ_2, ζ_3. (c),(f) show the true optimal mapping from features to best of the three tools $S(x) = \text{argmax}_a \zeta_{a,t}$.

regions. Applying the LHD to the ranks results in hypercube divisions that are narrower/wider in dense/sparse regions. The tasks with a rank nearest to the Latin Hypercube points are selected to be included in the design. As with the sequential methods, a Gaussian Process with the squared exponential kernel is used to predict the expected performance at all tasks. The best predicted tool is chosen for each task in the mapping and the true opportunity cost is then measured. A new random design is chosen for every new budget and the performance predictions are re-calculated, therefore this is not a sequential method.

4.4.3 Results

Figure 4.4 compares the opportunity cost for various budget sizes for different sampling policies and both task feature distributions. On average, the REVI policy provides the best mapping for both task distributions and all budget sizes. The NEVI and EVI policies make the assumption that maximizing the single task expected

Table 4.1: Final Average Opportunity Cost ± std. err. for different sampling policies. *Random* is the performance of a mapping that picks a random tool for each task. *Single Best* is the performance of the single truly best tool.

	Uniform		
Tools	3	5	8
Random	427.6 ± 0.06	583.8 ± 0.06	712.3 ± 0.07
Single Best	325.8 ± 0.31	448.4 ± 0.4	542.6 ± 0.46
LHD	15.06 ± 0.31	21.95 ± 0.4	26.44 ± 0.46
EVI	1.87 ± 0.06	2.12 ± 0.06	1.92 ± 0.07
NEVI	1.69 ± 0.05	1.78 ± 0.04	1.54 ± 0.04
REVI	1.61 ± 0.04	1.71 ± 0.04	1.46 ± 0.03
	Bimodal		
	3	5	8
Random	430.3 ± 0.03	583.4 ± 0.03	710.6 ± 0.04
Single Best	272.7 ± 0.23	377 ± 0.24	455 ± 0.25
LHD	10.13 ± 0.23	14.11 ± 0.24	17.1 ± 0.25
EVI	0.8 ± 0.03	0.83 ± 0.03	0.9 ± 0.04
NEVI	0.69 ± 0.02	0.7 ± 0.02	0.68 ± 0.02
REVI	0.63 ± 0.02	0.69 ± 0.02	0.64 ± 0.02

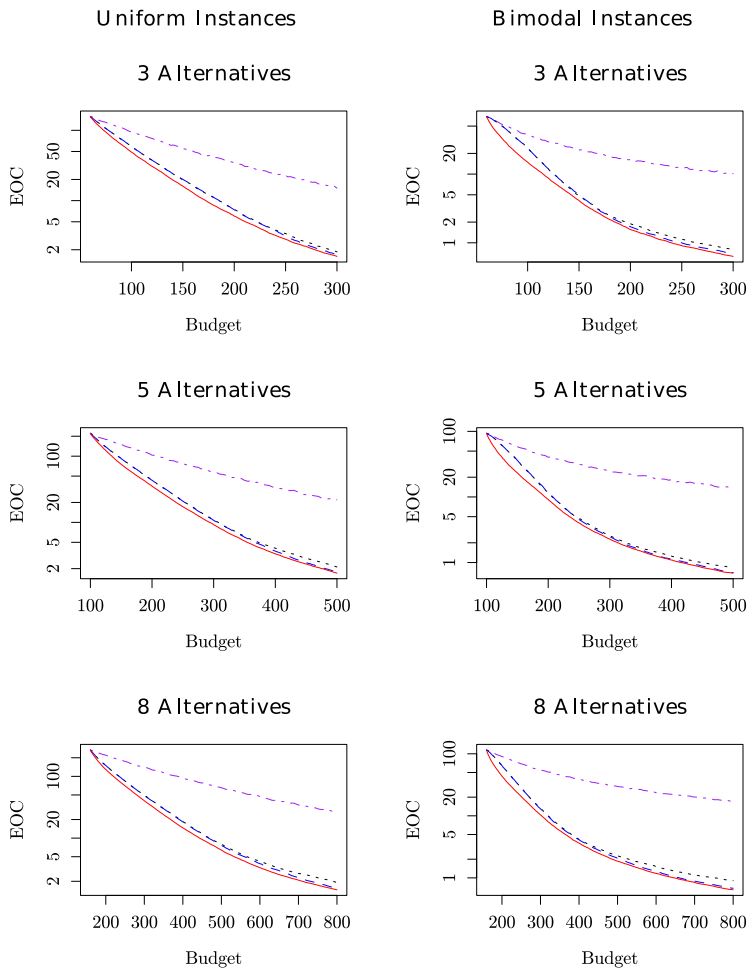


Figure 4.4: Opportunity cost of different sampling policies for various budget sizes averaged over 400 runs. In all plots: pink (dot-dash) is Latin Hypercube, black (dotted) is EVI, blue (dashed) is NEVI, and red (solid) is REVI. For all budgets, number of tools and feature distributions, the REVI policy produced the designs with the lowest opportunity cost on average.

improvement also maximizes the global expected mapping improvement. This may be approximately true in the uniform case where the effects of correlation are similar for most tasks. However, this assumption is less true in the bimodal case where there is greater variation in task density and therefore the expected improvement due to covariance varies more between tasks. In the bimodal case, after the initial design, we see a divergence in average opportunity cost between the REVI policy and the NEVI or EVI policies. At the initial stages, NEVI and EVI are more likely to allocate samples to unsampled outlying tasks providing smaller gains to portfolio

performance, whereas samples allocated by REVI or the Latin Hypercube do account for task density. After the outliers have been sampled, the efficiency of NEVI and EVI improves.

EVI assumes that the noise variance is negligible compared to the posterior variance for the (task, tool) pair that maximizes the expected improvement. This assumption becomes less true as the budget size increases and posterior variance for even the maximizing (task, tool) pair reduces and noise becomes non-negligible. Therefore, in all cases, we see a slight divergence in EOC between EVI and NEVI for large budget sizes.

The final opportunity cost and standard errors are reported in Table 4.1. As the number of tools increases, the opportunity cost for LHD increases whereas the sequential policies do not increase, suggesting that the policies given here scale with the number of tools and budget size much more favorably than the non-sequential design. In all cases the REVI policy produced the best performing mappings, and all policies were significantly better than the Latin Hypercube Designs with equivalent budget.

In the uniform features case, X_U, taking the final EOC of latin hypercube as a baseline level, the REVI policy achieved the same EOC in 49%, 53% and 58% of the sampling budget for 3, 5, and 8 tools respectively. The bimodal features experiment was 62%, 65% and 67%, respectively.

4.4.4 The Early/Tardy Machine Scheduling Problem

This dataset comes from a study of scheduling problem structure and heuristic performance [Smith-Miles et al., 2009] and consists of $M = 300$ problem instances and performance measurements for $A = 2$ two scheduling heuristics. The data was generated considering a single machine and a queue of jobs each with earliness and tardiness penalties, a processing time and a due date. The machine processes one job at a time and a scheduling heuristic determines which and when of the remaining jobs to process next whenever the machine is free. Processing a job such that it completes before it's due date yields an early penalty and a tardiness penalty for jobs completed after their due dates. The performance is the sum of penalties over the

range of jobs in an instance. A problem instance is a set of 100 randomly generated jobs, with set distributions of the penalties, processing times and due dates. Given a problem instance of 100 jobs, the features used for prediction were (1) tardiness factor (TF): the ratio of mean due date and sum total processing time, (2) due date range factor (DDR): ratio of range of due dates and sum processing time and (3) penalty ratio (PR): the largest ratio tardy to early penalties in the set of jobs.

The 300 instances were made by setting a desired tardiness factor to one of 6 levels from 0.2 to 1, due date range to one of 5 levels from 0.2 to 1, and penalty ratio to one of 10 levels from 1 to 10. In total, creating $6 \times 5 \times 10 = 300$ instances. The levels are equally spaced therefore problem instances are uniformly distributed.

The two heuristics applied were (1) Earliest Due Date: jobs are sorted by their due date, and process each job at a time determined using an optimal idle time insertion algorithm. And (2) Shortest Processing Time: jobs are sorted by their processing time in ascending order and the same optimal idle time insertion algorithm. Output values were centred and scaled.

We apply random sampling, NEVI, and REVI. The Gaussian process used a squared exponential kernel with length scale corresponding to each feature were set to half the largest value in the range of features, that is 0.5, 0.5, 5 for TF, DDR and PR respectively. The signal variance parameter was set to 1 (dataset Y values are scaled and centered), and the noise varaince set to 0.0001, or $1/100^{th}$ of the signal variance. All methods are initialized with $n_0 = 20$ samples per algorithm and sequential allocation is applied up to $N = 100$ samples. We record the opportunity cost as before and results are plotted in Figure 4.5.

As in the synthetic experiments, REVI converges fastest followed by NEVI and then uniform sampling. The distribution of tasks is uniform and equally spaced therefore there is less advantage to using REVI. To achieve the same level of opportunity cost as random sampling with 100 points, NEVI used 48 points while REVI used 46 points, saving 52% and 54% respectively. All methods found that 230 of the 300 problem instances are best served with the EDD method.

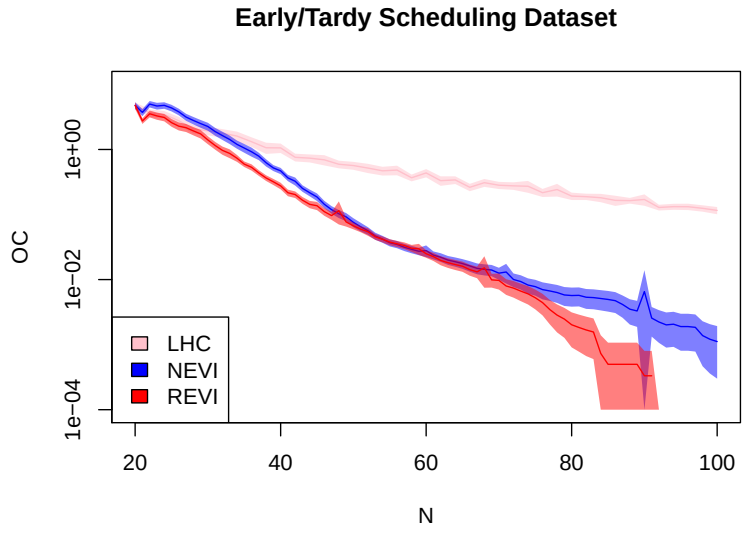


Figure 4.5: We apply REVI and NEVI and uniform random sampling. Shaded regions show two standard errors.

4.5 Conclusion

In this article, we extended the typical ranking and selection problem such that the performance of an alternative/tool may be described as a function over some input feature space, and the goal is to efficiently learn which tool performs best for each of a given set of tasks characterized by points in feature space. This has many applications, including algorithm selection, where we are given a set of problem instances and would like to learn which algorithm is best for each problem instance, or in personalized medicine where one must efficiently identify a mapping from patient characteristics to most effective treatment.

We proposed the Regional Expected Value of Improvement (REVI) policy which samples in a way that maximizes the expected increase in predicted performance over all the tasks. This method is myopically optimal by construction and also asymptotically optimal. We also proposed the NEVI and EVI sampling strategies that make some simplifying assumptions and no longer have the myopic optimality property, however they reduce the computational complexity and memory requirement and performed significantly better than Latin Hypercube Design in our experiments.

Chapter 5

Continuous Task Conditional Ranking and Selection

5.1 Introduction

In this chapter we extend the work of Chapter 4 by allowing the set of tasks to be continuous. The summation over tasks is replaced by an integral over tasks which must be approximated. We consider a simple Monte-Carlo approach with the aim of trading off computational overhead with bias in the estimated value of information. We also compare to the Gap-Stepwise Uncertainty Reduction algorithm [Hu and Ludkovski, 2015] observing significantly faster optimization with little or no added computational cost.

In the previous chapter, it was assumed that problem instances are given upfront. However, in simulation optimization, we may specifically choose a particular simulation setting (task) from a continuous range of settings. A user may desire the bespoke best input for each possible simulator setting.

5.2 Problem Definition

There exists a continuous distribution of tasks with features $x \in X \subset \mathbb{R}^D$, and the distribution $\mathbb{P}[x]$ of tasks in the feature space is specified. There exists a discrete set of tools A. Executing a given tool $a \in A$ on a given task $x \in X$

yields a stochastic performance measurement $Y_a(x) = f(a, x) = \theta_a(x) + \epsilon$ where $\theta_a(x) : \mathbb{R}^D \to \mathbb{R}$ is a deterministic latent function and $\epsilon \sim N(0, \sigma_{\epsilon,a})$ is independent and identically distributed noise with known variance (which in practice is measured). The objective of the user is to find a classifier, or mapping, from problem instance to the best alternative $S(x) : X \to A$ that approximates $S^*(x) = \underset{a}{\text{argmax}}\ \theta_a(x)$ given a finite budget of N performance measurements. The aim is to find a mapping that maximizes the overall expected performance across all problem instances

$$\text{True Performance} = F = \int_X \theta_{S(x)}(x)\mathbb{P}[x]dx. \tag{5.1}$$

Individual performance measurements may be collected sequentially so that after n measurements the user may determine the alternative and problem instance for the $(n+1)^{th}$ sample.

If there was only a single task, the problem reduces to finding the tool a with the highest θ_a, a standard ranking and selection problem. The formulation above extends ranking and selection to a continuous range of correlated ranking and selection tasks. The distribution of tasks $\mathbb{P}[x]$ ensures that frequent tasks (high $\mathbb{P}[x]$) contribute to performance more than rare tasks (low $\mathbb{P}[x]$) and as such a user would prefer to have a more accurate mapping $S(x)$ for common tasks. We refer to $\mathbb{P}[x]$ as the task feature density however, without loss of generality, any user defined weight function, $W(x)$, can be incorporated to give priority to particular tasks x.

5.3 Sampling Methods

We propose to use independent Gaussian process regression models for each a to predict expected performance $\theta_a(x)$ given task features x.

Given a dataset of (task, tool, performance) triplets $(x^1, a^1, y^1), .., (x^n, a^n, y^n)$, we define the subset of (task, performance) pairs associated with tool a as X_a^n, Y_a^n. We define the sigma algebra generated by the data as $\mathcal{F}^n$ that defines the sequence of filtrations used to condition the distributions of $\theta_1(x), ..., \theta_A(x)$ on the data. For a single tool a, a user defines prior mean and covariance functions $\mu_a^0(x)$, $k_a^0(x, x')$. After observing data the posterior distribution of $\theta_a(x)$, the Gaussian processes

Regression, is given by

$$\mathbb{E}[\theta_a(x)|\mathcal{F}^n] = \mu_a^0(x) + k_a^0(x, X_a^n)\big(k_a^0(X_a^n, X_a^n) + \mathbb{I}\sigma_{a\epsilon}^2\big)^{-1}(Y_a^n - \mu^0(X_a^n)), \tag{5.2}$$

$$\mathrm{Cov}[\theta_a(x), \theta_a(x')|\mathcal{F}^n] = k_a^0(x, x') - k_a^0(x, X_a^n)\big(k_a^0(X_a^n, X_a^n) + \mathbb{I}\sigma_{a\epsilon}^2\big)^{-1}k_a^0(X_a^n, x'), \tag{5.3}$$

where $k_a^0(X_a^n, X_a^n)$ is the $n \times n$ matrix with elements determined by evaluating the prior covariance function between all possible pairs of elements in X_a^n and $\mathbb{I}$ is the $n \times n$ identity matrix. We can then construct a mapping for any task x by taking the tool with the best predicted performance

$$S(x) = \underset{a}{\mathrm{argmax}}\ \mu_a^n(x)$$

and the model estimate of performance, P^n, of such a mapping is computed by

$$P^n = \int_X \mu_{S(x)}^n(x)\mathbb{P}[x]dx = \int_X \max_a \mu_a^n(x)\mathbb{P}[x]dx \tag{5.4}$$

which we aim to optimize as a surrogate for optimizing F. However this cannot be written analytically for arbitrary $\mathbb{P}[x]$, we will discuss approximations below in Section 5.3.2. Given the performance prediction, the Value of Information in this application is given by

$$\begin{aligned}
&\mathcal{I}(x, a) && (5.5)\\
&= \quad \mathbb{E}_{y^{n+1}}[P^{n+1} - P^n]\\
&= \quad \mathbb{E}_{y^{n+1}}\left[\int_X \left(\max_{a'} \mu_{a'}^{n+1}(x) - \max_{a''} \mu_{a''}^n(x)\right)\mathbb{P}[x]dx \big| \mathcal{F}^n, (x, a)^{n+1} = (x, a)\right]\\
&= \quad \int_X \left(\mathbb{E}_{y^{n+1}}\left[\max_{a'} \mu_{a'}^{n+1}(x) - \max_{a''} \mu_{a''}^n(x) \big| \mathcal{F}^n, (x, a)^{n+1} = (x, a)\right]\right)\mathbb{P}[x]dx
\end{aligned}$$

and we refer to $\mathcal{I}(x, a)$ as the *improvement*. At time n, $\mu_a^n(x)$ is known, for a given $(x, a)^{n+1}$ the Gaussian process gives a predictive distribution of y^{n+1} thus the statistical distribution of $\mu_a^{n+1}(x)$, P^{n+1} and the expectation $\mathcal{I}(x, a)$ can be calculated, we next derive these distributions.

Given only the n samples and the user's choice of $(x, a)^{n+1} = (x, a)$, the prior

predictive distribution of the next observation is

$$y^{n+1} \sim N(\theta_a(x), \sigma^2_{a,\epsilon}) = N(\mu^n_a(x), k^n_a(x,x) + \sigma^2_{a,\epsilon}) \tag{5.6}$$

Equation 5.2 gives the formula for the current posterior mean after n samples, the formula for the posterior after one sample allocated to tool a is

$$\mathbb{E}[\theta_a(x)|\mathcal{F}^1] = \mu^0_a(x) + \frac{k^0_a(x, x^1_a)}{k^0_a(x^1_a, x^1_a) + \sigma^2_{a\epsilon}}(y^1_a - \mu^0(x^1)). \tag{5.7}$$

Replacing $\mu^0_a(x)$ and $\mu^1_a(x)$ with $\mu^n_a(x)$ and $\mu^{n+1}_a(x)$, and the same for covariances, yields a consistent updating formula for the posterior mean. For each new observed y^{n+1}, only the sampled tool a^{n+1} is changed,

$$\mathbb{E}[\theta_a(x)|\mathcal{F}^{n+1}] = \mu^{n+1}_a(x) = \begin{cases} \mu^n_a(x) + \frac{k^n_a(x, x^{n+1}_a)}{k^n_a(x^{n+1}_a, x^{n+1}_a) + \sigma^2_{a\epsilon}}(y^{n+1}_a - \mu^n(x^{n+1})) & a = a^{n+1} \\ \mu^n_a(x) & a \neq a^{n+1} \end{cases} \tag{5.8}$$

Combining the distribution of y^{n+1} with the updating formula for the mean yields the predictive distribution

$$\mu^{n+1}_a(x) = \mu^n_a(x) + \tilde{\sigma}^n_a(x; x^{n+1})Z \tag{5.9}$$

where $Z \sim N(0,1)$ is the stochastic z-score of y^{n+1} value on its predictive distribution and

$$\tilde{\sigma}^n_a(x; x^{n+1}) = \frac{k^n_a(x, x^{n+1})}{\sqrt{k^n_a(x^{n+1}, x^{n+1}) + \sigma^2_{a,\epsilon}}} \tag{5.10}$$

is a deterministic function of x, parametrized by x^{n+1}, that is an additive update to the posterior mean caused by the new sample and the size of the update is scaled by the stochastic Z. The improvement $\mathcal{I}(x,a)$ is an integral over all $x \in X$. Next, we calculate the improvement for a single task x in the integral due to the new sample $(x^{n+1}, a^{n+1}, y^{n+1})$. Given task x, we define the best predicted tool as $\max_a \mu_a(x) = \mu^n_{(1)}(x)$ and for the following equation we drop (x). The expected

improvement in estimated performance is given by the integrand of Equation 5.5

$$\begin{aligned}
\mathbb{E}\big[\max_{a'} \mu^{n+1}_{a'} - \mu^n_{(1)}\big] &= \mathbb{E}\big[\max\big\{\max_{a' \neq a^{n+1}} \mu^n_{a'} - \mu^n_{(1)}, \mu^n_{a^{n+1}} - \mu^n_{(1)} + Z\tilde{\sigma}_{a^{n+1}}\big\}\big] \\
&= \begin{cases} \mathbb{E}[\max\{\Delta_a, Z\tilde{\sigma}_a\}] & a^{n+1} = (1) \\ \mathbb{E}[\max\{0, \Delta_a + Z\tilde{\sigma}_a\}] & a^{n+1} \neq (1) \end{cases} \\
&= \Delta_a \Phi\big(\Delta_a/\tilde{\sigma}_a\big) - \tilde{\sigma}\phi\big(\Delta_a/\tilde{\sigma}_a\big)
\end{aligned}$$

where $\Delta_a = -|\mu^n_a(x) - \max_{a \neq a^{n+1}} \mu^n_a(x)|$, $\tilde{\sigma}_a = \tilde{\sigma}^n_a(x, x^{n+1})$ and $\phi(x)$ and $\Phi(x)$ are the standard normal density and cumulative functions respectively and is the EI acquisition function. Here, we are measuring the expected improvement in the mean of one Gaussian process over the mean of other independent Gaussian processes. Finally the improvement across all $x \in X$ is the integral over x and y^{n+1},

$$\mathcal{I}(x, a) = \int_{x' \in X} \big(\Delta_a(x')\Phi\big(\Delta_a(x')/\tilde{\sigma}^n_a(x', x)\big) - \tilde{\sigma}^n_a(x', x)\phi\big(\Delta_a(x')/\tilde{\sigma}^n_a(x', x)\big)\big)\,\mathbb{P}[x']dx'.$$

In the following subsections, we propose several sampling policies for determining x^{n+1} and a^{n+1} based on the above value of information calculation.

5.3.1 Local Expected Value of Improvement

As a cheap simple heuristic for maximizing $\mathcal{I}(x, a)$ we propose the Local Expected Value of Improvement (LEVI) procedure which determines the next (task, tool) pair to sample based on the improvement that the sampled task alone contributes to the global improvement $\mathcal{I}(x, a)$. The LEVI method allocates the sample to maximize the argument to the VoI integral:

$$\begin{aligned}
\text{LEVI}(x, a) &= \mathbb{E}\big[\max_{a'} \mu^{n+1}_{a'}(x) - \max_{a''} \mu^n_{a''}(x)\big]\mathbb{P}[x] \\
&= \Big(\Delta_a(x)\Phi(\Delta_a(x)/\tilde{\sigma}_a(x; x)) - \tilde{\sigma}_a(x)\phi(\Delta_a(x)/\tilde{\sigma}_a(x; x))\Big)\mathbb{P}[x]
\end{aligned}$$

The LEVI method treats x as a single ranking and selection task and samples the single task that has the joint largest improvement and probability $\mathbb{P}[x]$. See Figure 5.1 for an illustration. It tends to allocate a sample to (task, tool) pairs for which the

difference in prediction between the chosen tool and the best tool, $\Delta_a(x)$ is small, where the possible change in prediction, $\tilde{\sigma}_a(x;x)$, is high and where the task density, $\mathbb{P}[x]$, is high. Let n_a be the number of the n samples so far allocated to tool a. Each call to LEVI(x,a) requires a call to all posterior means $\mu_1^n(x),...,\mu_A^n(x)$ which have complexity $O(n_1),..,O(n_A)$ and one call to the posterior covariance $k_a^n(x,x)$ which has complexity $O(n_a^2)$. This leading complexity is equivalent to one call for the EGO expected improvement, however for this problem the acquisition function must be optimized once for each tool in order to allocate a single sample. Optimization of LEVI(x,a) can be done by many methods and in Section 5.4 we use the squared exponential kernel that gives smooth predictions and Nelder-Mead algorithm with multiple starts.

When allocating samples according to LEVI, given an infinite sampling budget the true optimal mapping $S^*(x)$ will be determined perfectly.

5.3.2 Regional Expected Value of Improvement

The Regional Expected Value of Improvement (REVI) sampling procedure determines the next (task, tool) pair to sample based on a Monte-Carlo estimate of the global improvement $\mathcal{I}(x,a)$ and reduces to the REVI method of the previous Chapter when tasks are discrete. We use a Monte-Carlo integral over a fixed set of N_X tasks, $X_{MC}=\{x_1,...,x_{N_X}\}$, that are randomly distributed according to $\mathbb{P}[x]$, the integral becomes

$$\begin{aligned}\mathcal{I}(x,a) &\approx \hat{\mathcal{I}}(x,a) \\ &= \frac{1}{N_X}\sum_{x_i\in X_{MC}} \mathbb{E}\big[\max_a \mu_a^{n+1}(x_i) - \max_{a'}\mu_{a'}^n(x_i)\big] \\ &= \frac{1}{N_X}\sum_{x_i\in X_{MC}} \Delta_a(x_i)\Phi(\Delta_a(x_i)/\tilde{\sigma}_a(x_i;x)) - \tilde{\sigma}_a(x_i;x)\phi(\Delta_a(x_i)/\tilde{\sigma}_a(x_i;x)).\end{aligned} \tag{5.11}$$

The above integral $\hat{\mathcal{I}}(x,a)$ gives an estimate of the total improvement $\mathcal{I}(x,a)$. LEVI(x,a) gives the single point improvement therefore LEVI$(x,a)V_X$ is also an estimate of the overall improvement where $V_X=\int_x dx$ is the volume of the task space. Combining these two estimates of improvement weighted according their

sample sizes yields

$$\text{REVI}(x,a) = \frac{1}{N_X+1}\left(N_X\hat{\mathcal{I}}(x,a) + \text{LEVI}(x,a)V_X\right). \tag{5.12}$$

The REVI method allocates samples to (task, tool) pairs for which the difference between the chosen tool and the best remaining tool is close across similar tasks, where the prediction uncertainty is high across those tasks and where the task density is high across those tasks, see Figure 5.1. After each new sample, the set of fixed points, X_{MC}, are regenerated such that the mapping does not overfit to a single discretization. N_X may be seen as a parameter to be chosen by the user, and setting $N_X = 0$ the REVI method becomes equivalent to LEVI method. In Section 5.4 we choose $N_X = n$ such that the discretization density grows with the sample density.

A single call to $\text{REVI}(x,a)$ requires one call to $\text{LEVI}(x,a)$ and also the means of the X_{MC} tasks for all A tools and also all posterior covariances $k_a^n(X_{MC},x)$ to calculate $\tilde{\sigma}_a^n(X_{MC};x)$. The means for all N_X tasks and tools, $\mu_1^n(X_{MC}),...,\mu_A^n(X_{MC})$ can be precomputed and cached between $\text{REVI}(x,a)$ calls. Also the final terms of the matrix multiplication in Equation 5.3 for the posterior covariances may be precomputed and cached reducing each $k_a^n(x_i,x)$ call from $O(n_a^2)$ to $O(n_a)$,

$$k_a^n(x^{n+1},X_{MC}) = k_a^0(x^{n+1},X_{MC}) - k_a^0(x^{n+1},X_a^n)\underbrace{\left(k_a^0(X_a^n,X_a^n)\right)^{-1}k^n(X_a^n,X_{MC}).}_{\text{compute once and store for given } X_{MC}}$$

Therefore after the first call, each additional call to $\text{REVI}(x,a)$ has leading order complexity $O(n_a^2 + N_X n_a)$. Defining $\bar{n} = \max_a n_a$ and assuming $N_X = n \approx A\bar{n}$ the leading complexity is $O(\bar{n}^2)$ which is equivalent to $\text{LEVI}(x,a)$ and EGO. Again optimization of $\text{REVI}(x,a)$ can be done by many methods and in Section 5.4 we use the same Nelder-Mead algorithm with multiple starts.

If the task distribution is discrete, $X = \{x_1,..\}$ and $\mathbb{P}[x_i] = p_i$, then $\lim_{N_x\to\infty}\hat{\mathcal{I}} = \mathcal{I}(x_i,a)$ may be computed exactly by summing tasks weighted according to their probabilities and the $\text{LEVI}(x,a)$ term will vanish from $\text{REVI}(x,a)$. If all tasks were equally likely $p_i = 1/|X|$, the REVI method is equivalent to the REVI method given in Chapter 4.3 which is asymptotically and myopically optimal.

In Section 5.4 we show that the REVI method significantly outperforms LEVI and another method from the literature in our tests.

5.3.3 Neighbors Only Regional Expected Value of Improvement

The Monte-Carlo integral $\hat{\mathcal{I}}$ with N_X samples requires computing $\tilde{\sigma}^n_a(x_i, x)$, the change in the predicted performance of an task $x_i \in X_{MC}$ caused by the new sample at x, which requires computing $k^n_a(x_i, x)$. When covariance between x_i and x is low, the task x_i expected improvement is very small. As a result the summation in Equation 5.11 may be composed of a few large terms and many small negligible terms. To improve efficiency, one may use importance sampling, for example with a stationary kernel, setting $X_{MC} = \{x + \delta_i | i = 1, .., N_X\}$ with random $\delta_i \sim \mathcal{N}(0, 1)$, would significantly reduce error in $\hat{\mathcal{I}}(x, a)$. However in our experiments this lead to slower computation, it's not possible to precompute means and partial covariances, and the REVI(x, a) is very rugged and not easily optimized, fewer computations can be cached.

Instead we propose the Neighbors Only Regional Expected Value of Improvement method (N-REVI). The Monte-Carlo tasks, X_{MC}, are generated as with REVI, however, only tasks whose prior covariance is above a threshold are included in the Monte-Carlo integral. By filtering points according to the cheaply computed prior covariance, we can avoid the costly matrix multiplication in posterior covariance in Equation 5.3 for points that do not significantly contribute to $\hat{\mathcal{I}}$,

$$k^n_a(x^{n+1}, X_{MC}) = \underbrace{k^0_a(x^{n+1}, X_{MC})}_{\text{if very small}} - \underbrace{k^0_a(x^{n+1}, X^n_a)\big(k^0_a(X^n_a, X^n_a)\big)^{-1} k^n(X^n_a, X_{MC})}_{\text{then don't compute}}.$$

The acquisition function is given by

$$\begin{aligned} \mathcal{I}(x, a) &\approx \hat{\mathcal{I}}_N(x, a) \\ &= \frac{1}{N_X} \sum_{x_i \in X_{MC}} \mathbb{E}\big[\max_a \mu^{n+1}_a(x_i) - \max_{a'} \mu^n_{a'}(x_i)\big] \underbrace{\mathbb{1}\{k^0_a(x_i, x) > \delta\}}_{\text{filtering}} \end{aligned}$$

where $\mathbb{1}\{\}$ is the indicator function returning unity if the argument is true. Otherwise

the N-REVI acquisition function is the same as the REVI function

$$\text{N-REVI}(x,a) = \frac{1}{N_X+1}\left(N_X\hat{\mathcal{I}}_N(x,a) + V_X LEVI(x,a)\right)$$

After each new sample the X_{MC} is randomly regenerated. This sampling method has the same advantages as the LEVI and REVI methods however it gives a similarly accurate estimate of improvement to REVI while significantly reducing computational cost. The disadvantage is that the estimate is not consistent and biased, excluding outlying tasks will underestimate $\mathcal{I}(x,a)$,

$$\lim_{N_X\to\infty}\hat{\mathcal{I}}_N(x,a) \neq \mathcal{I}(x,a).$$

This compromise is required to achieve caching of computations with large sample sizes N_X. The N-REVI method has two parameters, N_X and δ. When using a stationary kernel, $k^0(x,x') = h(|x-x'|)$, such as the Matern class of kernels or squared exponential, points may be filtered according to $|x-x'| < r$ and the two free parameters are N_X and r. We show in Section 5.4 that computation time of N-REVI is drastically less than REVI without any significant loss in performance. By setting $N_X = 0$ or $\delta = \infty$ or $r = 0$, the N-REVI method is equivalent to LEVI, also by setting $\delta = 0$ or $r = \infty$ the N-REVI method is equivalent to REVI, therefore computational complexity is between REVI and LEVI depending on the choice of parameters, the kernel and $\mathbb{P}[x]$.

5.4 Numerical Experiments

We perform three sets pf experiments. First, we test N-REVI for a range of parameters N_x and r. Second we compare N-REVI, REVI, and LEVI against another method from the literature, comparing both convergence and computation time. Third, we investigate the effect of the design of X_{MC} in N-REVI.

We apply our methods to a set of four ($A = 4$) synthetic functions sampled from a Gaussian process prior with a squared exponential kernel, $\theta_1(x), ..., \theta_4(x) \sim GP(\mu^0(x) = 0, k^0(x,x') = \exp(-0.5||x-x'||_2^2/7^2)$. The input domain is $X = [0,100]^2$

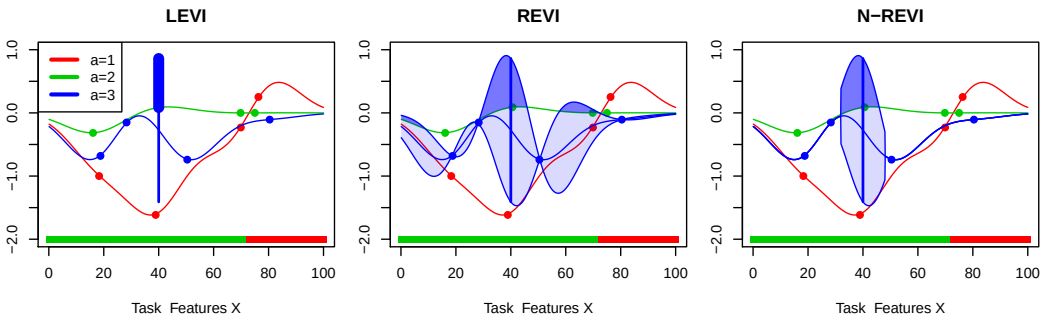


Figure 5.1: How the methods quantify acquisition value at $(x, a) = (40, 3)$. Left: LEVI computes how much the the current GP will improve upon the best at the current task (thick line). Centre: REVI computes how much all tasks will improve (shaded). Right:N-REVI truncates the integration over tasks to save computation. The colour bar at the bottom of the plot is the current mapping $S(x)$

and the output domain is real numbers. In practice, for each function, a multivariate normal sample is produced from the prior evaluated at a discretized X and the sample is used to condition the continuous prior, yielding a continuous posterior mean. The posterior mean functions are used as synthetic functions $\theta_1, \ldots, \theta_4$ and plotted with the optimal mapping in Figures 5.2 (a) and (b). The noise is set to $\sigma^2_{a,\epsilon} = 0.1^2$ for all tools. Given these four functions we use two task distributions: a) A uniform distribution (UNI) $\mathbb{P}_U[x] = 1/10,000$, so that we can compare directly with the Gap-SUR method proposed by Hu and Ludkovski [2015], and b) A "Wedge" task distribution, $\mathbb{P}_W[x] = x_1/5 * 10^5$, linearly increasing only in the x_1 direction that is adversarially designed to show the benefits of accounting for local task density and correlation.

To initialize sampling, a random design is used as described below with 10 samples per tool, 40 samples in total, Figure 5.2 e) show one such the sample allocation for the wedge distribution. In practice these samples would be used to learn hyperparameters for the Gaussian Processes however we treat the hyperparameters as constant throughout sampling for all methods such that the only difference between all methods is the acquisition function. Figures 5.2 c) and d) give the estimated performance functions and mapping after fitting a GP to the initial 40 samples.

After the initialization, the sequential methods are then applied until 200 samples in total have been allocated. Figure 5.2 f) shows the LEVI(x, a) after the

initialization and it can be seen that LEVI prioritizes high $\mathbb{P}[x]$, which is on the boundary of the domain. However REVI(x, a) shown in Figure 5.2 g) does not prioritize the highest $\mathbb{P}[x]$ but is closer to the mean x where the regional benefit of a new sample is greater as there are many tasks surrounding the new sample.

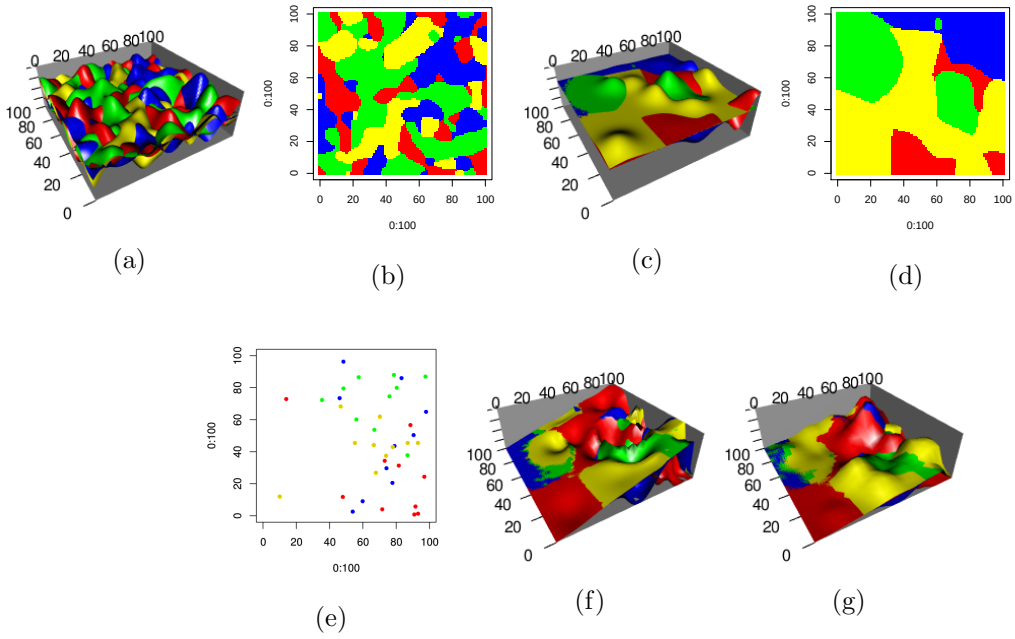


Figure 5.2: In all plots: x and y axis are task features, each color represents an tool. (a),(b) the true performance functions and the true optimal mapping. (c)(d) the predicted functions and mapping after the initial 40 samples when using the wedge distribution. (e) the allocation of the 40 samples. (f) the LEVI(x, a) and (g) REVI(x, a), note how LEVI(x, a) is greatly influenced by the $\mathbb{P}[x]$ and peaks on the edge of the domain, whereas REVI(x, a) is smoother and peaks further within the domain where more tasks can benefit from a new measurement.

To measure the quality of the mapping, for each budget size, $N \in [40, ..., 200]$, the mapping $S(x) = \underset{a}{\text{argmax}}\ \mu_a^N(x)$ is evaluated for 1000 points in the space X and the total opportunity cost is recorded. The 1000 test tasks, X_T, are distributed according to $\mathbb{P}_U[x]$ and $\mathbb{P}_W[x]$ for uniform and wedge respectively. For each distribution they are generated once and the same tasks are used to measure opportunity cost for all experiments. They are never used for the random sampling method nor

in X_{MC}, it is a left-out test set.

$$\text{Opportunity Cost} \quad = \quad \sum_{x_i \in X_T} \max_a \theta_a(x_i) - \theta_{S(x_i)}(x_i) \tag{5.13}$$

To measure the computational cost of the methods, the time taken to optimize the acquisition function is measured, all parameters for the Nelder-Mead algorithm are kept constant across all experiments $\alpha = 1$, $\beta = 0.5$, $\gamma = 2$ with 15 random restarts seeded by points sampled from the appropriate $\mathbb{P}[x]$. Each sampling method is applied 400 times with different initial samples and noise values. For the REVI methods we run experiments for $N_X \in \{0.5n, n, 2n, 4n\}$, and $r \in \{1l_X, 2l_X, 3l_X, \infty\}$ where $l_X = 7$ is the kernel length scale. In order to investigate the influence of the varying sample size N_X and overfitting, we also run experiments with REVI where the Monte-Carlo sample size is constant: $N_X = 100$ and $N_X = 200$ and where the MC samples are fixed for all time or randomized at each time step which we refer to as "jittering". We compare the final opportunity cost of these methods with original REVI where the MC sample size grows as the budget is consumed.

5.4.1 Alternative Methods

Random Sampling

Given a budget of N samples, N tasks are chosen by Latin Hypercube (uniform case) and by sampling $\mathbb{P}_W[x]$ (wedge case). The points are randomly divided into A groups whose sizes differ by at most 1. For all (task, tool) pairs performance, $f(x, a) = \theta_a(x) + \epsilon$, is measured and a Gaussian Process is fitted to the resulting dataset. This method represents the naive approach that does not exploit past evaluations to inform new ones.

Gap-SUR

Hu and Ludkovski [2015] consider the same problem with a uniform task distribution. The proposed method allocates samples to the (task, tool) pair that maximizes the

reduction in the upper bound of the highest performing tool:

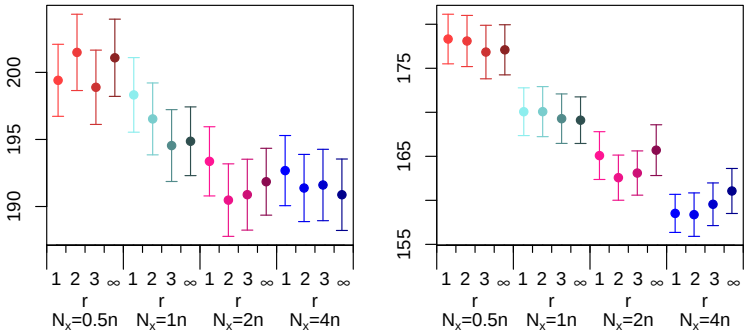
$$Gap - SUR(x,a) = \big(\ \mathbb{E}[\max_{a'} \theta_{a'}(x)|\mathcal{F}^n] - \mathbb{E}[\max_{a'} \theta_{a'}(x)|\mathcal{F}^{n+1}]\ \big)\mathbb{P}[x]$$

where $\theta_a \sim N(\mu^n(a), k_a^n(x,x))$ and $\mathcal{F}^{n+1}$ is the next filtration constructed by assuming the new sample will be equal to the current mean $(x^{n+1}, a^{n+1}, y^{n+1}) = (x, a, \mu_a^n(x))$. The maximum of multiple Gaussian random variables is calculated by approximating the maximum of any two Gaussian random variables as another Gaussian, for which first and second moments are known, and iterating over A. One evaluation of $GAP - SUR(x,a)$ requires one call each to $k_1^n(x,x), .., k_A^n(x,x)$, however once the posterior variances are known, evaluating $Gap - SUR(x,a)$ for all a with the same x is relatively cheap. Therefore we optimize the acquisition function over x once using Nelder-Mead with the same parameters as other methods and for each x, $Gap - SUR(x,a)$ is optimized over a by inspection, thereby reducing the number of posterior covariance calls.

5.4.2 Results

First we have examined the influence of the REVI parameters N_X (Monte Carlo sample size) and r (filtering radius) on the opportunity cost. Figure 5.3 shows the results for (a) the uniform distribution and (b) the wedge distribution. Increasing the N_X parameter significantly improves performance in all cases, and the improvement is larger for the wedge distribution than for the uniform distribution. Apparently larger N_X improves accuracy of the estimate of improvement more in a more complicated task distribution. Varying r from $r = 1$ to $r = \infty$ does not provide significant performance but does significantly increase computation time presented in Figure 5.4. As can be seen, LEVI is fastest, followed by $Gap - SUR$. REVI still scales linearly with the number of samples, but the slope is steeper than for the other methods. Standard REVI with $r = \infty, N_X = n$ has about the same runtime curve as N-REVI with $r = 1, N_X = 4n$. But as we have seen earlier in Figure 5.3, the latter version performs much better in terms of opportunity cost. Thus, for our final comparison with the other benchmark methods, we limit ourselves to N-REVI with

$r = 1, N_X = 4n$.



(a) UNI Final OC (b) Wedge Final OC

Figure 5.3: Opportunity cost of REVI depending on N_X and r at the end of the 200 samples taken.

Figure 5.5 displays how the opportunity cost decreases with the number of samples, for the different sampling methods. As expected, random sampling is significantly worse than all sequential methods for both task distributions. LEVI performs as good as or better than Gap-SUR, and as seen in Figure 5.4 is slightly faster to optimize in this implementation. In all benchmarks, N-REVI is significantly better than the local methods, LEVI and Gap-SUR, suggesting that even the simplest accounting of correlated tasks, $N_X = 4n$, $r = l_X$, does yield a significant performance benefit for increased computational cost. For most real world simulation applications the running time is dominated by the simulation, here we see that REVI improves performance and yet scales poorly to large sampling budgets N and computation time may balloon and assuming negligible computation overhead is unrealistic. Whereas N-REVI, in our experiments, has the performance benefits of REVI and scales to larger sampling budgets with the reduced risk of significantly growing run time.

The difference between the REVI and local methods is larger for the wedge distribution. The local methods prioritize the mode of $\mathbb{P}[x]$, which is at $x_1 = 100$ in the wedge case. The REVI methods prioritize where there are many correlating tasks, which will not be at $x_1 = 100$ due to lack of X_{MC} points to contribute to $\hat{\mathcal{I}}(x, a)$ for a sample near a boundary. Therefore REVI methods outperform LEVI even in the uniform case as LEVI has no incentive to avoid sampling near a boundary, tasks that yield only marginal global improvement.

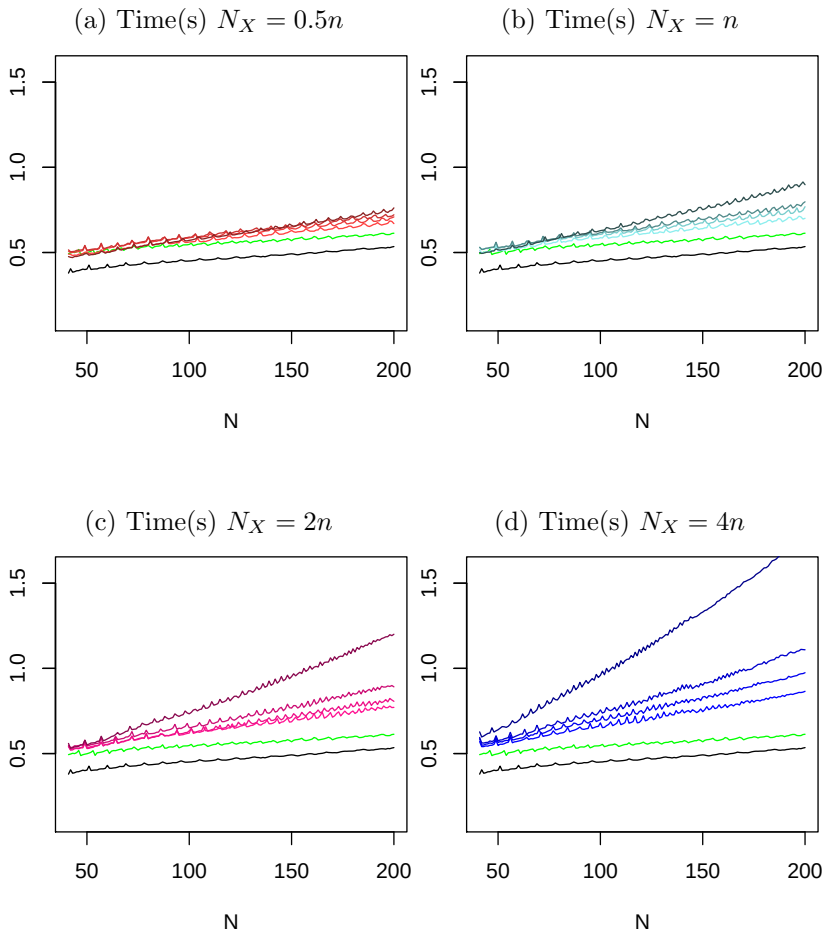
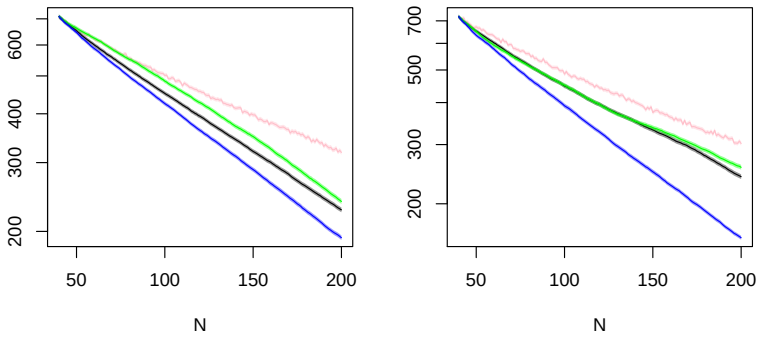


Figure 5.4: Optimization time in seconds per iteration for different sample sizes. Green is $Gap - SUR$, black is LEVI. N-REVI with r values increasing from light to dark. REVI corresponds to the darkest (most time consuming) lines of each plot, where $r = \infty$.

Finally, results comparing standard REVI ($N_X = n/2$) with REVI where X_{MC} is fixed and where X_{MC} is jittered (both $N_X = 100$) are given in Figure 5.6 a). Standard REVI performs worst presumably due to the small N_X for small budgets. This is followed by the fixed samples and then jittered samples suggesting the mapping $S(x)$ was overfitting to the fixed X_{MC} and therefore jittering is necessary. When the X_{MC} set is equal to the sampling budget of 200, Figure 5.6, there is no significant difference between the three versions, presumably there are not enough samples allocated to overfit, this may not be the case for larger budgets. Although not shown the standard REVI requires less computation while the sample size is still small.

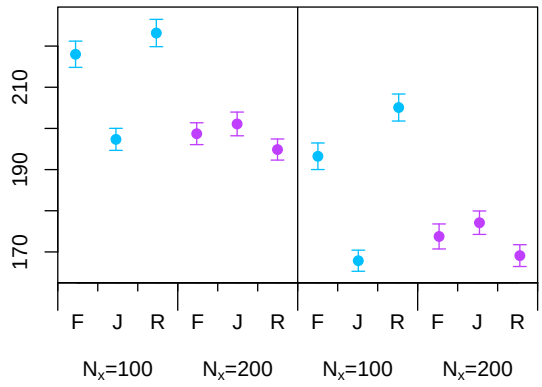


(a) UNI OC (b) Wedge OC

Figure 5.5: Opportunity cost over the course of the sampling, pink is random sampling, green is $Gap - SUR$, black is LEVI, and blue for N-REVI with $r = 1, N_X = 4n$.

5.5 Conclusion

We have considered the problem of finding the best of a finite set of tools for each task across a continuous task feature domain. We have proposed new myopic sampling methods that allow to efficiently derive an accurate mapping from task features to the best tool. In two synthetic benchmark problems, our methods show significantly better performance than random sampling as well as a recently published algorithm from the literature. Furthermore, we have compared various ways of numerically integrating the expected improvement over the task domain, and found that using a large sample size, but only taking into account samples with correlation yields the best results at lower computational cost. Future work includes applying the method to a real world task and providing a consistency guarantee.



(a) Final OC, UNI Wedge

Figure 5.6: Final opportunity cost, uniform problems left, wedge problems right, for fixed X_{MC} points (F), jittering X_{MC} points (J) and standard REVI (R). Blue for $N_X = 100$ and purple for $N_X = 200$ ($N_X = 0.5n$ and n for REVI).

Chapter 6

Multi-Task Conditional Bayesian Optimization

6.1 Introduction

In this chapter, we consider the problem where one is given multiple continuous optimisation problems (tasks) over a common domain where the learning about one function can inform other functions. Given a set of tasks that can be described by continuous features, and a tool with continuous parameters to be tuned, we would like to identify the unique optimal parameter setting for each task. This generalises the work in the previous chapters to both continuous task domain $x \in X$ and continuous tool parameters $a \in A$ (whereas a was previously an index). We can perform experiments to collect information, i.e., run the tool with a specific parameter setting on a specific task, and obtain a sample of a noisy performance measurement. Given a finite budget, the goal is to decide which sequence of experiments to perform that would allow us to construct the best mapping from task features to optimal tool setting.

This general problem occurs in many applications, including

- *Tuning of Optimisation Algorithms.* Many optimisation algorithms have continuous parameters that need to be tuned specific to the problem instance at hand. It is thus desirable to construct a mapping that suggests the best

parameter setting (tool) depending on features of the problem instance. In the machine learning community, training of deep neural networks requires setting the learning rate for gradient descent depending on the dataset and model. In the meta-learning community, much work has been done on deriving a mapping from problem features to best algorithm parameters based on a given set of performance data.

- *Dose-Finding Clinical Trials.* The dosage and the compound mixture of a drug may have a strong impact on a drug's effect, and different patients react differently to the drug. Our method can be used to design more efficient clinical trials to identify the best dosage or compound mixture for each patient, based on patient characteristics such as age or biomarker response. A related application has been considered by Krause and Ong [2011].

- *Online Operating Policies.* A complex system may best be operated in different ways depending on the context such as environmental conditions. For example, a factory may be using dispatching rules for real-time scheduling, and the dispatching rules have some parameters whose optimal setting depends on shop floor conditions such as utilisation level or product mix. Heger et al. [2016] use a large number of experiments to derive a policy for setting dispatching rule parameters depending on shop floor conditions.

In this chapter, we propose two myopic sequential sampling methods, REVI, a generalisation of REVI from previous chapters, and CLEVI, a modified LEVI from the previous chapter. We use a Gaussian processes to model the unknown function from the *joint* domain of task features and tool parameters (henceforth just "tool") to performance, $X \times A \to \mathbb{R}$. The method thus exploits covariance in the space of tasks and tools. We demonstrate empirically that the new methods outperform two recently published multi-task conditional Bayesian optimisation algorithms from the literature, the Profile Expected Improvement method of Ginsbourger et al. [2014] and the Surrogate Collaborative Tuning method of Bardenet et al. [2013]. Furthermore, to the best of our knowledge, we are also the first to explicitly consider two different ways for a decision maker to pick a solution: the solution with the best *sampled*

performance, and the solution with the best *predicted* performance.

The chapter is structured as follows. In Section 6.2 we provide the general problem formulation, and then describe the Gaussian Process model and our two new sampling procedures in Section 6.3. A baseline algorithm is described in Section 6.4. In Section 6.5 we empirically evaluate our methods on three synthetic benchmarks and we conclude in Section 6.6 with a summary and some ideas for future work.

6.2 Problem Formulation

We assume that there exists a (discrete or continuous) set of tasks described by D_X features, $x \in X \subset \mathbb{R}^{D_X}$, and the tasks are distributed according to a known density $\mathbb{P}[x]$. There is a tool with D_A tunable parameters $a \in A \subset \mathbb{R}^{D_A}$. Executing the tool a on a task with features x yields a performance measurement $Y_{x,a} = \theta(x, a) + \epsilon$ where $\theta : X \times A \to \mathbb{R}$ is a deterministic latent function from (task, tool) to expected performance and ϵ is independent and identically distributed observation noise $\epsilon \sim N(0, \sigma_\epsilon^2)$. Our aim is to find a mapping $S : X \to A$ from a given task to the optimal tool setting for this task that approximates the true optimal mapping $S^*(x) = \underset{a}{\text{argmax}}\ \theta(x, a)$ with incomplete information. The quality of the derived mapping $S(x)$ at the end of sampling is the corresponding true expected performance over all tasks:

$$\int_{x \in X} \theta(x, S(x)) \mathbb{P}[x] dx. \tag{6.1}$$

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

Given this formulation, if x is constant, the problem reduces to a single global optimisation over A. However, because there is a range of x, one must find the *conditional* optima given task x: $\max_a \theta(x, a)$. Examples of benchmark functions used later can be seen in Figure 6.1.

This formulation also accounts for task distribution because in practice, under a constrained budget, finding the optimal tools for tasks with unusual outlying features (low $\mathbb{P}[x]$) is less useful than finding the optimal tools for common tasks

(high $\mathbb{P}[x]$) and may be replaced by any user defined weight function $W(x)$.

In the clinical trials dose finding example given earlier, x may be a patient's cancer cell biomarker measurements, $\mathbb{P}[x]$ would be the target patient population distribution across biomarker space X, and a would be the quantities of compounds to use in a treatment. In this case, only patients in the trial can be tested while the mapping can be learnt for patients outside the trial. $\theta(x, a)$ would be the expected measured outcome of the treatment such as average reduction in tumour size. A clinician would then like to find the compound mixture a for each patient x that maximises the expected tumour reduction $\theta(x, a)$ across all patients $\mathbb{P}[x]$. The clinician can then create a mapping from a given patient x to the optimal compounds $a = S(x)$.

In simulation optimisation, for example finding optimal ambulance base location conditional on patient distribution, x would be the distribution parameters for the patient population over a city, $\mathbb{P}[x]$ is the distribution of cities, each one parameterised by is patients x. A is the set of valid ambulance base locations and $\theta(x, a)$ is journey time from ambulances to patients in a simulation run. $S(x)$ returns the optimal ambulance location based on patient distribution specified by x.

Using this framework, we consider two ways to derive a mapping $S(x)$. In general, for a risk neutral decision maker, $S(x)$ would return the solution with the best *predicted* performance based on the samples collected and the derived performance prediction model $\mu(x, a)$, such a mapping is given by

$$S^1(x) = \underset{a}{\operatorname{argmax}}\ \mu(x, a). \tag{6.2}$$

As a special case of the above framework, when the task distribution is discrete, $X = \{x_1, .., x_{N_X}\}$, the sampling budget is greater than the number of tasks, $N > N_X$, and the outcomes are deterministic, ($\sigma_\epsilon = 0$), one may also consider a risk averse decision maker. In this work, we define such a decision maker as one that only selects tools a for a task x_i from tools that have been measured, i.e., of which the performance is known with certainty. If the set of sampled tools is denoted $A^N = \{a^1, ..., a^N\}$ and the subset corresponding to task x_i is denoted $A_i^N = \{a^n | x^n = x_i, n \in \{1, ..., N\}\}$, a

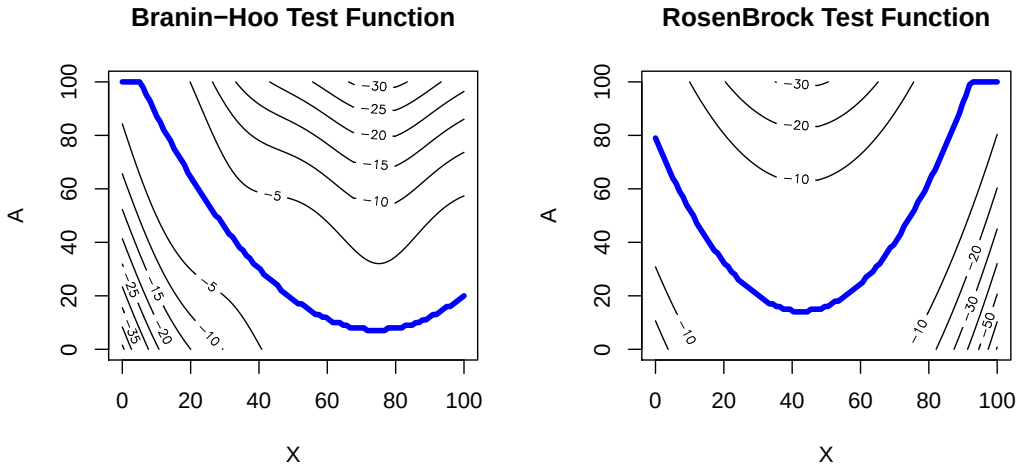


Figure 6.1: Given functions of multiple inputs, we aim to find the optimum of some inputs conditioned on the remaining inputs. In 2D, for each point along the horizontal axis we aim to find the optimal value along the vertical axis as shown by the thick blue lines.

mapping for such a decision maker is then given by

$$S^2(x_i) = \underset{a \in A_i^N}{\operatorname{argmax}}\ \theta(x_i, a). \tag{6.3}$$

where we have noted that the true objective function is known for past evaluated points $\{y^1, ..., y^n\} = \{\theta(x^1, a^1), ..., \theta(x^n, a^n)\}$ as there is no observation noise.

6.3 Myopic Sampling Methods

We use a Gaussian process regression model to predict the underlying latent function $\theta(x, a)$ based on the n data points collected so far $(x^1, a^1, y^1), ..., (x^n, a^n, y^n)$.

For this chapter, we use the notation $\tilde{x} = (x, a) \in X \times A$ and write functions of both input variables as $\mu^n(x, a) = \mu^n(\tilde{x})$. We define the set of sampled task features values $\{x^1, ..., x^n\} = X^n$, tools $\{a^1, ..., a^n\} = A^n$, the set of input pairs $\{(x^1, a^1), ..., (x^n, a^n)\} = \tilde{X}^n$ and column vector of outputs $(y^1, ..., y^n) = Y^n$. We define the sequence of filtrations, $\mathcal{F}^n$, as sigma algebras generated by the data collected up to time n, $\mathcal{F}^n = \sigma\{(x^1, a^1, y^1), \ldots, (x^n, a^n, y^n)\}$. Contrasting with the previous chapters, $a \in A$ is continuous and we make it an explicit argument, there is

now only one Gaussian process model over the joint domain $X \times A$,

$$\begin{aligned} \mathbb{E}[\theta(x,a)|\mathcal{F}^n] &= \mu^n(\tilde{x}) \\ &= \mu^0(\tilde{x}) + k^0(\tilde{x}, \tilde{X}^n)(K^n + \sigma_\epsilon^2 I)^{-1}(Y^n - \mu^0(\tilde{X}^n)) \\ \mathrm{Cov}[\theta(x,a), \theta(x',a')|\mathcal{F}^n] &= k^n(\tilde{x}, \tilde{x}') \\ &= k^0(\tilde{x}, \tilde{x}') - k^0(\tilde{x}, \tilde{X}^n)(K^n + \sigma_\epsilon^2 I)^{-1} k^0(\tilde{X}^n, \tilde{x}') \end{aligned} \tag{6.4}$$

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

As mentioned in Section 6.2, there are at least two choices of the derived mapping $S(x)$ that are not clearly distinguished in the current literature. The EGO and SKO algorithms were implicitly designed on the basis that the input corresponding to best sampled point will be returned to the user at the end of sampling, which corresponds to the mapping $S^2(x)$ from Equation 6.3. For the EGO algorithm, samples are allocated to maximise the expected improvement (EI) of the largest sampled value $\bar{y} = \max Y^n$, recall the expected improvement from Chapter 3.3.2,

$$\begin{aligned} \mathrm{EI}(a) &= \mathbb{E}\left[\max\{y^1, ..., y^n, y^{n+1}\} - \max\{y^1, \ldots, y^n\} | \mathcal{F}^n, a^{n+1} = a\right] \\ &= (\mu^n(a) - \bar{y})\Phi\left(\frac{\mu^n(a) - \bar{y}}{\delta^n(a)}\right) - \delta^n(a)\phi\left(\frac{\mu^n(a) - \bar{y}}{\delta^n(a)}\right) \end{aligned}$$

where $\Phi(z)$ and $\phi(z)$ are standard normal cumulative and density functions and $\delta^n(a) = \sqrt{k^n(a,a)}$ is the posterior standard deviation.

In the multi-task setting however, $S^2(x)$ and EGO can only be used if the task distribution is discrete and the sampling budget is greater than the number of tasks, therefore cannot be used in the continuous task case ($|X| = N_X = \infty$) or whenever the tasks outnumber the sampling budget($N_X > N$). Using a regression model over the joint domain $\mu(x,a)$, such as Gaussian process regression, one is able to predict the performance of any point. For a given task x, a risk-neutral decision maker is

more likely to choose the point with the best *predicted* (but not necessarily sampled) performance. This is the assumption used by Frazier et al. [2009a] in their paper on Knowledge Gradient for Correlated Normal Beliefs, although that algorithm was not designed specifically for continuous optimisation, it can be applied to a discretization of a continuous domain. In our case of multi-task optimisation, we would choose the mapping given by Equation 6.2, $S^1(x) = \underset{a}{\text{argmax}}\ \mu^N(x,a)$, which is defined for any sampled or unsampled task and thus applicable also if the task distribution is continuous. Our methods below will mainly target the mapping S^1, but we will also propose a new way to deal with S^2 in Section 6.3.3.

In order to allocate samples to maximise Value of Information, we require an estimate of the current performance upon which we must improve, and we note that as with constructing the mapping, it is not possible to use the highest sampled point as this does not exist for all tasks. Instead we use the model's prediction of expected performance, given the mapping $S^1(x)$, the current predicted performance on a task x is $\mu^n(x, S^1(x)) = \max_a \mu^n(x,a)$ and so the total predicted performance after n samples across the task distribution is given by

$$P^n = \int_{x\in X} \mu^n(x, S^1(x))\mathbb{P}[x]\,dx = \int_{x\in X} \max_a \mu^n(x,a)\mathbb{P}[x]\,dx$$

that is the same as Equation 6.1 with the true function, $\theta(x,a)$, replaced by the prediction $\mu^n(x,a)$ and is identical to previous chapters with a change of notation for a.

Given a measure of performance, we need to derive myopic sampling policies that Value of Information, $\mathbb{E}[P^{n+1} - P^n|\mathcal{F}^n, (x,a)^{n+1}]$ requiring the new posterior mean $\mu^{n+1}(x,a)$ that is given in previous chapters (Equations 5.9 and 5.10), repeated here

$$\mu^{n+1}(\tilde{x}) = \mu^n(\tilde{x}) + \tilde{\sigma}(\tilde{x};\tilde{x}^{n+1})\ Z^{n+1} \tag{6.5}$$

where Z^{n+1} is a standard normal random variable $Z^{n+1} \sim N(0,1)$. The remaining

term $\tilde{\sigma}(\tilde{x}; \tilde{x}^{n+1})$ is given by

$$\tilde{\sigma}(\tilde{x}; \tilde{x}^{n+1}) = \frac{k^n(\tilde{x}, \tilde{x}^{n+1})}{\sqrt{k^n(\tilde{x}^{n+1}, \tilde{x}^{n+1}) + \sigma_\epsilon^2}}$$

is the additive update to the posterior mean caused by the new sample at $\tilde{x}^{n+1}$ and scaled by the stochastic Z^{n+1}. The expectation, over $Z^{n+1} \sim N(0,1)$, of the performance prediction after the next sample is

$$\mathbb{E}[P^{n+1}|\mathcal{F}^n, \tilde{x}^{n+1}] \quad = \quad \int_{x \in X} \mathbb{E}\big[\max_a \{\mu^n(x,a) + \tilde{\sigma}((x,a); \tilde{x}^{n+1}) Z^{n+1}\}\big] \mathbb{P}[x]\, dx. \tag{6.6}$$

Finally, we may write the Value of Information, the difference in predicted performance between consecutive samples as follows

$$\begin{aligned} \mathcal{I}(\tilde{x}) &= \mathbb{E}\big[P^{n+1} - P^n | \mathcal{F}^n, \tilde{x}^{n+1} = \tilde{x}\big] \\ &= \int_{x' \in X} \mathbb{E}\big[\max_{a'} \{\mu^n(x',a') + \tilde{\sigma}((x',a'); \tilde{x}) Z^{n+1}\} - \max_{a'} \mu^n(x',a')\big] \mathbb{P}[x']\, dx'. \end{aligned} \tag{6.7}$$

which differs from the previous chapters because now the inner $\max_a$ is over a continuous set; we make this distinction from the previous chapters clearer in the next section. The integrand, for a fixed task x, is the same as the Knowledge Gradient acquisition function given in Chapter 3.3.1. The above expression can be evaluated exactly when X and A are finite discrete sets. However, when X and A are continuous sets, there is no solution for the expectation over Z^{n+1} of the max function, nor is it possible to integrate across tasks for arbitrary $\mathbb{P}[x]$. Next, we propose the CLEVI and REVI acquisition functions based on approximations to Equation 6.7.

6.3.1 CLEVI Sampling Policy

We aim to allocate samples in order to maximise $\mathcal{I}(\tilde{x})$, the expected improvement in predicted performance calculated across all tasks, however this integral must be

approximated. The Convolutional Local Expected Value of Improvement (CLEVI) policy makes two assumptions in order to evaluate the integral. Firstly, for each x', the maximisation over continuous A may be approximated by a maximisation over a finite set A_D. And secondly, the improvement for a task that is not the candidate sample task, $x' \neq x^{n+1}$, may be approximated by the improvement at the sampled task x^{n+1}, and the covariance across tasks $k_X(x', x^{n+1})$.

By replacing the maximisation over continuous A with a maximisation over discrete A_D containing $n_A = |A_D|$ points, the expectation within the integrand of $\mathcal{I}(\tilde{x})$ for a given x' may be written as:

$$\mathbb{E}\big[\max\{\mu^n(x', A_D) + \tilde{\sigma}((x', A_D); (x, a))Z\} - \max \mu^n(x', A_D)\big] \tag{6.8}$$

where $\mu^n(x', A_D) = (\mu^n(x', a_1), \ldots, \mu^n(x', a_{n_a})) \in \mathbb{R}^{n_A}$ is the vector of means and similarly for $\tilde{\sigma}((x', A_D); (x, a))$. Gathering terms, Equation 6.8 is thus of the form

$$\mathbb{E}\big[\max\{\mu_1 + \sigma_1 Z, \ldots, \mu_{n_A} + \sigma_{n_A} Z\}\big]$$

which is the expectation of the maximum of 1-dimensional linear functions with a Gaussian argument Z. This expectation can be cheaply evaluated using Algorithm 1 in Knowledge Gradient for Correlated Normal Beliefs [Frazier et al., 2009a] reproduced here in Algorithm 1. To summarise briefly, which of the linear functions is the largest varies with Z, finding the index of highest linear function for each Z and calculating the intersections (the epigraph) can be done in $O(n_A \log(n_A))$ time. Once the piecewise linear "ceiling" over all the linear functions is known, the expectation over Z is a sum of expectations of each linear piece, a sum of means of truncated normal distributions. For convenience, we define the function KG: $\mathbb{R}^{n_A} \times \mathbb{R}^{n_A} \to \mathbb{R}$ that takes a vector of means (intercepts) and a vector of additive updates (gradients) and returns the expectation as given by Algorithm 1. Given the $\mathrm{KG}(\underline{\mu}, \underline{\sigma})$ function we may write the Value of Information integral as

$$\mathcal{I}(x, a) \approx \int_X \mathrm{KG}\big(\, \mu^n(x', A_D),\, \tilde{\sigma}((x', A_D); (x, a))\,\big)\mathbb{P}[x']dx'.$$

First we require that the Gaussian process kernel is factorisable,

$$k^0((x,a),(x',a')) = \sigma_0^2 k_X(x,x')k_A(a,a'),$$

such as the Matern class or squared exponential kernels. The second assumption we make is that the expected improvement at an arbitrary task $x' \neq x^{n+1}$ may be approximated by the improvement at the next task x^{n+1} and the covariance between tasks $k_X(x', x^{n+1})$,

$$\begin{aligned}\mathrm{KG}\big(\,\mu^n(x', A_D), \quad \tilde{\sigma}((x', A_D); (x^{n+1}, a^{n+1}))\,\big) \\ \approx \mathrm{KG}\big(\,\mu^n(x^{n+1}, A_D), \quad \tilde{\sigma}((x^{n+1}, A_D); (x^{n+1}, a^{n+1}))\,\big) k_X(x', x^{n+1}).\end{aligned}$$

Plugging both of these assumptions into Equation 6.7 and rearranging yields the following formula

$$\mathcal{I}(x,a) \approx \mathrm{KG}\big(\,\mu^n(x, A_D),\, \tilde{\sigma}((x, A_D); (x,a))\,\big) \int_X k_X(x', x)\mathbb{P}[x']dx'.$$

where the integral of the right hand side is the convolution, or kernel smoothing, between the task covariance kernel and the underlying task distribution. We denote the convolved density $\tilde{\mathbb{P}}[x]$. This may be found analytically in special cases such as a Gaussian kernel and any piece-wise linear density function over X such as the uniform or triangle distributions. If no analytical expression can be found, then Monte-Carlo integration may be used, samples of $\mathbb{P}[x]$ used in a kernel density estimate using kernel $k_X(x, x')$. Given these two assumptions, we finally write the CLEVI acquisition function as

$$\mathrm{CLEVI}(x,a) \quad = \quad \mathrm{KG}\big(\,\mu^n(x, A_D),\, \tilde{\sigma}((x, A_D); (x,a))\,\big)\tilde{\mathbb{P}}[x] \tag{6.9}$$

The KG function need only be called once. The CLEVI sampling policy treats each task as a single global optimisation and uses the Knowledge Gradient for standard global optimisation over A within the single task x^{n+1} to quantify the value of a new sample at $(x,a)^{n+1}$. However this acquisition value is weighted by $\tilde{\mathbb{P}}[x]$ which

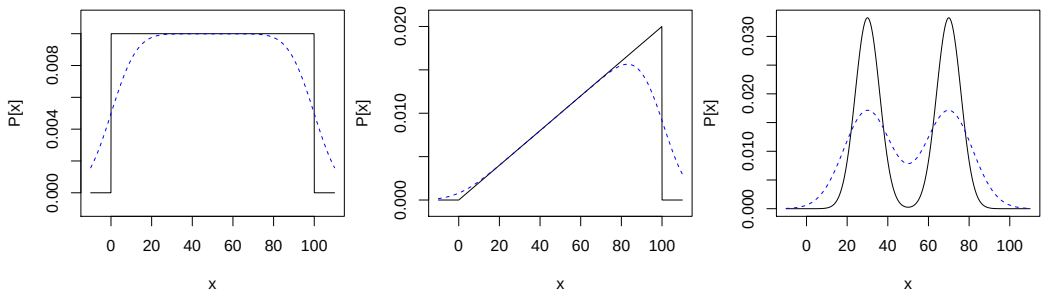


Figure 6.2: In all plots, black solid line: the task distribution $\mathbb{P}[x]$, blue dotted line: transformed task distribution $\tilde{\mathbb{P}}[x]$ with a Gaussian kernel with length scale $l_X = 10$. Using the transformed $\tilde{\mathbb{P}}[x]$ instead of $\mathbb{P}[x]$ down weighs sampling at sharp boundaries as shown in the left plots, and up weighs sampling between clusters as shown in the right plot.

gives the task a relative importance. If the untransformed $\mathbb{P}[x]$ were used instead, the policy may allocate samples to where the single point task density, $\mathbb{P}[x]$, is high as we saw LEVI in the previous Chapters. However this may not be ideal in certain cases. For example, if the mode of $\mathbb{P}[x]$ is close to $\mathbb{P}[x] = 0$, such as the triangular distribution, then the mode is also the boundary where not many correlated tasks can benefit from the new data. Sampling away from the boundary can provide improvement for more tasks. Further examples are given in Figure 6.2.

Next we discuss the choice of discretisation A_D. For single-task global optimisation (equivalently $\mathbb{P}[x]$ is a single point mass), the Knowledge Gradient for Continuous Parameters [Scott et al., 2011a] also calculates an estimate of the Value of Information by discretising over the decision variable, A. They propose to discretise A using past evaluated points and the current point $A_D = A^n \cup \{a^{n+1}\}$ therefore $n_A = n + 1$. This has the advantage that points will cluster around the global maxima, and if there is no observation noise the output of Algorithm 1 reduces to the popular EI function. However, we would like to use the same set of points A_D for every task value $x \in X$ for which the optimal a may be different. Therefore we use a uniform latin hypercube over A, denoted $A_{LHC} = \{a_1, ..., a_{n_a}\}$ specifically in order to avoid clustering. We freeze A_D, using the same discretisation for all tasks $x \in' X$. As a result the CLEVI function is a smooth deterministic function that is easily optimised. Like the Knowledge Gradient policy, the derivative of $CLEVI(x, a)$ with respect to (x, a) can be derived in closed form and used with multiple starts in

gradient ascent optimisers.

We next discuss n_A. For a given n_A and x^{n+1}, the discretisation is a set of reference points, $\{x^{n+1}\} \times A_D = \tilde{X}_{ref} \subset X \times A$ all with the same task value x^{n+1}. In contrast, recall the observed $\tilde{X}^n$ points are more spread out over the domain $X \times A$ all with unique task values X^n. Only a subset of $\tilde{X}^n$ points will have significant correlation with the current task, i.e. large $k_X(X^n, x^{n+1})$. Hence, by setting $n_A = n$, the reference points will outnumber the number of influential observed points, therefore we use $n_A = n + 1$ and $A_D = A_{LHC} \cup \{a^{n+1}\}$.

A single evaluation of the EGO expected improvement (EI) function requires evaluating both the posterior mean $O(n)$ and covariance $O(n^2)$. Evaluating the CLEVI function requires evaluating the posterior mean and covariance n_A times, and one call to the KG function which has complexity $O(n_A log(n_A))$, thus the complexity of one CLEVI call is $O(n_A n^2 + n_A log(n_A))$, strictly greater than one EI call. In this problem formulation, the Value of Information can only be measured using the posterior means which are *all* changed by the new sample. Thus, the problem formulation requires extra calls to the posterior covariance. However, as with the Neighbors-REVI in the previous chapter, this complexity may be reduced by assuming that reference points $\tilde{X}_{ref}$ that have low prior correlation with $(x, a)^{n+1}$ will have zero posterior correlation. This reduces computation with little loss of efficiency. This "zeroing" of posterior covariance is discussed in Section 6.3.4.

The CLEVI sampling policy is readily adapted to the case where the set of tasks is finite, $X = \{x_1, ..., x_{N_X}\}$ and each task has an associated probability. The convolution reduces to summation, and the CLEVI function can be optimised over A for each task individually. Likewise, if A is finite, one may set $A_D = A$, and continuously optimise over tasks X.

6.3.2 REVI Sampling Policy

The Regional Expected Value of Improvement policy (REVI) is the continouos tool generalisation of the same REVI method from all previous chapters. REVI improves upon the CLEVI policy by not making the second assumption and instead accounting for improvement of similar tasks by evaluation rather than the convolution

Algorithm 1 The KG Function The following algorithm takes a vector of intercepts and gradients finds the piece-wise linear epigraph, or "ceiling", of all the overlapping linear functions and calculates the expectation over a normally distributed input Z. $\tilde{Z}$ is the vector of Z values at the vertices of the epigraph, I is the vector of indices of the corresponding linear functions that are part of the epigraph. The algorithm starts with an epigraph of two lines with the lowest gradients, and at each step, adds a steeper line and updates the epigraph. All vector indices to start from 1.

Require: $\underline{\mu}, \underline{\sigma} \in \mathbb{R}^{n_A}$

Remove dominated pairs from $\underline{\mu}$ and $\underline{\sigma}$

Sort the elements of $\underline{\mu}$ and $\underline{\sigma}$ in order of increasing $\underline{\sigma}$

Initialize $\underline{\mu} \leftarrow \underline{\mu} - \max\{\underline{\mu}\}$, $I \leftarrow [1, 2]$, $\tilde{Z} \leftarrow [-\infty, \frac{\mu_1 - \mu_2}{\sigma_2 - \sigma_1}]$

for $i = 3$ **to** $|\underline{\mu}|$ **do**

(1) $j \leftarrow last(I)$, $z \leftarrow \frac{\mu_i - \mu_j}{\sigma_j - \sigma_i}$

if $z < last(\tilde{Z})$ **then** Delete last element of I, last element of $\tilde{Z}$, return to (1)

Add i to end of I and z to end of $\tilde{Z}$

end for

$\tilde{Z} \leftarrow [\tilde{Z}, \infty]$

return $\sum_{i=1}^{length(I)} \mu_{I_i}(\Phi(\tilde{Z}_{i+1}) - \Phi(\tilde{Z}_i)) + \sigma_{I_i}(\phi(\tilde{Z}_i) - \phi(\tilde{Z}_{i+1}))$

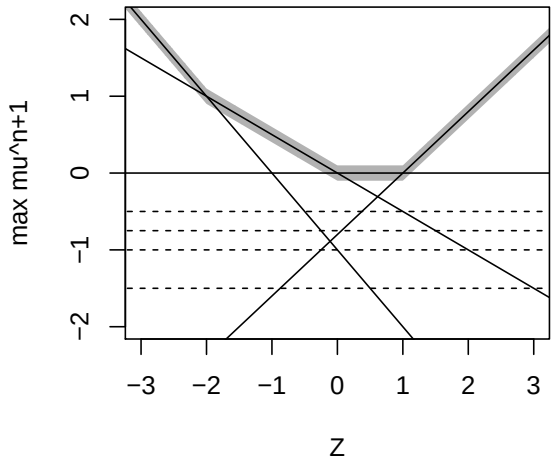


Figure 6.3: Each line represents the posterior mean at a given point after a new sample y^{n+1} with z-score given by Z. Algorithm 1 removes the dominated functions (dotted), finds the epigraph (highlighted), and calculates the expectation of the epigraph over $Z \sim N(0, 1)$.

approximation, however this requires greater computation. Equation 6.6 gives the value of information for each task integrated over the distribution of tasks $\mathbb{P}[x]$. However this cannot be evaluated exactly if X and A are continuous sets.

By discretising over A as described above in the CLEVI policy, for a given x the expectation over Z^{n+1} can be computed exactly using the KG function. We discretise over X by replacing it with a Monte-Carlo set X_{MC} of n_X task feature values distributed according to $\mathbb{P}[x]$. We define the Monte-Carlo estimate as

$$\begin{aligned} \mathcal{I}(x,a) &\approx \hat{\mathcal{I}}(x,a) \qquad (6.10) \\ &= \frac{1}{n_X} \sum_{x_i \in X_{MC}} \mathbb{E} \max_{a_j \in A_D} \{\mu^n(x_i,a_j) + \tilde{\sigma}((x_i,a_j);(x,a))Z^{n+1}\} - \max_{a_j \in A_D} \mu^n(x_i,a_j) \end{aligned}$$

that tends to the true value $\mathcal{I}(x,a)$ as n_X and n_A tend to infinity, it is a consistent estimator. In the summation of Equation 6.10, each term is the expectation over the maximum of linear functions of Z^{n+1}, therefore each term may be calculated using the KG function and the summation yields the REVI acquisition function

$$\text{REVI}(x,a) = \frac{1}{n_X} \sum_{x_i \in X_{MC}} \text{KG}\big(\, \mu^n(x_i, A_D),\, \tilde{\sigma}((x_i,A_D);(x,a))\,\big). \qquad (6.11)$$

The above Monte-Carlo integral does not usually include the proposed sample task x^{n+1}, because x^{n+1} is not a sample from $\mathbb{P}[x]$. It may be included by adding CLEVI(x,a) such that REVI is a mix of two estimates of $\mathcal{I}(\tilde{x})$ although we do not consider such approximations here. The set of reference points used to compute REVI is

$$\tilde{X}_{ref} \;=\; X_{MC} \times A_{LHC} \cup \{a^{n+1}\} \;\subset\; X \times A$$

which has $n_X(n_A+1)$ points. The (task, tool) pair that maximises the REVI function is chosen for sampling. At each time step, the random sets X_{MC} and A_D are generated and held constant until the next time step when they are regenerated. Jittering the discretisation in both domains ensures the learnt mapping does not overfit to one particular discretisation and in the long term the learnt mapping converges to the true optimal mapping. We discuss more efficient choice of n_X and X_{MC} in Section 6.3.4.

One call to REVI requires the computation of $\hat{\mathcal{I}}(x^{n+1},a^{n+1})$ which can be decomposed. There are $n_X n_A$ fixed reference points in the discretisation. For $n_X(n_A-1)$ of the points that do not vary with $(x,a)^{n+1}$, the posterior means and

final two terms of the posterior covariance can be precomputed and stored between REVI calls

$$k^n(\tilde{x}^{n+1}, \tilde{X}_{ref}) = k^0(\tilde{x}^{n+1}, \tilde{X}_{ref}) - k^0(\tilde{x}^{n+1}, \tilde{X}^n) \underbrace{\left(k^0(\tilde{X}^n, \tilde{X}^n)\right)^{-1} k^0(\tilde{X}^n, \tilde{X}_{ref})}_{\text{precompute and store}}.$$

Therefore, only the first two terms of the matrix multiplication for the posterior covariance are necessary, resulting in an $O(n_X n_A n)$ computation per call. The remaining posterior means and covariances for the n_X points corresponding to (X_{MC}, a^{n+1}) must be computed resulting in a cost of $O(n_X n^2)$ and the KG function must be called n_X times. Overall, one call to REVI requires $O(n_X n^2 + n_X n_A n + n_X n_A log(n_A))$. In our experiments we set $n_A = n$ and $n_X = 4\sqrt{n}$, each REVI call has leading order complexity $O(n^{2.5})$ which is greater than one call to EI, $O(n^2)$, however less than the $O(n^3)$ required to fit a Gaussian process. Much of the computation may be reduced by assuming points in the discretisation that are uncorrelated with the new sample may be set to 0 which is discussed in Section 6.3.4.

6.3.3 Discrete Task Distributions

Adapting the CLEVI policy to discrete tasks or discrete tools is easily done as explained in Section 6.3.1. Adapting REVI to finite sets X where $\mathbb{P}[x]$ is a probability mass function, the expected improvement over all tasks can be computed and weighted according to the relative probabilities of each task, or equivalently $\lim_{n_x \to \infty} \text{REVI}(x_j, a)$, the new acquisition function becomes

$$\text{REVI}_D(x_j, a) \quad = \quad \sum_{x_i \in X} \mathbb{P}[x_i]\, KG\big(\, \mu^n(x_i, A_D),\, \tilde{\sigma}((x_i, A_D), (x_j, a))\,\big) \quad (6.12)$$

If we further assume that the tool space is discrete with no correlation between tools, the KG function reduces to the EI function and $\text{REVI}_D(x, a)$ is equivalent to the REVI acquisition function of Chapter 5.

In the case of a risk averse decision maker who only chooses from sampled points when selecting tools for a given task, it is important to ensure that good solutions for each task are actually sampled. As discussed in Section 6.2, the mapping

for such a decision maker is defined as

$$S^2(x_i) = \underset{a \in A_i^N}{\text{argmax}}\ \mu^N(x_i, a)$$

where $A_i^N = \{a^n | x^n = x_i, n \in \{1, ..., N\}\}$ is the subset of a values that have been measured on task x_i and we have utilised that $y^n = \mu^N(x^n, a^n)$. The REVI and CLEVI policies aim to maximise the peak posterior mean for each x_i, which can result in some tasks not being sampled. For example, if two tasks are very similar, it is only necessary to sample one of them to learn about both. However, for such an unsampled task, the decision maker would be restricted to select the best of the few randomly allocated initial samples. The overall performance of CLEVI and REVI is possibly rather poor when using $S^2(x_i)$.

In order to gain the performance advantages of using the posterior mean, while only selecting tools from sampled points, we propose here to sample according to REVI and CLEVI. However allocate the *final* n_X samples in a single pass, one sample for each of the n_X tasks and within each task, determine a^{n+1} using the EGO algorithm. For task x_i, allocate a^{n+1} to maximise the *conditional* expected improvement of y^{n+1} over the best y value sampled on task x_i. The procedure is outlined in Algorithm 2.

As we show in the next section, maximising the posterior mean before applying EGO for the final samples is superior when compared to using EGO for all samples. We apply EGO for the last samples, instead of pure exploitation, because this is more efficient due to Jensen's inequality. Maximising the peak of the new dataset yields better samples than sampling the point with highest expectation $\mathbb{E}[\max\{y^1, ..., y^{n+1}\}] = \mathbb{E}[\max\{y^1, ..., \mu^n(x, a) + \sqrt{k^n(\tilde{x}, \tilde{x})}Z\}] > \max\{y^1, ..., y^n, \max_a \mu^n(x_i, a)\}$.

6.3.4 Efficient Monte Carlo Integration

We reuse the same approach as in the previous chapter where we proposed Neighbours-REVI, adapted for correlation over the joint domain $X \times A$. For reference points $\tilde{x} \in \tilde{X}_{ref}$ that are not highly correlated with the new sample point (small $k^0(\tilde{x}, \tilde{x}^{n+1})$), we propose to make the approximation $\tilde{\sigma}^n(\tilde{x}; \tilde{x}^{n+1}) \approx 0$. Enforcing sparsity on the

Algorithm 2 Risk averse REVI

Initialise n_0 samples using Latin Hypercube Design
update functions μ^{n_0}, k^{n_0}
for $n = n_0 + 1$ **to** $N - n_X$ **do**
  $(x, a)^{n+1} \leftarrow \text{argmax } REVI_D(x, a)$
  $y^{n+1} \leftarrow \theta(x^{n+1}, a^{n+1})$
  update functions μ^{n+1}, k^{n+1}
end for
$i \leftarrow 1$
for $n = N - n_X$ **to** N **do**
  $a^{n+1} \leftarrow \text{argmax } EGO(x_i, a)$
  $y^{n+1} \leftarrow \theta(x_i, a^{n+1})$
  update functions μ^{n+1}, k^{n+1}
  $i \leftarrow i + 1$
end for
return $S^1(x_i) = \underset{a \in A_i^N}{\text{argmax}}\ \mu^N(x_i, a)$

vector of additive updates, $\tilde{\sigma}^n(\cdot)$, results in two computational speed-ups. Firstly, for reference points, $\tilde{x}_r \in \tilde{X}_{ref}$, that are largely unaffected by the sample $\tilde{x}^{n+1}$, we may avoid the costly matrix multiplication involved in computing the numerator of $\tilde{\sigma}^n(\tilde{x}_r; \tilde{x}^{n+1})$,

$$k^n(\tilde{x}_r, \tilde{x}^{n+1}) = \underbrace{k^0(\tilde{x}_r, \tilde{x}^{n+1})}_{\text{if very small}} - \underbrace{k^0(\tilde{x}_r, \tilde{X}^n)\big(k^0(\tilde{X}^n, \tilde{X}^n)\big)^{-1}k^0(\tilde{X}^n, \tilde{x}^{n+1})}_{\text{then don't compute}}$$

and simply assume $\tilde{\sigma}^n(\tilde{x}_r; \tilde{x}^{n+1}) = 0$. Secondly, for any given task in the Monte-Carlo set, $x_i \in X_{MC}$, if multiple elements of the vector $\tilde{\sigma}((x_i, A_D); (x, a)^{n+1})$ are zero, then all but the element with highest corresponding $\mu^n(x_i, \cdot)$ can be removed. The function $\text{KG}(\mu, \tilde{\sigma})$ performs a sort and a loop over the elements in the vectors $\tilde{\sigma}((x_i, A_D); (x, a)^{n+1})$, by removing such zeroed elements, calls to the KG function can be much cheaper. If for a given task, all elements $\tilde{\sigma}((x_i, A_D); (x, a)^{n+1})$ are zero, the KG function need not be called at all! An example is given in Figure 6.3. Stationary kernels have intrinsic length scales and so we "sparsify" reference points that are beyond $r = 3$ length scales, i.e., where the Mahanalobis distance $\sqrt{(\tilde{x} - \tilde{x}^{n+1})D(\tilde{x} - \tilde{x}^{n+1})} > 3$ with $D = diag(1/l_1^2, ..., l_d^2)$ being a diagonal matrix of the square inverse of the GP length scales over dimensions of $X \times A$. Computation

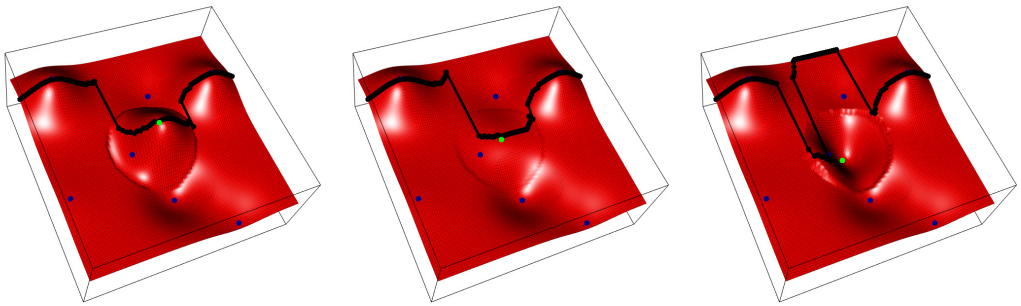


Figure 6.4: Horizontal axis $X = [0, 100]$, vertical axis $A = [0, 100]$. Surface: $\mu^n(x, a) + \tilde{\sigma}^n(x, a; 50, 50)Z$ for $Z = 2$ (L), 0.1 (C), -2 (R). Green point: new sample at $(50, 50)$. Blue points: observations. Black line: predicted conditional optima. The circular ridge is the radius around the new point of the sparse approximation $\tilde{\sigma}^n(x, a; 50, 50) = 0$. Each fixed x slice is a global optimisation problem computed by Knowledge Gradient s in Figure 3.2. Computing REVI$(50, 50)$ is the average over tasks of the KG for each task.

of REVI is illustrated in Figure 6.4.

6.4 Comparison with the Profile Expected Improvement Algorithm

The Profile Expected Improvement (PEI) algorithm of Ginsbourger et al. [2014] considers the same problem with the added assumption that the task distribution is uniform. The algorithm that they propose is a modification of the EGO algorithm where a new sample maximises the expected improvement over a target value for the given task,

$$\text{PEI}(x, a) = \mathbb{E}[\max\{y^{n+1} - T(x), 0\}]$$

where the new sample is given by the Gaussian process,

$$y^{n+1} \sim N(\mu^n(x, a), k^n((x, a), (x, a)).$$

The target of improvement is given by the peak posterior mean for the given task x, however capped by the highest value of the data seen so far $T(x) = \min\{\max_a \mu^n(x, a), \max Y^n\}$. We now show that this is a slightly modified simplification of the CLEVI algorithm. If we take the CLEVI acquisition function, firstly

assume that the task distribution is uniform and the convolution is not applied so that the $\mathbb{P}[x]$ term can be discarded as constant. Secondly, set the sparsity approximation such that all points have zero additive update except for the sampled point, $\tilde{\sigma}((x, A_D \setminus \{a\}); (x, a)) = 0$. Thirdly, by assuming no noise in function observations the posterior standard error and the update to the mean at the sampled point are equal $\tilde{\sigma}((x, a); (x, a)) = \sqrt{k^n((x, a), (x, a))}$. Finally, if we augment the set A_D with the highest mean of the current task, $\text{argmax}_a \mu^n(x, a) \in A_D$, the CLEVI function simplifies to

$$\begin{aligned}
\text{CLEVI}'(x, a) &= \mathbb{E}\big[\max\{\underbrace{\mu^n(x, a) + \sqrt{k^n((x, a), (x, a))}Z}_{y^{n+1}}, \underbrace{\max_a \mu^n(x, a)}_{T(x)}\}\big] - \max_a \mu^n(x, a) \\
&= \mathbb{E}\big[\max\{y^{n+1} - T(x), 0\}\big]
\end{aligned}$$

The only differences between the PEI acquisition function and the CLEVI function for a uniform task distribution without convolution, zero noise and maximum sparsity, is the addition of $\text{argmax}_a \mu^n(x, a)$ to the set A_D and the capping of the target value $T(x)$. By augmenting the set A_D, this has the advantage that the target level for improvement is more accurately measured. The disadvantage is that this requires an extra optimisation over a with fixed x for each call to $\text{PEI}(x, a)$, though in our benchmarks we found this to be negligible. Another consequence of this optimisation is that the $\text{PEI}(x, a)$ acquisition function is no longer differentiable with respect to (x, a) since $dT(x)/dx$ is not analytically tractable and therefore PEI cannot be optimised by gradient descent with multiple starts which may cause excess evaluation in high dimensions, although this also may easily be remedied by taking the max over A_D instead of A as with CLEVI and REVI.

By assuming maximum sparsity, the effect one sample has on other predictions and on the target level itself is neglected and sampling is less efficient as we show in Section 6.5, particularly when there are long length scales in the Gaussian process. The advantage of maximum sparsity however is that there are fewer posterior covariance calls which are each $O(n^2)$. Although the PEI algorithm was not designed with noisy problems in mind, the authors note that the Gaussian process model

may easily be adapted to account for noise and the acquisition function itself is still applicable. In our benchmarks we find that when the assumptions are satisfied, negligible observation noise, uniform task distribution and unaffected target level, the performances of PEI, CLEVI and REVI are similar. However, on more varied scenarios PEI performs significantly worse than CLEVI and furthermore REVI significantly outperforms both. In our numerical experiments with non-uniform task distribution we modify the PEI algorithm to account for task density by weighting the acquisition function according to the point-wise task density,

$$\mathrm{PEI}_{dens}(x, a) = \mathbb{P}[x]\mathbb{E}[\max\{y^{n+1} - T(x), 0\}]$$

such that high density tasks are given priority when sampling and low density tasks are only sampled if their improvement over their target level is high enough.

6.5 Numerical Experiments

We perform numerical three sets of experiments. In the first benchmark we use a continuous distribution of tasks and the popular Rosenbrock optimisation benchmark function comparing our algorithms against PEI and latin hypercube sampling. In the second benchmark we investigate the effect of dimensionality upon the REVI and CLEVI acquisition functions, we generate random functions from a Gaussian process prior with dimensions varying from two to six, and again compare our algorithms against PEI and uniform sampling. Finally we consider the discrete task case and compare CLEVI and REVI against the SCoT algorithm [Bardenet et al., 2013] on risk neutral and risk averse scenarios.

6.5.1 Rosenbrock Test Function

For the first continuous benchmark problem, we use the Rosenbrock test function scaled such that it has domain $X \times A = [0, 100]^2$ and takes values in the range $y \in [-45, 0]$ and we add noise of variance $\sigma_\epsilon \in \{0.1^2, 1.0^2\}$. We test two different task distributions, a uniform distribution $\mathbb{P}[X] = 1/100$ and a triangular distribution $\mathbb{P}[X] = X * 2 * 10^{-4}$ that will adversarially penalise methods that don't take task

correlation into account. Two noise levels and two task distributions yield four different experimental setups and for each setup we apply each algorithm 100 times with different initial design and noise values. For each application, an initial budget of 20 samples is allocated by latin hypercube over the $X \times A$ domain after which a Gaussian process with a squared exponential kernel is fitted. The hyper parameters of the Gaussian process are estimated via maximum likelihood and updated after every new sample. Samples are sequentially added to the initial design according to four algorithms, PEI, CLEVI, REVI and finally LEVI which is the CLEVI algorithm however without the convolution applied to the task distribution such that it is simply the integrand of Equation 6.7. Each method is applied until a sampling budget of 80 samples has been exhausted. We also compare against latin hypercube sampling. To measure the quality of a mapping learnt by each method, for each experiment, a test set of 250 tasks values, X_{test}, are generated from $\mathbb{P}[x]$, and the difference in performance between the true optimal a and the performance of the a value determined by the mapping is averaged over all $x_i \in X_{test}$

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

The resulting average opportunity cost over 100 simulation runs for each algorithm for each budget is given in Figure 6.5 as well as one exemplary final sample design from each sampling method. All acquisition functions were maximised using the Nelder-Mead optimisation algorithm with $\min\{2n, 120\}$ random restarts and all default parameters in R's "optim" function with the exception that the number of iterations was reduced to 50.

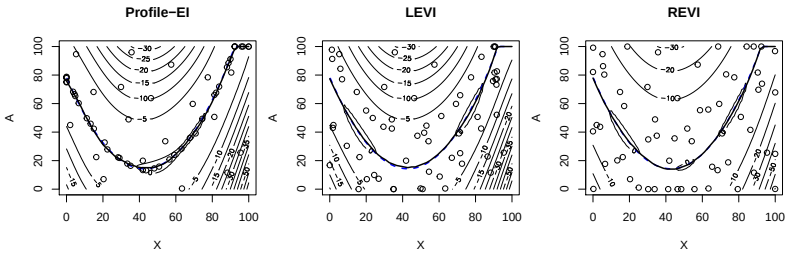
We see that in all cases REVI is the quickest to converge to the true optimal mapping and the CLEVI/LEVI methods are either similar or slightly worse. PEI frequently converges more slowly and in experiments not shown here this performance is replicated when using the CLEVI algorithm with maximum sparsity, therefore it is probably the assumption of a fixed target level that prevents PEI from converging as quickly. It is proven that in the infinite sample limit the PEI algorithm will converge, however the finite time behaviour is apparently different. The length scale in the tool

in the domain A is typically $l_A \approx 122$ while the largest possible distance between two points is 100, thus a sample at $(x, a)^{n+1}$ will affect the model $\mu^{n+1}(x^{n+1}, a)$ for all $a \in A$. Therefore, on this test function, the fixed target assumption is violated. The same behaviour was observed on the Branin-Hoo function that also has a length scale $l_A > 100$ when the domain is scaled to $A = [0, 100]$. The increase in noise reduces the speed of convergence, however does not change the relative ranking of algorithms. Comparing the uniform distribution with the triangular distribution, we see that the LEVI algorithm performs marginally worse due to its failure to account for the difference between the mode of the task distribution (which is also a boundary) and the maximum influence of a sample over the task distribution which is away from the boundary.

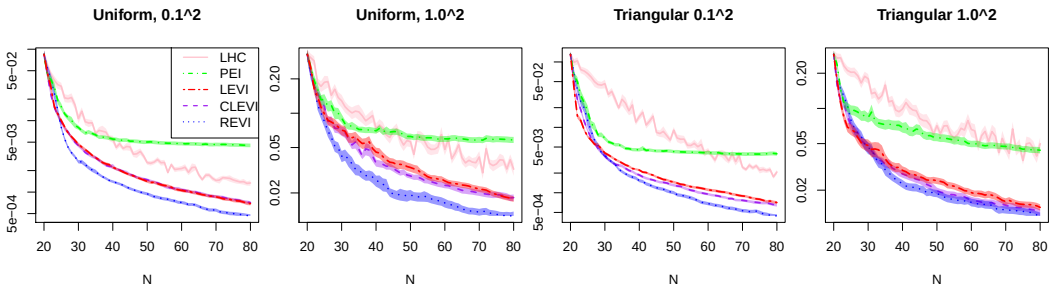
In Figure 6.5, we see the final allocation of points over $X \times A$. The PEI algorithm allocates more samples to the predicted peak of each task however the CLEVI and REVI algorithms allocate samples more evenly. REVI and CLEVI aim to maximise the posterior mean of the model and therefore allocate samples such that the whole model is updated to accurately predict the true peak and samples are scattered around $\pm 0.1 l_A$ of the true peak. This has the advantage that convergence is quicker and since the samples are more spread out there will be less chance of numerical issues when inverting the covariance matrix whilst fitting the Gaussian process. This extra convergence must be traded off with the extra uncertainty over the *predicted* peaks and in this problem setting with a risk neutral decision maker REVI and CLEVI perform as expected. The CLEVI and REVI algorithms may be easily modified to maximise a lower confidence bound instead of the posterior mean (as done by Picheny et al. [2013a]) but we do not consider this case in our problem formulation.

6.5.2 High Dimensional Test Functions

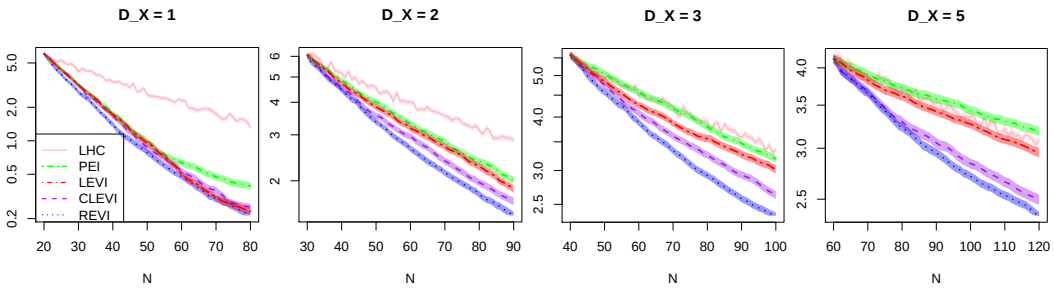
In our second benchmark we generate test functions from a Gaussian process prior where the dimension of the task space X varies from one to five dimensions, $D_X \in \{1, 2, 3, 5\}$ and we fix $D_A = 1$. We do this for two reasons. Firstly, we aim to create a scenario where the assumptions of PEI are met and show that it performs well in



(a) Final sampling allocations



(b) Rosenbrock test function opportunity cost



(c) GP test functions opportunity cost

Figure 6.5: In all cases, REVI produces the best mappings for all experiments and budget sizes, CLEVI and PEI are often equal, however diverge for large budgets where noise variance becomes significant. For the Triangular distribution, REVI outperforms other methods due to its ability to account for regional effects.

this case, that is, where length scales are smaller than the domain and the target level is not always changed by the new sample. This may be done by increasing D_A or by reducing l_A, and to avoid sparsity we chose the latter. Secondly, the REVI algorithm requires a Monte-Carlo integral which can perform poorly as the number of dimensions increases. We initialise each sampling procedure with $10(D_X + 1)$ samples in a latin hypercube. In all experiments, we use a uniform task distribution.

Gaussian processes are well known to struggle in higher dimensions due to

either data sparsity in high dimensional space or the n^3 computational cost or matrix ill-conditioning when data is dense in high dimensional space. In preliminary experiments, we found that all algorithms performed equal with latin hypercube sampling for $D_X = 3, 5$ when all length scales were $l_X = l_A = 10$. The initial design of points was too sparse and the initial samples were allocated to fill empty space and the advantages of REVI and CLEVI provided no significant benefit over PEI or uniform. To create a scenario without the data sparsity we increase the length scale with dimension such that the nearest neighbour in the initial design set is approximately $1.3 * L$ where L is the length scale of the kernel used for all dimensions. The resulting length scales are 10, 16, 22, 28 and 33 for 1, 2, 3 and 5 dimensions, respectively, where the $X \times A$ space is $[0, 100]^{D_X+1}$ and the generating process signal variance is $\sigma^2 = 10^2$. The kernel parameters for generating the functions were also used when fitting the Gaussian process, therefore the only difference between experiments is the acquisition functions. We apply all the same algorithms as from the previous benchmark, however the optimiser has more restarts $\min\{(3+D_X)n, 90+30D_X\}$ and for REVI we set $n_X = (3 + D_X)\sqrt{n}$ to be consistent with the previous experiment. All function evaluations have a noise added, $\epsilon \sim N(0, 1)$.

For the lowest dimensional case we see that all algorithms perform equally. We see that as dimensions increases, the methods that neglect covarying tasks, PEI and LEVI, get worse, and when $D_X = 5$, they do not significantly differ from latin hypercube sampling. Likewise, the REVI and CLEVI algorithms do not suffer as much with increasing dimension. The tasks are uniformly distributed in a hypercube, samples on the boundary of the hypercube have fewer neighbouring tasks which may be improved by the sample. As the number of dimensions increases, there are more edges, vertices and boundaries to avoid, and with increasing length scale the boundary affects more space within the hypercube. The REVI function measures the improvement at tasks X_{MC}, and at boundaries there are fewer tasks and thus smaller improvement in the mapping. Consequently, such areas are less favourable to sample. For CLEVI, by taking the convolution of the task distribution, the sharp boundaries in the true task distribution are rounded and reduced and CLEVI also tends to sample away from boundaries.

6.5.3 Finite Tasks

In our third benchmark, we compare the discrete task versions of CLEVI and REVI against the Surrogate based Collaborative Tuning algorithm, (SCoT) proposed by Bardenet et al. [2013]. The SCoT algorithm tackles the complex problem of predicting good hyper parameters (tool parameters) for training machine learning algorithms (learning rate, batch size, etc.) based on the features of the dataset to which the algorithm is applied. Therefore, X is the space of dataset features, A is the hyper parameters of a machine learning optimisation algorithm and $\theta(x, a)$ is the test set accuracy of the trained algorithm with the given parameters on the given dataset. The proposed algorithm fits a Gaussian process to predict the test accuracy using algorithm parameters and dataset features. The method sequentially executes a parameter setting on a dataset in order to learn the optimal parameter setting for each dataset. At each iteration, the task is pre-determined in a round robin fashion x_i and for a given task, the parameter setting a^{n+1} is determined by maximising the expected improvement of a new measurement over the current best measurement for the current task, that is by the EGO algorithm. Therefore the SCoT algorithm is equivalent to repeated application of the second stage of Algorithm 2. The authors note that the framework can accommodate any acquisition function and we exchange the component with the Knowledge Gradient acquisition function to determine a^{n+1} while determining the task by round robin allocation. Therefore, this algorithm is equivalent to the CLEVI algorithm without the approximation to account for the influence on other tasks and where the task sampling sequence is predetermined. We also again compare with random allocation. We generate random test functions as with the $D_X = D_A = 1$ case described above, however when sampling and measuring opportunity cost, task values are restricted to a finite set of randomly generated numbers $X = X_{MC} = X_{test} \in \mathbb{R}^{20}$ that are distributed according to $x_1, ..., x_{10} \sim N(20, 10^2)$ and $x_{11}, ..., x_{20} \sim N(50, 5^2)$. We measure the opportunity cost using the two mappings discussed in Section 6.2: the risk neutral mapping, executing the best predicted parameter for each task, and the risk averse mapping, executing the best evaluated parameter for each task.

As can be seen in Figure 6.6, in the risk neutral case, REVI and CLEVI

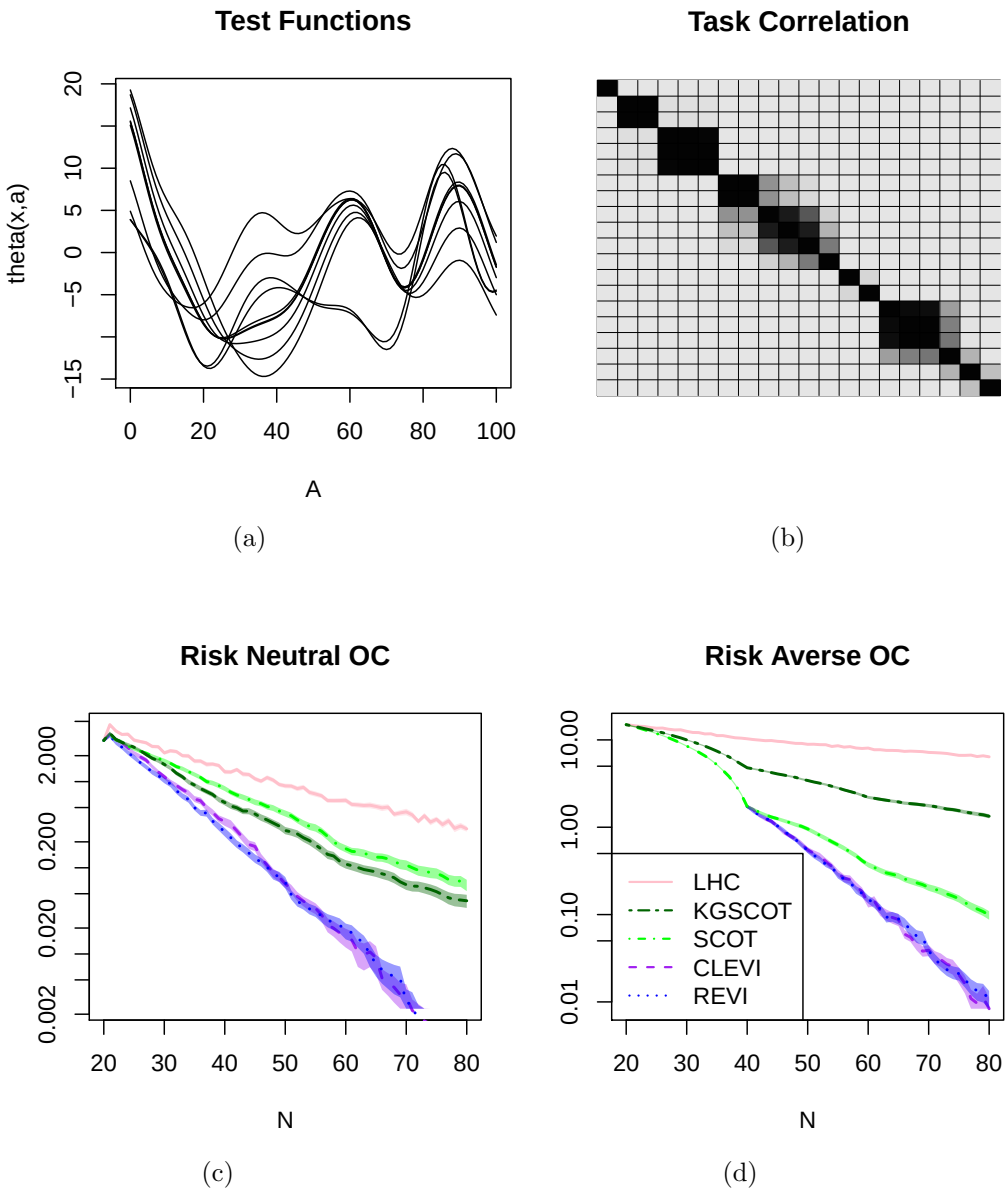


Figure 6.6: (a) one test function realisation, (b) one example task covariance matrix, (c) and (d) opportunity cost for a risk neutral/averse user averaged over 200 test function realisations.

perform best for all budget sizes. Replacing EGO with Knowledge Gradient in the SCoT algorithm did not yield much improvement suggesting that forcing the task allocation to be round robin accounts for most of the performance difference between CLEVI and SCoT. The REVI and CLEVI algorithms without the risk averse modification perform worse than both variations of SCoT and are not shown on the plot. However, with the risk averse modification, there is a very large improvement

and therefore optimising the posterior mean before optimising the sample values yields a great benefit. In general, the risk averse opportunity cost is higher than the risk neutral, demonstrating the price a risk averse user must pay. In both cases, REVI and CLEVI do not differ significantly, suggesting a small number of positively correlated tasks in low dimensions do not benefit from the added accuracy of REVI, although further investigation is required.

6.6 Conclusion and Future Work

We have considered the problem of simultaneously identifying the optimal parameters for a set of tasks with correlation across tasks and where the performance of a particular parameter on a particular task has to be inferred from (potentially noisy) samples. To this end, we provide a general problem formulation, and propose two myopic information collection policies, CLEVI and REVI, that both aim to approximate the overall improvement across all tasks. CLEVI aims to maximise the expected improvement at the sampled task, weighted according to the regional influence the sample is expected to have, whereas REVI more accurately takes into account the regional influence, the information gain for other tasks due to the correlation structure. As expected, while CLEVI is computationally cheaper, REVI performs better, and both methods have equal leading order complexity (REVI has a strict . We show that an alternative algorithm developed for the same problem, Profile Expected Improvement, that we consider state of the art from the literature, is a special case of our CLEVI algorithm and under certain conditions its performance is comparable. However, in almost all cases CLEVI and REVI converge toward the true optimal mapping much faster. Further empirical tests show that on discrete task sets, CLEVI and REVI also significantly outperform the SCoT algorithm, another algorithm from the literature, by a wide margin.

Furthermore, we have pointed out that the problem can be considered with two possible goals: identifying a mapping that predicts the best parameter setting for any given task, and identifying a mapping that selects the best *sampled* parameter setting for each task. The latter is sensible in particular for a risk averse decision

maker under a deterministic setting with a small number of tasks. We demonstrate that for such a setting, one should still collect information based on REVI, but switch to round robin allocation with the task conditional EI (SCoT algorithm) for selecting the last sample for each task.

There are several possible avenues for future work. In this work we have not considered model mismatch, real-world applications essentially always have model mismatch and this can affect the relative performance of Bayesian ptimisation algorithms as demonstated by Schulz et al. [2016]. The REVI algorithm in particular maximally exploits covariance across tasks and parameters and will likely suffer the most from inaccurately estimated covariance structure. Therefore the question remains, can exploiting poorly estimated covariance (REVI) be worse than ignoring only task covariance (CLEVI) or ignoring all covariance (PEI, SCoT)? To this end the proposed algorithms should be applied to various real world problems, including those applications mentioned in the introduction, to reveal any application specific flaws or benefits. An extension to batch parallel sampling should be straightforward and speed up optimisation in practice. The distinction between searching for the solution with the best estimated performance, and searching for the solution with the best sampled performance, applies to all types of problems where Bayesian optimisation is used, and should be examined also in other contexts. Finally, one might consider other notions of risk aversion, such as lower confidence bound, instead of the single extreme case we consider.

Chapter 7

Bayesian Optimization with Uncertain Inputs

7.1 Introduction

Simulation optimization, i.e., the search for a design or solution that optimizes some output value of a simulation model, allows to automate the design of complex systems and has many real-world applications. However, a simulation is never a perfect model of reality [Sargent, 2013]. When constructing the simulation model, the decision maker often faces the challenge of defining proper distributions for the stochastic components within a simulator e.g. the mean of an arrival time distribution. In particular, if the parameters for these distributions are estimated from real world data, multiple possible parameters may be a suitable fit. This issue, generally known as "input uncertainty" of "input distributions", has obtained increasing interest of the simulation community in recent years.

In this chapter, we adapt the two successful and well-known simulation optimization methods introduced in Chapter 3, namely Efficient Global Optimization (EGO) and Knowledge Gradient (KG), to allow for input uncertainty. We assume that the design as well as the possible input distributions can be described by some continuous parameters and that one has a probability distribution over the parameter determining the input distribution. For example, if the input distribution parameters

are estimated by a group of experts, and the different opinions are aggregated into one probability distribution over the input distribution parameter. We furthermore assume that the output of the simulation model is correlated across designs as well as input distribution parameters. Then, given a budget of simulation runs each with a given design and input distribution parameter, the goal is to identify the design with the best expected performance over the distribution of likely input distribution parameters.

We start with a formal definition of the problem in Section 7.2. Section 7.3 explains the newly proposed methods for simulation optimization in the presence of input uncertainty, which are then empirically tested and compared to some synthetic benchmarks in Section 7.4. We then conclude with a summary and some suggestions for future work.

7.2 Problem Definition

Given a simulation model with a tunable parameter that specifies the possible *solutions* $a \in A$ (as opposed to "tool" used in previous chapters) and an input distribution with parameter $x \in X$ (henceforth simply called "input") following an assumed distribution $\mathbb{P}[x]$ independent of a. When running a simulation with solution a and input x, we observe an output following an unknown noisy function $f(x, a) = \theta(x, a) + \epsilon$ where $\theta(x, a) = \mathbb{E}[f(x, a)]$ is a deterministic latent function defining the expected outcome and $\epsilon \sim N(0, \sigma_\epsilon^2)$ is independent white noise with constant variance. For many distributions, the input x may be variance of distributions used within simulations and may lead to variable noise variance over x. Homoscedasticity may be enforced by repeating evaluations, although in this preliminary work we restrict ourselseves to the homeostatic case. The objective of the user is to identify the solution a that maximizes the expected performance simultaneously across the input parameter distribution

$$F(a) = \int_X \theta(x, a)\mathbb{P}[x]dx. \tag{7.1}$$

We assume we have a fixed budget of N samples (simulation runs with a specific solution and input), and that we can sample iteratively, i.e., we can select the solution and the input from which to sample performance $f(x, a)$ based on the information collected so far.

Although for reasons of simplicity we use scalar notation, the approach applies to multi-dimensional inputs and solutions.

If $\mathbb{P}[x]$ is an empirical distribution, the integral may be replaced by a summation.

As an aside, we may make the random number stream, s, explicit in the simulator, $f(x, a, s)$, and therefore $\theta(x, a) = \int_s f(x, a, s)\mathbb{P}[s]ds$. Hence, if x itself represents the complete random number stream, $\theta(x, a)$ may be viewed as a deterministic function and Equation 7.1 reduces to a standard simulation optimisation problem.

7.3 Sampling Methods

In this section, we show how to modify two well known global optimization methods, Efficient Global Optimization (EGO) and Knowledge Gradient with Continuous Parameters (KG) to the case of input uncertainty. We assume that we can use a Gaussian process to model the underlying latent performance function $\theta(x, a)$ given the data observed. For simplicity, let us denote the location and value of the n-th observation by x^n, a^n and y^n. Given all the data collected $(x^1, a^1, y^1), ..., (x^n, a^n, y^n)$, define $\tilde{X}^n = \{(x^1, a^1), ..., (x^n, a^n)\}$ and $Y^n = (y^1, ..., y^n)$, and the sigma algebra $\mathcal{F}^n$ generated by all the data $\sigma\{(x^1, a^1, y^1), \ldots, x^n, a^n, y^n)\}$. Again, we restate the

posterior Gaussian process mean and kernel,

$$\begin{aligned}
\mathbb{E}[\theta(x,a)|\mathcal{F}^n] &= \mu^n(x,a) \\
&= \mu^0(x,a) \qquad (7.2) \\
&\quad -k^0((x,a),\tilde{X}^n)(k^0(\tilde{X}^n,\tilde{X}^n)+I\sigma)^{-1}(Y^n-\mu^0(\tilde{X}^n)) \\
\mathrm{Cov}[\theta(x,a),\theta(x',a')|\mathcal{F}^n] &= k^n((x,a),(x',a')) \\
&= k^0((x,a),(x',a')) \qquad (7.3) \\
&\quad -k^0((x,a),\tilde{X}^n)(k^0(\tilde{X}^n,\tilde{X}^n)+I\sigma)^{-1}k^0(\tilde{X}^n,(x',a'))
\end{aligned}$$

where $\mu^0(x,a)$ is the prior mean which is typically set to $\mu^0(x,a)=0$ and $k^0((x,a),(x',a'))$ is the kernel of the Gaussian process. In Section 7.4, we use the popular squared exponential kernel, though also the Matérn class of kernels has been widely used for simulation optimization.

7.3.1 Efficient Global Optimization for Input Uncertainty

The well-known EGO algorithm sequentially collects objective function evaluations and was originally designed for deterministic global optimization problems. It starts by evaluating the function at a randomly distributed set of inputs to the function. Then, the location of the next sample is determined by maximizing the expected improvement of a new function evaluation over the current best evaluation. The predictive distribution of the new noiseless sample is given by $y^{n+1} \sim N(\mu^n(a), k^n(a,a))$ and so the expected improvement over the current best of a hypothetical sample at parameter a is given by

$$\begin{aligned}
\mathrm{EGO}(a) &= \mathbb{E}[\max\{y^1,...,y^{n+1}(a)\}] - \max\{y^1,...,y^n\} \qquad (7.4) \\
&= \mathbb{E}[\max\{0, y^{n+1}(a) - \max\{y^1,...,y^n\}\}] \qquad (7.5) \\
&= \Delta(a)\Phi(\Delta(a)/\sigma(a)) - \sigma(a)\phi(\Delta(a)/\sigma(a)) \qquad (7.6)
\end{aligned}$$

where $\phi(a)$ and $\Phi(a)$ are the standard normal density and cumulative functions, $\Delta(a) = \mu^n(a) - \max\{y^1,...,y^n\}$ and $\sigma(a) = \sqrt{k^n(a,a)}$. Note that in Equation 7.5, the expectation is over the maxima of two linear functions of y^{n+1}, one is the identity

and the other is the constant function, and y^{n+1} is normally distributed. When function evaluations are deterministic, the previous best sample value is the "target level", $\max Y^n$, and it does not change with the new sample. The EGO acquisition function given in Equation 7.6 is easily and cheaply optimised to find the location of the most promising new function evaluation, a^{n+1}, and then the objective function is evaluated and a $y^{n+1} = f(a^{n+1})$ is observed. The Gaussian process model is updated and the search for the location of the next sample is performed again and the algorithm repeats until the budget of function evaluations is exhausted.

Adapting this method to allow for noise and input uncertainty we need to answer two questions, namely what is the value of the current best upon which we aim to improve, and what is the predictive distribution of the value after a new hypothetical sample is generated. We calculate a prediction of the current best upon which we aim to improve by adapting Equation 7.1, replacing the unknown $\theta(x, a)$ with the model prediction, $\mu^n(x, a)$, and the continuous integral over X can be replaced by a Monte-Carlo integral, a summation over N_X random inputs $X_{MC} = \{x_1, ..., x_{N_X}\} \sim \mathbb{P}[x]$

$$\hat{F}^n(a) = \frac{1}{N_X} \sum_{x_i \in X_{MC}} \mu^n(x_i, a).$$

Then, the current best upon which we aim to improve, the target level for this application, is found by maximizing $\hat{F}^n(a)$ over a. This maximization can be done cheaply using any off-the-shelf optimization algorithm as the function is based only on posterior means. This is further explained in Section 7.4. N_X is a parameter that may be chosen by the user to determine accuracy and in Section 7.4 we use $N_X = n$, so that accuracy increases over the run.

In order to answer the second question, we require an updating formula for the posterior mean to derive the predictive distribution of $\hat{F}^{n+1}(a)$ given only the data available at time n. Exactly as derived in Chapter 3.3.1, this is simply a change of indices in the posterior mean equation, 7.2, and the predictive distribution of the

new posterior mean is then given by

$$\mu^{n+1}(x,a) \sim N(\mu^n(x,a),\ \tilde{\sigma}((x,a);(x,a)^{n+1})^2)$$

The predicted performance after a new sample can then be written as

$$\begin{aligned}
\hat{F}^{n+1}(a;(x,a)^{n+1}) &= \frac{1}{N_X}\sum_{x_i\in X_{MC}} \mu^{n+1}(x_i,a) \\
&= \frac{1}{N_X}\sum_{x_i\in X_{MC}} \mu^{n}(x_i,a) \\
&\quad + Z\frac{1}{N_X}\sum_{x_i\in X_{MC}} \tilde{\sigma}^n(x_i,a;(x,a)^{n+1}) \\
&= \hat{F}^n(a) + Z\ \hat{\Sigma}^n(a;(x,a)^{n+1}).
\end{aligned} \tag{7.7}$$

where $\hat{\Sigma}^n(a,(x,a)^{n+1})$ is given by the final term in Equation 7.7. The predictive distribution of the new design a after a random sample at $(x,a)^{n+1}$ is then given by

$$\hat{F}^{n+1}(a;(x,a)^{n+1}) \sim N\left(\hat{F}^n(a),\ \hat{\Sigma}^n(a;(x,a)^{n+1})^2\right).$$

The above summation does not include the sampled input x^{n+1} however this can be included with a unique weighting and is discussed in Section 7.3.3. The expression gives the updated value for arbitrary a caused by a new sample at $(x,a)^{n+1}$. If we only consider the updated value at the sampled design $a^{n+1}=a$ then the expected improvement over the previous optimal predicted design is given by

$$\begin{aligned}
\mathrm{EGO}(x,a) &= \mathbb{E}[\max\{\max_{a'}\hat{F}^n(a'),\hat{F}^{n+1}(a;(x,a))\}] - \max_{a'}\hat{F}^n(a') \\
&= \mathbb{E}[\max\{0,\hat{F}^n(a) - \max_{a'}\hat{F}(a') + Z\hat{\Sigma}^n(a;(x,a))\}] \\
&= \mathbb{E}[\max\{0,\Delta(a) + Z\hat{\Sigma}^n(a;(x,a))\}] \\
&= \Delta(a)\Phi(\Delta(a)/\hat{\Sigma}^n(a)) - \hat{\Sigma}^n(a)\phi(\Delta(a)/\hat{\Sigma}^n(a))
\end{aligned}$$

where $\Delta(a) = \hat{F}^n(a) - \max_{a'}\hat{F}^n(a')$ and $\hat{\Sigma}^n = \hat{\Sigma}^n(a;(x,a))$. Samples are sequentially allocated to (x,a) pairs that maximise $\mathrm{EGO}(x,a)$. After each sample, the Monte-Carlo points, X_{MC}, may be regenerated to avoid overfitting to one particular

discretization of the distribution $\mathbb{P}[x]$.

7.3.2 Knowledge Gradient for Input Uncertainty

The EGO algorithm compares the value at the current best design (which is assumed to be fixed) with the value at the new sampled design. The Knowledge Gradient compares the values across a range of designs (all are correlated with y^{n+1}) with the new highest value across the same range of designs. We define the set of previously evaluated designs as $A^n = \{a^1, ..., a^n\}$. Hence A^{n+1} includes the next sampled design a^{n+1}. Traditional Knowledge Gradient for Continuous Parameter using Gaussian processes Scott et al. [2011a] allocates samples to maximize the following acquisition function

$$\begin{aligned}
\text{KG}(a) &= \mathbb{E}[\max_{a'' \in A^{n+1}} \{\mu^{n+1}(a'')\}] - \max_{a' \in A^{n+1}} \{\mu^n(a')\} \\
&= \mathbb{E}[\max\{\mu^{n+1}(a^1), ..., \mu^{n+1}(a^n), \mu^{n+1}(a)\}] - \max_{a' \in A^{n+1}} \{\mu^n(a')\} \\
&= \mathbb{E}[\max\{\mu^n(a^1) + Z\tilde{\sigma}(a^1; a), ..., \mu^n(a) + Z\tilde{\sigma}(a; a)\}] - \max_{a' \in A^{n+1}} \{\mu^n(a')\} \\
&= \mathbb{E}[\max\{c_1 + Zm_1, ..., c_{n+1} + Zm_{n+1}\}]
\end{aligned}$$

where $c_i = \mu^n(a^i) - \max_{a' \in A^{n+1}} \mu^n(a')$ and $m_i = \tilde{\sigma}^n(a^i, a)$. The final expectation is the maximum of $(n+1)$ linear functions with a normally distributed argument and may be computed using Algorithm 1 in Chapter 6.3. In contrast, EGO(a) is the maximum over only two linear functions with a normally distributed argument as a result of not accounting for changes in the posterior mean at unsampled designs $a \neq a^{n+1}$. We adapt KG(a) to the input uncertain case KG(x, a) by replacing $\mu^n(a)$ and $\tilde{\sigma}(a, a^{n+1})$ in the above equations with their Monte-Carlo counterparts $\hat{F}^n(a)$ and $\hat{\Sigma}^n(a; (x, a)^{n+1})$.

$$\begin{aligned}
\text{KG}(a, x) &= \mathbb{E}[\max_{a \in A^{n+1}} \{\hat{F}^{n+1}(a)\}] - \max_{a' \in A^{n+1}} \{\hat{F}^n(a')\} \\
&= \mathbb{E}[\max\{\hat{F}^n(a^1) + Z\hat{\Sigma}^n(a^1; (x, a))), ..., \hat{F}^n(a) + Z\hat{\Sigma}^n(a; (x, a))\}] \\
&\qquad - \max_{a' \in A^n} \{\hat{F}^{n+1}(a')\} \\
&= \mathbb{E}[\max\{c_1 + Zm_1, ..., c_{n+1} + Zm_{n+1}\}]
\end{aligned}$$

where $c_i = \hat{F}^n(a^i) - \max_{a' \in A^{n+1}} \hat{F}^n(a')$ and $m_i = \hat{\Sigma}^n(a^i, (x, a))$ and the average is still computed using the same Algorithm 1 in Chapter 6.3. The equations are a "drop-in" replacement! As with the adapted EGO algorithm, this KG for input uncertainty algorithm also has a single parameter N_X that determines the granularity of the Monte-Carlo integration and can be chosen by the user. In our benchmarks we again set $N_X = n$ so that the accuracy increases with the sample size over the run.

7.3.3 Including the Sampled Input in the Monte-Carlo Integral

The proposed Monte-Carlo integral may be improved by importance sampling by setting $X_{MC} \sim G[x]$ where $G[x]$ is a proposal distribution. For example, $G[x|(x, a)^{n+1}] \propto \mathbb{P}[x]\hat{\Sigma}^n((x, a), (a, x)^{n+1})$. Meaning that in order to minimise error the Monte-Carlo integral should focus samples where the density of input parameters is high, large $\mathbb{P}[x]$, and also where the new function evaluation has great effect on the prediction of other the model at other locations, large $\tilde{\sigma}((x, a), (a, x)^{n+1})$. Instead, the above proposed methods set $X_{MC} \sim \mathbb{P}[x]$, focusing the integration only where $\mathbb{P}[x]$ is high. Alternatively, when using a stationary kernel, one may set X_{MC} to be a cluster around x^{n+1} so that samples are allocated to where $G[x] \propto \tilde{\sigma}((a, x), (a, x)^{n+1})$. However, as discussed in Section 6.3.4, in practice this second approach leads to expensive computation and the EGO and KG functions become less smooth and harder to optimise. In order to appropriately focus the Monte-Carlo integration whilst still being generalisable to any input uncertainty distribution and kernel we therefore propose a third way, a mix of these two possible approaches. Using a standard Monte-Carlo integral as well as the sampled point x^{n+1} which may be seen as a cluster of size 1 focused where $\hat{\Sigma}^n(a, (a^{n+1}, x^{n+1}))$ is likely to be greatest. The sampled input is not a sample from $\mathbb{P}[x]$ therefore simply including the input in the Monte-Carlo sum would lead to bias, for example if $\mathbb{P}[x^{n+1}] = 0$, the sampled input should not be included at all. Therefore we include the sampled input with a unique weight that assumes it is from a single point from a uniform distribution $G[x] = 1/V_X$ where $V_X = \int_X dx$ is the volume of the input parameter domain. Therefore the importance weight is simply $\mathbb{P}[x^{n+1}]/G[x^{n+1}] = \mathbb{P}[x^{n+1}]V_X$. Therefore

we may adjust the Monte-Carlo integrals as follows

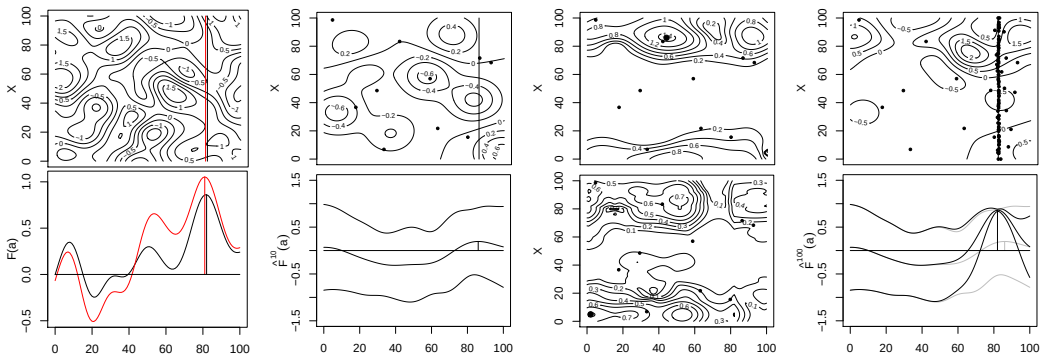
$$\begin{aligned}
\hat{F}^n(a; x^{n+1}) &= \frac{1}{N_X + 1}\left(\sum_{x_i \in X_{MC}} \mu^n(x_i, a) + \mathbb{P}[x^{n+1}] V_x \mu^n(x^{n+1}, a)\right) \\
\hat{\Sigma}^n(a; (x,a)^{n+1}) &= \frac{1}{N_X + 1}\Bigg(\sum_{x_i \in X_{MC}} \tilde{\sigma}^n((x_i, a); (x,a)^{n+1}) \\
&\qquad + \mathbb{P}[x^{n+1}] V_x \tilde{\sigma}^n((x^{n+1}, a); (x,a)^{n+1})\Bigg).
\end{aligned}$$

Secondly, we combine the original Monte-Carlo integral and the single sample Monte-Carlo integral according to their sample size N_X and 1. The modified Monte-Carlo integrals may be directly used in the EGO(x, a) and KG(x, a) and are used for the numerical experiments in Section 7.4.

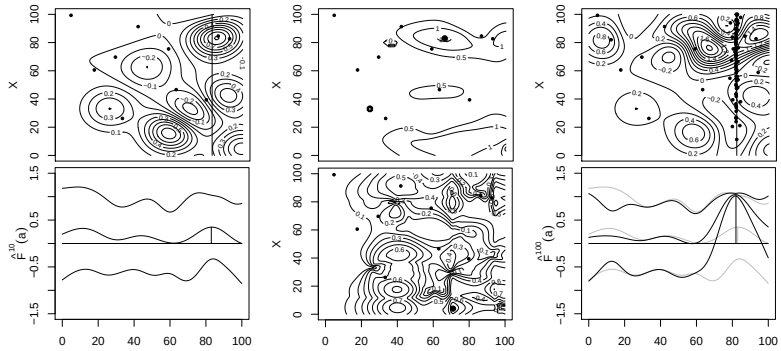
7.4 Numerical Experiments

We apply the new algorithms to two benchmarks based on the same test function but with different assumed distributions of the input. The set of inputs is $X = [0, 100]$, the set of designs is $A = [0, 100]$ and the test function $\theta : X \times A \to \mathbb{R}$. We generate a synthetic test function by sampling from a Gaussian Process with a squared exponential kernel with hyper-parameters $l_X = 10$, $l_A = 10$, $\sigma_0^2 = 1$, $\sigma_\epsilon^2 = (0.1)^2$, the test function $\theta(x, a)$ is shown in Figure 7.1 (a) top. The first input parameter distribution is uniform $\mathbb{P}[x] = 1/100$, thus the sampling procedure must sample across all inputs to learn about the best alternative. The second distribution is a triangular distribution $\mathbb{P}[x] = x\frac{2}{10{,}000}$ such that the mode input is $x = 100$ and the mean input is $x = 66.67$ and the sampling procedure must prioritize high $\mathbb{P}[x]$. Given these input parameter distributions, the true $F(a)$ is calculated via numerical integration and shown in Figure 7.1 (a) bottom.

At the start of sampling, 10 samples are allocated by the random sampling methods described below, the Gaussian process prediction of $\theta(x, a)$ and $\hat{F}^{10}(a)$ after the initial allocation are shown in Figures 7.1 (b) and (e) for the uniform and triangular distribution cases respectively. Then the sequential methods are used to allocate an additional 90 samples reaching a full budget of 100, Figures 7.1 (d) and



(a) $\theta(x,a), F(a)$ (b) UNI: $\mu^{10}(x,a)$ (c) UNI: EGO, KG (d) UNI: $\mu^{100}(x,a)$



(e) TRI: $\mu^{10}(x,a)$ (f) TRI: EGO, KG (g) TRI: $\mu^{100}(x,a)$

Figure 7.1: In all plots, A is on the horizontal axis, small points represent function evaluations. (a) $\theta(x,a)$ and $F(a)$ measured using the uniform test inputs (black) and triangular test inputs (red). $\mu^{10}(x,a)$ and $\hat{F}^{10}(a)$ with upper and lower confidence bounds after 10 samples are shown for uniform inputs (b) and triangular inputs (e). (c) and (f) top EGO(x,a), bottom KG(x,a) after the initial 10 samples with uniform inputs (c) and triangular inputs (f), large points show the peaks. (d) and (g) $\mu^{100}(x,a)$ and $\hat{F}^{100}(a)$ (with $\hat{F}^{10}(a)$ in grey) after 100 samples allocated by EGO, (d) uniform and (g) triangular.

(g) show $\theta(x,a)$ and $\hat{F}^{100}$ after 100 samples have been allocated according to EGO. Then, based on the learned Gaussian Process model, the design a^* with the largest predicted performance over a sample of 1000 inputs, X_R, is recommended to the user

$$a^* = \underset{a'}{\text{argmax}} \;\; \hat{F}^N(a') = \underset{a'}{\text{argmax}} \;\; \frac{1}{1000} \sum_{x_i \in X_R} \mu^N(x_i, a')$$

where $\hat{F}^N(a)$ is optimized by evaluating for all integer values $a \in 0, 1, ..., 100$ and the highest value is used as a seed for sequential parabolic interpolation to find the optimal a^* to high accuracy.

The quality of the sampling procedure is determined by the opportunity cost, the difference in true performance between the design with the highest predicted value and the true best design, which is measured over a separate random sample of 1000 inputs X_{test} that have not been used in the algorithm. The true value of a given alternative is calculated by

$$F(a) = \frac{1}{1000} \sum_{x_i \in X_{test}} \theta(x_i, a).$$

Therefore the opportunity cost is given by

$$\text{Opportunity Cost} \quad = \quad \max_{a'} F(a') - F(a^*). \tag{7.8}$$

Code is available online for all experiments and benchmarks at `http://www2.warwick.ac.uk/fac/cross_fac/complexity/people/students/dtc/students2013/pearce/`.

7.4.1 Benchmark Methods

- *Random Sampling* Given a budget N, samples are randomly distributed over the joint input-design space by Latin Hyper Cube (in the uniform input case) or by sampling from the input distribution and latin hypercube in the A space. We consider this the simplest uninformed brute-force approach.

- *EGO on the mean and mode input* We apply a single parameter EGO to the mean input and to the mode input. This represents a typical approach used in practice, where the input uncertainty is simply reduced to using the most likely or average input parameter value. Technically this is equivalent to using the EGO algorithm described above with only one constant sample in the Monte-Carlo integral. In the uniform case we only use the mean input $x = 50$, and in the triangular case we use the mean $x = 66.67$ and the mode $x = 100$. At the end of sampling the best a on the single input alone is recommended while opportunity cost is measured over all test inputs.

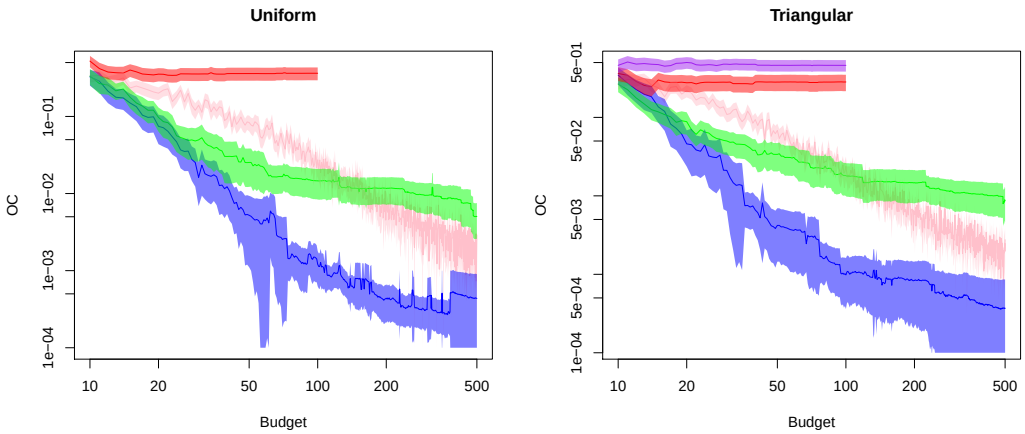


Figure 7.2: the opportunity cost when the input distribution is uniform (left) and when the input distribution is triangular (right). EGO on the mean input (red) and mode input (purple) perform worst and stop early. For small budgets, Random Sampling (pink) which is significantly worse than EGO (green) and KG (blue). For large budgets, even EGO for Input Uncertainty struggles and performs worse than Random.

7.4.2 Results

$\hat{F}^n(a)$ provides a point estimate of the true performance, $\sum \theta(x_i, a)$, averaged over a given set of inputs, X_{MC}, the posterior variance of the estimate is given by

$$\frac{1}{N_X^2} \sum_{x_i \in X_{MC}} \sum_{x_j \in X_{MC}} k^n((x_i, a), (x_j, a))$$

and is plotted in Figure 7.1 as confidence bounds. In Figures 7.1 (d) and (g) it can be seen that samples are focussed around the true optimal a and the error in $\hat{F}^{100}(a)$ is much lower around the optimal a.

In Figure 7.2, in both cases applying EGO to the mean input and the mode input results in the opportunity cost not decreasing to zero and the algorithm converges to the wrong design, so reducing input uncertainty to just a typical input parameter value leads to inferior solutions. In this example, the EGO focusing on the mode input even converges to a solution that is worse than the solution obtained after the initial 10 samples of the methods that take input uncertainty into account.

Of the methods that account for input uncertainty, KG works best. In the

small budget case, EGO outperforms random sampling but EGO does not perform as well in the large budget case. Our modified version of the EGO algorithm requires a fixed current best quality value upon which to improve and we use $\max_a \hat{F}^n(a)$ at point $a^* = \underset{a}{\text{argmax}}\ \hat{F}^n(a)$. However, the predicted value of point a^* is also changed by the new sample at $(x, a)^{n+1}$ and this change is ignored by the EGO method. We saw in Chapter 6.5 that PEI uses EGO and makes the same simplifying assumption. We saw that PEI performs worse than REVI which uses KG and does not make the oversimplifying assumption and performs much better.

7.5 Conclusion and Future Work

When building a simulation model, the user usually faces the challenge to define proper input distributions. Input uncertainty arises when one is not completely certain what distributions and/or parameters to use. In this paper, we proposed two simulation optimization methods based on EGO and KG ideas that are able to take into account input uncertainty, and identify the design that has the best expected performance over the assumed known distribution of input distribution parameters.

Numerical experiments demonstrated that the new algorithms indeed sample the search space very efficiently in the small budget regime where they are more efficient than random sampling. The approach to simply use EGO on a typical input distribution parameter such as the mode or the mean of the assumed distribution clearly performed worse, which demonstrates the importance of properly accounting for input uncertainty. We also saw that EGO with input uncertainty performs worse for large budget and we hypothesize that this is also due to oversimplifying the problem.

There are various avenues for future work. While we assumed in this chapter that the design space and the input distribution parameter space can each be described by a single continuous parameter, the proposed methods should also be tested in higher dimensions and with discrete parameters. It would be interesting to examine the impact of parameter N_x. Finally, one could consider worst case performance rather than expected performance.

In the next Chapter, we extend the KG for input uncertainty to simulation optimisation with common random numbers.

Chapter 8

Bayesian Optimization with Common Random Numbers

8.1 Introduction

We consider the problem of expensive stochastic optimization with limited evaluations,

$$\arg\max_{x \in X} \mathbb{E}[\theta(x, s)] \tag{8.1}$$

where $\theta(x, s)$ is a real valued output, $X \subset \mathbb{R}^d$ is the solution space, usually given by box constraints for continuous variables, or a set of discrete alternatives. The parameter s represents all of the stochasticity in the objective, i.e., $\theta(x, s)$ is deterministic. For example, s may be the seed of a pseudo random number generator that is called within a simulator. Hence evaluating multiple x with the same s will reuse a set of common random numbers (CRN). Alternatively, the seed s and random number stream uniquely define a "scenario" passed to the objective function, and the aim of optimization is to find an $x \in X$ that is the best averaged over all possible randomly generated scenarios. Example applications include

- **Control and Reinforcement Learning:** x are parameters of a control policy, s defines a randomly generated environment (e.g. maze, race track, terrain) and $\theta(x, s)$ is final reward.

- **Machine Learning:** x are hyperparameters of a machine learning algorithm or model, s defines a random split of training data into train and validation, and $\theta(x, s)$ is test set accuracy.
- **Simulation Optimization:** In many optimization problems, a solution can only be evaluated by a stochastic simulator whose seed s we may choose.

In this work we empirically investigate the the following two simulation optimization applications.

- **Inventory Management:** x are target inventory levels below which more stock is ordered, s defines a random stream of customers and $\theta(x, s)$ is profit.
- **Base Location:** x are spatial locations of ambulance bases, s defines times and locations of patients randomly appearing across the map, and $\theta(x, s)$ is average ambulance journey time.

From a surrogate modelling perspective, as a result of using CRN, the noise corrupting the objective output has covariance for outputs with the same seed. This is in contrast to the common assumption of independent noise for the objective outputs. For example, the seed s may influence the difficulty of a randomly generated scenario, and the performance of all solutions $x \in X$ degrades for difficult scenarios and improves for easy scenarios.

Traditionally, CRN has been exploited by considering the reduction in variance of performance differences, $\theta(x, s) - \theta(x', s)$, as CRN typically induces a positive correlation in noise, and

$$\mathrm{Var}(\theta(x, \cdot) - \theta(x', \cdot)) = \mathrm{Var}(\theta(x, \cdot)) + \mathrm{Var}(\theta(x', \cdot)) - 2\mathrm{Cov}(\theta(x, \cdot), \theta(x', \cdot)).$$

There have been several previous works that focus on evaluating pairs of solutions or multiple comparisons either "with CRN" or "without CRN".

In this work we take a different perspective. The domain of the objective is the cross-product of a solution space and a set of synchronised random number stream. We index one set of stream by a positive integer we refer to as the seed, thus the domain of the objective is $X \times \{1, 2, ...\}$. Therefore, the surrogate model

is defined over the same $X \times \mathbb{N}^+$. Similarly, the algorithm needs to propose the next $(x, s) \in X \times \mathbb{N}^+$ and evaluate $\theta(x, s)$. The target of optimisation is to learn $\bar{\theta}(x) = \mathbb{E}[\theta(x, s)]$ hence data is collected to learn $\underset{x}{\text{argmax}}\ \bar{\theta}(x)$. Given this perspective, we emphasize that the benefit in using CRN comes from the *emergent structure in the noise*, i.e., how the output for a single seed is uniquely different from the average over seeds,

$$\epsilon_s(x) = \theta(x, s) - \bar{\theta}(x). \tag{8.2}$$

In particular, if $\epsilon_s(x) = o_s$ is the constant function, this implies that $\underset{x}{\text{argmax}}\ \bar{\theta}(x) = \underset{x}{\text{argmax}}\ \theta(x, s)$ and it is sufficient to optimize a single seed s. Thus, first we propose a Gaussian process model for $\theta(x, s)$ that also yields a method for inferring $\bar{\theta}(x)$ and is a generalization of standard models. Second, we propose the Knowledge Gradient for Common Random Numbers (KG^{CRN}) that quantifies the value of a new point in $X \times \mathbb{N}^+$ for learning the optimizer of the average over infinitely many seeds, $\text{argmax}\ \bar{\theta}(x)$. Optimizing KG^{CRN} determines the most beneficial combination of solution x executed with seed s to efficiently learn $\underset{x}{\text{argmax}}\ \bar{\theta}(x)$. The KG^{CRN} algorithm is therefore able to automatically trade-off the benefits of evaluating x with a previously evaluated seed, thereby utilizing CRN, and of evaluating x with a fresh new seed, by simply maximizing the expected benefit. This removes both the need to observe multiple x simultaneously in a batch with CRN or the need to consider differences in pairs of outputs evaluated with CRN. However, we point out that our KG^{CRN} algorithm can easily be extended to batch acquisition, e.g., using the technique of Wu et al. [2017].

In the following section we formally define the problem, Section 8.2. In Section 8.3, we describe and motivates the proposed surrogate model and Section 8.4 derives the new acquisition procedure and discuses practicalities. In Section 8.5 we draw parallels with a previous approach based on pairwise sampling. An empirical evaluation on both synthetic experiments and two of the applications mentioned above are presented in Section 8.6. The paper concludes in Section 8.7.

8.2 Problem Definition

A user is given an expensive-to-evaluate, real valued function $\theta : X \times \mathbb{N}^+ \to \mathbb{R}$ with arguments composed of a solution $x \in X \subset \mathbb{R}^d$ from *solution space* (the tool a in previous chapters) and a nominal positive integer seed $s \in \mathbb{N}^+$. We refer to $\theta(x, s)$ as the *objective function*. The random seed s controls all stochasticity in the function, i.e., $\theta(x, s)$ is deterministic. The aim is to identify the solution x that maximizes the expectation of the objective over random number streams

$$\arg\max_x \bar{\theta}(x) = \arg\max_x \mathbb{E}[\theta(x, \cdot)]$$

and we refer to $\bar{\theta}(x)$ as the *target*. There is a limited budget of N objective function calls, and for each call, the user can choose an *input pair* (x, s) from the *acquisition space*, then observe $y = \theta(x, s)$. Objective function evaluations may be collected sequentially so that after n measurements the user may determine the x and s for the $(n + 1)^{th}$ function evaluation.

If every call to the function uses a new unique random seed, the problem reduces to standard stochastic optimization and the user only needs to determine x values for each evaluation of $\theta(x, s)$. The problem considered here is therefore a more general setting that allows the reuse of random number seeds by making the argument s explicit.

8.3 A Surrogate Model for Simulation with Common Random Numbers

Given a budget of N calls to $\theta(x, s)$, the proposed Bayesian optimization algorithm has two phases, an initialization phase where we evaluate a small number of candidates $n_{init} \ll N$, chosen as a space filling design in $X \times \{1, 2, 3, 4, 5\}$. That is, we instantiate five (randomly chosen) seeds to collect data points that are then used to fit a Gaussian process model. The GP model is combined with an acquisition function (infill criteria) to sequentially allocate the remaining $N - n_{init}$ points of the budget, updating the model after each new point and determining the next point. We first describe the

model of $\theta(x, s)$ and then propose the Knowledge Gradient for Common Random Numbers in Section 8.4.

8.3.1 The Gaussian Process Generative Model

A generative model is a probability distribution over all observable and unobservable quantities and such a model can be sampled to generate realizations of all variables thereby synthesizing data. Inference is the task of estimating the unobserved variables that are consistent with the generative model and the observed quantities. In the case of optimization with CRN, we desire a generative model with two properties. First, sampling outputs from the generative model assuming each output comes from a different seed must recover a model used without CRN. Second, the seeds are labeled with arbitrary numbers, in particular, there is no exploitable "neighborhood" between integer seed values. Alternatively, there might be similarity in the random number streams. However such an approach requires specifying a stream feature extractor and knowing the distribution of stream features in the corresponding feature space. The integer seed approach has fewer specifications and is therefore a strictly more general setting.

Following previous works without CRN, we first assume that the target, $\bar{\theta}(x)$, is a realization of a Gaussian process with constant prior mean and covariance given by a kernel such as a $\frac{5}{2}$-Matérn or squared exponential,

$$\bar{\theta}(x) \sim \mathrm{GP}\big(\,\bar{\mu},\ k_{\bar{\theta}}(x, x')\,\big). \tag{8.3}$$

When all seeds are unique, e.g., $s^i = i$, output y values are generated by adding independent and identically distributed Gaussian noise $y \sim N(\bar{\theta}(x), \sigma^2_\epsilon(x))$. Given n solutions $X^n = (x^1, ..., x^n)$, the vector of outputs, $Y^n = (\theta(x^1, 1), ..., \theta(x^n, n))$, is assumed to be a single multivariate Gaussian random vector with constant mean $\bar{\mu}$ and a covariance matrix composed of a kernel matrix and diagonal noise matrix

$$Y^n \sim N\big(\,\bar{\mu},\ k_{\bar{\theta}}(X^n, X^n) + \mathrm{diag}(\sigma^2_\epsilon(X^n))\,\big). \tag{8.4}$$

For $\theta(x, s)$ in the CRN setting, we require a kernel over acquisition space $\tilde{X} = X \times \mathbb{N}^+$

that when evaluated for unique seeds recovers the above covariance matrix. To satisfy all zero off-diagonal elements for unequal seeds, we require a Kronecker delta function over seeds (white noise), to model covariance in outputs for the same seed we require another kernel over $X \times X$. We propose the following model for the objective,

$$\theta(x, s) \sim \mathrm{GP}\big(\, \bar{\mu},\ k_{\bar{\theta}}(x, x') + \delta_{s's} k_\epsilon(x, x')\big) \tag{8.5}$$

where $k_\epsilon(x, x')$ is the kernel of the *difference function* between the target and the objective function for a particular seed and must also must satisfy $k_\epsilon(x, x) = \sigma^2_\epsilon(x)$. We return to design of $k_\epsilon(x, x')$ shortly. $\mu^0(x, s) = \bar{\mu}$ is the constant prior mean. Given a tuple of input pairs $\tilde{X}^n = ((x, s)^1, ..., (x, s)^n)$, the generative distribution of Y^n is thus

$$Y^n \sim N\big(\, \bar{\mu},\ k_{\bar{\theta}}(X^n, X^n) + \mathbb{1}_{S^n} \circ k_\epsilon(X^n, X^n)\,\big) \tag{8.6}$$

where $\circ$ denotes matrix element-wise (Hadamard) product and $\mathbb{1}_{S^n} \in [0, 1]^{n \times n}$ is a binary masking matrix with elements equal to one at i, j when $s^i = s^j$. Hence for the noise matrix, $\mathbb{1}_{S^n} \circ k_\epsilon(X^n, X^n)$, the diagonal and also any off-diagonal pairs where $s^i = s^j$ are non-zero with corresponding covariance $k_\epsilon(x^i, x^j)$. The model encodes the functional form of the objective as target and *difference functions*, $\epsilon_s(x)$,

$$\theta(x, s) \quad = \quad \bar{\theta}(x) + \epsilon_s(x) \tag{8.7}$$

where the $\epsilon_s(x)$ are independent and identically distributed GP realizations

$$\epsilon_1(x), \epsilon_2(x), ... \sim \mathrm{GP}\big(\, 0,\ k_\epsilon(x, x')\,\big). \tag{8.8}$$

This model structure has multiple desirable properties. Firstly, by design it mirrors the standard model assumed for non-CRN use cases, $y = \bar{\theta}(x) + \epsilon$, where it is commonly assumed that all ϵ are independent Gaussian *variable* realizations. With CRN, the "noise" terms $\epsilon_s(x)$ are independent Gaussian *process* realizations. Secondly, $k_\epsilon(x, x')$ dictates the covariance in differences from the target at x and x' induced by CRN that may be chosen by the user for a given application, we discuss our model next. Thirdly, $k_\epsilon(x, x')$ is typically a parametric function whose hyperparameters are

learnt from multiple realizations, $\epsilon_1(x), \epsilon_2(x), ...$, of a single GP and each seed may be viewed as a task in a multi-task model. This differs slightly from other multi-task models commonly used for multi-fidelity optimization, [Poloczek et al., 2017, Swersky et al., 2013], or for multi-objective optimization, [Picheny, 2015], where one task is not necessarily the same as others and a unique GP model for each task may be more suitable. However, because all $\epsilon_s(x)$ come from a single common GP, the kernel $k_\epsilon(x, x')$ must have the flexibility to model how the objective for any seed may differ from the target. We assume a decomposition of the difference functions, $\epsilon_s(x)$, into three parts: a constant offset o_s, a bias function $b_s(\cdot)$, and white noise $w_s(\cdot)$:

$$\theta(x, s) \quad = \quad \bar{\theta}(x) + \epsilon_s(x) \quad = \quad \bar{\theta}(x) + o_s + b_s(x) + w_s(x). \tag{8.9}$$

Firstly, to capture the notion that some seeds may result in scenarios that are "easy" and others "hard" for all solutions x, $\epsilon_s(x)$ may contain a global offset modeled by the constant kernel,

$$o_s(x) \sim \mathrm{GP}(0, k(x, x') = \eta^2), \tag{8.10}$$

where the sample function is constant for all x and hence denoted by $o_s \sim N(0, \eta^2)$. Secondly, to capture the notion that similar solutions should have similar outputs given the same seed, we include a "bias" function modelled with another Matérn or squared exponential kernel,

$$b_s(x) \sim \mathrm{GP}\left(0, k(x, x') = k_b(x, x')\right). \tag{8.11}$$

Thirdly, to capture any other effects not modelled by o_s and $b_s(x)$, such as discontinuities, we follow Chen et al. [2012] and Xie et al. [2016] and include a realization of white noise

$$w_s(x) \sim \mathrm{GP}\left(0, k(x, x') = \delta_{x'x}\sigma_w^2\right). \tag{8.12}$$

Therefore, this functional form of $\theta(x, s)$ is a realization of the Gaussian process

$$\theta(x, s) \quad \sim \quad \mathrm{GP}\big(\bar{\mu},\ k_{\bar{\theta}}(x, x') + \delta_{ss'}(\eta^2 + k_b(x, x') + \sigma_w^2\delta_{xx'})\big) \tag{8.13}$$

$$\quad = \quad \mathrm{GP}\big(\bar{\mu},\ k(x, s, x', s')\big). \tag{8.14}$$

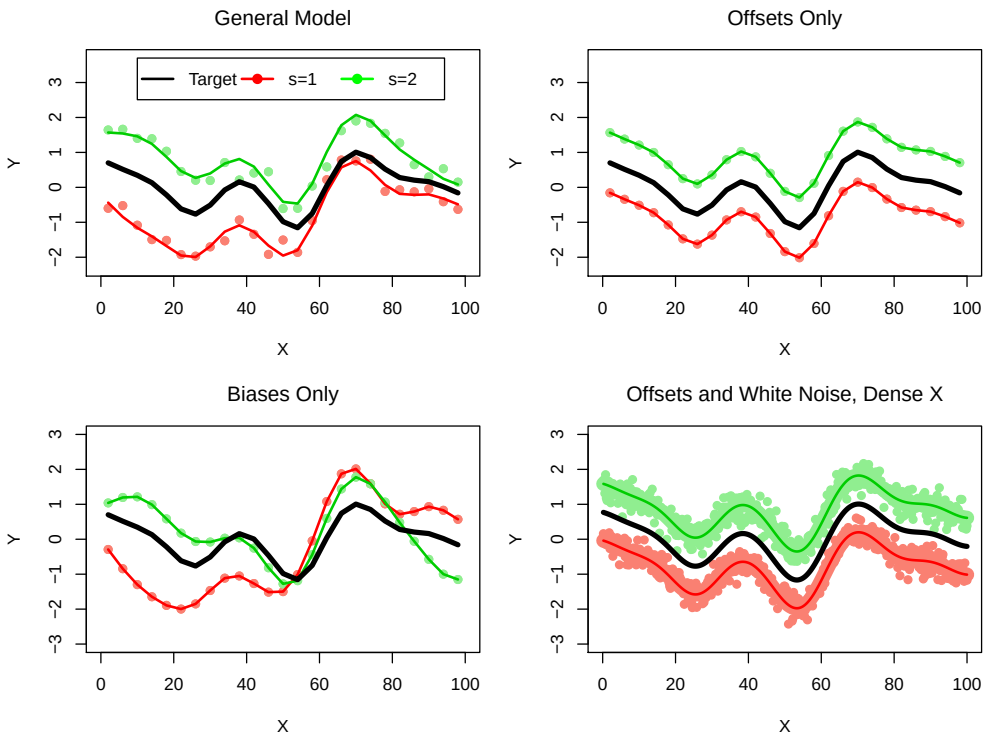


Figure 8.1: Samples from the generative model. In all plots, lines show $\bar{\theta}(x)$ and $\bar{\theta}(x) + o_s + b_s(x)$ (no white noise), points show $\theta(x, s)$ (including white noise). Left plots: an algorithm must evaluate multiple seeds to find optimum. Right plots: an algorithm can optimize one seed to find $\arg\max \bar{\theta}(x)$.

See Figure 8.1 for example realizations. Although this is a general model, to simplify parameter learning in practice we assume parameter sharing between $k_{\bar{\theta}}(x, x')$ and $k_b(x, x')$ such that a CRN model has only two more hyperparameters than its corresponding non-CRN model. We discuss in more detail in Section 8.4.2. For the rest of this section, we assume that all kernels are known functions and the unknown $\theta(x, s)$ are to be inferred.

8.3.2 Inferring the Objective $\theta(x, s)$

We denote an observation at time n as (x^n, s^n, y^n), the sequence of observed solutions as $(x^1, ..., x^n) = X^n$, the sequence of observed seed values as S^n and the sequence of input pairs, $\tilde{x}^i = (x^i, s^i)$, as $(\tilde{x}^1, , ..., \tilde{x}^n) = \tilde{X}^n$. The vector of observed outputs is denoted $(y^1, ..., y^n) = Y^n$. And, abusing notation, we also treat these as sets e.g. $\tilde{x} \in \tilde{X}^n$, and use both (x, s) and $\tilde{x}$ interchangeably to represent an input pair.

The dataset of observed inputs and outputs we denote $D^n = ((\tilde{x}^1, y^1), ..., (\tilde{x}^n, y^n))$. Inferring the underlying realization of $\theta(x, s)$ can be done analytically using the Bayesian update equations for multivariate Gaussian random variables,

$$\theta(x,s)|D^n \sim \text{GP}\big(\mu^n(x,s),\ k^n(x,s,x's')\big) \tag{8.15}$$

$$\mu^n(x,s) = \mu^0(x,s) - k^0(x,s,\tilde{X}^n)K^{-1}(Y^n - \mu^0(\tilde{X}^n)) \tag{8.16}$$

$$k^n(x,s,x',s') = k^0(x,s,x',s') - k^0(x,s,\tilde{X}^n)K^{-1}k^0(\tilde{X}^n,x',s') \tag{8.17}$$

where $k^0(x, s, x's')$ is any positive semi-definite kernel over $X \times \mathbb{N}^+$. The matrix $K = k^0(\tilde{X}^n, \tilde{X}^n)$ is the generative covariance for Y^n. Note that there is no added identity matrix as in Equation 8.4 thus the model assumes deterministic outputs for any given input pair (x, s). Intuitively, the white noise kernel is the squared exponential kernel with an infinitely short length scale and observations at one location do not inform predictions at any other location. The posterior mean predicts a sum of GP realizations $\mu^n(x, s) = \mathbb{E}_n\left[\bar{\theta}(x) + o_s + b_s(x) + w_s(x)\right]$. At observed input pairs, $(x^i, s^i) \in \tilde{X}^n$, the predicted white noise realization is informed by data and $\mathbb{E}_n[w_{s^i}(x^i)] \neq 0$ (almost surely), while at unobserved input pairs $\mathbb{E}_n[w_s(x)] = 0$, returning the prior of the white noise GP. As a result, the posterior mean discontinuously interpolates the data as shown in Figure 8.2.

8.3.3 Inferring the Target $\bar{\theta}(x)$

The aim of the optimization is to maximize $\bar{\theta}(x)$ over solution space X however the model of $\theta(x, s)$ and collected data is over the acquisition space $X \times \mathbb{N}^+$. The expectation of the objective is also the average over infinite seeds and therefore the model of $\theta(x, s)$ induces another GP over the target $\bar{\theta}(x)$ as follows.

Proposition 8.3.0.1 *For any given kernel over the domain $X \times \mathbb{N}^+$ that is of the form $k_{\bar{\theta}}(x, x') + \delta_{ss'}k_\epsilon(x, x')$, and a dataset of n input-output triplets D^n, the posterior*

over the target is a Gaussian process given by

$$\bar{\theta}(x)|D^n \sim GP(\mu^n_{\bar{\theta}}(x), k^n_{\bar{\theta}}(x,x')) \tag{8.18}$$

$$\mu^n_{\bar{\theta}}(x) = \mu^n(x,s') \tag{8.19}$$

$$k^n_{\bar{\theta}}(x,x') = k^n(x,s',x',s'') \tag{8.20}$$

where $s', s'' \in \mathbb{N}^+ \setminus S^n$ *with* $s' \neq s''$ *any two unobserved unequal seeds.*

The intermediate steps and proof are given in Appendix B.0.1. For the sake of a simple notation, we assume that seeds are labeled by positive integers, and let $s' = 0$ and $s'' = -1$. Then $\mu^n(x, 0)$ is the posterior expectation of the target $\bar{\theta}(x)$.

8.4 Knowledge Gradient for Common Random Numbers

8.4.1 Acquisition Function

Evaluations of $\theta : X \times \mathbb{N}^+ \to \mathbb{R}$ are collected in order to optimize $\bar{\theta} : X \to \mathbb{R}$. Given a model of both functions, the acquisition function quantifies the value of a new hypothetical observation at (x, s). This is then optimized to obtain the best $(x, s)^{n+1}$ and the objective is evaluated $y^{n+1} = \theta(x^{n+1}, s^{n+1})$. The surrogate model is defined over the space of non-negative seeds $X \times \{0, 1, ...\}$, the target of optimization is over the space $X \times \{0\}$ and the acquisition is over the space $X \times \{1, 2, ..\}$. Therefore we require a 'correlation aware' acquisition function that computes the value of a sample at $(x, s)^{n+1}$ for $s^{n+1} > 0$ by measuring changes in the model at other locations $(x', 0) \neq (x, s)^{n+1}$. This requirement excludes certain acquisition functions in their unmodified form such as Expected Improvement [Jones et al., 1998b], Upper Confidence Bound [Srinivas et al., 2009] and Thompson sampling [Kandasamy et al., 2018]. Two popular families of acquisition functions that naturally account for how the whole surrogate model changes include Entropy Search [Villemonteix et al., 2009b], and Correlated Knowledge Gradient [Scott et al., 2011b]. Entropy Search measures mutual information between the distribution of the next output, $\mathbb{P}[y^{n+1}|D^n, x^{n+1}]$, and the induced distribution of the location of the maximizer $\mathbb{P}[x^*|D^n]$. Correlated

Knowledge Gradient measures the expected incremental increase in the predicted outcome for the user, peak posterior mean $\mathbb{E}[\max_x \mu^{n+1}(x) - \max_x \mu^n(x)|D^n, x^{n+1}]$. In this work we adopt the Knowledge Gradient for its principled Value of Information derivation and provable guarantees.

In the use case we consider, the value of information is the expected increase in the predicted peak of the target, $\max \mu^{n+1}(x, 0) - \max \mu^n(x, 0)$, caused by a new sample $(x, s)^{n+1}$. For the rest of this work, we use the shorthand $\mathbb{E}_n[\cdot] = \mathbb{E}[\cdot|D^n]$. The Knowledge Gradient for Common Random Numbers is given by

$$\begin{aligned} \mathrm{KG}_n^{\mathrm{CRN}}(x, s) &= \mathbb{E}_n\left[\max_{x' \in X} \mu^{n+1}(x', 0) - \max_{x'' \in X} \mu^n(x'', 0) \middle| (x, s)^{n+1} = (x, s)\right] \\ &= \mathbb{E}_n\left[\max_{x' \in X} \mu^n(x', 0) + \tilde{\sigma}^n(x', 0; x, s)Z - \max_{x'' \in X} \mu^n(x'', 0)\right] \quad (8.21) \end{aligned}$$

where, conditioned on D^n, the expectation is only over $Z \sim N(0, 1)$ and

$$\tilde{\sigma}^n(x, 0; (x, s)^{n+1}) = \frac{k^n(x, 0, (x, s)^{n+1})}{\sqrt{k^n((x, s)^{n+1}, (x, s)^{n+1})}}.$$

A full derivation can be found in multiple previous works [Frazier et al., 2009b, Pearce and Branke, 2017a]. The next input to the objective, $(x, s)^{n+1}$, is determined by optimizing the above acquisition function $(x, s)^{n+1} = \arg\max_{x,s} \mathrm{KG}_n^{\mathrm{CRN}}(x, s)$. Evaluation of $\mathrm{KG}^{\mathrm{CRN}}$ is the expectation of a maximization and can be evaluated analytically when X is a finite set, however in general approximations are required which are discussed in Section 8.4.2 and we discuss numerically finding $\arg\max_{x,s} \mathrm{KG}_n^{\mathrm{CRN}}(x, s)$ in Section 8.4.2.

The acquisition space, $X \times \mathbb{N}^+$, contains an infinite number of seeds. However as a result of the assumed form of the GP, the posterior mean and correlation are identical for all *unobserved* seeds $s \in \mathbb{N}^+ \setminus S^n$. Thus, $\mu^n(s, 0)$ can be used as the target estimate and also the value under the acquisition criterion is identical for all new seeds, $\mathrm{KG}_n^{\mathrm{CRN}}(x, s) = \mathrm{KG}_n^{\mathrm{CRN}}(x, s')$ for all $s, s' \in \mathbb{N}^+ \setminus S^n$. Thus, it suffices to evaluate the acquisition criterion on all observed seeds $s \in S^n$ and only a single new seed $s = \max\{S^n\} + 1$. Over multiple iterations, new seeds may be evaluated and added to the set of observed seeds and the acquisition space grows accordingly by

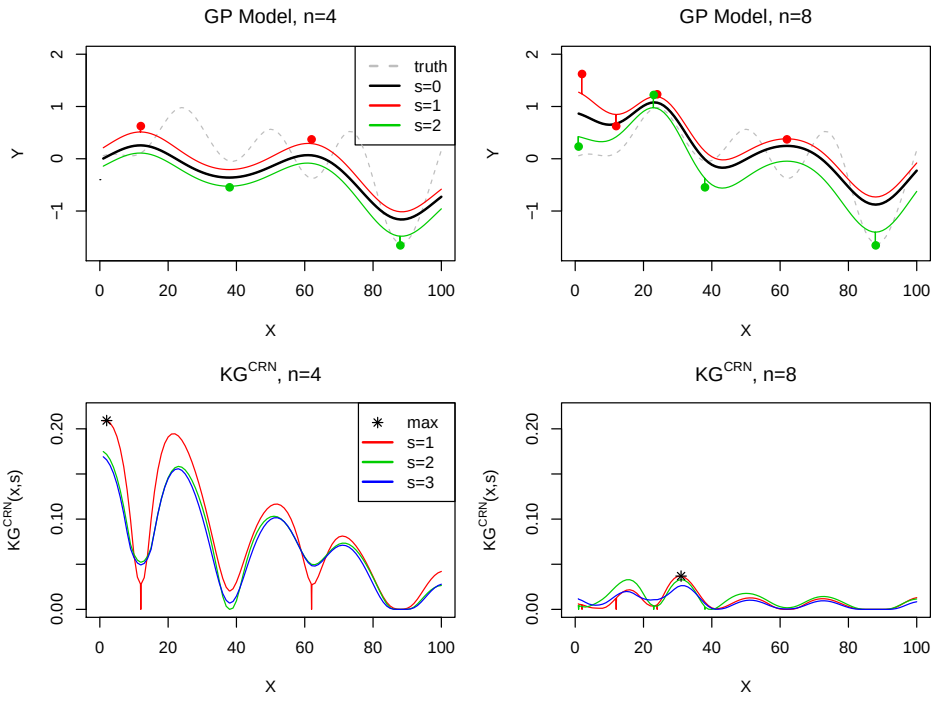


Figure 8.2: (TL, TR) The GP model with offsets, bias functions and white noise. The model discontinuously interpolates the white noise. (BL, BR) $\mathrm{KG}^{\mathrm{CRN}}$ after 4 initial points on seeds $s = 1, 2$ (L) and an added 4 sequential points by $\mathrm{KG}^{\mathrm{CRN}}$ (R). All new points were allocated to seeds $s = 1, 2$ and the next point will be allocated to $s = 1$.

always including one new seed. Note that that the acquisition criterion is maximized jointly over the old and new seeds. In particular, no heuristic or user specified controls are used to make the exploration-exploitation trade-off over old and new seeds.

A connection can be drawn between our algorithm and recent work on multi-information source optimization [Poloczek et al., 2017, Swersky et al., 2013]. At a given iteration, each seed in the acquisition space may be viewed as an information source and $s = 0$ is the target, and a user must choose a solution x and an information source s in order to optimize a target $s = 0$. However in the CRN case, the target itself cannot be observed, all tasks have equal budget consumption, and the number of available sources is infinite.

8.4.2 Implementation Practicalities

In Section 8.3 we assumed that $k_{\bar{\theta}}(x, x')$ and $k_{\epsilon}(x, x')$ are known. Note that in practice their hyperparmeters are estimated from data. Also in Section 8.4 we assume $\mathrm{KG}_n^{\mathrm{CRN}}(x, s)$ can be evaluated and maximized however there is computational cost. In practice, we desire a CRN method that does not introduce an unrealistic computational requirement when compared to an equivalent non-CRN method and in this method we discuss such solutions.

Gaussian Process Hyperparameters.

In this work we assume that the target is modeled with the popular squared exponential (SE) kernel

$$k_{\bar{\theta}}(x, x') = \sigma_{\bar{\theta}}^2 \exp(-(x - x')^\intercal L(x - x')/2)$$

where $L = \mathrm{diag}(1/l_1^2, ..., 1/l_d^2)$ is a diagonal matrix of inverse length scales. We also assume that the bias functions come from a squared exponential kernel $k_b(x, x') = \sigma_b^2 \exp(-(x - x')^\intercal L(x - x')/2)$ that shares the diagonal matrix L. For any $k_{\bar{\theta}}(x, x')$, one may simply use $k_b(x, x') \propto k_{\bar{\theta}}(x, x')$ where the ratio is a hyperparameter. The constant kernel and white noise kernel each have a single parameter η^2 and σ_w^2. The CRN model has parameters $L, \sigma_{\bar{\theta}}, \eta^2, \sigma_b^2, \sigma_w^2$, two more than a non-CRN model. All parameters are learnt by maximizing the marginal likelihood by first learning a non-CRN model (i.e. clamping $\eta^2 = \sigma_b^2 = 0$). However, the added computation for the CRN model comes from fine tuning the hyperparameters of the difference kernel, η^2, σ_b^2, keeping total noise constant to the learnt noise of the non-CRN model $\eta^2 + \sigma_b^2 + \sigma_w^2 = \sigma_{w,\text{non-CRN}}^2$. We then perform further fine tuning of all parameters simultaneously. For details, see the Appendix B.2. In future work, especially with more complex models, we will study a Bayesian treatment of the hyperparameters: such an approach that can improve algorithm performance especially for very small budgets when hyperparameters are most uncertain.

Evaluation of $\mathbf{KG}_n^{\mathbf{CRN}}(x,s)$.

The acquisition function, Equation 8.21, is composed of an expectation over Z of maximizations over X. This may be evaluated analytically when X is a feasibly small finite set using Algorithm 1 from Frazier et al. [2009b]. Alternatively, when X is a continuous set, one may replace the expectation over the infinite Z with a Monte-Carlo average. For each Z sample, the inner maximization is performed over X numerically, yielding a stochastic unbiased estimate of $\mathrm{KG}_n^{\mathrm{CRN}}(x,s)$ [Wu et al., 2017].

In this work, we follow Poloczek et al. [2017] and Xie et al. [2016] that use a deterministic approximation. This allows us to reliably test a conjecture and allows direct comparison with prior work both described in Section 8.5.2. The inner maximization over X may be replaced with a smaller random subset A that is frozen between iterations thus approximating $\mathrm{KG}^{\mathrm{CRN}}$ with

$$\mathrm{KG}_n^{\mathrm{CRN}}(x,s;A) = \mathbb{E}_n\left[\max_{x'\in A\cup\{x\}} \mu^n(x',0) + \tilde{\sigma}^n(x',0;x,s)Z - \max_{x''\in A\cup\{x\}} \mu(x'',0)\right] \quad (8.22)$$

The random subset is union of a latin hypercube over X with n points, A_{LHC}^n, and random perturbations of previously sampled points $A_P^n = \{x^i + \gamma | x^i \in X^n\}$ where $\gamma \sim N(\underline{0}, I)$ is Gaussian noise scaled for the application at hand. Finally, we let $A^n = A_{LHC}^n \cup A_P^n$.

Optimization over the Acquisition Space.

In general, KG is a multi-modal differentiable acquisition function over X and typically maximized by applying random search followed by using the best points for multi-start gradient ascent. For $\mathrm{KG}_n^{\mathrm{CRN}}(x,s)$, the acquisition space is $\tilde{X}_{acq}^n = X \times \{1, ..., \max S^n + 1\}$, suggesting $\mathrm{KG}_n^{\mathrm{CRN}}(x,s)$ needs to be independently optimized over X for each s. However, recall the fundamental CRN modelling assumption that all seeds have the same latent $\bar{\theta}(x)$ or "backbone". As a result, $\mathrm{KG}_n^{\mathrm{CRN}}(x,s)$ for each seed typically has peaks and troughs in similar locations, see Figure 8.2. Therefore, to maximize $\mathrm{KG}_n^{\mathrm{CRN}}(x,s)$, we use the same initial random search budget however distributed over $\tilde{X}_{acq}^n$ and the best points are used for gradient ascent over

X with the corresponding seed fixed. The added computation unique to CRN is as follows, the best (x, s) is evaluated for all s and finally x is fine-tuned with the best s fixed. Consequently, the cost of acquisition optimization for CRN is only marginally greater than for non-CRN.

When the solution space X is discrete, such as integers, one may simply perform all operations in continuous space and round any x values to their corresponding nearest neighbors in X when necessary, x_r^n, x^{n+1}, A^n.

Algorithm 3 The KG^{CRN} Algorithm.

Require: $\theta(x, s)$, X, n_{init}, N, $k_{\bar{\theta}}(x, x')$, algorithm to evaluate $\mathbb{E}[\{\max_{x'} a(x') + b(x', x)Z\}]$ and $\nabla_x \mathbb{E}[\{\max_{x'} a(x') + b(x', x)Z\}]$, `Optimizer()` over $X \times \mathbb{N}^+$

$\tilde{X}^{n_{init}} \leftarrow n_{init}$ sampled points by LHC over $X \times \{1, 2, 3, 4, 5\}$
$Y^{n_{init}} \leftarrow \theta(\tilde{X}^{n_{init}})$

for $n = n_{init}$ **to** $N - 1$ **do**

$\mu^n(x, s), k^n(x, s, x', s') \leftarrow \text{GP}(\theta(x, s) | \tilde{X}^n, Y^n, L, \sigma_b^2, \eta^2, \sigma_b^2, \sigma_w^2)$
$\text{KG}_n^{\text{CRN}}(x, s) \leftarrow \mathbb{E}[\{\max_{x'} \mu^n(x', 0) + \tilde{\sigma}^n(x', 0, x, s)Z\}] - \max_{x''} \mu^n(x'', 0)$

$(x, s)^{n+1} \leftarrow$`Optimizer`$(\text{KG}_n^{\text{CRN}}(x, s))$

$y^{n+1} \leftarrow \theta(x^{n+1}, s^{n+1})$
$\tilde{X}^{n+1}, Y^{n+1} \leftarrow (\tilde{X}^n, (x, s)^{n+1}), (Y^n, y^{n+1})$

end for

$\mu^N(x, s) \leftarrow \text{GP}(\theta(x, s) | \tilde{X}^N, Y^N, L, \sigma_b^2, \eta^2, \sigma_b^2, \sigma_w^2)$

return $x_r^N = \underset{x}{\text{argmax}}\ \mu^N(x, 0)$

8.4.3 Algorithm Properties

The value of the information by sampling decision x with seed s is the expected gain in the quality of the best decision that can be selected given all the available information. In this regard, the KG^{CRN} is myopically optimal by construction. The following observation is trivial yet worth highlighting: standard Knowledge Gradient (KG) is reproduced by artificially constraining KG^{CRN} to only acquire data for a

new seed in each iteration. Thus, we have

$$\max_{x,s\in\mathbb{N}^+} \mathrm{KG}_n^{\mathrm{CRN}}(x,s) \geq \max_{x,s\in\mathbb{N}^+\setminus S^n} \mathrm{KG}_n^{\mathrm{CRN}}(x,s) = \max_x \mathrm{KG}(x) \tag{8.23}$$

thus sampling without any CRN may be viewed as a lower bound on the information gain achievable by $\mathrm{KG}^{\mathrm{CRN}}$.

Given an infinite budget, it is a desirable property for any algorithm to be able to discover the true optimum $x^{OPT} = \underset{x\in X}{\operatorname{argmax}}\, \bar{\theta}(x)$ (assuming the optimizer is unique). Here we give an additive bound on the loss when applying $\mathrm{KG}^{\mathrm{CRN}}$ to a finite subset, A, of continuous space X. Let $k_{\bar{\theta}}(x,x')$ be a Matérn class kernel, and $d = \max_{x'\in X} \min_{x\in A} \mathrm{dist}(x,x')$ the largest distance from any point in the continuous domain X to its nearest neighbor in A.

Theorem 8.4.1 *Let $x_r^N \in A$ be the point that KG^{CRN} recommends in iteration N. For each $p \in [0,1)$, there is a constant K_p such that with probability p*

$$\lim_{N\to\infty} \bar{\theta}(x_N^r) > \bar{\theta}(x^{OPT}) - K_p d$$

Proof is given in Appendix B.0.2. Note that this establishes consistency for the finite case as $A = X$ and $d = 0$. Note that this bound is conservative as A is randomized at each iteration to avoid "overfitting".

8.5 Comparison with Previous Work

8.5.1 Compound Sphericity

We show how to recover the generative model considered by Xie et al. [2016] and Chen et al. [2012] as a special case of our proposed model. If there are no bias functions, $k_b(x,x') = 0$, the differences kernel reduces to $k_\epsilon(x,x') = \eta^2 + \sigma_w^2\delta_{xx'}$ and each difference function $\epsilon_s(x)$ is an offset and white noise. Thus, the differences matrix $k_\epsilon(X^n,X^n)$ is $\eta^2+\sigma_w^2$ on the diagonal and constant η^2 for all off-diagonal terms, this matrix composition is referred to as compound sphericity. The correlation in differences may be written as $\rho = \eta^2/(\eta^2+\sigma_w^2)$. Let $\Delta^n = Y^n - \mu^0(\tilde{X}^n)$ and

$\mathbb{1}_s = \mathbb{1}_{s \in S^n} \in \{0,1\}^n$ be a binary masking vector. $\mathbb{1}_x$ is defined analogously. Then the posterior mean has the following simple form:

$$\begin{aligned}
\mu^n(x,s) &= \mu^0(x) - (k_{\bar{\theta}}(x, X^n) + \eta^2 \mathbb{1}_s + \sigma_w^2 \mathbb{1}_s \mathbb{1}_x) K^{-1} \Delta^n \\
&= \underbrace{k_{\bar{\theta}}(x, X^n) K^{-1} \Delta^n}_{\mu^n(x,0)} + \underbrace{\eta^2 \mathbb{1}_s K^{-1} \Delta^n}_{\text{independent of } x} + \underbrace{\sigma_w^2 \mathbb{1}_s \mathbb{1}_x K^{-1} \Delta^n}_{=0 \text{ except for } (x^i, s^i) \in \tilde{X}^n} \\
&= \mu^n(x,0) + A_s + B_s \mathbb{1}_{(x,s) \in \tilde{X}^n} \qquad (8.24)
\end{aligned}$$

and the posterior mean function for a given seed, $s > 0$, differs from the target, $s = 0$, by two additive terms. The first is a constant A_s and the second is non-zero for singletons $(x, s) \in \tilde{X}^n$. This leads to the following two Lemmas, both cases correspond to the second additive term equating to zero. Firstly, if there is no white noise ($\sigma_w^2 = 0$) then for all seeds $\epsilon_s(x) = o_s$ is only a constant offset and a user may simply optimize a single seed to learn $\arg\max \bar{\theta}(x)$. This corresponds to compound sphericity with full correlation, $\rho = 1$, and may be viewed as a "best case" scenario for CRN.

Lemma 8.5.1 *Let the function $\theta(x, s)$ be a realization of a Gaussian process with compound sphericity with full correlation, $\rho = 1$. Then for all $s \in \mathbb{N}^+$, the posterior mean functions have the same optimizer as the target estimate*

$$\arg\max_{x \in X} \mathbb{E}_n[\bar{\theta}(x)] = \arg\max_{x \in X} \mu^n(x, s') \qquad \forall s' \in \mathbb{N}^+.$$

Proof By setting $\sigma_w^2 = B_s = 0$ in Equation 8.24, the posterior means for all seeds differ by only an additive constant, A_s, therefore the maximizer of any two seeds is the same and by Proposition 8.3.0.1 the same maximizer as the estimate of $\mathbb{E}_n[\bar{\theta}(x)]$. □

Secondly, when there is white noise and the set of solutions X is large and dense, a user may simply optimize a single seed to learn $\arg\max \bar{\theta}(x)$ as above.

Lemma 8.5.2 *Let the function $\theta(x, s)$ be a realization of a Gaussian process with compound sphericity over a continuous set of solutions X, then for all $s \in \mathbb{N}^+$,*

the posterior mean functions have the same optimizer excluding past observation singletons $\tilde{X}^n$

$$\arg\max_{x \in X \setminus X^n} \mathbb{E}_n[\bar{\theta}(x)] = \arg\max_{x \in X \setminus X^n} \mu^n(x, s') \qquad \forall s' \in \mathbb{N}^+.$$

Proof By excluding singletons $x \in X^n$, the second additive term in Equation 8.24 vanishes ($B_s \mathbb{1}_{(x,s) \in \tilde{X}^n} = 0$). The posterior means for all seeds differ by only an additive constant, A_s, therefore the maximizer of any two seeds is the same and by Proposition 8.3.0.1 the same as $\mathbb{E}_n[\bar{\theta}(x)]$. □

The right column of Figure 8.1 illustrates example functions for these cases and top row of Figure 8.2 shows how the posterior mean is discontinuous at evaluated points. If there are no bias functions and discontinuities are excluded, the posterior mean is has the same shape for all seeds. Consequently, for a function that is a realization of a GP with the compound spheric noise model, if there is high correlation or a large and dense number of solutions X, allocating samples to a single seed can be much more efficient than allocating to multiple seeds. This result agrees with those found by Chen et al. [2012]: in the case $\rho = 1$ with data collected on seed $s = 1$, the intercept of the function $\bar{\theta}(x)$ is less accurately known while derivatives $\nabla_x \bar{\theta}(x)$ are more accurately known. This is because in the $\rho = 1$ case, the generative modelling assumption imposes the functional form as $\theta(x, s) = \bar{\theta}(x) + o_s$ implying $\nabla_x \theta(x, s) = \nabla_x \bar{\theta}(x)$.

It is due to the presence of the *bias* functions, $b_s(x)$, that the optimizer of one seed, $\arg\max_x \theta(x, s)$, is not an accurate estimate of the optimizer of the target function, $\arg\max x\bar{\theta}(x)$, and an optimization algorithm must evaluate multiple seeds.

Next, in Lemma 8.5.3 we show that if all solutions of a finite set X have been evaluated there is nothing left to gain according to KG^{CRN}.

Lemma 8.5.3 *Let* $\theta(x, s)$ *be a realization of a Gaussian process with the compound spheric kernel and* $\rho = 1$. *Let* $X = \{x_1, ..., x_d\}$ *and evaluated points* $\tilde{X}^n = \{(x_1, 1), ..., (x_d, 1)\}$, *then for all* $(x, s) \in X \times \mathbb{N}^+$, *there is no more value*

of any measurement

$$KG_n^{CRN}(x,s) = 0 \tag{8.25}$$

and the maximizer of $\bar{\theta}(x)$ *is known.*

Proof is given in the Appendix B.0.3.

Next, KG^{CRN} may be evaluated according to the method proposed by Scott et al. [2011b] that we used in Chapter 7.3.2. Recall the method discretizes the inner maximization over X with past evaluated points, X^n, and the new proposed point so that the integral over Z is analytically tractable. This may be viewed as a noise-generalized EI because it reduces to Expected Improvement (EI) [Jones et al., 1998b] when outputs are deterministic. By augmenting this KG evaluation method with the ability to choose the seed, in the full correlation case it is guaranteed to never evaluate a new seed and the $\text{KG}_n^{\text{CRN}}(x,s)$ function again simplifies to EI. Therefore the KG^{CRN} algorithm naturally reduces to the EGO algorithm for deterministic functions applied to seed $s = 1$.

Lemma 8.5.4 *Let* $\theta(x,s)$ *be a realization of a Gaussian process with the compound spheric kernel with* $\rho = 1$*. Let* $X \subset \mathbb{R}^d$ *be the set of possible solutions,* $\tilde{X}^n = \{(x^1,1),...,(x^n,1)\}$ *be the set of sampled locations and* $X^n = (x^1,...,x^n)$*. Define*

$$KG_n^{CRN}(x,s;A) = \mathbb{E}_n\left[\max_{x' \in A \cup \{x\}} \mu^{n+1}(x',0) - \max_{x' \in A \cup \{x\}} \mu^n(x',0) \middle| (x,s)^{n+1} = (x,s)\right] \tag{8.26}$$

Then for all $x \in X$

$$KG_n^{CRN}(x,1;X^n) > KG_n^{CRN}(x,2;X^n)$$

and therefore $\max_x KG_n^{CRN}(x,1;X^n) > \max_x KG_n^{CRN}(x,2;X^n)$ *and seed* $s = 2$ *will never be evaluated. Further*

$$KG_n^{CRN}(x,1;X^n) = \mathbb{E}\big[\max\{0, y^{n+1} - \max Y^n\} \big| D^n, x^{n+1} = x, s^{n+1} = 1\big].$$

The proof is given in the Appendix B.0.3.

In the more general case, evaluating $\text{KG}_n^{\text{CRN}}(x,s)$ by any method, when using

compound spheric with either full correlation or in a continuous domain X, we conjecture that the true myopically optimal behaviour is to never go to a new seed,

$$\max_{x\in X, s_{old}\in S^n} \mathrm{KG}_n^{\mathrm{CRN}}(x, s_{old}) > \max_{x\in X, s_{new}\notin S^n} \mathrm{KG}_n^{\mathrm{CRN}}(x, s_{new})$$

and a new seed $s \notin S^n$ will never be sampled. However, the above inequality cannot be proven because $\max_{x\in X} \mathrm{KG}^{\mathrm{CRN}}(x, s)$ has no analytic expression and must be found numerically via gradient ascent algorithms. (Note that $\mathrm{KG}_n^{\mathrm{CRN}}(x, s_{old}) > \mathrm{KG}_n^{\mathrm{CRN}}(x, s_{new})$ is not true in general, $x^i \in X^n$ are counter examples.) Therefore we numerically demonstrate this conjecture in Section 5.4.

However, this conjectured behaviour comes with the risk that if the modelling assumption is incorrect for a given application, the algorithm will try to optimize a single seed and never find the true optimum of $\bar{\theta}(x)$. We observe this phenomenon in Section 5.4 where compound sphericity on a continuous search space encourages greedy resampling of only observed seeds. However this does not happen with the inclusion of bias functions, bias functions allow for more intelligent modelling of noise structure that can then be exploited more appropriately.

8.5.2 Comparison with Knowledge Gradient with Pairwise Sampling

Extending Bayesian optimization to account for correlation in noise has been considered by Xie et al. [2016]. The proposed method considers the case when the search space X is a large finite subset of a continuous search space, enabling the use of GP regression. For the generative model, the method assumes that $\bar{\theta}(x)$ is a realization of a GP and considers compound spheric covariance for difference functions. For acquisition, the standard Knowledge Gradient acquisition function quantifies the value of a single observation without CRN (on a new seed) and this is extended with a second acquisition function that quantifies the value of a pair of observations with CRN (on the same new seed), the acquisition space is thus $\{X, X \times X\}$. The method switches between the serial mode and the batch mode depending on which mode promises the larger value per sample. Since the value of a pair cannot be computed analytically, a lower bound is given by considering the *difference* between the pair of

outcomes

$$\begin{aligned}
\mathrm{KG}_n^{\mathrm{PW}}(x_i, x_j) &= \frac{1}{2}\left(\mathbb{E}_n\left[\max_{x' \in X} \mu^n(x', 0) + \tilde{\tilde{\sigma}}^n(x', 0; x_i, x_j)Z - \max_{x'' \in X} \mu^n(x'', 0)\right]\right) \\
\tilde{\tilde{\sigma}}^n(x, 0; x_i, x_j) &= \frac{k^n(x, 0, x_i, s') - k^n(x, 0, x_j, s')}{\sqrt{k^n(x_i, s', x_i, s') + k^n(x_j, s', x_j, s') - 2k^n(x_i, s', x_j, s')}}
\end{aligned}$$

where $s' = s^{n+1} = n+1$ is a new seed and $\mathrm{KG}_n^{\mathrm{PW}}(x, x')$ is optimized over $X \times X$. Note we have adapted the notation from the original work (where the seed is not an explicit argument) to the formulation presented in this work. In the original work, numerical evaluation of $\mathrm{KG}^{\mathrm{PW}}$ is performed by discretizing the inner maximization, as discussed in Section 8.4.2. One call to $\mathrm{KG}^{\mathrm{PW}}$ requires evaluating both $k^n(x, 0, x_i, s^{n+1})$ and $k^n(x, 0, x_j, s^{n+1})$ for each x and is thus marginally more expensive than one call to KG or $\mathrm{KG}^{\mathrm{CRN}}$.

In the large $|X|$ setting, it is efficient to use GP regression, with compound sphericity in the high ρ setting it is efficient to use CRN. Within both of these regimes, it is doubly beneficial to resample old seeds as implied by both Lemmas 8.5.1 and 8.5.2. Therefore, the Knowledge Gradient with Pairwise Sampling combines an acquisition procedure that can only sample new seeds with a differences model for which it is efficient to only sample old seeds. Also, from a value of information perspective, both serial and batch modes of $\mathrm{KG}^{\mathrm{PW}}$ are guaranteed to yield equal or lower expected value than sequential allocation by $\mathrm{KG}^{\mathrm{CRN}}$.

Proposition 8.5.4.1 *Let D^n be a dataset of observation triplets. For a Gaussian process with a kernel of the form $k_{\bar{\theta}}(x, x') + \delta_{ss'}k_{\epsilon}(x, x')$, the expected increase in value after two steps allocated according to KG^{CRN} is at least as big as two steps allocated according to KG^{PW},*

$$\begin{aligned}
& \mathbb{E}_n\left[\max_{x'} \mu^{n+2}(x', 0) - \max_{x''} \mu^n(x'', 0) \middle| (x, s)^{n+1}, (x, s)^{n+2} \sim KG^{CRN}\right] \\
\geq\ & \mathbb{E}_n\left[\max_{x'} \mu^{n+2}(x', 0) - \max_{x''} \mu^n(x'', 0) \middle| (x, s)^{n+1}, (x, s)^{n+2} \sim KG^{PW}\right]
\end{aligned}$$

Proof The suboptimality of one or two steps of the serial mode of $\mathrm{KG}^{\mathrm{PW}}$ is clear by noting it is constrained to a new seed, a subset of the same acquisition space

considered by $\mathrm{KG}^{\mathrm{CRN}}$ as mentioned in Equation 8.23. We focus on the suboptimality of one step of the batch mode

$$
\begin{aligned}
& \mathbb{E}_n\left[\max_{x'}\mu^{n+2}(x',0)-\max_{x''}\mu^{n}(x'',0)\Big|(x,s)^{n+1},(x,s)^{n+2}\sim \mathrm{KG}^{\mathrm{CRN}}\right] \\
= & \max_{(x,s)^{n+1}}\mathbb{E}_n\left[\max_{(x,s)^{n+2}}\mathbb{E}_{n+1}\big[\max_{x'}\mu^{n+2}(x',0)\big|(x,s)^{n+2}\big]-\max_{x''}\mu^{n}(x'',0)\Big|(x,s)^{n+1}\right] \\
\geq & \max_{x^{n+1}}\mathbb{E}_n\left[\max_{x^{n+2}}\mathbb{E}_{n+1}\big[\max_{x'}\mu^{n+2}(x',0)\big|x^{n+2}\big]-\max_{x''}\mu^{n}(x'',0)\right. \\
& \qquad\qquad \left.\Big|x^{n+1},s^{n+1}=s^{n+2}=n+1\right] \qquad (8.27) \\
\geq & \max_{x^{n+1},x^{n+2}}\mathbb{E}_n\left[\max_{x'}\mu^{n+2}(x',0)-\max_{x''}\mu^{n}(x'',0)\right. \\
& \qquad\qquad \left.\Big|x^{n+1},x^{n+2},s^{n+1}=s^{n+2}=n+1\right] \qquad (8.28) \\
\geq & \mathbb{E}_n\left[\max_{x'}\mu^{n+2}(x',0)-\max_{x''}\mu^{n}(x'',0)\right. \\
& \qquad\qquad \left.\Big|(x^{n+1},x^{n+2})=\arg\max \mathrm{KG}_n^{\mathrm{PW}}(x,x'),s^{n+1},s^{n+2}=n+1\right] \\
= & \mathbb{E}_n\left[\max_{x'}\mu^{n+2}(x',0)-\max_{x''}\mu^{n}(x'',0)\Big|(x,s)^{n+1},(x,s)^{n+2}\sim \mathrm{KG}^{\mathrm{PW}}\right]
\end{aligned}
$$

where the first inequality is sub-optimality due to constraining the acquisition space to a new seed, the second is by Jensen's inequality and the convexity of the max operator implying sub-optimality due to batch pre-allocation, and the third inequality is due to the approximation with differences used in $\mathrm{KG}^{\mathrm{PW}}$ that introduces sub optimally by not allocating to truly maximize the batch. □

Sequentially allocating two singles to the same new seed is guaranteed to have higher value per sample than a corresponding batch mode pre-allocating a pair to a single seed as shown by Equations 8.27 and 8.28. However in the $\mathrm{KG}^{\mathrm{PW}}$ algorithm, the serial mode is constrained to allocate to unique seeds whereas the batch mode is constrained to allocate to same seeds. Each mode computes the value over a different subset of the full acquisition space and therefore occasionally the batch mode can return higher value per sample.

Instead, we make explicit the domain for the objective function as both

a decision variable x and a seed s and build a surrogate model and acquisition procedure over the same space. This approach has many advantages. Firstly there is no need to consider batches/pairs, drastically reducing the space for the acquisition from $X \times X$, also reducing the cost per call to the acquisition function, whilst being provably more efficient. Secondly the structure in the noise, difference functions, can be more aggressively exploited by allocating budget to only a few seeds or allocating to new seeds as necessary. Thirdly, note that the framework allows a user to replace Knowledge Gradient with any multi-fidelity/multi-information source [Huang et al., 2006b, Poloczek et al., 2017] or 'correlation aware' serial acquisition procedure and a corresponding parallel batch acquisition function (or an analytic lower bound) is not required.

On the other hand, when enabling resampling of old seeds, assuming compound sphericity incentivises sampling of old seeds. The KG^{CRN} includes bias functions enabling accurate modelling and the appropriate trade-off between old and new seeds. The KG^{PW} does not encounter such pitfalls as it does not sample old seeds.

8.6 Numerical Experiments

We perform three sets of experiments, first using synthetic GP sample functions and known hyperparameters, allowing perfect comparison of just the acquisition procedures under laboratory controlled conditions, though such test problems can be more multi modal and complex than real world problems. The next two problems are taken from the SimOpt library (`http://simopt.org`), the Assemble-to-order problem (ATO) and the Ambulances in a Square problem (AIS). The code for all experiments will be made public upon publication.

8.6.1 Compared Algorithms and Variants

We aim to investigate the empirical effects of including bias functions and the ability of the acquisition procedure to revisit old seeds whilst holding all other experimental factors constant. Therefore we consider the following five algorithms.

- **Knowledge Gradient (KG):** A GP model with independent homoskedastic noise is fitted, $\eta^2 = \sigma_b^2 = 0$, $\sigma_w^2 > 0$. Acquisition is according to KG^{CRN} artificially constrained to a new seed.

- **KG with Pairwise ($\mathbf{KG^{PW}}$):** The algorithm proposed by Xie et al. [2016]. A GP with the compound spheric differences kernel is fitted $\sigma_b^2 = 0$, $\eta^2, \sigma_w^2 \geq 0$. For acquisition, the value of single sample is given by KG^{CRN} and pairs by KG^{PW}, both are constrained to a new seed.

- **KG with Pairwise and Bias Functions ($\mathbf{KG^{PW}}$-bias):** A GP with both offsets and bias functions is fitted, $\sigma_b^2, \eta^2, \sigma_w^2 \geq 0$. Acquisition is the same as above.

- **KG for Common Random Numbers and Compound Sphericity ($\mathbf{KG^{CRN}}$-CS):** A GP with $\sigma_b^2 = 0$ and $\eta^2, \sigma_w^2 \geq 0$ is fitted. Acquisition can sample any seed according to KG^{CRN}.

- **KG for Common Random Numbers ($\mathbf{KG^{CRN}}$):** A GP with both offsets and bias functions is fitted, $\sigma_b^2, \eta^2, \sigma_w^2 \geq 0$. Acquisition can sample any seed according to KG^{CRN}.

8.6.2 Synthetic Data, no Bias Functions

We set $X = \{1, .., 100\}$ and generate synthetic data from a Multivariate Gaussian $\bar{\theta}(X) \sim N(\underline{0}, k_{\bar{\theta}}(X, X))$ where $k_{\bar{\theta}}(x, x') = 100^2 \exp\left(-\frac{(x-x')^2}{2 \cdot 5^2}\right)$. The offsets are sampled $o_s \sim N(0, \rho 50^2)$ and the white noise $w_s(x) \sim N(0, (1-\rho)50^2)$. We vary $\rho \in \{0, 0.1, ..., 0.9, 1.0\}$ holding the total noise constant such that standard KG will always perform the same. For algorithms we compare normal KG, KG^{PW} and KG^{CRN} all without bias functions. For each method we evaluate the KG by Equation 8.22 and set $A = X$. We optimize the acquisition function by exhaustive search. In all cases we fit the GP regression model with known kernel hyperparameters except for KG where we force $\rho = 0$. This allows us to fully focus on differences in the generative model and acquisition function. We measure opportunity cost, let

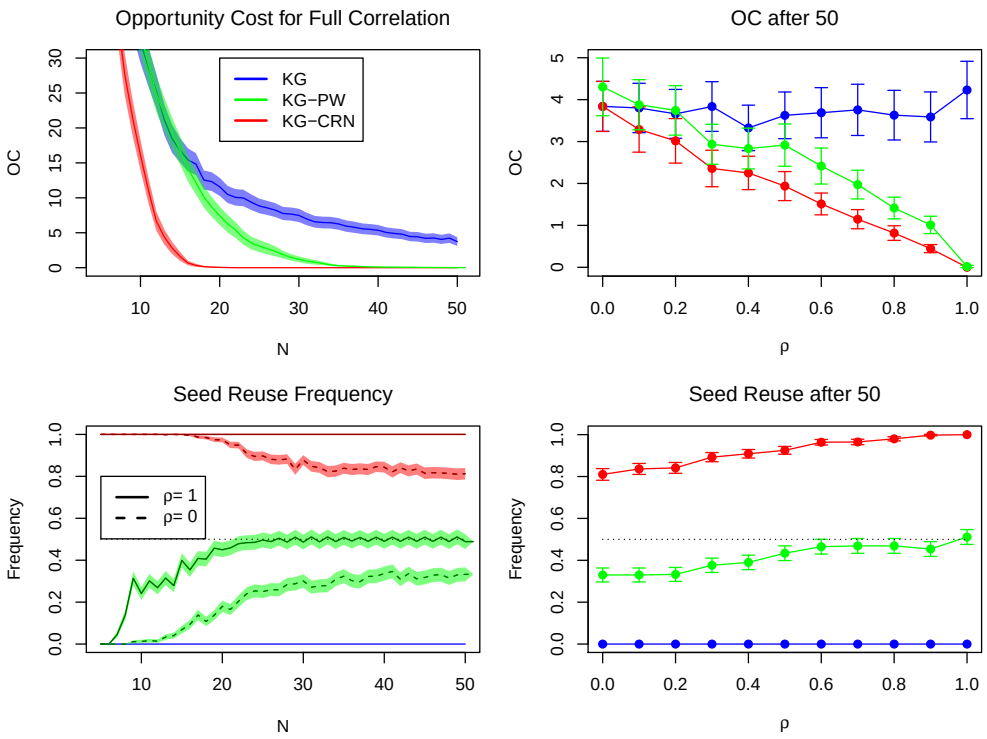


Figure 8.3: TL: Opportunity Cost for the $\rho = 1$ case, the $\rho = 0$ case all algorithms equal KG. KG^{CRN} aggressively optimizes a single seed. TR: final OC for a range of ρ values. For increasing ρ both CRN methods improve. BL: the average seed reuse for the cases $\rho = 0, 1$. For large ρ, KG^{PW} is upper bounded by 0.5, KG^{CRN} never samples a new seed. BR: final seed reuse over a range of ρ.

$x_r^n = \arg\max_x \mu^n(x, 0)$,

$$\text{Opportunity Cost at time } n = \max \bar{\theta}(x) - \bar{\theta}(x_r^n). \tag{8.29}$$

We report the frequency of seed reuse, how often at an iteration n the next sampled seed s^{n+1} was in the current history of observed seeds S^n. If KG^{PW} samples a pair for every iteration, the first sample of each pair would be new and the second would be old hence the average reuse frequency is upper bounded by 0.5.

From top row plots of Figure 8.3, for low ρ values, all algorithms have similar opportunity cost as there is no exploitable CRN structure. As ρ increases there is more CRN structure to exploit and KG^{PW} performance improves for larger budgets while KG^{CRN} performance improves for all budgets.

The bottom row plots of Figure 8.3 show seed reuse which we interpret as how much an algorithm uses CRN. For all ρ, KG^{CRN} starts by resampling old seeds, utilizing CRN, and later samples more new seeds only for low ρ, seed reuse dropping to 0.8, or querying new seeds 20% of the time. We see that this results in significantly faster convergence in the $\rho = 1$ case plotted.

KG^{PW} instead starts by sampling singles on new seeds, ignoring CRN reproducing KG. For larger budgets KG^{PW} uses more pairs and improves upon KG for the range of ρ. However for the best case for CRN, $\rho = 1$, KG^{PW} quickly hits its seed reuse upper bound of 0.5, querying new seeds 50% of the time, and cannot fully utilize CRN.

In the Appendix B.1, we present the same experiment using only bias functions, see B.2, and observe no improvement over standard KG, suggesting that local differences correlation is not as beneficial as global, i.e. constant, correlation.

8.6.3 Assemble to Order Benchmark

The Assemble to Order (ATO) simulator was introduced by Xu et al. [2010] and a slightly modified version has been used in Xie et al. [2016] to test the KG^{PW} algorithm. A shop sells five products assembled from eight items held in inventory. A random stream of customers arrives into the shop, each buying a product and consuming inventory. When an item in inventory drops below a user defined threshold, an order for more is placed. The shop aims to maximize profit (product sales minus storage cost), by optimizing the reorder thresholds for each item. A seed defines the stream of customers and the item delivery times. For this problem, the solution space is $X = \{1, .., 20\}^8$.

$\text{KG}_n^{\text{CRN}}(x, s)$ is evaluated and optimized as described in Section 8.4.2. The expectation of the maximizations in $\text{KG}^{\text{PW}}(x^{n+1}, x^{n+2})$ is evaluated exactly the same way. The location of the single sample is found by using standard KG $x^{n+1} = \underset{x}{\text{argmax}}\ \text{KG}_n^{\text{CRN}}(x, s_{new})$. The pair of samples is searched for in two ways. First, $\text{KG}^{\text{PW}}(x^{n+1}, x^{n+2})$ is optimized for x^{n+2} only with the same multi-start gradient ascent optimizer. Second, we *jointly* optimize the pair by searching over the full $X \times X$. Finally, the results of both phases are locally fine tuned over $X \times X$.

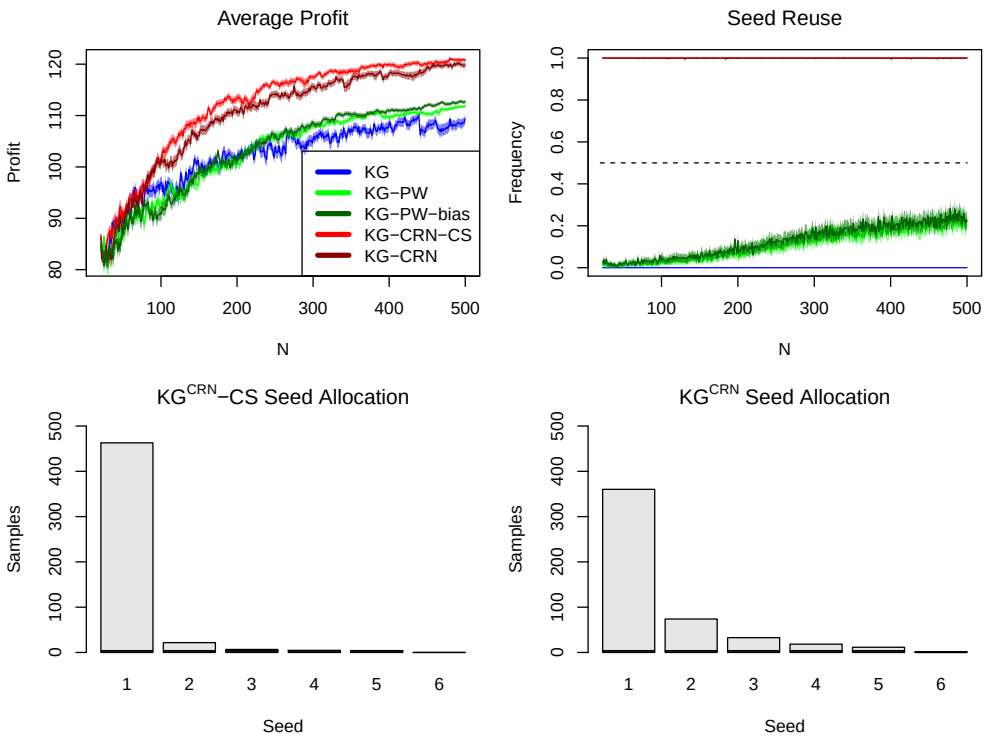


Figure 8.4: Top left: profit of x_r^N evaluated on a held-out set of 2,000 test seeds. Top right: average seed reuse over iterations. Bottom: seed allocation for KG^{CRN} without bias functions (left) and with bias functions (right). Both KG^{CRN} variants mostly sample a single seed.

All methods start with $n_{init} = 20$. All hyperparameters are learnt by maximum likelihood and fine tuned after each new sample. We record the quality of the recommended $x_r^n = \underset{x}{\text{argmax}}\ \mu^n(x, 0)$ on a held-out test set of seeds. ATO results are reported in Figure 8.4.

Both algorithms with KG^{CRN} acquisition yield the largest profits and the KG^{PW} variants are only marginally improving upon KG. In this application, the KG^{CRN} variants *never* use new seeds after the initial five seeds, instead allocating almost all budget to a single seed suggesting that this ATO problem is an ideal use case for the compound spheric assumption. From the previous experiment we observed that KG^{CRN} samples old seeds early and moves onto new seeds for large budgets. In this learnt hyperparameter case, in the Appendix B.1, the offset hyperparameter grows over time as model fit improves and data collection focuses on

the peak. Consequently, for larger budgets KG^{CRN} is even more likely to resample old seeds. With KG^{PW}, the early behavior samples singles on new seeds which cannot inform any CRN hyperparameters and the algorithm never learns a larger offset parameter. As a result it allocates very little of the budget to pairs failing to significantly exploit the CRN structure and hence producing marginally superior results to KG. In this application, the ability to revisit old seeds clusters observations on fewer seeds which allows for more robust learning of CRN hyperparameters.

Marginalization of GP hyperparameters, accounting for uncertainty around the maximum likelihood estimate, may affect performance differences, however we don't investigate such an approach in this study. Also a true estimate of the improvement due to a pair of observations by Monte-Carlo (not the lower bound by KG^{PW}) would likely increase sampling of pairs.

8.6.4 Ambulances in a Square Problem

This simulator (AIS) was introduced by Pasupathy and Henderson [2006]. Given a city over a 30km by 30km square, one must optimize the location of three ambulance bases to reduce the journey time to patients that appear across the city as a Poisson point process. The seed defines the times and locations of patients. The solution space is $X = [0, 30]^6$, the valid (x,y) locations for each of three ambulance bases. We run the simulator for 1800 simulated time units in which on average 30 patients appear. This problem is over a continuous search space and the optimal result for each realization of patients is to place the ambulance bases near the patients hence the peak of one seed is not the same as the average of seeds and bias functions are required. Results are summarized in Figure 8.5

Both algorithms with the surrogate model that includes bias functions provide the best results in this benchmark, improving upon KG. The KG^{CRN} with the compound spheric assumption in a continuous search space leads to excessive sampling of observed seeds agreeing with Lemma 8.5.2 and the conjectured behaviour of KG^{CRN} acquisition. Our proposed KG^{CRN} with bias functions on the other hand does not suffer and automatically queries many new seeds. Again, both KG^{PW} variants sample far more seeds by their construction which is less penalized in this benchmark. We

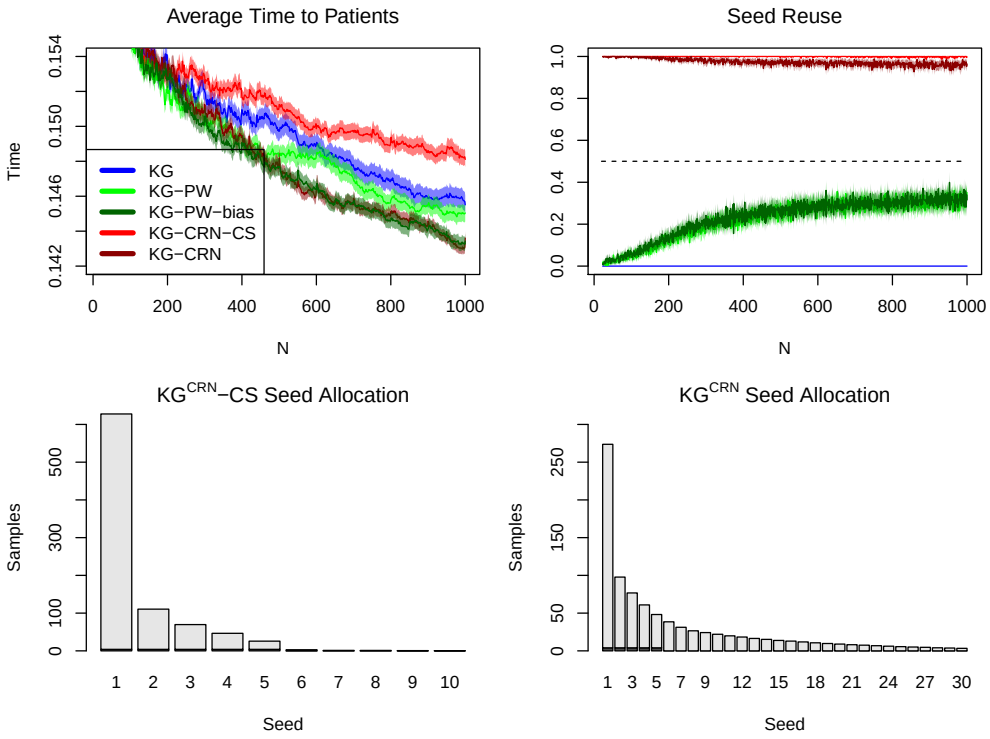


Figure 8.5: Top left: average journey time to patients. Top right: seed reuse over iterations. Bottom: seed allocation by KG^{CRN} without (left) and with (right) bias functions. The algorithms with bias functions provide the best results. The compound spheric assumption, which is violated in this benchmark, leads to greedy sampling of observed seeds and sub optimal performance.

also performed experiments where the sum of ambulance journey times was optimized and where the number of patients was fixed. All results, including ATO are reported in Table 8.1. In all experiments, the KG^{CRN} without bias functions never sampled a new seed. We also report running time of all experiments and in all cases KG was quickest, followed by the KG^{CRN} variants and the KG^{PW} variants used the most computational time.

Table 8.1: Mean ± 2 standard errors of average performance for all benchmarks, results that do not significantly differ from the best are in bold. The ability to revisit seeds improves the ATO results and including bias functions improves AIS results (or compound sphericity significantly harms AIS).

	KG	KG^{PW}	KG^{PW}-bias
ATO, N=500	109.35 ± 1.88	111.86 ± 0.65	112.69 ± 0.67
AIS, N=500	$.1498 \pm .0011$	**.1483 ± 0.0010**	**.1477 ± .0010**
AIS, N=1000	$.1455 \pm .0010$	$.1450 \pm 0.0010$	**.1435 ± .0009**
AIS, sum time	4.66 ± 0.33	$4.611 \pm .045$	**4.449± .030**
AIS, 30 patients	$.1498 \pm .0009$	**.1468 ± .0008**	**.1467 ± .0009**

	KG^{CRN}-CS	KG^{CRN}
ATO, N=500	**120.99 ± 0.71**	**119.84 ± 1.13**
AIS, N=500	$.1512 \pm .0010$	**.1482± .0010**
AIS, N=1000	$.1481 \pm .0009$	**.1436 ± .0008**
AIS, sum time	$4.515 \pm .035$	**4.430 ± .034**
AIS, 30 patients	.1482 ± .0008	**.1467 ± .0009**

Therefore both the ability to revisit old seeds and the modelling of bias functions are necessary to make a robust algorithm that works across a variety of problems.

8.7 Conclusion

We proposed an approach to simulation optimization with common random numbers where the seed of the random number generator used within a stochastic objective function is an input to be chosen by the optimization algorithm. We augment a standard Gaussian process model with two extra hyperparameters to model structured noise (scenario influence), while maintaining the ability to predict the

average output of the target function in closed form. Matching this augmented model, we propose KG^{CRN} that quantifies the expected value of evaluating the objective for given decision variable and seed, providing a clean framework that allows Bayesian optimization to automatically exploit CRN where this is beneficial, and resort to standard KG where not. Moreover, the proposed KG^{CRN} algorithm structure does not add significant complexity over the equivalent non-CRN Knowledge Gradient due to the fundamental structure of CRN.

In future work we plan to augment other problem settings with common random numbers, such as multi-fidelity optimization, simulations with input uncertainty and multi-objective optimization.

Chapter 9

Conclusions and Future Work

9.1 Conclusion

We have considered a range of problems, and in every case using a simple principled Value of Information approach popularised by the Knowledge Gradient family of algorithms. In Chapter 4 we showed how choosing a tool from a set for many points in a domain is a generalisation of ranking and selection. We proposed the first incarnation of the REVI algorithm that exploited the correlated posterior over the domain enabled specifically by a Gaussian process. We applied this to the problem of efficient selection of scheduling heuristics. Chapter 5 empirically demonstrated the utility of freezing and caching computations enabling much larger Monte-Carlo sample sizes without impacting computation time. We combined this with the algorithm from the previous chapter to form Neighbours-REVI that outperformed a recently published baseline with only marginal computational cost. In Chapter 6 we proposed two new methods, first, CLEVI uses a convolution trick to account for local task density and therefore avoid sampling on boundaries of the task space. Second, REVI generalised the Knowledge Gradient to the conditional multi-task setting and, building on the previous chapter, used a computationally efficient Monte-Carlo integral over tasks. We then moved onto optimisation of a weighted average or integral of a function in Chapter 7 where we proposed to modify the Expected Improvement and standard Knowledge Gradient acquisition functions to this case. The result was to simply replace terms in the standard acquisition functions with Monte-Carlo

estimators. Simulation optimisation with common random number and Gaussian processes has been largely overlooked. In Chapter 8, we apply the same value of information procedure (arriving at a different algorithm to a previous attempt) to combine Knowledge Gradient with Common Random Numbers. We show that a particular assumption considered by previous works, compound sphericity, leads to poor modeling and degraded performance in certain applications. We overcome this by including a single extra parameter to model seed specific bias functions allowing the algorithm to appropriately sample old and new seeds.

9.2 Future Work

There remain problems where a value of information approach can lead to fast efficient algorithms.

For simulation optimisation with input uncertainty, we assumed the uncertainty distribution is fixed over time. However, one may consider a setting in which a user with a given budget can choose between improving the distribution over the uncertain input, or choose to collect more simulation data.

We investigated common random numbers for global optimisation. We plan to investigate the multi-task conditional optimisation problem setting with CRN to incorporate covariance in noise across solutions as well as tasks, such that multiple optimisation problems can simultaneously benefit from the same variance reduction.

We have not looked at simulation optimisation with input uncertainty to optimise the worst case, i.e., the minimum of functions instead of the average of functions. A traditional value of information derived procedure will not yield an algorithm that is asymptotically consistent as shown in Chapter 3.3.4. Hence there may be potential in researching an *approximate* value of information recipe for deriving *approximately* myopically optimal acquisition functions that are still asymptotically consistent.

Appendix A

Proofs from Chapter 4

A.1 Proof of Theorem 4.1

The REVI, NEVI, and EVI policies are all asymptotically optimal, meaning that given an infinite sampling budget they will always find the best tool for each task. Here we only prove the case for the REVI policy, however these results also apply to the NEVI and EVI policies.

Theorem 4.1 *When sampling according to the REVI policy, as the budget goes to infinity, $N \to \infty$, the sequence of mappings converges almost surely to $S^N(x_t) = \underset{a}{\text{argmax}}\ \zeta_{t,a}$ for all i.*

We prove Theorem 4.1 in five parts. The first part shows that the expected improvement, the REVI function, of sampling a (task, tool) pair is non-negative, collecting an extra sample is always expected to improve, or maintain, the predicted portfolio performance. The second part shows that the expected improvement of a (task, tool) pair is zero if and only if the posterior variance is zero. The third part shows that if infinite samples are allocated to a given (task, tool) pair, the posterior variance and expected improvement are zero and the true performance is known for the (task, tool) pair. The fourth part simply states that if there is no expected improvement in a pair, then the true performance must be known. Finally, we show that in the limiting case of an infinite sampling budget, the case in which only a subset of pairs has non-zero expected improvement would imply a pair outside the subset must have been sampled whilst it did not maximize expected improvement

thus contradicting the REVI policy. Hence to be consistent with the REVI policy, all (task, tool) pairs must be sampled until there is no expected improvement and the true expected performance values are known and equal to the posterior means for all pairs. Therefore, the mapping chooses the correct tool for all tasks. These four parts are broken down into the following three propositions and one final proof.

A.1.1 Preparatory Material

The following proposition states that the expected improvement in the predicted portfolio performance is always non-negative. The result applies to NEVI(t, a) and EVI(t, a) since they are both subsets of the same summation that is used to calculate REVI(t, a).

Proposition A.1.0.1 *REVI*$^n(t, a) \geqslant 0$ *for all* $(t, a) \in \{1, .., M\} \times \{1, .., A\}$.

Proof Substituting the appropriate terms, the REVI function of Equation (4.9) is defined as

$$\text{REVI}^n(t,a) = \sum_j w_j \mathbb{E}\left[\max_b \mu_{j,b}^{n+1} - \max_b \mu_{j,b}^n \middle| \mathcal{F}^n, (t,a)^{n+1} = (t,a)\right]. \qquad \text{(A.1)}$$

Taking a single term from the summation in Equation (A.1) and temporarily dropping the j subscripts and the conditioning terms for clarity, we can define the highest and second highest means $\mu_{(1)}^n = \max_b \mu_b^n$ and $\mu_{(2)} = \max_{b \neq (1)} \mu_b^n$, such that a single term becomes

$$\begin{aligned}
\mathbb{E}\left[\max_b \mu_b^{n+1} - \mu_{(1)}^n\right] &= \mathbb{E}\left[\max\left\{\max_{b \neq a} \mu_b^n - \mu_{(1)}^n\,,\ \mu_a^n - \mu_{(1)}^n + \tilde{\sigma}_j^n(t,a)Z\right\}\right] \\
&= \begin{cases} \mathbb{E}\left[\max\{0\,,\ \mu_a^n - \mu_{(1)}^n + \tilde{\sigma}_j^n(t,a)Z\}\right] & a \neq (1) \\ \mathbb{E}\left[\max\{\mu_{(2)} - \mu_{(1)}\,,\ \tilde{\sigma}_j^n(t,a)Z\}\right] & a = (1) \end{cases}
\end{aligned}$$

where Z is a standard normal random variable. The case for $a = (1)$ may be rearranged to be of the same form as the $a \neq (1)$ case:

$$\begin{aligned}
\mathbb{E}\left[\max\{\mu_{(2)} - \mu_{(1)}\,,\ \tilde{\sigma}_j^n(t,a)Z\}\right] &= \mathbb{E}\left[\max\{0\,,\ \mu_{(1)} - \mu_{(2)} + \tilde{\sigma}_j^n(t,a)Z\}\right] \\
&\quad -\mathbb{E}[\mu_{(1)} - \mu_{(2)} + \tilde{\sigma}_j^n(t,a)Z] \\
&= -\mathbb{E}\left[\min\{0\,,\ \mu_{(1)} - \mu_{(2)} + \tilde{\sigma}_j^n(t,a)Z\}\right] \\
&= \mathbb{E}\left[\max\{0\,,\ \mu_{(2)} - \mu_{(1)} + \tilde{\sigma}_j^n(t,a)Z\}\right],
\end{aligned}$$

where the second line comes from the difference between the partial expectation and the full expectation and the third line is due to the symmetry of the normal distribution. Therefore we may write the original expectation for both cases $a = (1)$ and $a \neq (1)$ as

$$\mathbb{E}\left[\max_b\ \mu_b^{n+1}\right] - \mu_{(1)}^n = \mathbb{E}\left[\max\{0\,,\ -|\mu_a - \max_{b\neq a}\mu_b| + \tilde{\sigma}_j^n(t,a)Z\}\right]. \tag{A.2}$$

By the convexity of the max operator and Jensen's inequality the expectation in Equation (A.2) must be non-negative. All of the weights are non-negative, therefore $\text{REVI}^n(t,a)$ is a sum of the expectations of non-negative random variables therefore $\text{REVI}^n(t,a) \geqslant 0$. $\square$

We next show that is is only zero when the posterior variance of a given pair is zero, again this result applies to NEVI and EVI since they are a subset of the same summation as REVI. Let ${}_{tj,a}$ be the (t,j) element of the posterior covariance for the tool a.

Proposition A.1.0.2 *$REVI^n(t,a) = 0$ if and only if $\Sigma_{tj,a}^n = 0$ for all $j \in \{1,..,M\}$.*

Proof We first prove that $\text{REVI}^n(t,a) = 0 \Rightarrow \Sigma_{tj,a} = 0$ for all j. If the sum in Equation (A.1) is equal to zero, each and every expectation of a non-negative random variable must also be equal to zero. Therefore the non-negative random variable is identically zero implying $-|\mu_a^n - \max_{b\neq a}\mu_b| + \tilde{\sigma}_j^n(t,a)Z \leqslant 0$ for all $Z \in \mathbb{R}$ which implies $\tilde{\sigma}_j^n(t,a) = 0$. Therefore $\text{REVI}^n(t,a) = 0 \Rightarrow \tilde{\sigma}_j^n(t,a) = 0$ for all j. It is easily shown that $\tilde{\sigma}_j^n(t,a) = 0 \Leftrightarrow \Sigma_{tj,a} = \Sigma_{jt,a} = 0$ and we have that

$\text{REVI}^n(t,a) = 0 \Rightarrow \Sigma^n_{tj,a} = 0$ for all j.

We now prove the reverse direction, $\Sigma^n_{tt,a} = 0 \Rightarrow \text{REVI}^n(t,a) = 0$. Since Σ^n_a is a positive semi-definite (PSD) matrix all principal 2×2 sub-matrices are also PSD and Hadamard's inequality implies that $\Sigma_{tt,a} = 0 \Rightarrow \Sigma_{tj,a} = \Sigma_{jt,a} = 0$ for all j and $\tilde{\sigma}^n_j(t,a) = 0$ for all j. Evaluating the REVI function yields

$$\text{REVI}^n(t,a) = \sum_j w_j h(\Delta^n_{j,a}, 0) = \sum_j 0 = 0. \tag{A.3}$$

□

We next show that when a pair is sampled infinitely often, the variance reduces to zero along with any expected improvement. The result does not depend on $REVI(t,a)$ but only on the previous two propositions, therefore it applies to NEVI and EVI with only a change of notation.

Proposition A.1.0.3 *If a sampling procedure samples (task, tool) pair (t,a) infinitely often, then $REVI^\infty(t,a) = 0$ and $\mu^\infty_{t,a} = \zeta_{t,a}$ almost surely.*

Proof The sigma algebra generated by the data $\mathcal{F}^\infty$ contains infinitely many samples of $Y_{t,a}$ with finite variance, $\sigma^2_{\epsilon,a} < \infty$. The strong law of large numbers implies $\theta_{t,a}$ is $\mathcal{F}^\infty$-measurable. Therefore the expectation of $\theta_{t,a}$ conditioned on $\mathcal{F}^\infty$ is the true mean, $\mathbb{E}[\theta_{t,a}|\mathcal{F}^\infty] = \mu^\infty_{t,a} = \mathbb{E}[Y_{t,a}] = \zeta_{t,a}$, and the posterior variance is zero, $\text{Var}[\theta_{t,a}|\mathcal{F}^\infty] = \Sigma^\infty_{tt,a} = 0$. Proposition A.1.0.2 implies that $\text{REVI}^\infty(t,a) = 0$. □

The following proposition simply states that if expected improvement is zero, the true performance is already known. However, it does not assume that infinite samples have been allocated to a given pair. Again this proof does not rely on the REVI function therefore applies to the NEVI and EVI policies, too.

Proposition A.1.0.4 *For a given pair (t,a) with positive prior variance $\Sigma^0_{tt,a} > 0$, if $REVI^n(t,a) = 0$ then the posterior mean is the true expected performance $\mu^n_{t,a} = \zeta_{t,a}$.*

Proof By Proposition A.1.0.2 the posterior variance is zero $\Sigma^n_{tt,a} = 0$. Proof of the implication $\Sigma^n_{tt,a} = 0 \Rightarrow \mu^n_{t,a} = \zeta_{t,a}$ is omitted as it simply follows from the consistency

of the conjugate Gaussian posterior distribution which may be demonstrated in this instance using Equation (4.5). The posterior variance of a pair, $\Sigma^n_{tt,a}$, will be zero if either there is no observation noise $\sigma^2_{\epsilon,a} = 0$ and one sample of $Y_{t,a}$ is collected, or if infinite samples are collected of a noisy observation. In both cases it is easily shown that $\mu^n(t,a) = \zeta_{t,a}$.

A.1.2 Proof of Theorem 4.1

The above four propositions are results relating to a single (task, tool) pair and show that there is always an expected improvement from sampling a pair unless infinite samples are allocated to the pair and the true performance is known. Below, the final part of the proof relates to all the (task, tool) pairs when sampling according to the REVI/NEVI/EVI policies in the limit of infinite samples. The proof shows that a limiting state in which a subset of pairs has non zero expected improvement leads to a contradiction. Therefore all pairs must have zero improvement, the true expected performance is known, and the mapping selects the true best tool for every task maximizing portfolio performance.

Proof When sampling according to the REVI policy, assume that in the infinite limit $N \to \infty$ there exists a set of (task, tool) pairs, I, for which the expected improvement is strictly positive, $I = \{(t,a); \text{REVI}^\infty(t,a) > 0\}$, and by the non-negativity of REVI given in Proposition A.1.0.1 we denote the complement $I^C = \{(i',a'); \text{REVI}^\infty(i',a') = 0\}$.

By the contrapositive of Proposition A.1.0.3, each pair $(t,a) \in I$ must have been sampled finitely, and therefore there exists a finite time in the sampling history after which $(t,a) \in I$ is no longer sampled, $\tilde{n}(t,a) = \min\{n; (t,a) \notin \{(t,a)\}_n^\infty\}$. We now denote the latest stage in the sampling history at which a finitely sampled pair was selected by the policy $\tilde{n} = \max\{\tilde{n}(t,a); (t,a) \in I)\}$. Therefore we may rewrite the set of finitely sampled pairs as those that are never sampled after $\tilde{n}$, $I = \{(t,a); (t,a) \notin \{(t,a)\}_{\tilde{n}}^\infty\}$.

The assumption that in the infinite limit there exist pairs with positive improvement $\text{REVI}^\infty(t,a) > 0$ for $(t,a) \in I$ implies that there exists a time $n^* > \tilde{n}$

at which a finitely sampled pair has the largest expected improvement

$$\min_{(t,a)\in I} \text{REVI}^{n^*}(t,a) > \max_{(i',a')\in I^C} \text{REVI}^{n^*}(i',a'),$$

therefore $(t,a) \in I$ would be sampled by the REVI policy at time $n^* > \tilde{n}$. However by assumption it is shown that all pairs in $(t,a) \in I$ are not sampled after time $\tilde{n}$. This contradiction implies that in the limit of infinite sampling budget, a case in which a subset of pairs has strictly positive improvement contradicts the REVI policy.

Therefore, under the REVI policy, in the infinite sampling limit all pairs must have zero expected improvement $REVI^{\infty}(t,a) = 0$ for all (t,a). This implies that $\Sigma^{\infty}_{i,a} = 0$ for all (t,a) and therefore by Proposition A.1.0.4 we have that $\mu^{\infty}_{t,a} = \zeta_{t,a}$ for all pairs and the mapping selects the true best tool for every task

$$S^{\infty}(x_t) = \underset{a}{\text{argmax}}\ \mu^{\infty}_{t,a} = \underset{a}{\text{argmax}}\ \zeta_{t,a}.$$

□

A.2 Dynamic Programming Formulation

For each tool, the posterior performance distribution θ_a is given by a multivariate normal that is parametrised by a mean and covariance matrix, therefore we define a state of sampling as the tuple of all the multivariate normal parameters, one set for each tool:

$$s = ((\mu_1, \Sigma_1), ..., (\mu_A, \Sigma_A)) \in \mathbb{S}$$

where $\mu_a \in \mathbb{R}^M$ and $\Sigma_a \in \mathbb{R}^{M\times M}$. We define the state at a stage n during sampling $S^n \in \mathbb{S}$, where the state is updated after each new sample according to Equations (6) and (7) in the main document that can be used to define the state transition function. A policy, π, is a function giving the next (task, tool) to be sampled, and we denote the set of all possible policy functions Π. The terminal value function is given by $V^N(s) = P^N = \sum_t w_t \max_a \mu^N_{t,a}$, however this quantity is only $\mathcal{F}^N$ measurable. For

any time $n < N$ during sampling, for any state $s \in \mathbb{S}$, the expectation of the terminal value assuming the remaining $N - n$ samples are allocated according to π can be calculated:

$$V^{n,\pi}(s) = \mathbb{E}\left[\sum_t w_t \max_a \mu^N_{t,a} \middle| \mathcal{F}^n, S^n = s, \pi\right]. \tag{A.4}$$

Maximizing the expectation of the final predicted portfolio performance over all possible policies for all $s \in \mathbb{S}$ gives the value function of the optimal policy, the optimal value function:

$$V^n(s) = \sup_\pi \mathbb{E}\left[\sum_{t=1}^M w_t \max_a \mu^N_{t,a} \middle| \mathcal{F}^n, S^n = s, \pi\right].$$

Similarly a policy π^* that maximizes the final value given by Equation A.4, for any state $s \in \mathbb{S}$ and at any stage during sampling $n < N$, is said to be optimal,

$$\pi^* \in \operatorname*{argmax}_\pi V^{n,\pi}(s). \tag{A.5}$$

Next we define the Q-values for this particular problem that give the expected optimal final value starting from any state s given that the next decision $(t, a)^{n+1}$ will be (t, a),

$$Q^n(s, t, a) = \mathbb{E}[V^{n+1}(S^{n+1})|\mathcal{F}^n, (t, a)^{n+1} = (t, a), S^n = s]. \tag{A.6}$$

A policy that determines samples by maximizing the Q-values is an optimal policy, by choosing to maximize the future value function in the current step for all $s \in \mathbb{S}$,

$$(t, a)^{n+1} = \operatorname*{argmax}_{t,a} Q^n(s, t, a). \tag{A.7}$$

A.2.1 Myopic Optimality of the REVI Policy

This result is similar to the result for the Knowledge Gradient Policy for Correlated Normal Beliefs (Frazier et al. [2009a], Remark 1). Samples allocated according to the REVI policy are determined by maximizing Equation 4.9 from the main text

restated here,

$$(t,a)^{n+1} = \underset{t,a}{\text{argmax}}\ \mathbb{E}\left[\sum_j w_j \max_b \mu_{j,b}^{n+1} \middle| \mathcal{F}^n, (t,a)^{n+1} = (t,a)\right] - \sum_j w_j \max_b \mu_{j,b}^n \tag{4.9}$$

In the special case where there is only one sample left, $n = N-1$, we may rewrite the above equation

$$\begin{aligned}
(t,a)^N &= \underset{t,a}{\text{argmax}}\ \mathbb{E}\left[\sum_j w_j \max_b \mu_{j,b}^N \middle| \mathcal{F}^{N-1}, (t,a)^N = (t,a)\right] \\
&\quad - \sum_j w_j \max_b \mu_{j,b}^{N-1} \\
&= \underset{t,a}{\text{argmax}}\ Q^{N-1}(S^{N-1}, t, a) - V^N(S^{N-1}) \\
&= \underset{t,a}{\text{argmax}}\ Q^{N-1}(S^{N-1}, t, a). \qquad \text{(A.8)}
\end{aligned}$$

The second line is using the definition given by Equation (A.6), the final term on the right hand side does not depend on $(t,a)^N$ and so may be dropped. Equation (A.8) satisfies Equation (A.5) that defines an optimal policy. Therefore the REVI policy is an optimal policy in the case when there is only one sample left, REVI policy is myopically optimal:

Theorem A.2.1 *For each state $s \in \mathbb{S}$,*

$$V^{N-1,REVI}(s) = \sup_{(t,a)^N} V^{N-1,(t,a)^N}(s) = V^{N-1}(s).$$

A.2.2 Asymptotic Optimality

We have already provided a proof for the infinite sampling limit of the REVI policy and so we briefly restate the same theorem in a Dynamic Programming setting here. Proofs of the following assumptions are easily adapted from the propositions given above and proofs found in Frazier et al. [2008] and Frazier et al. [2009a] with the inclusion of appropriate summation symbols, therefore we do not repeat them here. The state, $s \in \mathbb{S}$, value functions, $V^n(s)$, and Q-values are defined above. We denote the optimal value function with a variable final

budget N as $V^n(s;N)$, we assume the state after infinite samples, S^∞, exists and is finite (see Frazier et al. [2009a] Lemma A.5) and that there exists an upper bound on the value of the optimal policy as the sampling budget tends to infinity, $\lim_{N\to\infty} V^n(s;N) = U(s) = \mathbb{E}\left[\sum_t w_t \max_a \theta_{t,a} \middle| S^0 = s\right]$ (see Frazier et al. [2009a] Lemma A.4). We further assume that as the number of samples for a given pair (i,a) approaches infinity, the Q-value for that pair is equal to the current value, $Q^{N-1}(s,t,a) = V^N(s)$, meaning there is no expected improvement in sampling from (t,a) (Frazier et al. [2009a] Lemma A.7). If the Q-value equals the terminal value for all (task, tool) pairs then the value is equal to the upper bound (Frazier et al. [2009a] Lemma A.6). As the sampling budget goes to infinity, the value of an optimal policy approaches the upper bound, the REVI policy also approaches the same upper bound therefore must equal the value of the optimal policy (Frazier et al. [2009a] Theorem 4),

Theorem A.2.2 *For each* $s \in \mathbb{S}$ $\lim_{N\to\infty} V^{0,REVI}(s;N) = \lim_{N\to\infty} V^0(s;N)$.

A.2.3 Bound on Sub-Optimality, Proof of Theorem 4.3.2

For the REVI policy, we may calculate an upper bound on the difference in value functions between the optimal policy and the REVI policy. Equation (4.14) of Theorem 4.3.2 from the main text is restated here,

$$\max_\pi \mathbb{E}[P^N|\mathcal{F}^n,\pi] - \mathbb{E}[P^N|\mathcal{F}^n,\text{REVI}] \leqslant \max_{(t,a)_{n+1}^{N-1}} \sum_{k=n}^{N-2} \sqrt{2\pi} \sum_{j=1}^{M} w_j |\tilde{\sigma}_j^k(t^{k+1},a^{k+1})|. \tag{4.14}$$

For convenience, we introduce a vector norm function $\|u\| = \sum_{t=1}^M w_t|u_t|$ and make explicit the state in $\tilde{\sigma}^n(t,a)$, which is replaced by $\tilde{\sigma}(S^n,t,a)$. By replacing the terms in the above equation, we may rewrite it in the dynamic programming formulation:

Theorem A.2.3 *For each* $s \in \mathbb{S}$ *and* $n \leqslant N$*, the difference in value between the optimal and REVI policies is bounded by*

$$V^n(s) - V^{n,REVI}(s) \leqslant \max_{(t,a)_{n+1}^{N-1}} \sum_{k=n}^{N-2} \sqrt{2\pi} \|\tilde{\sigma}(\tilde{S}^k, t^{k+1}, a^{k+1})\|$$

where $\tilde{S}^k$ *is the state where only the covariance matrices are updated according to* $\{(t,a)\}_1^k$.

This result is similar to the result of Frazier et al. [2009a] Theorem 5. The proof has three parts, firstly the difference in one step of the optimal policy is derived, secondly, by induction this can be applied to multiple steps, thirdly, we can substitute the REVI value function into the previous results to yield a sub-optimality upper bound.

A.2.4 Preparatory Material

This first result is based on Frazier et al. [2009a] Lemma A.8 and derives the difference in value for one step. We use this later given that the value for one step ahead of the REVI policy and the optimal policy are equal.

Proposition A.2.3.1 *Let* $s \in \mathbb{S}$, *then* $V^{N-1}(s) \leqslant V^N(s) + \max_{(t,a)^N} \|\tilde{\sigma}(s, t^N, a^N)\|/\sqrt{2\pi}$.

Proof Bellman's Equation implies

$$V^{N-1}(s) = \max_{(t,a)^N} \mathbb{E}[V^N(S^N)|S^{N-1} = s, (t,a)^N].$$

We may find an upper bound for the inner term on the right hand side:

$$\begin{aligned} V^N(S^N) &= \sum_j w_j \max_a \mu^N_{j,a} \\ &= \sum_j w_j \max\{\mu^{N-1}_{j,a^N} + Z\tilde{\sigma}_j(S^{N-1}, t^N, a^N) \ , \ \max_{a' \neq a^N} \mu^{N-1}_{j,a'}\} \\ &\leq \sum_j w_j \max_a \{\mu^{N-1}_{j,a}\} + |Z||\tilde{\sigma}_j(S^{N-1}, t^N, a^N)| \\ &\leq V^N(S^{N-1}) + |Z|\|\tilde{\sigma}(S^{N-1}, t^N, a^N)\|. \end{aligned}$$

Substituting this back into the original proposition:

$$\begin{aligned} V^{N-1}(s) &\leq \max_{(t,a)^N} \mathbb{E}\left[V^N(S^{N-1}) + |Z|\|\tilde{\sigma}(S^{N-1}, t^N, a^N)\| \middle| S^{N-1} = s\right] \\ &= V^N(s) + \max_{(t,a)^N} \mathbb{E}\left[|Z|\|\tilde{\sigma}(S^{N-1}, t^N, a^N)\| \middle| S^{N-1} = s\right] \\ &= V^N(s) + \mathbb{E}[|Z|] \max_{(t,a)^N} \|\tilde{\sigma}(s, t^N, a^N)\| \\ &= V^N(s) + \max_{(t,a)^N} \|\tilde{\sigma}(s, t^N, a^N)\|/\sqrt{2\pi}. \end{aligned}$$

The following result is modified from Frazier et al. [2009a] Lemma A.9 which generalises the previous single step difference in value to multiple steps with induction.

Proposition A.2.3.2 *For a given state $S^n \in \mathbb{S}$, then*

$$V^n(S^n) \leqslant V^{N-1}(S^n) + \frac{1}{\sqrt{2\pi}} \max_{(t,a)_{n+1}^{N-1}} \sum_{k=n}^{N-2} \|\tilde{\sigma}(\tilde{S}^k, t^{k+1}, a^{k+1})\|$$

where $\tilde{S}^k$ is the state where only the covariance matrices Σ_a^k are sequentially updated according to the sampling sequence $\{(t,a)\}_n^k$ and Equation (7) in the main paper.

Proof We prove this by induction, the base case for $n = N-1$ is trivially true. For the following terms we first replace the term in Bellman's Equation using the induction hypothesis

$$\begin{aligned} V^n(s) &= \max_{(t,a)^{n+1}} \mathbb{E}[V^{n+1}(S^{n+1})|S^n = s] \\ &\leq \max_{(t,a)^{n+1}} \mathbb{E}\left[V^{N-1}(S^{n+1}) + \frac{1}{\sqrt{2\pi}} \max_{(t,a)_{n+2}^{N-1}} \sum_{k=n+1}^{N-2} \|\tilde{\sigma}(\tilde{S}^k, t^{k+1}, a^{k+1})\| \middle| S^n = s\right]. \end{aligned}$$

We may replace the $V^{N-1}(S^{n+1})$ using Proposition A.2.3.1

$$\begin{aligned} V^n(S^n) &\leq \max_{(t,a)^{n+1}} \mathbb{E}\Big[V^N(S^n) + \|\tilde{\sigma}(\tilde{S}^n, t^{n+1}, a^{n+1})\|/\sqrt{2\pi} \\ &\qquad + \max_{(t,a)_{n+2}^{N-1}} \sum_{k=n+1}^{N-2} \|\tilde{\sigma}(\tilde{S}^k, t^{k+1}, a^{k+1})\|/\sqrt{2\pi} \Big| S^n = s\Big] \\ &\leq \max_{(t,a)^{n+1}} \mathbb{E}\left[V^N(S^n) + \max_{(t,a)_{n+2}^{N-1}} \sum_{k=n}^{N-2} \|\tilde{\sigma}(\tilde{S}^k, t^{k+1}, a^{k+1})\|/\sqrt{2\pi} \Big| S^n = s\right] \\ &\leq V^{N-1}(S^n) + \max_{(t,a)_{n+1}^{N-1}} \sum_{k=n}^{N-2} \|\tilde{\sigma}(\tilde{S}^k, t^{k+1}, a^{k+1})\|/\sqrt{2\pi}. \end{aligned}$$

A.2.5 Proof of Theorem A.2.3

Theorem A.2.3 *For each $S^n \in \mathbb{S}$ and $n \leqslant N$,*

$$V^n(S^n) - V^{\mathrm{REVI},n}(S^n) \leqslant \max_{(t,a)_{n+1}^{N-1}} \sum_{k=n}^{N-2} \|\tilde{\sigma}(\tilde{S}^k, t^{k+1}, a^{k+1})\|/\sqrt{2\pi}.$$

Proof Since the REVI policy is myopically optimal we have that $V^{N-1}(s) = V^{\mathrm{REVI},N-1}(s)$. However, since the REVI policy is only myopically optimal we also have that $V^{\mathrm{REVI},N-1}(s) \leq V^{\mathrm{REVI},n}(s)$, substituting into the inequality in Propostion A.2.3.2 yields the above formula.

Appendix B

Proofs and Further Experiments from Chapter 8

B.0.1 Estimating the Target

This following result is a simple consequence of the symmetry of the model across seeds proven in Lemma B.0.2. This is consistent with other CRN and non-CRN methods that do not make the seed explicit but do incorporate off-diagonal noise matrix covariance. We first derive the posterior mean and then posterior covariance.

Proposition B.0.1 (Proposition 8.3.0.1) *For any given kernel over the domain $X \times \mathbb{N}^+$ that is of the form $k_{\bar{\theta}}(x, x') + \delta_{ss'} k_\epsilon(x, x')$, and a dataset of n input-output triplets D^n, the posterior over the target is a Gaussian process given by*

$$\begin{aligned} \bar{\theta}(x)|D^n &\sim GP(\mu^n_{\bar{\theta}}(x), k^n_{\bar{\theta}}(x, x')) \\ \mu^n_{\bar{\theta}}(x) &= \mu^n(x, s') \\ k^n_{\bar{\theta}}(x, x') &= k^n(x, s', x', s'') \end{aligned}$$

where $s', s'' \in \mathbb{N}^+ \setminus S^n$ with $s' \neq s''$ any two unobserved unequal seeds.

Lemma B.0.2 *Let $\theta(x, s)$ be a realization of a Gaussian Process with $\mu^0(x, s) = 0$ and any positive semi-definite kernel of the form $k(x, s, x', s') = k_{\bar{\theta}}(x, x') + \delta_{ss'} k_\epsilon(x, x')$. For all $x \in X$, $s_{obs} \in S^n$, and $s, s', s'' \in \mathbb{N}^+ \setminus S^n$, the posterior mean and kernel*

satisfy

$$\begin{aligned} \mu^n(x,s) &= \mu^n(x,s') \\ k^n(x,s_{obs},x',s) &= k^n(x,s_{obs},x',s') \qquad \text{(B.1)} \\ k^n(x,s,x',s') &= k^n(x,s,x',s'') = k^n(x,s',x',s'') \qquad \text{(B.2)} \end{aligned}$$

Proof Writing out the posterior mean in full from Equation 8.16,

$$\begin{aligned} \mu^n(x,s) &= k^0(x,s,\tilde{X}^n)K^{-1}Y^n \\ &= \begin{cases} \left(k_{\bar{\theta}}(x,X^n) + (\mathbb{1}_{s=S^n}^{\intercal} \circ k_\epsilon(x,X^n))\right)K^{-1}Y^n & s \in S^n \\ k_{\bar{\theta}}(x,X^n)K^{-1}Y^n & s \in \mathbb{N}^+ \setminus S^n \end{cases} \end{aligned}$$

where $\circ$ is element-wise product and $\mathbb{1}_{s=S^n}$ is a column vector of zeros for all $s \in \mathbb{N}^+ \setminus S^n$. The proofs for Equations B.1 and B.2 follow similarly from Equation 8.17. $\square$

We can now prove Proposition 8.3.0.1.

Proof The objective of optimization, $\bar{\theta}(x)$, is given by the average output over infinitely many seeds which may be written as the limit

$$\bar{\theta}(x) = \lim_{N_s \to \infty} \frac{1}{N_s} \sum_{s=1}^{N_s} \theta(x,s). \qquad \text{(B.3)}$$

Adopting the shorthand $\mathbb{E}_n[...] = \mathbb{E}[...|D^n]$, we first consider the posterior expected performance,

$$\begin{aligned} \mathbb{E}_n[\bar{\theta}(x)] &= \mathbb{E}_n\left[\lim_{N_s \to \infty} \frac{1}{N_s} \sum_{s=1}^{N_s} \theta(x,s)\right] \\ &= \lim_{N_s \to \infty} \frac{1}{N_s} \sum_{s=1}^{N_s} \mathbb{E}_n\left[\theta(x,s)\right] \\ &= \lim_{N_s \to \infty} \frac{1}{N_s} \sum_{s=1}^{N_s} \mu^n(x,s) \end{aligned}$$

Let $n_s = \max\{S^n\}$ be the largest observed seed. The sum of posterior means can be split into sampled seeds $s \in \{1, ..., n_s\}$ and unsampled seeds $s \in \{n_s + 1, ..., N_s\}$,

$$\begin{aligned}
\mathbb{E}_n[\bar{\theta}(x)] &= \lim_{N_s \to \infty} \frac{1}{N_s} \left(\sum_{s=1}^{n_s} \mu^n(x, s) + \sum_{s'=n_s+1}^{N_s} \mu^n(x, s') \right) \\
&= \lim_{N_s \to \infty} \frac{1}{N_s} \left(\sum_{s=1}^{n_s} \mu^n(x, s) + (N_s - n_s)\mu^n(x, n_s + 1) \right) \\
&= \lim_{N_s \to \infty} \frac{1}{N_s} \left(\sum_{s=1}^{n_s} \mu^n(x, s) - n_s\mu^n(x, n_s + 1) \right) + \mu^n(x, n_s + 1) \\
&= \mu^n(x, n_s + 1).
\end{aligned}$$

where we have used Lemma B.0.2 to simplify. Similarly for the covariance, writing each $\bar{\theta}(x)$ term as the limit of a sum over seeds,

$$\begin{aligned}
& \mathbb{E}_n \Big[\big(\bar{\theta}(x) - \mathbb{E}_n[\bar{\theta}(x)] \big) \big(\bar{\theta}(x') - \mathbb{E}_n[\bar{\theta}(x')] \big) \Big] \\
= \; & \mathbb{E}_n \left[\left(\lim_{N_s \to \infty} \frac{1}{N_s} \sum_{s=1}^{N_s} \theta(x, s) - \mu(x, s) \right) \left(\lim_{N_t \to \infty} \frac{1}{N_t} \sum_{s'=1}^{N_t} \theta(x', s') - \mu(x', s') \right) \right] \\
= \; & \lim_{N_s, N_t \to \infty} \frac{1}{N_s N_t} \sum_{s,s'=1}^{N_s, N_t} \mathbb{E}_n \left[(\theta(x, s) - \mu(x, s)) \big(\theta(x', s') - \mu(x', s') \big) \right] \\
= \; & \lim_{N_s, N_t \to \infty} \frac{1}{N_s N_t} \sum_{s,s'=1}^{N_s, N_t} k^n(x, s, x's')
\end{aligned}$$

The domain in the limit of the summation, $\mathbb{N}^+ \times \mathbb{N}^+$, is unaffected by setting $N_t = N_s$.

The summation decomposes into four terms,

$$\begin{aligned}
\sum_{s,s'=1}^{N_s} k^n(x,s,x',s') &= \underbrace{\sum_{s,s'=1}^{n_s} k^n(x,s,x',s')}_{\text{observed seeds full covariance}} + \underbrace{\sum_{s'=n_s+1}^{N_s}\sum_{s=1}^{n_s} k^n(x,s,x',s')}_{\text{observed-unobserved covariance}} \\
&\quad + \underbrace{\sum_{s=n_s+1}^{N_s} k^n(x,s,x',s)}_{\text{unobserved seeds variance}} + \underbrace{\sum_{n_s<s\neq s'\leq N_s} k^n(x,s,x',s')}_{\text{unobserved seeds covariance}} \\
\sum_{s,s'=1}^{N_s} k^n(x,s,x',s') &= \underbrace{\sum_{s,s'=1}^{n_s} k^n(x,s,x',s')}_{\text{constant with } N_s} + \underbrace{2(N_s-n_s)\sum_{s=1}^{n_s} k^n(x,s,x',s')}_{\text{linear with } N_s} \\
&\quad + \underbrace{(N_s-n_s)k^n(x,s',x',s')}_{\text{linear with } N_s} + \underbrace{(N_s-n_s)^2 k^n(x,s',x',s'')}_{\text{quadratic with } N_s}
\end{aligned}$$

where s' and s'' are two unequal unobserved seeds. Dividing the final Equation by N_s^2 and taking the limit $N_s \to \infty$, only the final term remains. □

Given the assumed kernel with independent and identically distributed difference functions, the average of all seeds includes finite observed seeds and infinitely many identical unobserved seeds that dominate. Hence any one unobserved seed is an estimate for the objective. Likewise the covariance across any two unique unobserved seeds dominates the summation of full posterior covariance for the objective. Also note that the prior kernel is unchanged, $\bar{k}^0(x,x') = k^0(x,1,x',2) = k_{\bar{\theta}}(x,x')$ as desired.

B.0.2 Proof of Theorem 8.4.1

Theorem B.0.3 (Theorem 8.4.1) *Let $x_r^N \in A$ be the point that KG^{CRN} recommends in iteration N. For each $p \in [0,1)$ there is a constant K_p such that with probability p*

$$\lim_{N\to\infty} \bar{\theta}(x_r^N) > \bar{\theta}(x^{OPT}) - K_p d$$

First we define $V^n(x,x') = \mathbb{E}_n[\bar{\theta}(x)\bar{\theta}(x')]$.

Lemma B.0.4 *Let $x, x' \in X$, the limits of the series $(\bar{\mu}^n(x))_n$ and $(V^n(x,x'))_n$ exist. Denote then by $\bar{\mu}^\infty(x)$ and $V^\infty(x,x')$, respectively. We have*

$$\lim_{n\to\infty} \bar{\mu}^n(x) = \bar{\mu}^\infty(x) \tag{B.4}$$

$$\lim_{n\to\infty} V^n(x,x') = V^\infty(x,x') \tag{B.5}$$

almost surely.

Proof $\bar{\theta}(x)$ and $\bar{\theta}(x)\bar{\theta}(x')$ are integrable random variables for all $x, x' \in X$ by choice of $\bar{\theta}$. Proposition 2.7 in Çınlar [2011] states that any sequence of conditional expectations of an integrable random variable under an increasing filtration is uniformly integrable martingale. Thus, both sequences converge almost surely to their respective limit. □

Lemma B.0.5 *$KG_n^{CRN}(x,s) \geq 0$ for all $(x,s) \in X \times \mathbb{N}^+$.*

Proof Adopting the shorthand $x_r^n = \underset{x\in X}{\operatorname{argmax}}\, \mu^n(x,0)$,

$$\begin{aligned}
\mathrm{KG}_n^{\mathrm{CRN}}(x,s) &= \mathbb{E}\left[\max_{x'\in X} \mu^n(x',0) + \tilde{\sigma}^n(x',0;x,s)Z - \mu^n(x_r^n,0)\right] \\
&= \mathbb{E}\left[\max_{x'\in X} \mu^n(x',0) + (\tilde{\sigma}^n(x',0;x,s) - c)Z - \mu^n(x_r^n,0)\right]
\end{aligned}$$

for any arbitrary constant c. By setting $c = \tilde{\sigma}^n(x_r^n,0;x,s)$, the inner expression satisfies $\mu^n(x_r^n,0) + (\tilde{\sigma}^n(x_r^n,0;x,s) - c)Z - \mu^n(x_r^n,0) = 0$ for all $Z \in \mathbb{R}$ and

$$\begin{aligned}
&\max_{x'\in X}\{\mu(x',0) + (\tilde{\sigma}^n(x',0;x,s) - c)Z - \mu_0\} \\
&\qquad \geq \mu(x_r^n,0) + (\tilde{\sigma}^n(x_r^n,0;x,s) - c)Z - \mu^n(x_r^n,0) = 0
\end{aligned}$$

for all Z and $\mathrm{KG}_n^{\mathrm{CRN}}(x,s)$ may be written as the expectation of a non-negative random variable. □

Lemma B.0.6 *Given deterministic simulation outputs, there is no improvement in*

re-sampling a sampled point.

$$KG_n^{CRN}(x^i, s^i) = 0$$

for all $(x^i, s^i) \in \tilde{X}^n$.

Proof The posterior covariance between the output at any point and the output at an observed point is zero,

$$\begin{aligned}
k^n(x^i, s^i; x, s) &= k^0(x^i, s^i; x, s) - k^0(x^i, s^i; \tilde{X}^n) \left(k^0(\tilde{X}^n; \tilde{X}^n)\right)^{-1} k^0(\tilde{X}^n; x, s) \\
&= k^0(x^i, s^i; x, s) - \underbrace{[k^0(\tilde{X}^n; \tilde{X}^n)]_i}_{i^{th} \text{ row}} \left(k^0(\tilde{X}^n; \tilde{X}^n)\right)^{-1} k^0(\tilde{X}^n; x, s) \\
&= k^0(x^i, s^i; x, s) - e_i{}^\intercal k^0(\tilde{X}^n; x, s) \\
&= k^0(x^i, s^i; x, s) - k^0(x^i, s^i; x, s) \\
&= 0
\end{aligned}$$

where e_i is the i^{th} row of the $n \times n$ identity matrix. Therefore $\tilde{\sigma}^n(x, s; x^i, s^i) = 0$ for all (x, s) and $\mathrm{KG}_n^{\mathrm{CRN}}(x, s) = 0$. □

Let ω denote an arbitrary sample path and note that ω determines an observation for each query to a seed, as $n \to \infty$. Lemmas B.0.5 and B.0.6 and noting that $(x, s)^{n+1} = \operatorname{argmax} \mathrm{KG}_n^{\mathrm{CRN}}(x, s)$ together imply that no input (x, s) will be sampled more than once when using $\mathrm{KG}^{\mathrm{CRN}}$ and so we only consider sample paths ω where all elements are unique. Given finite X, there must be an $x \in X$ that is observed for infinite seeds on ω. We study the asymptotic behaviour $\mathrm{KG}_n^{\mathrm{CRN}}(x, s)$ for $n \to \infty$ as a function of $\mu^n(x, 0)$, $\tilde{\sigma}^n(x', 0, x, s)$.

Lemma B.0.7 *If* x *is sampled for infinitely many seeds, then* $\tilde{\sigma}^\infty(x', 0; x, s) = 0$ *for all* $x' \in X$ *and* $KG_\infty^{CRN}(x, s) = 0$ *for all* $s \in \mathbb{N}^+$ *almost surely.*

Proof Setting $x^{n+1} = x$ and assuming (x^i, s^i) pairs are arranged such that s^{n+1} is

always a new seed,

$$\begin{aligned}
\lim_{n\to\infty} \tilde{\sigma}^n(x', 0; x, s^{n+1}) &= \lim_{n\to\infty} \frac{|k^n(x', 0, x, s^{n+1})|}{\sqrt{k^n(x, s^{n+1}, x, s^{n+1})}} \\
&= \lim_{n\to\infty} \frac{|\bar{k}^n(x', x)|}{\sqrt{\bar{k}^n(x, x) + k_\epsilon(x, x)}} \\
&\leq \lim_{n\to\infty} \sqrt{\bar{k}^n(x', x')} \frac{\sqrt{\bar{k}^n(x, x)}}{\sqrt{\bar{k}^n(x, x) + k_\epsilon(x, x)}} \\
&= 0
\end{aligned}$$

where the final line is by noting that $\bar{k}^n(x, x) + k_\epsilon(x, x) > 0$ for all n and x. □

Lemma B.0.8 *Let (x, s) be an input pair for which $KG_n^{CRN}(x, s) = 0$. Then for all $x' \in X$*

$$\tilde{\sigma}^n(x', 0; x, s) = c$$

where c is a constant.

Proof From Equation B.6, $\mathrm{KG}_n^{\mathrm{CRN}}(x, s)$ can be written as the expectation of a non-negative random variable. Therefore the random variable itself must equate to zero almost surely implying

$$\begin{aligned}
\max_{x'\in X}\{\mu(x', 0) + (\tilde{\sigma}^n(x', 0; x, s) - c)Z - \mu^n(x_r^n, 0)\} &= 0 \\
\max_{x'\in X}\{\mu(x', 0) + (\tilde{\sigma}^n(x', 0; x, s) - c)Z\} &= \max_{x''\in X}\{\mu(x'', 0)\}
\end{aligned}$$

for all $Z \in \mathbb{R}$. This implies $\tilde{\sigma}^n(x', 0; x, s) = c$ for all $x' \in X$. □
Note the case where $(x, s) \in \tilde{X}^n$ we have that $\tilde{\sigma}(x', 0; x, s) = 0$ for all $x' \in X$.

Lemma B.0.9 *Let $s \in \mathbb{N}^+ \setminus S^n$ be an unobserved seed, if $KG_n^{CRN}(x, s) = 0$ for all $x \in X$, then* $\operatorname{argmax}_x \mu^n(x, 0) = \operatorname{argmax}_x \bar{\theta}(x)$

Proof By Lemma B.0.8, we have that $\bar{k}^n(x, x') = c$ for all $x, x' \in X$ and the covariance matrix $\bar{k}^n(X, X)$ is proportional to the all ones matrix. Hence $\bar{\theta}(x) - \mu^n(x, 0)$ is a normal random variable that is constant across all $x \in X$ and $\operatorname{argmax}_{x\in X} \mu(x, 0) =$

$\underset{x \in X}{\text{argmax}}\ \bar{\theta}(x)$ holds. $\square$

Lemmas B.0.6, B.0.7, consider evaluating KG^{CRN} as the sampling budget increases in a specific way. More generally, recall that KG^{CRN} picks $(x,s)^{n+1} \in \underset{x,s}{\text{argmax}}\ \text{KG}_n^{\text{CRN}}(x,s)$ in each iteration n. Since $\theta(x,\cdot)$ is evaluated infinitely often (by choice of x), $\text{KG}_n^{\text{CRN}}(x,\cdot) \to 0$ for all $x \in A$ holds almost surely and by Lemma B.0.9 the true optimizer is known. There exist many proofs for KG like policies showing that sample paths ω with finitely evaluated $\theta(x,\cdot)$ are impossible, or all x are evaluated infinitely often almost surely [Frazier et al., 2008, 2009b, Salemi et al., 2019, Xie et al., 2016] and hence we refrain from duplicating a proof here. Most (if not all) proofs follow a similar argument by contradiction, if data is sequentially allocated to *maxima* of the non-negative function $\text{KG}^{\text{CRN}}(x,s)$, then data will never be re-allocated to an (x,s) for which $\bar{\theta}(x)$ is known as such points are known *minima* of $\text{KG}^{\text{CRN}}(x,s) = 0$.

Next we consider a bound on the loss due to discretization of a continuous search space. Suppose that $X \subset \mathbb{R}^d$ is a compact infinite set and $A \subset X$ is a finite set of discretization points. Suppose that $\bar{\mu}^0(x) = 0$ for all x, and $k_{\bar{\theta}}(x,x')$ is a four times differentiable Matern kernel e.g. the popular squared exponential kernel. Suppose that $\bar{\theta}(x)$ is drawn from the prior, i.e. let $\bar{\theta}(x) \sim GP(\bar{\mu}^0(x), k_{\bar{\theta}}(x,x'))$ then the sample $\bar{\theta}(x)$ over the set of functions is itself twice differentiable in X with probability one. Let $x^{OPT} = \underset{x \in X}{\text{argmax}}\ \bar{\theta}(x)$ and $d = \max_{x' \in X} \min_{x \in A} \text{dist}(x,x')$ be the largest distance from any point in the continuous domain X to its nearest neighbor in A.

Proof The extrema of $\frac{\delta}{\delta x_i}\bar{\theta}(x)$ over X are bounded, the partial derivatives of $\bar{\theta}(x)$ are also GPs for our choice of $k_{\bar{\theta}}(x,x)$. Thus we can compute for every $p \in [0,1)$ a constant K_p such that $\bar{\theta}(x)$ is K_p Lipschitz continuous on X with probability at least p, then there exists an $\bar{x} \in A$ with $\text{dist}(\bar{x}, x^{OPT}) \leq d$ and

$$\bar{\theta}(\bar{x}) > \bar{\theta}(x^{OPT}) - K_p d$$

holds with probability p. Finally the point recommended by KG^{CRN} is the maximizer

of $x_r^N \in \underset{x \in A}{\text{argmax}}\ \bar{\theta}(x)$ and therefore is not worse than $\bar{x}$

$$\begin{aligned}\lim_{N\to\infty} \bar{\theta}(x_r^N) &\geq \bar{\theta}(\bar{x}) \\ &\geq \bar{\theta}(x^{OPT}) - K_p d\end{aligned}$$

□

B.0.3 Proof of Propositions 8.5.3 and 8.5.4

We next provide proofs for algorithm behaviour in the case of compound sphericity, recall this corresponds to the difference functions reducing to constant offsets. Lemma 8.5.1 states that the difference $\mu^n(x,s) - \mu^n(x,s') = A_s - A_{s'}$ is constant for all x. Likewise the same relationship applies to $\tilde{\sigma}^n(x',s';x,s)$ that quantifies changes in the posterior mean and therefore must also maintain the symmetry over seeds s',

Lemma B.0.10 *Let $x, x' \in X$, $s, s' \in \mathbb{N}^+$, then the difference in posterior mean updates satisfies*

$$\tilde{\sigma}^n(x',s';x,s) = \tilde{\sigma}^n(x',0;x,s) + h^n(s',x,s).$$

Proof

$$\begin{aligned}\tilde{\sigma}^n(x',s';x,s) &= \frac{k^n(x',s';x,s)}{\sqrt{k^n(x,s,x,s)}} \\ &= \frac{1}{\sqrt{k^n(x,s,x,s)}}\Bigg(k_{\bar{\theta}}(x',x) + \eta^2\delta_{ss'} \\ &\qquad -\Bigg(k_{\bar{\theta}}(x',X^n) + \eta^2 \mathbb{1}_{s'=S^n}^\intercal\Bigg)K^{-1}\Bigg(k_{\bar{\theta}}(X^n,x) + \eta^2\mathbb{1}_{s=S^n}\Bigg)\Bigg) \\ &= \tilde{\sigma}(x',0;x,s) + \frac{\eta^2\delta_{ss'} - \eta^2\mathbb{1}_{s'=S^n}^\intercal K^{-1}\Bigg(k_{\bar{\theta}}(X^n,x) + \eta^2\mathbb{1}_{s=S^n}\Bigg)}{\sqrt{k^n(x,s,x,s)}} \\ &= \tilde{\sigma}(x',0;x,s) + h(s',x,s)\end{aligned}$$

□

As a result of the symmetry over seeds it is possible to use any seed $s \in \mathbb{N}^+$ as the target of optimization formalized in the following Lemma.

Lemma B.0.11 *Let $x \in X$, $s, s' \in \mathbb{N}^+$, then*

$$KG_n^{CRN}(x,s) = \mathbb{E}[\max_{x' \in X} \mu^n(x',s') + \tilde{\sigma}^n(x',s';x,s)Z - \max_{x'' \in X} \mu^n(x'',s')].$$

Proof

$$\begin{aligned}
\mathrm{KG}_n^{\mathrm{CRN}}(x,s) &= \mathbb{E}[\max_{x' \in X} \mu^n(x',0) + \tilde{\sigma}^n(x',0;x,s)Z - \max_{x'' \in X} \mu^n(x'',0)] \\
&= \mathbb{E}[\max_{x' \in X} \mu^n(x',s') - A_{s'} + (\tilde{\sigma}^n(x',s';x,s) - h(s',x,s))Z \\
&\qquad - \max_{x'' \in X} \mu^n(x'',s') - A_{s'}] \\
&= \mathbb{E}[\max_{x' \in X} \mu^n(x',s') + \tilde{\sigma}^n(x',s';x,s)Z - \max_{x'' \in X} \mu^n(x'',s')] - h(s',x,s)\mathbb{E}[Z] \\
&= \mathbb{E}[\max_{x' \in X} \mu^n(x',s') + \tilde{\sigma}^n(x',s';x,s)Z - \max_{x'' \in X} \mu^n(x'',s')]
\end{aligned}$$

□

Proposition B.0.12 (Proposition 8.5.3) *Let $\theta(x,s)$ be a realization of a Gaussian process with the compound spheric kernel and $\rho = 1$. If $X = \{x_1, ..., x_d\}$ and $\tilde{X}^n = \{(x_1,1), ..., (x_d,1)\}$ then for all $(x,s) \in X \times S$*

$$KG^{CRN}(x,s) = 0$$

and the maximizer of $\bar{\theta}(x)$ is known.

Proof Lemma B.0.11 shows that any seed can be used as the target of optimization. Therefore we may choose $s = 1$ as the target. All x have been sampled for $s = 1$ therefore $\tilde{\sigma}^n(x,1;x',s') = 0$ for all $x \in X$ and $s' \in \mathbb{N}^+$. Hence

$$\begin{aligned}
\mathrm{KG}^{\mathrm{CRN}}(x,s) &= \mathbb{E}[\max_{x' \in X} \mu^n(x',1) + 0Z - \max_{x'' \in X} \mu^n(x'',1)] \\
&= 0
\end{aligned}$$

for all $x, s \in X \times \mathbb{N}^+$. By Lemma B.0.9 the maximizer $\operatorname{argmax}_{x \in X} \bar{\theta}(x)$ is known (although its underlying value, $\max \bar{\theta}(x)$, is not known). □

Proposition B.0.13 (Proposition 8.5.4) *Let $\theta(x,s)$ be a realization of a Gaussian process with the compound spheric kernel with $\rho = 1$. Let $X \subset \mathbb{R}^d$ be a set of possible solutions and let $\tilde{X}^n = \{(x^1,1),...,(x^n,1)\}$ be the set of sampled locations and $X^n = (x^1,...,x^n)$. Define*

$$KG_n^{CRN}(x,s;A) = \mathbb{E}\left[\max_{x' \in A \cup \{x\}} \mu^{n+1}(x',0) - \max_{x' \in A \cup \{x\}} \mu^n(x',0) \middle| D^n, (x,s)^{n+1} = (x,s)\right].$$

Then for all $x \in X$

$$KG_n^{CRN}(x,1;X^n) > KG_n^{CRN}(x,2;X^n)$$

and therefore $\max_x KG_n^{CRN}(x,1;X^n) > \max_x KG_n^{CRN}(x,2;X^n)$ and seed $s = 2$ will never be evaluated.

Proof By Lemma B.0.11, we may set $s = 1$ as the target of optimization. For all sampled points $i = 1,...,n$, we have that $\tilde{\sigma}^n(x^i,1;x,s) = 0$ and $\mu^n(x^i,1) = y^i$. Let $\bar{Y}^n = \max Y^n$. The expression for Knowledge Gradient becomes

$$\begin{aligned}
\mathrm{KG}_n^{\mathrm{CRN}}(x,s;X^n) &= \mathbb{E}\big[\max\{\bar{Y}^n, \mu^n(x,1) + \tilde{\sigma}^n(x,1;x,s)Z\}\big] - \max\{\bar{Y}^n, \mu^n(x,1)\} \\
&= \mathbb{E}\big[\max\{0, \mu^n(x,1) - \bar{Y}^n + \tilde{\sigma}^n(x,1;x,s)Z\}\big] \\
&= \Delta(x)\Phi\left(\frac{\Delta(x)}{|\tilde{\sigma}^n(x,1;x,s)|}\right) - |\tilde{\sigma}^n(x,1;x,s)|\phi\left(\frac{\Delta(x)}{|\tilde{\sigma}^n(x,1;x,s)|}\right) \\
&= f\big(\Delta(x),\ |\tilde{\sigma}^n(x,1;x,s)|\big)
\end{aligned}$$

where $\Phi(\cdot), \phi(\cdot)$ and cumulative and density functions of the Gaussian distribution, $\Delta(x) = \mu^n(x,1) - \bar{Y}^n$ and $f(a,b)$ is the well known expected improvement acquisition function derived from the expectation of a truncated Gaussian random variable. Note that the function $f(a,b)$ is monotonically increasing in b, $\frac{d}{db}f(a,b) = \phi(-a/b) > 0$. Hence, to prove the proposition, it is sufficient to show $|\tilde{\sigma}^n(x,1;x,1)| > |\tilde{\sigma}^n(x,1;x,2)|$

for all $x \in X$. Firstly we may simplify $\tilde{\sigma}^n(x,1;x,1)$ as follows

$$\begin{aligned} \tilde{\sigma}^n(x,1;x,1) &= k^n(x,1,x,1)/\sqrt{k^n(x,1,x,1)} && \text{(B.6)} \\ &= \sqrt{k^n(x,1,x,1)}. && \text{(B.7)} \end{aligned}$$

Substituting this into the inequality yields

$$\begin{aligned} |\tilde{\sigma}^n(x,1;x,1)| &> |\tilde{\sigma}^n(x,1;x,2)| \\ \sqrt{k^n(x,1,x,1)} &> \frac{|k^n(x,1,x,2)|}{\sqrt{k^n(x,2,x,2)}} \\ 1 &> \frac{|k^n(x,1,x,2)|}{\sqrt{k^n(x,2,x,2)k^n(x,1,x,1)}} \\ -1 &< \text{corr}(\theta(x,1),\theta(x,2)|D^n) \leq 1 \end{aligned}$$

where the last line is true by the positive semi-definiteness of the kernel, the correlation between two random variables cannot be greater than one. □

The above proof demonstrates that allocating samples according to KG^{CRN} will always sample seed $s = 1$. The target is stochastic however the objective is deterministic and the new output $y^{n+1} \sim N(\mu^n(x,1), k^n(x,1,x,1))$. The acquisition function simplifies to

$$\begin{aligned} \text{KG}_n^{\text{CRN}}(x,1;X^n) &= \mathbb{E}\big[\max\{0, \mu^n(x,1) + \sqrt{k^n(x,1,x,1)}Z - \bar{Y}^n\}\big] \\ &= \mathbb{E}\big[\max\{0, y^{n+1} - \bar{Y}^n\}\big|D^n, x^{n+1} = x, s^{n+1} = 1\big] \end{aligned}$$

where the last line is exactly the EI acquisition function of the famous EGO algorithm of Jones et al. [1998b].

B.1 Further Experimental Results

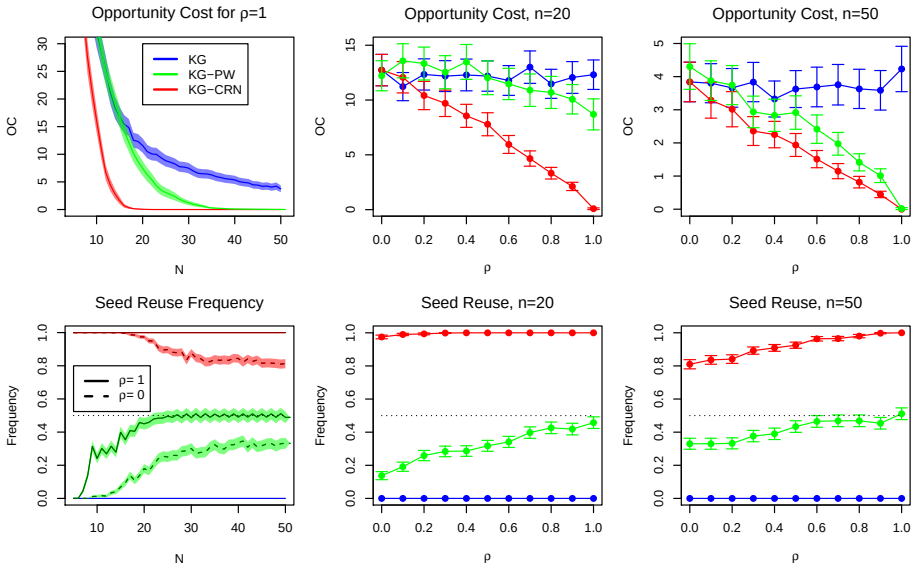


Figure B.1: Synthetic GP data with offsets and white noise only (compound spheric). For low ρ, all algorithms perform similarly. As ρ increases, KG^{CRN} samples more old seeds and outperforms other methods, KG^{PW} samples singles first, duplicating KG and sampling doubles later improving upon KG.

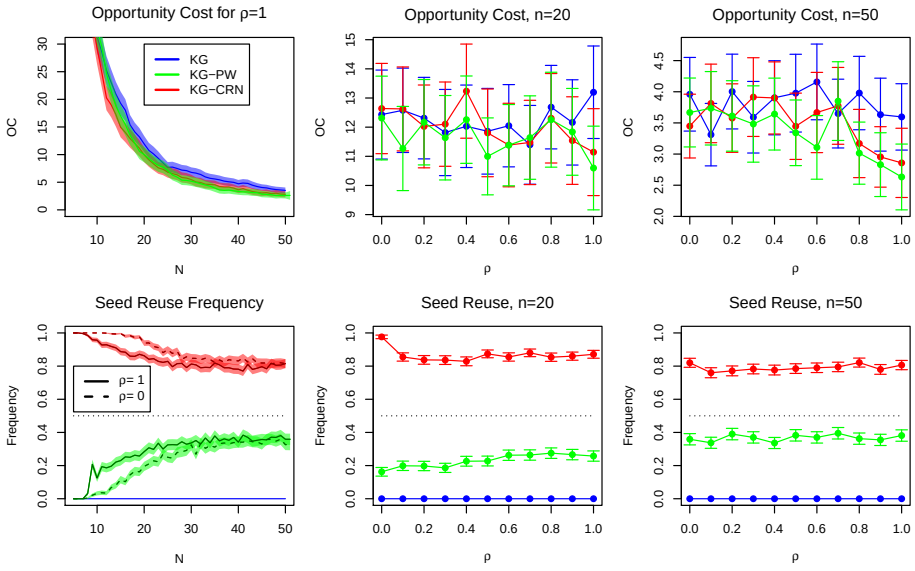


Figure B.2: GP synthetic data generated with $\eta^2 = 0$ and $\rho = \sigma_b^2/(\sigma_b^2 + \sigma_w^2)$ holding $\sigma_b^2 + \sigma_w^2 = 50^2$ constant. There is no significant benefit from bias functions alone. Local difference correlation, is only useful when combined with global difference correlation.

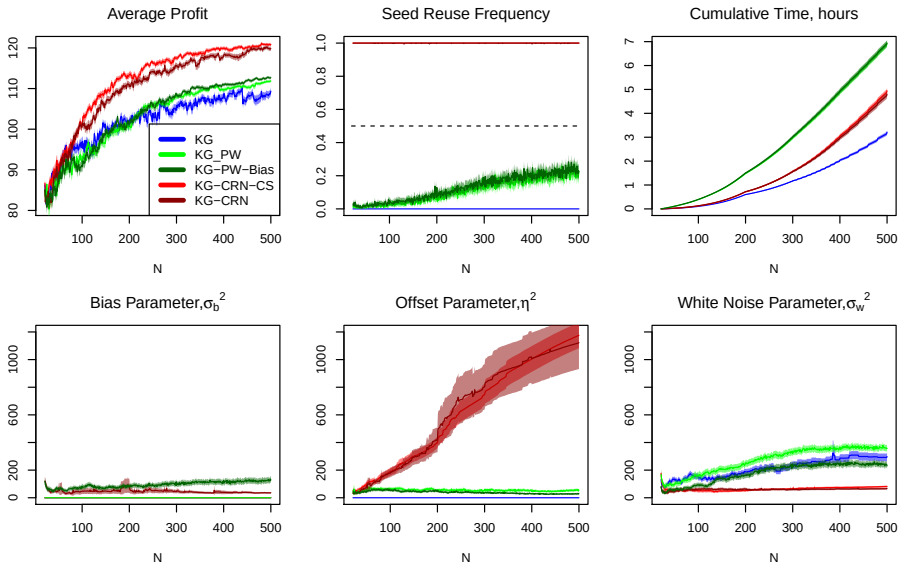


Figure B.3: ATO results. The KG^{PW} algorithm samples singles early on, and never learns a large offset parameter η^2. KG^{CRN} samples old seeds and eventually learns a large offset parameter and never samples any new seeds. KG has smallest runtime, followed by KG^{CRN} variants then KG^{PW} variants.

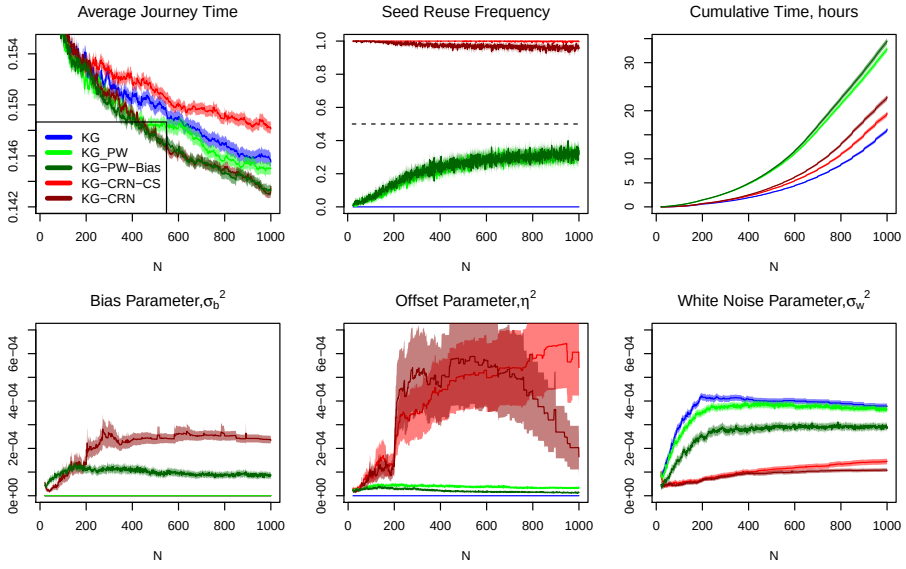


Figure B.4: The bias functions provide significant benefit to both KG^{CRN} and KG^{PW}. Excluding bias functions, $\text{KG}^{\text{CRN}} - CS$, leads to significant detriment sampling only old seeds. The KG^{CRN} variants learn larger offset parameters and require much much less computation time.

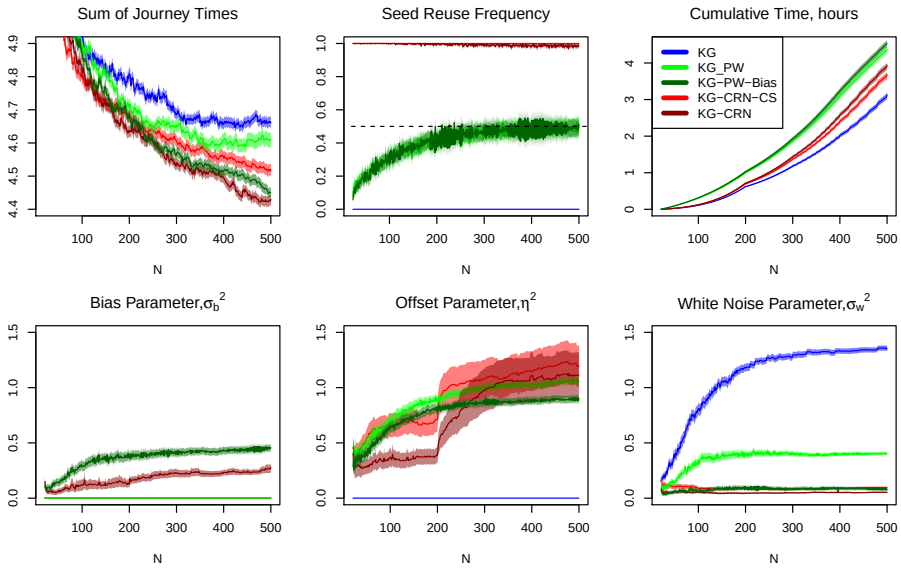


Figure B.5: The AIS problem with the sum of journey times in a simulation as the objective. KG^{CRN} variants improve performance over KG, and bias functions improve performance over compound spheric variants. Seed reuse is almost maximized by all methods.

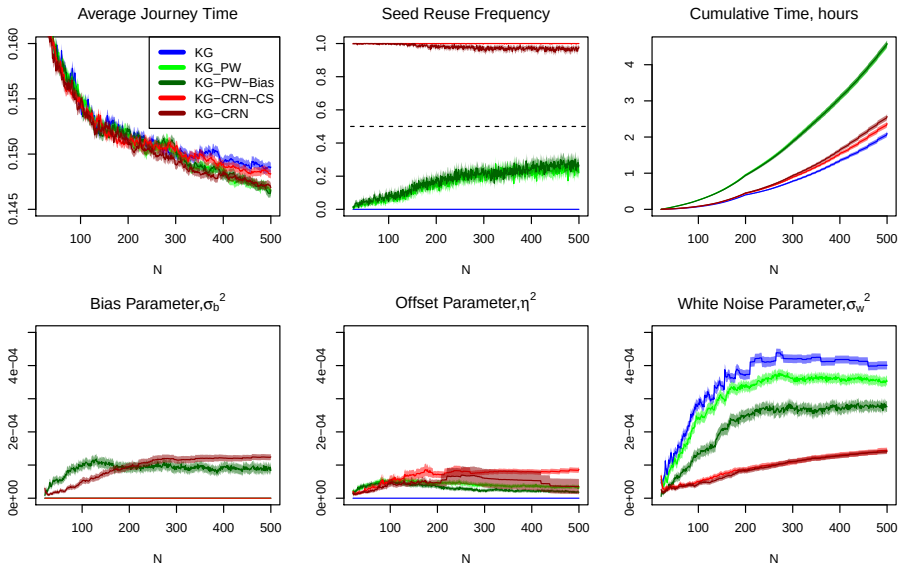


Figure B.6: All algorithm variants perform similarly and the offset and bias parameters are much lower than the white noise parameter suggesting there is little exploitable structure in the noise for this problem.

B.2 Algorithm Implementation Details

B.2.1 Hyperparameter Learning

All parameters are learnt by multi-start conjugate gradient ascent of the marginal likelihood Rasmussen [2003]

$$\begin{aligned}\mathbb{P}[Y^n|\tilde{X}^n, L, \sigma_{\bar{\theta}}^2, \eta^2, \sigma_b^2, \sigma_w^2] &= -\frac{1}{2}\Big((Y^n-\bar{Y})^\intercal K^{-1}(Y^n-\bar{Y}) + \log(|K|) + n\log(2\pi)\Big) \\ K_{ij} &= \sigma_{\bar{\theta}}^2 \exp\left(-\frac{1}{2}(x^i-x^j)^\intercal L(x^i-x^j)\right) \\ &\quad +\mathbb{1}_{s^i=s^j}\left(\eta^2+\sigma_b^2\exp\left(-\frac{1}{2}(x^i-x^j)^\intercal L(x^i-x^j)\right)\right. \\ &\quad \left. +\mathbb{1}_{x^i=x^j}\sigma_w^2\right).\end{aligned}$$

Firstly, an independent noise model is fitted by clamping $\eta^2 = \sigma_b^2 = 0$ to yield

$$L^{IND}, {\sigma_{\bar{\theta}}^2}^{IND}, {\sigma_w^2}^{IND} = \text{argmax}\ \mathbb{P}[Y^n|\tilde{X}^n, L, \sigma_{\bar{\theta}}^2, \eta^2 = \sigma_b^2 = 0, \sigma_w^2] \tag{B.8}$$

Secondly, the noise parameters $\eta^2, \sigma_b^2, \sigma_w^2$ are optimized whilst keeping keeping the total noise fixed $\eta^2 + \sigma_b^2 + \sigma_w^2 = {\sigma_w^2}^{IND}$ which is a two-dimensional optimization, we reparameterize as follows

$$\begin{aligned}\eta^2(\alpha,\beta) &= \beta(1-\alpha){\sigma_w^2}^{IND} \\ \sigma_b^2(\alpha,\beta) &= (1-\beta)(1-\alpha){\sigma_w^2}^{IND} \\ \sigma_w^2(\beta) &= \alpha{\sigma_w^2}^{IND} \\ \alpha,\beta &= \underset{[0,1]^2}{\text{argmax}}\ \mathbb{P}[Y^n|\tilde{X}^n, L^{IND}, {\sigma_{\bar{\theta}}^2}^{IND}, \eta^2(\alpha,\beta), \sigma_b^2(\alpha,\beta), \sigma_w^2(\beta)]\end{aligned}$$

Thirdly, the final MLE estimates of all parameters are simultaneously fine-tuned by gradient ascent. This three-stage method guarantees that the found likelihood is greater than the equivalent non-CRN parameter estimates and the second extra step of optimization is only 2-dimensional.

B.2.2 Optimization of $\mathbf{KG}_n^{\mathbf{CRN}}(x, s)$

Derivatives of $\mathrm{KG}^{\mathrm{CRN}}$ and $\mathrm{KG}^{\mathrm{PW}}$, when evaluated by discretization over X as we do, are easily (but tediously) derived and can be found in multiple previous works such as Scott et al. [2011b], Xie et al. [2016]. Alternatively, any automatic differentiation package, (Autograd, TensorFlow, PyTorch) may be used as the mathematical operations are all common functions. We propose the following optimization procedure. Firstly, $\mathrm{KG}^{\mathrm{CRN}}(x, s)$ is evaluated across an initial Latin Hypercube design with 1000 points over the acquisition space $\tilde{X}_{acq} = X \times \{1, ..., \max S^n + 1\}$. Secondly, the top 20 initial points are used to initialize 100 steps of conjugate gradient ascent over X holding the seed constant within each run. Thirdly, for the largest (x, s) pair found, $\mathrm{KG}^{\mathrm{CRN}}(x, s)$ is evaluated for the same x on all seeds $s \in \{1, ..., \max S^n + 1\}$ and finally 20 steps of gradient ascent are applied to fine tune the x from the best seed. When not using common random numbers, stages one and two all use the same new seed and stages three and four are skipped.

Bibliography

Raul Astudillo and P Frazier. Multi-attribute bayesian optimization under utility uncertainty. In *Proceedings of the NIPS Workshop on Bayesian Optimization*, 2017.

R. Bardenet, Mátyás Brendel, Balázs Kégl, and Michele Sebag. Collaborative hyperparameter tuning. In *International Conference on Machine Learning*, pages 199–207, 2013.

Mauro Birattari, Thomas Stützle, Luis Paquete, and Klaus Varrentrapp. A racing algorithm for configuring metaheuristics. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 11–18. Morgan Kaufmann Publishers Inc., 2002.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

Edwin V Bonilla, Kian M Chai, and Christopher Williams. Multi-task gaussian process prediction. In *Advances in neural information processing systems*, pages 153–160, 2008.

J. Branke, S. E. Chick, and C. Schmidt. Selecting a Selection Procedure. *Management Science*, 53(12):1916–1932, 2007.

Jürgen Branke and Jawad Elomari. Racing with a fixed budget and a self-adaptive significance level. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7997 LNCS: 272–280, 2013. ISSN 03029743. doi: 10.1007/978-3-642-44973-4{_}29.

Kuo-Hao Chang, L Jeff Hong, and Hong Wan. Stochastic trust-region response-surface method (strong)a new response-surface framework for simulation optimization. *INFORMS Journal on Computing*, 25(2):230–243, 2013.

C.-H. Chen, S. E. Chick, L. H. Lee, and N. A. Pujowidianto. Ranking and selection: Efficient simulation budget allocation. In *Handbook of Simulation Optimization*, pages 45–80. Springer, 2015.

Chun-Hung Chen. A lower bound for the correct subset-selection probability and its application to discrete-event system simulations. *IEEE transactions on automatic control*, 41(8):1227–1231, 1996.

Xi Chen, Bruce E Ankenman, and Barry L Nelson. The effects of common random numbers on stochastic kriging metamodels. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 22(2):7, 2012.

C. Chevalier, J. Bect, D. Ginsbourger, E. Vazquez, V. Picheny, and Y. Richet. Fast parallel kriging-based stepwise uncertainty reduction with application to the identification of an excursion set. *Technometrics*, 56(4):455–465, 2014.

Stephen E Chick and Koichiro Inoue. New two-stage and sequential procedures for selecting the best simulated system. *Operations Research*, 49(5):732–743, 2001.

Stephen E. Chick, Jürgen Branke, and Christian Schmidt. Sequential Sampling to Myopically Maximize the Expected Value of Information. *INFORMS Journal on Computing*, 22(1):71–80, feb 2010. doi: 10.1287/ijoc.1090.0327. URL `http://pubsonline.informs.org/doi/abs/10.1287/ijoc.1090.0327`.

Tinkle Chugh. Scalarizing functions in bayesian multiobjective optimization. *arXiv preprint arXiv:1904.05760*, 2019.

Erhan Çınlar. *Probability and stochastics*, volume 261. Springer Science & Business Media, 2011.

Lawrence Davis. Handbook of genetic algorithms. 1991.

Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

Nathaniel Eldredge. Analysis and probability on infinite-dimensional spaces. *arXiv preprint arXiv:1607.03591*, 2016.

Alexander IJ Forrester, András Sóbester, and Andy J Keane. Multi-fidelity optimization via surrogate modelling. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 463:2088, pages 3251–3269. The Royal Society, 2007.

P. Frazier, W. Powell, and S. Dayanik. The knowledge-gradient policy for correlated normal beliefs. *INFORMS Journal on Computing*, 21(4):599–613, 2009a.

P. I. Frazier and A. M. Kazachkov. Guessing preferences: A new approach to multi-attribute ranking and selection. In *Proceedings of the 2011 Winter Simulation Conference (WSC)*, pages 4319–4331, Dec 2011. doi: 10.1109/WSC.2011.6148119.

P. I. Frazier, W. B. Powell, and S. Dayanik. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47 (5):2410–2439, sep 2008.

Peter Frazier, Warren Powell, and Savas Dayanik. The knowledge-gradient policy for correlated normal beliefs. *INFORMS journal on Computing*, 21(4):599–613, 2009b.

Michael C Fu. Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing*, 14(3):192–215, 2002.

Michael C Fu, J-Q Hu, C-H Chen, and Xiaoping Xiong. Optimal computing budget allocation under correlated sampling. In *Proceedings of the 2004 Winter Simulation Conference, 2004.*, volume 1. IEEE, 2004.

D. Ginsbourger, J. Baccou, C. Chevalier, F. Perales, N. Garland, and Y Monerie. Bayesian adaptive reconstruction of profile optima and optimizers. *SIAM/ASA Journal on Uncertainty Quantification*, 2(1):490–510, 2014.

Björn Görder and Michael Kolonko. Ranking and selection: A new sequential bayesian procedure for use with common random numbers. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 29(1):2, 2019.

R. B. Gramacy and H. K. H. Lee. Optimization under unknown constraints. In *Bayesian Statistics 9*, 2011.

Matthew Groves, Michael Pearce, and Jürgen Branke. On parallelizing multi-task bayesian optimization. In *2018 Winter Simulation Conference (WSC)*, pages 1993–2002. IEEE, 2018.

S. S. Gupta and K. J. Miescke. Bayesian look ahead one-stage sampling allocations for selecting the best population. *Journal of Statistical Planning and Inference*, 54 (2):229–244, 1996.

William W Hager. Updating the inverse of a matrix. *SIAM review*, 31(2):221–239, 1989.

J. Heger, J. Branke, T. Hildebrandt, and B. Scholz-Reiter. Dynamic adjustment of dispatching rule parameters in flow shops with sequence-dependent set-up times. *International Journal of Production Research*, pages 6812–6824, 2016.

Daniel Hernández-Lobato, Jose Hernandez-Lobato, Amar Shah, and Ryan Adams. Predictive entropy search for multi-objective bayesian optimization. In *International Conference on Machine Learning*, pages 1492–1501, 2016.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

J. M. Hernandez-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems*, pages 918–926. Curran Associates, Inc., 2014.

Ruimeng Hu and Mike Ludkovski. Sequential design for ranking response surfaces. *arXiv preprint arXiv:1509.00980*, 2015.

D. Huang, T. T. Allen, W. I. Notz, and N. Zeng. Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of Global Optimization*, 34(3):441–466, mar 2006a.

Deng Huang, TT Allen, WI Notz, and RA Miller. Sequential kriging optimization using multiple-fidelity evaluations. *Structural and Multidisciplinary Optimization*, 32(5):369–382, 2006b.

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

Christian Igel. No free lunch theorems: Limitations and perspectives of metaheuristics. In *Theory and principled methods for the design of metaheuristics*, pages 1–23. Springer, 2014.

Christian Igel and Marc Toussaint. Recent results on no-free-lunch theorems for optimization. *ArXiv*, cs.NE/0303032, 2003.

Hamed Jalali, Inneke Van Nieuwenhuyse, and Victor Picheny. Comparison of kriging-based algorithms for simulation optimization with heterogeneous noise. *European Journal of Operational Research*, 261(1):279–301, 2017.

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998a.

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, 1998b.

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

Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. *arXiv preprint arXiv:1605.07079*, 2016.

Joshua Knowles. Parego: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006.

Andreas Krause and Cheng S Ong. Contextual gaussian process bandit optimization. In *Advances in Neural Information Processing Systems*, pages 2447–2455, 2011.

Loo Hay Lee, Ek Peng Chew, Suyan Teng, and David Goldsman. Optimal computing budget allocation for multi-objective simulation models. In *Proceedings of the 2004 Winter Simulation Conference, 2004.*, volume 1. IEEE, 2004.

S. Morales-Enciso and J. Branke. Tracking global optima in dynamic environments with efficient global optimization. *European Journal of Operational Research*, 242: 744–755, 2015.

Iain Murray and Ryan P Adams. Slice sampling covariance hyperparameters of latent gaussian models. In *Advances in neural information processing systems*, pages 1732–1740, 2010.

Barry L Nelson and Frank J Matejcik. Using common random numbers for indifference-zone selection and multiple comparisons in simulation. *Management Science*, 41(12):1935–1945, 1995.

Mihai Oltean. Searching for a practical evidence of the no free lunch theorems. In *International Workshop on Biologically Inspired Approaches to Advanced Information Technology*, pages 472–483. Springer, 2004.

Raghu Pasupathy and Shane G Henderson. A testbed of simulation-optimization problems. In *Proceedings of the 2006 winter simulation conference*, pages 255–263. IEEE, 2006.

Michael Pearce and Jürgen Branke. Bayesian simulation optimization with input uncertainty. In *2017 Winter Simulation Conference (WSC)*, pages 2268–2278. IEEE, 2017a.

Michael Pearce and Jürgen Branke. Efficient expected improvement estimation for continuous multiple ranking and selection. In *Proceedings of the 2017 Winter Simulation Conference*, pages 171–183. IEEE Press, 2017b.

Michael Pearce and Jürgen Branke. Efficient information collection on portfolios. *Warwick Research Archive Portal*, 2017c.

Michael Pearce and Jürgen Branke. Continuous multi-task bayesian optimisation with correlation. *European Journal of Operational Research*, 270(3):1074–1085, 2018.

Michael Pearce, Matthias Poloczek, and Branke Jürgen. Bayesian simulation optimization with common random numbers. In *2019 Winter Simulation Conference*, page to appear, 2019.

V. Picheny, D. Ginsbourger, Y. Richet, and G. Caplin. Quantile-based optimization of noisy computer experiments with tunable precision. *Technometrics*, 55(1):2–13, 2013a.

V. Picheny, T. Wagner, and D. Ginsbourger. A benchmark of kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization*, 48 (3):607–626, 2013b.

Victor Picheny. A stepwise uncertainty reduction approach to constrained global optimization. In *Artificial Intelligence and Statistics*, pages 787–795, 2014.

Victor Picheny. Multiobjective optimization using gaussian process emulators via stepwise uncertainty reduction. *Statistics and Computing*, 25(6):1265–1280, 2015.

Matthias Poloczek, Jialei Wang, and Peter I Frazier. Multi-information source optimization. *arXiv preprint arXiv:1603.00389*, 2016a.

Matthias Poloczek, Jialei Wang, and Peter I Frazier. Warm starting Bayesian optimization. In *Winter Simulation Conference*, pages 770–781. IEEE, 2016b.

Matthias Poloczek, Jialei Wang, and Peter Frazier. Multi-information source optimization. In *Advances in Neural Information Processing Systems*, pages 4289–4299, 2017.

W. H. Press, S. A. Teukolsky, S. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*, volume 2. Cambridge University Press, Cambridge, 1996.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2004. ISBN 026218253X.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

Peter L Salemi, Eunhye Song, Barry L Nelson, and Jeremy Staum. Gaussian markov random fields for discrete optimization via simulation: Framework and algorithms. *Operations Research*, 67(1):250–266, 2019.

Diariétou Sambakhé, Lauriane Rouan, Jean-Noël Bacro, and Eric Gozé. Conditional optimization of a noisy function using a kriging metamodel. *Journal of Global Optimization*, 73(3):615–636, 2019.

Robert G Sargent. Verification and validation of simulation models. *Journal of simulation*, 7(1):12–24, 2013.

E Schulz, M Speekenbrink, JM Hernández-Lobato, Z Ghahramani, and SJ Gershman. Quantifying mismatch in bayesian optimization. In *Nips workshop on bayesian optimization: Black-box optimization and beyond*, 2016.

W. Scott, P. Frazier, and W. Powell. The correlated knowledge gradient for simulation optimization of continuous parameters using gaussian process regression. *SIAM Journal on Optimization*, 21(3):996–1026, jul 2011a.

Warren Scott, Peter Frazier, and Warren Powell. The correlated knowledge gradient for simulation optimization of continuous parameters using gaussian process regression. *SIAM Journal on Optimization*, 21(3):996–1026, 2011b.

Haihui Shen, L Jeff Hong, and Xiaowei Zhang. Ranking and selection with covariates for personalized decision making. *arXiv preprint arXiv:1710.02642*, 2017.

K. A. Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys*, 41(1):1–25, dec 2008.

Kate Smith-Miles, Davaatseren Baatar, Brendan Wreford, and Rhyd Lewis. Towards objective measures of algorithm performance across instance space. *Computers and Operations Research*, 45:12–24, 2014. doi: 10.1016/j.cor.2013.11.015. URL `http://dx.doi.org/10.1016/j.cor.2013.11.015`.

Kate A Smith-Miles, Ross JW James, John W Giffin, and Yiqing Tu. A knowledge discovery approach to understanding relationships between scheduling problem structure and heuristic performance. In *International Conference on Learning and Intelligent Optimization*, pages 89–103. Springer, 2009.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning*, pages 1015–1022, 2010.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Shinya Suzuki, Shion Takeno, Tomoyuki Tamura, Kazuki Shitara, and Masayuki Karasuyama. Multi-objective bayesian optimization using pareto-frontier entropy. *arXiv preprint arXiv:1906.00127*, 2019.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012, 2013.

Saul Toscano-Palmerin and Peter I Frazier. Bayesian optimization with expensive integrands. *arXiv preprint arXiv:1803.08661*, 2018.

J. Villemonteix, E. Vazquez, and E. Walter. An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, 44(4):509–534, 2009a.

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, 2009b.

Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

David H. Wolpert. Ubiquity symposium: Evolutionary computation and the processes of life: What the no free lunch theorems really mean: How to improve search algorithms. *Ubiquity*, 2013(December):2:1–2:15, December 2013. ISSN 1530-2180. doi: 10.1145/2555235.2555237. URL `http://doi.acm.org/10.1145/2555235.2555237`.

David H Wolpert, William G Macready, et al. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

Jian Wu, Matthias Poloczek, Andrew G Wilson, and Peter Frazier. Bayesian optimization with gradients. In *Advances in Neural Information Processing Systems*, pages 5267–5278, 2017.

Jing Xie, Peter I Frazier, and Stephen E Chick. Bayesian optimization via simulation with pairwise sampling and correlated prior beliefs. *Operations Research*, 64(2): 542–559, 2016.

Jie Xu, Barry L Nelson, and JEFF Hong. Industrial strength compass: A comprehensive algorithm and software for optimization via simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 20(1):3, 2010.

Y. Xu, L. Trippa, P. Müller, and Y. Ji. Subgroup-based adaptive (SUBA) designs for multi-arm biomarker trials. *Statistics in Biosciences*, 8(1):159–180, 2014.

Xiaowei Zhang, Haihui Shen, L Jeff Hong, and Liang Ding. Knowledge gradient for selection with covariates: Consistency and computation. *arXiv preprint arXiv:1906.05098*, 2019.

Li Zhou. A survey on contextual multi-armed bandits. *CoRR*, abs/1508.03326, 2015. URL `http://arxiv.org/abs/1508.03326`.

Marcela Zuluaga, Guillaume Sergent, Andreas Krause, and Markus Püschel. Active learning for multi-objective optimization. In *International Conference on Machine Learning*, pages 462–470, 2013.