

# Accelerating Bayesian Optimization Structure Search with Transfer Learning

Nuutti Akilles Sten

## School of Science

Thesis submitted for examination for the degree of Master of  
Science in Technology.

Espoo 27.11.2020

## Supervisor

Prof. Patrick Rinke

## Advisors

D.Phil. Milica Todorović  
Aalto University  
Department of Applied Physics

D.Sc. Ulpu Remes  
University of Helsinki  
Department of Mathematics and  
Statistics

Copyright © 2020 Nuutti Akilles Sten

---

**Author** Nuutti Akilles Sten

---

**Title** Accelerating Bayesian Optimization Structure Search with Transfer Learning

---

**Degree programme** Life Science Technologies

---

**Major** Complex Systems**Code of major** SCI3060

---

**Supervisor** Prof. Patrick Rinke

---

**Advisors** D.Phil. Milica Todorović

Aalto University

Department of Applied Physics, D.Sc. Ulpu Remes

University of Helsinki

Department of Mathematics and Statistics

---

**Date** 27.11.2020**Number of pages** 76+6**Language** English

---

**Abstract**

In this thesis I studied the use of transfer learning for reducing the computational cost of Bayesian optimization structure search (BOSS). BOSS combines Gaussian process surrogate models and first principle simulations to model the potential energy surface of atomistic structures. The aim is to find the global minimum of the surface at highest possible accuracy while minimizing consumed resources. BOSS can provide accurate results with limited number of simulations. However, significant number of simulations are still required for studying many-atom structures. I studied whether transfer learning can reduce the number of expensive, high fidelity simulations in BOSS without compromising the accuracy of the global minimum estimate. In transfer learning, output from a machine learning problem is used to initialize another one. The idea in this work was to take low fidelity simulation data and use it to initialize BOSS on high fidelity search. I studied a particular transfer learning solution for Gaussian processes implemented in BOSS in my earlier work. In this thesis I refined and evaluated the application of the method in BOSS with a series of computational tests. I found that the method can significantly reduce the computational costs of finding the potential energy minimum of a high fidelity task.

---

**Keywords** Gaussian process regression, transfer learning, Bayesian optimization structure search, linear model of coregionalization

---

---

**Tekijä** Nuutti Akilles Sten

---

**Työn nimi** Bayesiläisen optimoinnin nopeuttaminen siirto-oppimisella aineen rakenteen laskennallisessa tutkimuksessa

---

**Koulutusohjelma** Life Science Technologies

---

**Pääaine** Complex Systems**Pääaineen koodi** SCI3060

---

**Työn valvoja** Prof. Patrick Rinke

---

**Työn ohjaajat** FT Milica Todorović, TKT Ulpu Remes

---

**Päivämäärä** 27.11.2020**Sivumäärä** 76+6**Kieli** Englanti

---

**Tiivistelmä**

Diplomityössäni tutkin, voidaanko siirto-oppimisella vähentää laskentaresurssien tarvetta aineen rakenteen laskennallisessa tutkimuksessa, kun tavoitteena on määrittää alhaisimman potentiaalienergian omaava rakenne mahdollisimman tarkkoja simulaatioita käyttäen. Bayesiläiseen optimointiin perustuvassa menetelmässä rakennemuuttujien vaikutusta tutkittavan aineen potentiaalienergiaan mallinnetaan simulaatioiden perusteella muodostettavalla surrogaattimallilla. Menetelmä on osoittautunut hyvin tehokkaaksi. Jokainen mallinnettu rakennemuuttuja lisää uuden ulottuvuuden hakuavaruuteen, mikä tekee simulaatioista raskaampia, kasvattaa tarvittavien simulaatioiden määrää, ja tekee mallin sovittamisesta haasteellisempää. Tarkimpien simulaatiomenetelmien käyttö moniatomisten rakenteiden tutkimisessa vaatii edelleen liikaa laskentaresursseja, mikä asettaa rajoitteita mahdollisille tutkimusasetelmille. Työssäni tutkin siirto-oppimista ratkaisuna tähän ongelmaan. Siirto-oppimisessa koneoppimiskokeen tuloksia hyödynnetään toisen kokeen alustamisessa. Tässä työssä ajatuksena oli käyttää matalan tarkkuuden simulaatioita bayesiläisen optimoinnin alustamiseen korkean tarkkuuden rakennehaussa. Tutkin tarkemmin tiettyä Gaussin prosesseihin soveltuvaa siirto-oppimismenetelmää. Tutkimukseni osoittaa, että siirto-oppimisella voidaan merkittävästi vähentää tarvittavien laskentaresurssien määrää aineen rakenteen laskennallisessa tutkimuksessa.

---

**Avainsanat** Gaussin prosessi, siirto-oppiminen, bayesiläinen optimointi, rakennetutkimus

## Foreword

I want to thank professors Patrick Rinke and Jukka Corander for the thesis project. I am grateful for my instructors Ulpu Remes and Milica Todorović on their feedback and support throughout the thesis work. In addition, I wish to acknowledge CSC – IT Center for Science, Finland, University of Helsinki and Code Refinery for support and training in HPC and scientific computing.

Lyhyt, mutta ytimekäs - näillä sanoilla voisi kuvata tähän diplomityöhön huipentunutta taivaltani Aalto-yliopistossa. Tämä ei olisi onnistunut ilman laajaa tukiverkkoa, perhettä, ystäviä, opiskelijayhteisöä tai teekkariutta. Kiitos Hannalle, joka on käskenyt minut nukkumaan ja syömään diplomityötä tehdessäni. Erityisesti haluan kiittää Tiinaa, joka sai minut uskomaan yhteisön voimaan. Lisäksi haluan mainita rakkaan kiltani Inkubion, tylsän ja asiallisen Polyteekkarimuseon, ihanan ja ylpeän Otangon, tynnyrissä kasvaneen WhisKYn, sekä TeekkariKiviYhdistyksen amestiteineen. Kiitos kaikille niille, jotka ovat olleet rakentamassa näitä ja muita yhteisöjä projekteineen.

Nuutti Sten

Otaniemi, 27.11.2020

# Contents

<b>Abstract</b>	<b>3</b>
<b>Abstract (in Finnish)</b>	<b>4</b>
<b>Foreword</b>	<b>5</b>
<b>Abbreviations and Notation</b>	<b>8</b>
<b>1 Introduction</b>	<b>9</b>
1.1 Research Objectives . . . . .	11
1.2 Structure of the Thesis . . . . .	11
<b>2 Computational Structure Search</b>	<b>13</b>
2.1 Simulating Electronic Structure . . . . .	13
2.2 Potential Energy Surface . . . . .	14
2.3 Bayesian Optimization Structure Search . . . . .	14
<b>3 Gaussian Processes and Multi-task Modelling</b>	<b>16</b>
3.1 Random Function Modelling and Coregionalization . . . . .	16
3.2 Gaussian Process Regression . . . . .	18
3.3 Linear Model of Coregionalization . . . . .	20
3.4 Intrinsic Coregionalization Model . . . . .	21
3.5 Valid Coregionalization Kernels . . . . .	21
3.6 Combining Basic Kernels with LMC . . . . .	21
3.7 Previous Work . . . . .	22
<b>4 Applying Linear Coregionalization to BOSS</b>	<b>24</b>
4.1 Basic Kernel as a Special Case of ICM . . . . .	24
4.2 ICM and Selection of Hyperparameters for a Two Task Problem . . . . .	24
4.3 Selection of Priors . . . . .	26
<b>5 Computational Approach</b>	<b>32</b>
5.1 HPC Environment . . . . .	32
5.2 Simulators . . . . .	32
5.3 Evaluation Methods . . . . .	32
5.3.1 Convergence Speed . . . . .	33
5.3.2 Detecting Outliers . . . . .	34
5.3.3 Loss Functions . . . . .	34
5.3.4 Comparing Distributions with Nonparametric Tests . . . . .	35
5.4 Alanine Molecule as a Test Subject . . . . .	36
5.5 Experiment Design . . . . .	37

<b>6</b>	<b>Experiments 1: Basic Analysis</b>	<b>42</b>
6.1	Results of Basic Analysis . . . . .	43
6.1.1	Sobol experiments . . . . .	43
6.1.2	Baselines . . . . .	46
6.2	Summary of Basic Analysis . . . . .	48
<b>7</b>	<b>Experiments 2: Model Refinement</b>	<b>51</b>
7.1	Results of Model Refinement . . . . .	51
7.2	Summary of Model Refinement . . . . .	54
<b>8</b>	<b>Experiments 3: Evaluation</b>	<b>57</b>
8.1	Results of Evaluation . . . . .	58
8.1.1	Initialization Strategies for TL . . . . .	58
8.1.2	TL Experiments . . . . .	61
8.1.3	Experiments on Alanine 2D . . . . .	61
8.1.4	Experiments on Alanine 4D . . . . .	62
8.1.5	Loss Functions . . . . .	67
8.2	Summary of Evaluation . . . . .	67
<b>9</b>	<b>Conclusions</b>	<b>70</b>
9.1	Research Summary . . . . .	70
9.2	Practical Implications on Using TL with BOSS . . . . .	71
9.3	Limitations . . . . .	71
9.4	Suggestions and Future Work . . . . .	72
9.5	Reproducibility . . . . .	73
<b>A</b>	<b>Table of Loss Function Values</b>	<b>77</b>

# Abbreviations and Notation

## Abbreviations

AMBER	Assisted Model Building with Energy Refinement simulation package
BO	Bayesian optimization
BOSS	Bayesian optimization structure search
CPU	Central processing unit
DFT	Density functional theory
FF	Force field
FHI-aims	Fritz-haber <i>ab initio</i> molecular simulation package
GP	Gaussian process
GPR	Gaussian process regression
HF	High fidelity
ICM	Intrinsic coregionalization model
LF	Low fidelity
LMC	Linear model of coregionalization
TL	Transfer learning
VHF	Very high fidelity

## Notation

$\mathbf{B}_q$ :	coregionalization kernel
$b_{i,i'}^q$ :	elements of $\mathbf{B}^q$ , cross covariance between tasks $i$ and $i'$ , or autocovariance if $i = i'$
$D$	search space
$F_i$	random variable, output value space for task $i$
$f_i$	regionalized random variable
$F_i(\mathbf{x})$	random function values in vector space
$f_i(\mathbf{x})$	value of a random function
$g_i(\mathbf{x})$	probabilistic surrogate model of $f_i(\mathbf{x})$ (probability distribution)
$i$	task, output dimension, source of observations (simulator)
$\mathbf{K}$	basic kernel matrix, a matrix of joined covariance functions $k(\mathbf{x}, \mathbf{x}')$
$k(x, x')$	covariance function for scalar space
$k(\mathbf{x}, \mathbf{x}')$	covariance function for vector space
$L$	lengthscale, a hyperparameter of covariance functions encoding smoothness
$N$	number of observations
$p$	period, a hyperparameter of covariance functions for periodic dimensions
$Q$	number of basic kernels used in coregionalization
$R$	rank of a matrix
$T$	number of tasks
$\mathbf{W}$	hyperparameter of $\mathbf{B}_q$
$x_d$	scalar value of dimension $d$ in search space $D$
$\mathbf{x}$ or $\mathbf{x}'$	point of coordinates in search space $D$
$\mathbf{x}_j$	$j^{\text{th}}$ acquisition location
$\kappa$	kappa, independence hyperparameter of $\mathbf{B}_q$
$\sigma^2$	variance, scaling hyperparameter of covariance functions

# 1 Introduction

Materials science attempts to find new materials for medicine, electronics, construction and other industries. However, in the study of novel complex materials, trial-and-error lab experiments have reached the limits of their efficacy, both in terms of efficiency and cost effectiveness, thereby restricting their scalability (Curtarolo et al. 2013). In addition, some properties of complex materials are difficult to measure directly. This and the rapid development of both hardware and algorithms over the last decades have led to computational methods becoming an integral tool in material science (Schmidt et al. 2019).

Following the rapid integration of computational methods in materials research, some of the pressing issues of the field have essentially become questions of computer science and machine learning (ML). For example, the 2020 issue of the *Annual Review of Materials Research* featured 21 articles, out of which six considered topics of ML. In the same issue, Morgan and Jacobs show how the number of materials science studies featuring ML methods have grown exponentially during the last two decades (Morgan and Jacobs 2020). One topic of particular interest is ML methods for active learning of material properties that are determined by the electronic structure.

A materials's electronic structure defines most of its functional properties (Lewars 2010). A structure can be a molecule or some other material, e.g. crystal. Potential energy plays a key role in the stability of structures and thus is a common interest of structure search studies. Potential energy as a function of structural variables is called a potential energy surface (PES). The minima of a PES determine the stable states for a structure. These define the internal structure that determines functional properties. Thus, the target of a structure search is often finding the PES minima. Potential energy is hard to measure directly, but simulative methods provide results with high accuracy. Because these simulations are computationally demanding, direct mathematical study or an exhaustive search of the PES are not viable options for analysis. Instead, computational models are used.

Bayesian optimization structure search (BOSS) is a ML method for active learning of atomistic properties (Todorović et al. 2019). BOSS combines Bayesian optimization (BO) and first principle simulations for an efficient and accurate structure search. In BOSS, a Gaussian process (GP) surrogate model of the PES is built and BO methods are used to select the acquisition locations. This allows BOSS to model uncertainty and iteratively improve the predictions. Consequently, BOSS has proven a good method for reducing computational resources in structure search (Todorović et al. 2019; Fang et al. 2020). In their study, Fang et al. 2020 showed that BOSS only required 10% of the computational resources on cysteine PES minimum search compared to state-of-the-art methods. BOSS has also proven effective in the study of complex materials. Egger et al. 2020, and Järvi, Rinke, and Todorović 2020 used BOSS to study stable organic/inorganic heterostructures. In addition Järvi, Alldritt, et al. 2020 studied structure of a molecule-surface adsorbate. These are tasks where

lab experiments or conventional structure search are ineffective. However, even with BOSS the study of structures with high fidelity simulations is expensive, and methods for reducing the resource requirements are needed.

Different methods are available for electronic structure simulations with different levels of accuracy and cost (Lewars 2010). Fast low fidelity simulations use classical mechanics, but do not perfectly describe physics at an atomistic scale. By contrast, resource intensive higher fidelity simulations, also known as *first principle* simulations, are approximations of the Schrödinger equations. Ideally, for an accurate structure search, very high fidelity first principle simulations would be used. In practice, this is not possible even with BOSS because of the poor scalability of the simulations on structures with many atoms. Therefore, the complexity of the structure limits the accuracy of the simulation method. However, there is some correlation between PES of different fidelities. This means that there could be a way to replace some of the high fidelity simulations with lower fidelity data to guide the active learning process. Then, the PES minimum of a high fidelity simulation could be found with reduced computational resources.

A possible solution for the issue is transfer learning (TL). It is a constrained setup of multi-task learning, a class of ML methods that combine multiple different sources of data that can not be directly mixed together. It can be used in active learning to either supplement for missing data, or to initialize models. Many examples of transfer learning in materials science are found from the past few years, and it appears that there is a growing interest in the topic. For example X. Zheng, P. Zheng, and Zhang 2018 reduced the amount of training data required on prediction on stable full-Heusler compounds using convolutional neural networks. Yamada et al. 2019 demonstrated how TL can be utilized in mixed materials property prediction. According to them TL can even bridge disciplines of materials science in a new way. They name transfer learning an indispensable tool for the success of ML focused materials research, enabling wider utilization of existing results in novel research. Lee and Asahi 2020 utilized TL with neural networks by supplementing for missing data in crystal structure modelling. Their TL method outperformed other regression methods with small data problems.

Common themes in earlier research include speeding up neural network training with pre-existing datasets before generating new, or improving models by supplementing for missing data. In addition, Kale and Liu 2013 showed that TL can be used for accelerating active learning by initializing a classifier with data from related but unidentical source.

In this thesis I study a TL solution for initializing higher fidelity structure search in BOSS with lower fidelity data. More specifically I evaluate a particular TL method called linear model of coregionalization (LMC) and perform computational tests with a constrained version of it, the intrinsic coregionalization model (ICM). I

selected this particular method, because of the wide applicability and scalability of it. In the coregionalization methods, it is assumed that surrogate models with different but correlated output data and shared search space can be treated as linear transformations of the same latent process. The application of the method for BOSS was first introduced and demonstrated in my earlier seminar work, where I implemented the ICM method into BOSS and found that it has potential for reducing computational cost (Sten 2020a). This thesis is a direct continuation of that work.

## 1.1 Research Objectives

Where the earlier work focused on the implementation and proof-of-concept, in this thesis I continue the work by improving the application of ICM for BOSS, evaluating the performance with computational tests and providing implications on practical use. My research questions are:

1. Can computational cost of BOSS be decreased with TL through ICM?
2. If so, how large reduction can be expected and on which conditions?
3. How ICM hyperparameters should be treated when applied to BOSS?
4. How to select the secondary, lower fidelity data to be reused for TL?
  - If lower fidelity simulation data from an active learning scheme is readily available, should it be used for transfer learning instead of generating new data for TL?
  - Else, how this secondary data should be generated?

I address the first two questions by comparing current performance of BOSS to variety of TL optimization setups. These involve six conditions: two optimization problems and three simulator pairs. To study the conditions, I do a separate experiment to compare the simulators and evaluate correlation between them. To answer the third question I first study how hyperparameters for ICM in BOSS should be treated in theory. I propose prior heuristics and do computational tests on them to see which performs the best. This hyperparameter setup I then use when evaluating performance of TL. For the fourth question I evaluate four different methods for selecting the secondary data.

## 1.2 Structure of the Thesis

The thesis consists nine sections. First is the general introduction, followed by background information of computational structure search and BOSS. Then, the mathematical background of Gaussian processes regression with one and multiple tasks is introduced. The fourth section covers the mathematical aspects of applying ICM to BOSS. It is followed by the computational approach section, where the computational tests are explained. The experimental part consists of three stages: basic analysis, model refinement and evaluation. Experiments and results for each

stage are presented in their own sections. Finally, the last section covers conclusions of the thesis work.

## 2 Computational Structure Search

In this section I explain simulation methods used in this thesis, the concept of a PES, and structure search with BOSS. This background information is important for understanding the problem of modelling high dimensional data, how BOSS works, and why we would like to replace high-fidelity data with lower fidelity simulations.

### 2.1 Simulating Electronic Structure

Computational models are widely used to study structures, properties, and processes in materials science (Lewars 2010). Computational electronic structure search focuses on the electronic structure of matter. Compared to laboratory research, simulation studies are more accessible, cheaper, faster and safer to conduct. However, computational research can not completely replace experiments, but can serve as a prior for them. There are many tools for computational materials science. In this thesis I use with force field (FF) FF and density functional theory (DFT) simulations. FF can be used for very large molecules, because other methods would require exhaustive resources. DFT has great resource-accuracy ratio, but is much more expensive than FF. Here I briefly introduce FF, ab initio and DFT methods. Through mathematical inspection of the methods is irrelevant for the thesis work and is thus bypassed.

Molecular mechanics, also known as force field (FF) or fully empirical methods are analytical potential energy functions used to model atomistic structures and molecules based on classical mechanics (Cornell et al. 1996; Lewars 2010). These simulations are fast and allow modelling even millions of atoms at a time. The downside is compromise of accuracy and less information compared to ab initio. Classical mechanics does not reveal electronic properties, but allows estimating potential energy and IR spectra. Assisted Model Building with Energy Refinement (AMBER) are force fields designed for simulating potential energy of organic molecules and structures.

Ab initio calculations are completely based on quantum theory, where matter is treated as wave functions instead of particles (Parr and Weitao 1989; Lewars 2010). The Schrödinger equations accurately describes any problem in electronic structure of matter, including the time. Solving the Schrödinger equation gives energy and wave function for a structure. The wave function can be used to determine for example spectra, electron distribution and reactivity that depends mainly on the latter. However, the equation can not be solved exactly for systems with more than one electron (Lewars 2010). Instead, approximations are used. The level of accuracy can be tuned with selection of approximation method.

Density functional theory (DFT) is a commonly used first principle approximation method (Parr and Weitao 1989). It is a a simplification of quantum mechanics. In DFT, the complex wave function and the Schrödinger equation are replaced by

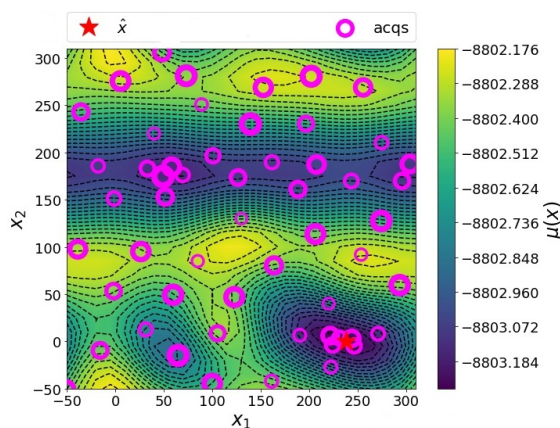


Figure 1: 2D slice of FHI-aims alanine PES as modelled by BOSS. Circles indicate acquisition locations and red star is the global minimum. The curvature is inferred from the observations.

electron density and a simpler calculation scheme (Parr and Weitao 1989; Lewars 2010). There are different methods for DFT, with different levels of accuracy and cost. Exchange-correlation functional is a key approximation in DFT. Generally the higher the accuracy of the approximation, the more accurate and slower it is. DFT can be applied to fairly large organic molecules like nucleic acids.

## 2.2 Potential Energy Surface

Potential energy describes the energy stored in the bonds of a structure under stress and rotation. Potential energy can only be evaluated relative to another state. Potential energy surface (PES) describes the relationship between the potential energy of a structure and its geometric conformation in the phase space of structural variables (Lewars 2010). PES minima are stable states of a structure. Other stationary points may be used for example to determine transition states in a reaction.

The problem with PES is, that each structural variable modelled adds another dimension to it. Consequently, the number of observations required for any model grow exponentially as the size of the structure increases. If some of these observations could be replaced with computationally cheaper data, significant resource savings could be observed. Effective visualization of the PES is also limited to 2D slices. Figure 1 illustrates a 2D slice of alanine molecule PES as modelled by BOSS.

## 2.3 Bayesian Optimization Structure Search

BOSS is a modelling tool for computational materials science designed for efficient search of property landscape minimas (Todorović et al. 2019; *BOSS project* n.d.; *BOSS python module* n.d.). BOSS builds a surrogate model of a structural property of interest using GP regression, iteratively refines the model through sequential

**Result:** Find surrogate model minima with limited number of observations;  
 Initial observations;  
**while**  $iteration < N$  **do**  
     Fit a surrogate model to observations;  
     Select new acquisition locations by minimizing an acquisition function;  
     Run simulation in the selected acquisition location;  
     Update observations;  
**end**

**Algorithm 1:** BOSS algorithm

sampling and BO. The BOSS algorithm is described in Algorithm 1. BO is used for active learning with balanced exploration and exploitation of the search space. Exploration means sampling from where there is least information from, usually as far from the current observations as possible. Exploitation means sampling from the most promising place considering the optimization target. The selection of the acquisition locations are controlled by an acquisition function. BOSS has currently two acquisition function alternatives. The default method is exploratory lower confidence bound acquisition (eLCB) (Brochu, Cora, and Freitas 2010) and the alternative is expected improvement (EI) (Zhan and Xing 2020). The methods can be extended to active multi-task learning of multiple objectives, but can be used as-is in sequential TL, as in this thesis. The topic of multi-task acquisition functions was outside the scope of this thesis.

### 3 Gaussian Processes and Multi-task Modelling

In this section I explain the background of random function modelling and coregionalization-based transfer learning to link the ICM method to GP scheme. Then, I explain GP regression, and explain how coregionalization works with GPs. This part contains the key mathematical concepts and equations for understanding the thesis work. In addition, I introduce previous work on the topic.

#### 3.1 Random Function Modelling and Coregionalization

In this thesis I adopt the idea and notation of a random function model common in geostatistics to structural search, especially following Journel and Huijbregts 1978 and Goovaerts 1997. The methods used have origins in geostatistics and have been used to model ore deposits for decades. However, the same ideas can be utilized in a new context of structure search.

Although the concepts and theory have evolved over time, I found that the original sources had a practical viewpoint on the topic and they provided a useful introduction into transfer learning through linear model of coregionalization. I encourage reader without extensive background in mathematics with further interest in random function modelling and coregionalization outside the scope of this thesis to read Journel's book (Journel and Huijbregts 1978) before getting into the the GP scheme, thoroughly explained by Rasmussen and Williams 2006, or the GP coregionalization by Bonilla, Chai, and Williams 2008. A more practical view on GP modelling can be found for example in Murphy 2013.

Let us have a random variable  $F$  that is defined as the set of possible values for a simulation model of potential energy of an atomistic structure in a search space  $D$ . In structure search  $D$  is typically spanned by dihedral angles of a molecule or interatomistic distances of a structure. All the possible values for the observed simulation outcome  $f(\mathbf{x})$  in a specific coordinates  $\mathbf{x}$  are modelled as a random function, that is a set of random variables in the search space  $\{F(\mathbf{x}) \forall \mathbf{x} \in D\}$ . The objective is to build a reliable estimate of the structural distribution of possible outputs throughout  $D$ . The objective function does not need to be truly random in nature. However, the true form is either unknown or complex, or the structural variability is very irregular. Direct mathematical study is therefore bypassed. Instead a surrogate model is built based on sampling. Each observation comes with a nonzero computational cost. Therefore the possible set of observations  $\{f(\mathbf{x}_j), j = 1, \dots, N\}$  can not be exhaustive. Either an exhaustive amount of observations, or extended knowledge of the random function would be needed to apply a parametric model. Instead, probabilistic regression is used. In a probabilistic model each prediction is then associated with known uncertainty.

Let us assume the RF autocorrelated, i.e. smooth so, that if  $\mathbf{x}$  is close to  $\mathbf{x}'$ , then  $f(\mathbf{x})$  is likely to be close to  $f(\mathbf{x}')$ . The probability distribution  $g(\mathbf{x}) = P(f(\mathbf{x}^*) = y)$

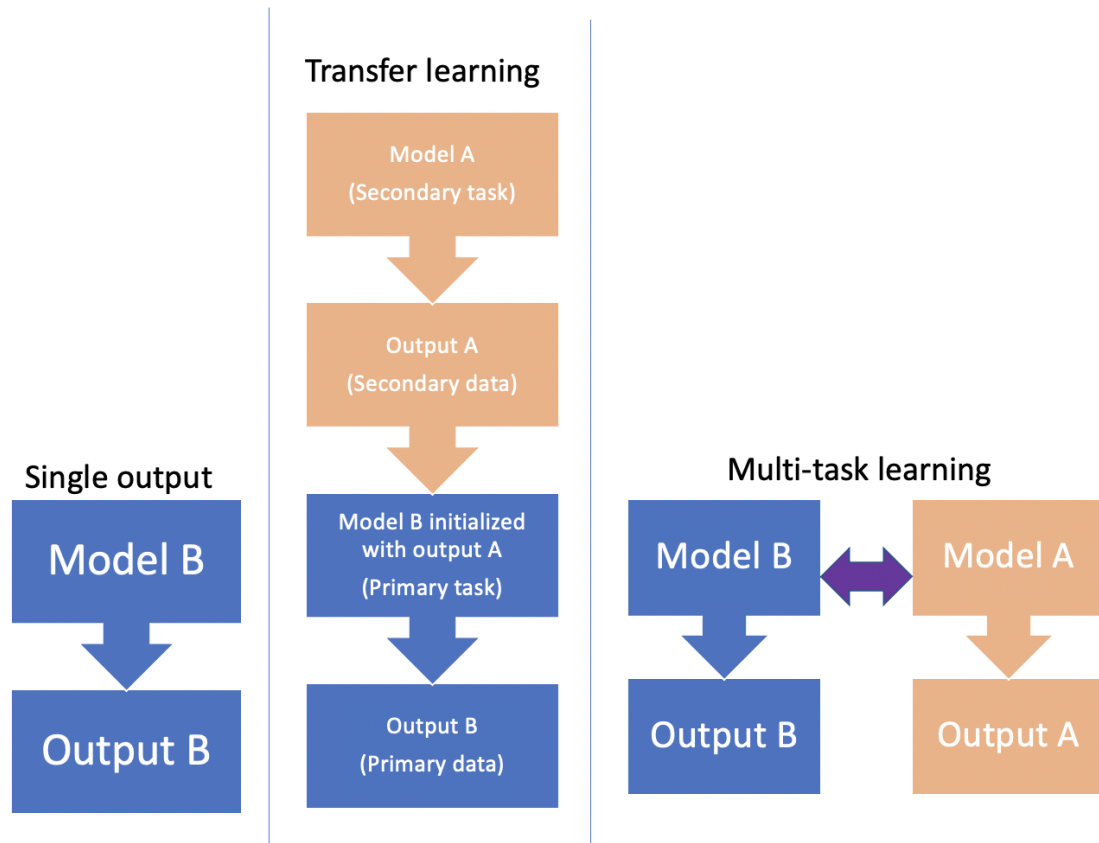


Figure 2: Single output modelling, transfer learning and multi-task learning. In TL, the tasks are performed in a sequential order. The arrows describe information flow.

for unobserved locations  $\mathbf{x}^*$  can then be estimated on any resolution with a finite number of observations. If we assume the distribution to be Gaussian, we have a multivariate Gaussian distribution or a Gaussian process (GP) over the set of observations.

An optimization task in structure search is to find the global minimum of the search space, i.e. the location where the distribution has lowest mean value. If there are two simulators modelling the same structure, they should have high correlation. Even though appealing, the observations are not directly interchangeable between the two tasks. We need a model that can transfer information between multiple tasks, i.e. a multi-task model.

The potential benefit of multi-task learning compared to modelling each task independently depends on the relationship between the tasks and the selected method's capability of transferring mutual information between them. This transfer of information can be utilized to replace some of the data from one task with the other. If there is a sequential order for taking data from different sources, we can also call this transfer learning. The task that provides supplementary data is called secondary

task, and the task that it is used to initialize is called the primary task. Figure 2 visualizes the difference between single output, TL and multi-task modelling.

Let us have  $T$  auto-correlated tasks  $F_i(\mathbf{x})$ , where  $i : \{1, T\}$  is the task index, that share search space  $\mathbf{x} \in D$ , i.e. they are coregionalized, and have realizations  $f_i(\mathbf{x})$ . By characterizing the correlation between the outputs  $f_i(\mathbf{x})$ , the probability distribution of a random function can be inferred utilizing observations from others (Journal and Huijbregts 1978). This means, that the primary task can be initialized with secondary data from the secondary task. In linear model of coregionalization (LMC), this is done by assuming a linear relationship between the tasks.

### 3.2 Gaussian Process Regression

Let us have a set of observations  $\{f(\mathbf{x}_j), \forall \mathbf{x}_j \in D\}$  from an autocorrelated random function  $F(\mathbf{x})$ . The aim is to predict the distribution  $g(\mathbf{x}^*)$  for the function value  $f(\mathbf{x})$  in unobserved locations  $\mathbf{x}^* \in D$ . First, we need to define covariance function to describe the autocorrelation. A covariance function describes dependence of inputs, independently from the outputs (Rasmussen and Williams 2006). Covariance functions are based on the assumption of similarity of close observations. This means, that if the difference between two inputs is small, then the difference between corresponding outputs is small

$$\mathbf{x} - \mathbf{x}' \approx 0 \implies f(\mathbf{x}) - f(\mathbf{x}') \approx 0. \quad (1)$$

Two simple examples of covariance functions include radial basis function (RBF) also known as *Gaussian, squared exponential* or *exponential quadratic* covariance function, and a periodic adaptation from it, standard periodic (STDP) covariance function (ibid.). The RBF covariance function is defined as

$$k_{\text{RBF}}(x, x') = \sigma^2 e^{-\frac{(x-x')^2}{2L^2}} \quad (2)$$

where  $\sigma^2$  is variance that acts as a scaling factor and  $L$  is lengthscale that determines 'smoothness' of the output. The larger the lengthscale is, the smoother the function is. Now, if the function is believed to be periodic with regard to some input dimension, a periodic covariance can be used.

The STDP covariance function is

$$k_{\text{STDP}}(x, x') = \sigma^2 e^{-\frac{2\sin^2(\pi|x-x'|)}{pL^2}} \quad (3)$$

where  $L$  and  $\sigma^2$  are defined as above, and  $p$  is period.

The complexity of possible covariance functions is unlimited. Covariance functions can also be joined together for increased complexity or multiple dimensions, for example by multiplication

$$k(\mathbf{x}, \mathbf{x}') = \prod_{d=1}^D k(x_d, x'_d). \quad (4)$$

A product of covariance functions is a valid covariance function (Rasmussen and Williams 2006).

A basic kernel matrix  $\mathbf{K} \in \mathbf{R}^{N \times N}$  is a matrix of valid covariance functions for set  $\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{pmatrix}$  of inputs  $\mathbf{x} \in \mathbf{R}^D$

$$\mathbf{K} = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \dots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \dots & k(\mathbf{x}_N, \mathbf{x}_N) \end{pmatrix} \quad (5)$$

where  $N$  is the total number of inputs.

The remaining question is, how can this kernel function be utilized for regression. Let us model the function  $f(\mathbf{x})$  as a GP

$$f(\mathbf{x}) \sim \text{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \quad (6)$$

that is defined by the mean  $m(\mathbf{x}) = \mathbf{E}[f(\mathbf{x})]$  and the covariance functions  $k(\mathbf{x}, \mathbf{x}') = \mathbf{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$  (Rasmussen and Williams 2006; Murphy 2013). The mean function is commonly assumed zero everywhere for simplicity of notation, and the kernel let to describe the process alone.

The output distribution conditional to known inputs  $p(\mathbf{f}(\mathbf{X})|\mathbf{X})$  is a multivariate normal distribution. The prior predictive distribution for  $j$  observations is

$$p(f(\mathbf{X})|\mathbf{X}) = N \left( \begin{pmatrix} m(\mathbf{x}_1) \\ \vdots \\ m(\mathbf{x}_j) \end{pmatrix}, \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \dots & k(\mathbf{x}_1, \mathbf{x}_j) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_j, \mathbf{x}_1) & \dots & k(\mathbf{x}_j, \mathbf{x}_j) \end{pmatrix} \right) = N(\mathbf{m}, \mathbf{K}). \quad (7)$$

Now let us consider prior predictive for observation  $j + 1$  based on  $j$  previous observations. The unobserved location  $j + 1$  is marked with an asterisk  $*$  to simplify

notation. The prior predictive distribution becomes

$$p(f(\mathbf{x}_*), f(\mathbf{X})|\mathbf{x}_*, \mathbf{X}) \sim N \left( \begin{array}{c} m(\mathbf{x}_1) \\ \vdots \\ m(\mathbf{x}_j) \\ m(\mathbf{x}_*) \end{array} \middle| \begin{array}{c} \left( \begin{array}{ccc} k(\mathbf{x}_1, \mathbf{x}_1) & \dots & k(\mathbf{x}_1, \mathbf{x}_j) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_j, \mathbf{x}_1) & \dots & k(\mathbf{x}_j, \mathbf{x}_j) \\ k(\mathbf{x}_*, \mathbf{x}_1) & \dots & k(\mathbf{x}_*, \mathbf{x}_j) \end{array} \right) \left( \begin{array}{c} k(\mathbf{x}_1, \mathbf{x}_*) \\ \vdots \\ k(\mathbf{x}_j, \mathbf{x}_*) \\ k(\mathbf{x}_*, \mathbf{x}_*) \end{array} \right) \end{array} \right) \quad (8)$$

$$= N \left( \begin{array}{c} \mathbf{m} \\ m(\mathbf{x}_*) \end{array} \middle| \begin{array}{cc} \mathbf{K} & \mathbf{K}_*^\top \\ \mathbf{K}_* & \mathbf{K}_{**} \end{array} \right). \quad (9)$$

The posterior predictive distribution for observation  $j + 1$  is a multivariate normal distribution

$$g(x_*) = p(f(x_*)|\mathbf{x}_*, \mathbf{X}, f(\mathbf{X})) = N(\hat{m}, \hat{v}). \quad (10)$$

For which the mean and variance parameters can be solved as

$$\hat{m} = \mathbf{K}_*^\top \mathbf{K}^{-1} (f(\mathbf{X}) - \mathbf{m}) + m(\mathbf{x}_*) \quad (11)$$

$$\hat{v} = \mathbf{K}_*^\top \mathbf{K}^{-1} \mathbf{K}_* + \mathbf{K}_{**}. \quad (12)$$

### 3.3 Linear Model of Coregionalization

In LMC, a linear relationship is assumed between  $T$  coregionalized random functions  $F_i(\mathbf{x}), i = [1, 2, \dots, T]$  (Journal and Huijbregts 1978). Above I defined a kernel as a function that describes dependence of inputs independently from particular functions  $F_i(\mathbf{x})$ . I will refer to these kernels (5) as *basic kernels* to distinguish them from coregionalization kernels  $\mathbf{B}$  with values  $b_{i,j}^q, q = [1, 2, \dots, Q]$ . A coregionalization kernel is a positive definite real  $T \times T$  matrix, that describes linear dependence of outputs of  $T$  auto-correlated, coregionalized random functions.

An LMC kernel  $\mathbf{K}_{\text{LMC}}$  is constructed as sum of  $Q$  Kronecker products of basic and coregionalization kernel matrices

$$\mathbf{K}_{\text{LMC}} = \sum_{q=1}^Q \mathbf{B}_q \otimes \mathbf{K}_q. \quad (13)$$

A Kronecker product, also known as tensor product

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \otimes \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} a1 & a2 & b1 & b2 \\ a3 & a4 & b3 & b4 \\ c1 & c2 & d1 & d2 \\ c3 & c4 & d3 & d4 \end{pmatrix} \quad (14)$$

is a matrix outer product (Harville 1997).

### 3.4 Intrinsic Coregionalization Model

ICM is a restricted version of LMC where only one basic kernel matrix is applied. It can be seen as a special case of LMC with  $Q = 1$ . From (13) we get then the ICM kernel matrix for GPs

$$\mathbf{K}_{\text{ICM}} = \mathbf{B}_1 \otimes \mathbf{K}_1 \quad (15)$$

Practical application of ICM is limited to *heterotopic* cases where the acquisition locations  $\mathbf{x}_j$  are at least partially different (Journel and Huijbregts 1978 chapter V.A.4. *Cokriging*). In an *isotopic* case, where both tasks are sampled identically, a cancellation of transfer of information occurs. Then, coregionalization has effectively no difference over modelling a single task alone, but will be require more computational resources. This means, that the sampling locations should not be completely identical for different sources.

### 3.5 Valid Coregionalization Kernels

A valid coregionalization kernel matrix  $\mathbf{B}$  can be built from

$$\mathbf{B}_q = \mathbf{W}_q \mathbf{W}_q^\top + \kappa I \quad (16)$$

where  $\mathbf{W}$  is a  $T \times T$  real matrix of rank  $R$  that encodes correlation between tasks, and  $\kappa = [\kappa_1, \dots, \kappa_T]$  are optional variables that allow modifying independence of the tasks even when  $\mathbf{W}$  is not full rank.  $I$  is the identity matrix. Row rank (here *rank*  $R$ ) describes number of linearly independent rows in a matrix. Row and column rank of a matrix are always equal (see Harville 1997 chapter 4.4 *Rank of a Matrix*). The elements of  $\mathbf{W}$  and  $\kappa$  may be found similarly to other hyperparameters, with the requirement of positive semidefiniteness of  $\mathbf{B}$ .

The matrix  $\mathbf{W}$  is always a valid basis for  $\mathbf{B}$ , because  $\mathbf{W}\mathbf{W}^\top$  is always positive semidefinite

$$\mathbf{z}^\top \mathbf{W}\mathbf{W}^\top \mathbf{z} = (\mathbf{W}^\top \mathbf{z})^\top (\mathbf{W}^\top \mathbf{z}) = \|\mathbf{W}^\top \mathbf{z}\|_2^2 \geq 0, \forall \mathbf{z} \in \mathbb{R}^T. \quad (17)$$

If values of  $\kappa$  are positive,  $\mathbf{W}_q \mathbf{W}_q^\top + \kappa I$  is positive semidefinite.

### 3.6 Combining Basic Kernels with LMC

Each basic kernel adds to the complexity of the surrogate model. The basic kernels should be selected such, that their number is small while they still capture the relevant properties of the underlying RFs. In their book, Goovaerts lists four rules to consider when combining basic kernels with LMC (Goovaerts 1997).

First, if a basic kernel can be applied for transfer of information between tasks, it will be a descriptive kernel on all tasks alone. This means that if there is nonzero cross-covariance between two tasks for a basic kernel, corresponding autocovariances are nonzero

$$b_{i,i'}^q \neq 0 \Rightarrow b_{i,i}^q \neq 0 \text{ and } b_{i',i'}^q \neq 0. \quad (18)$$

Second, if a basic kernel does not describe a task, it can not be used to infer from that task to others, or vice versa. This means that if autocovariance of a basic kernel is zero, all cross-covariances for the basic kernel involving that task are zero

$$b_{i,i}^q = 0 \Rightarrow b_{i,i'}^q = 0, \forall i. \quad (19)$$

Third, a basic kernel does not have to be descriptive for all the tasks

$$b_{i,i'}^q \text{ may be zero, } \forall i, i', q. \quad (20)$$

Fourth, even if a basic kernel is descriptive for two tasks alone, it does not have to transfer information between them

$$b_{i,i}^q \neq 0 \text{ and } b_{i',i'}^q \neq 0 \Rightarrow b_{i,i'}^q = 0 \text{ or } b_{i,i'}^q \neq 0. \quad (21)$$

### 3.7 Previous Work

In my previous seminar work (Sten 2020a) I explored the potential of multi-task learning for BOSS. I implemented coregionalization functionality to BOSS, and conducted a proof-of-concept simulation study. The goal of the experiments was to verify successful implementation and to see if TL has any potential benefit over single output BOSS. The results showed that data from multiple simulators can be combined with coregionalization. I also found that TL may speed up the convergence of the surrogate model. Figure 3 shows how TL found the PES global minimum faster compared to single output learning. The structure was alanine 2D model with d4 and d13 as free variables. Primary task was FHI-aims and secondary data from AMBER simulation. TL appears to reach lower levels of convergence faster up to unreasonably high accuracy. The addition in speed was more than the acquisition cost of the secondary data. However, statistical testing and model refinement was left for the thesis work.

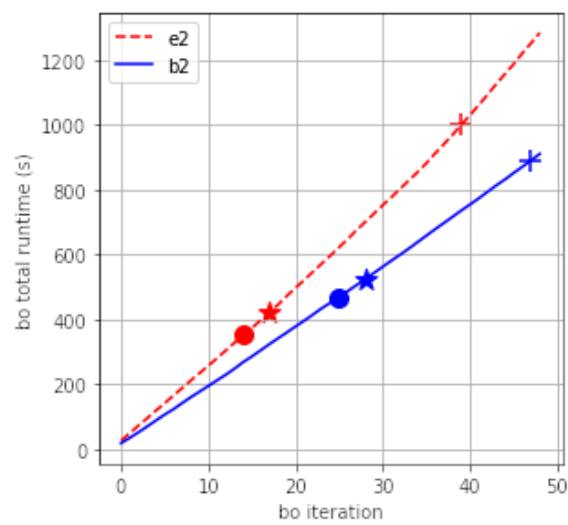


Figure 3: Results from the earlier experiments. Horizontal axis is BO iterations, describing the amount of data in the models. Vertical axis is total runtime, describing the computational resources used. Global minimum distance from true minimum at levels  $10^{-2}$ ,  $10^{-3}$  and  $10^{-4}$  eV are marked with the symbols  $\bullet$ ,  $\star$  and  $\times$ , describing the achieved accuracy of the prediction. TL method is 'e2' marked with red, and 'b2' in blue denotes single output BOSS.

## 4 Applying Linear Coregionalization to BOSS

In this section I present and discuss novel theoretical work considering the treatment of hyperparameters when applying ICM in BOSS. The purpose of this section is to explain and reason efficient practical application of TL in GP and BOSS scheme. First, I link LMC kernel to GP framework and discuss the selection of hyperparameters. Then, I introduce heuristics for selecting priors for ICM hyperparameters in BOSS. These I then test with a series of experiments explained in the following section.

### 4.1 Basic Kernel as a Special Case of ICM

The assumption is that a basic kernel such as BOSS uses is the current best behaviour. Before moving on to actual transfer learning, we should be able to replicate that performance with the ICM model. To do this, I study the relationship of 1-task ICM kernel and the basic kernel.

A basic kernel  $\mathbf{K}_2$  can be presented as a special case of ICM with just one task, and a single basic kernel  $\mathbf{K}_1$

$$\begin{aligned}\mathbf{W} &= [w] \\ \mathbf{kappa} &= [\kappa] \\ \mathbf{B} = [b] &= [w^2 + \kappa] \\ \mathbf{K}_{\text{ICM}} &= \mathbf{B}_1 \otimes \mathbf{K}_1 = b\mathbf{K}_1 = \mathbf{K}_2.\end{aligned}\tag{22}$$

Now, if the variance  $\sigma^2$  of  $\mathbf{K}_1$  is fixed to one

$$\mathbf{K}_{\text{ICM}} = [w^2 + \kappa]\mathbf{K}_1 = \mathbf{K}_2, \sigma_1^2 = 1\tag{23}$$

the variance of  $\mathbf{K}_2$  will be captured by  $w$  and  $\kappa$

$$[w^2 + \kappa] = \sigma_2^2.\tag{24}$$

and there are two variables describing variance, which may not be optimal. Because there is only one task, we can think of it as not having  $\mathbf{W}$  with full rank  $T > R = 0$  and adding  $\kappa$  for independence. A matrix with zero rank is a null matrix. Then,  $\mathbf{W} = [0]$  and  $\kappa$  is equal to  $\sigma_2^2$ . Alternatively, it can be thought of as having  $\mathbf{W}$  with full rank  $R = T = 1$ . Then,  $w^2 = \sigma_2^2$  and  $\kappa$  can be set to zero.

### 4.2 ICM and Selection of Hyperparameters for a Two Task Problem

Least possible number of hyperparameters should be used to tune a property of a model. I showed that basic kernel variance as a scaling factor can be fixed to zero in a single task case, because the same property is described by the coregionalization kernel. We can easily see that this holds for LMC models with any number of tasks and basic kernels. I also showed that the single variable  $\sigma^2$  is replaced by two

variables  $w$  and  $\kappa$ . Either one of them is to be fixed to zero to minimize the number of hyperparameters. In this section explain the hyperparameter selection when there are two tasks.

Let us have two tasks and  $\mathbf{W}$  of full rank  $R = 2$

$$\mathbf{W} = \begin{pmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{pmatrix}. \quad (25)$$

and respectively

$$\mathbf{B} = \mathbf{W}\mathbf{W}^\top = \begin{pmatrix} w_{1,1}^2 + w_{1,2}^2 & w_{1,1}w_{2,1} + w_{1,2}w_{2,2} \\ w_{1,1}w_{2,1} + w_{1,2}w_{2,2} & w_{2,1}^2 + w_{2,2}^2 \end{pmatrix}. \quad (26)$$

Covariance of outputs has no other restrictions in addition to positive definiteness. Therefore,  $\kappa$  is not needed for autocovariance, and can be fixed to zero. In total, there are four variables  $w_{1,1}, w_{1,2}, w_{2,1}$ , and  $w_{2,2}$  contributing to  $\mathbf{B}$ . Let us try to reduce the number of hyperparameters even further. Since the cross-diagonal elements of  $\mathbf{B}$  are always equal, three hyperparameters should be enough. This can be achieved by limiting the rank of  $\mathbf{W}$ . However, rank must be nonzero for  $\mathbf{W}$  to be able to encode cross covariance.

Let us have  $\mathbf{W}$  of rank  $R = 1$

$$\mathbf{W} = \begin{pmatrix} w_1 & w_2 \\ cw_1 & cw_2 \end{pmatrix}, \quad (27)$$

where  $c \in \mathbb{R}$  is a scaling factor. For the coregionalization matrix, we then compute

$$\mathbf{B} = \mathbf{W}\mathbf{W}^\top = \begin{pmatrix} w_1^2 + w_2^2 & c(w_1^2 + w_2^2) \\ c(w_1^2 + w_2^2) & c^2(w_1^2 + w_2^2) \end{pmatrix}. \quad (28)$$

This is always a valid coregionalization kernel and can be calculated from  $w = (w_1^2 + w_2^2)$  and  $c$ . Even further, either column of  $\mathbf{W}$  can be set to zero to reduce number of hyperparameters. Then  $\mathbf{B}$  is has only two hyperparameters  $w$  and  $c$

$$\mathbf{B} = \begin{pmatrix} w^2 & cw^2 \\ cw^2 & c^2w^2 \end{pmatrix}. \quad (29)$$

However, now there is an order of the covariances  $b_{i,i}$  such, that

$$\begin{aligned} b_{1,1} &= b_{1,2} = b_{2,1} = b_{2,2}, & \text{if } c = 1, & \text{ or} \\ b_{1,1} &> b_{1,2} = b_{2,1} > b_{2,2}, & \text{if } c < 1, & \text{ or} \\ b_{1,1} &< b_{1,2} = b_{2,1} < b_{2,2}, & \text{if } c > 1. & \end{aligned} \quad (30)$$

This limits coregionalization so, that the cross covariance is always between or equal to the autocovariances. If compared to the rules of basic kernel selection in 3.6 it is clear that these are strict restrictions and may not apply well in some cases, possibly even corrupting the model.

To increase independence of the tasks, vector  $\kappa$  is added to the diagonal of  $\mathbf{B}$

$$\mathbf{B} = \mathbf{W}\mathbf{W}^\top + \kappa\mathbf{I}. \quad (31)$$

Because the diagonal now has independent elements and cross covariances are always equal, the multiplier  $c$  can be set to one and one column of  $\mathbf{W}$  to zero

$$\mathbf{W} = \begin{pmatrix} w & 0 \\ w & 0 \end{pmatrix} \quad (32)$$

and the coregionalization kernel

$$\mathbf{B} = \begin{pmatrix} w^2 + \kappa_1 & w^2 \\ w^2 & w^2 + \kappa_2 \end{pmatrix} \quad (33)$$

can be constructed from three hyperparameters:  $\mathbf{W} = [w]$  and  $\kappa = [\kappa_1, \kappa_2]$ . The complexity of the model increases, and both tasks are now allowed to have autocovariance higher than the cross covariance.

When modelling two tasks with ICM, the preferable choice to minimize number of hyperparameters would be to limit rank of  $\mathbf{W}$  to one, set one column to zero, limit row-ratio to one, and use  $\kappa$  to modify independence of the tasks. Unfortunately this was not possible in this work, because hyperparameter tying was not enabled in GPY, the python module used for GPs in BOSS. Instead, a rank 1  $\mathbf{W}$  was used as

$$\mathbf{W} = \begin{pmatrix} w_1 & 0 \\ w_2 & 0 \end{pmatrix} \quad (34)$$

with and without  $\kappa$  and also compared to a full rank  $\mathbf{W}$  without  $\kappa$ .

### 4.3 Selection of Priors

Hyperparameters are the parameters of the kernel and the mean function, and they tune the smoothness and sensitivity of the model. They can be fixed to some constant value, but this does not usually give the optimal model. Because of this, they are refined so that the model best describes the training data.

In BOSS, the hyperparameters are fitted using type II maximum likelihood estimation (MLE-II), where the hyperparameters that maximize the marginal likelihood are selected (Rasmussen and Williams 2006). In addition, a prior distribution may be assigned to the hyperparameters. This is assuming that some assumptions of the possible solutions can be made. Then, a type II maximum a posteriori (MAP-II) method is used instead of MLE-II, maximising the marginal likelihood given the priors.

In fact, MLE equals to using MAP with a uniform prior over all possible solutions. Good selection of priors may speed up the hyperparameter optimization significantly and improve the model, especially if the amount of data is limited. However, a too restrictive prior can slow down or even completely corrupt the model.

In BOSS, a Gamma prior has been used for basic kernel hyperparameters

$$\text{PDF}_{Ga(\alpha,\beta)}(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}, x \in \mathbb{R}_+, \alpha > 0, \beta > 0 \quad (35)$$

where  $\alpha$  is shape parameter and  $\beta$  is rate parameter. For variance prior the prior parameters were selected with heuristic

$$\alpha = 2, \beta = \frac{2}{\hat{A}^2} \quad (36)$$

where  $\hat{A}$  is the expected amplitude of outputs  $f(\mathbf{x})$ . For lengthscales a constant heuristic was used

$$\alpha = 3.3678, \beta = 9.0204. \quad (37)$$

These have been found good priors in previous work, and suffice for baseline experiments. However, priors need to be re-evaluated for the new hyperparameters introduced in ICM. The priors used do not have to be optimal or better than the current heuristic. It suffices that they are as close to the current standard as possible given the differences in the models. There is no need for new priors for lengthscales, Hyperparameters should be selected and treated such, that the experiments best reveal effects of secondary data used and not only inheritent differences between the models, while taking full use of the transfer of information between the tasks.

I refined the ICM hyperparameter treatment in a two-step process. First, I attempted to find suitable prior candidates where I could replicate behaviour of a basic kernel with ICM kernel with a 1-task problem, without secondary data. This is an unlikely use case, and the results are unlikely to be directly useful for future experiments. However, this step was crucial for minimizing all other differences in the models except that of secondary data in the later TL experiments. In addition, I assumed that the solutions for 1 task ICM priors would be special cases of any general solutions, if they exist, and would help in finding good priors for problems with higher number of tasks. The second step was then to find good priors for 2-task problems, based on the results of the first step. Finally, all suitable prior heuristics were tested against the baselines for comparing behaviour of the hyperparameters, and performance metrics. The performance was tested in measures of convergence against BO iterations, CPU time and distraction rate. These metrics are explained in detail later in this work. All hyperparameter prior heuristics are listed in Table 1 and futher explained in this section below. The computational approach and experiment results are further explained in following sections.

Prior selection in 1-task case is simplified, because cross covariance does not need to be accounted for. It suffices that a good prior is found for autocovariance. In 4.1 I showed that in this case there are three possible ways to select the free hyperparameters for autocovariance. I found tree potential heuristics for selecting hyperparameters and setting priors. Example distributions of the prior alternatives are visualized in Figure 4.

Table 1: Prior heuristics. The first column is the number of the heuristic, and then the hyperparameter setup and priors listed for lengthscale, variance,  $\mathbf{W}$  and  $\kappa$ . For the hyperparameters the first column shows whether it is free, or the value it is fixed to, and then the prior used where  $\beta = \frac{2}{\hat{A}^2}$  when  $\hat{A}$  is the expected amplitude of the primary task. For  $\mathbf{W}$  the rank  $R$  is listed in between the two. Dash symbol ‘-’ denotes *not applicable*. The lengthscale hyperparameter is always free and has the prior  $\text{Ga}(3.3678, 9.0204)$  for all dimensions.

heur.	$\sigma^2$	$\sigma^2$ prior	$\mathbf{W}$	$R$	$\mathbf{W}$ prior	$\kappa$	$\kappa$ prior
0	free	$\text{Ga}(2, \beta)$	-	-	-	-	-
1.0	1	-	free	1	-	free	-
1.1	1	-	0	0	-	free	$\text{Ga}(2, \beta)$
1.2	1	-	free	1	$N\left(0, \frac{1}{\sqrt{\beta}}\right)$	free	$\text{Ga}(1, \beta)$
1.3	1	-	free	1	$N\left(0.9\sqrt{\frac{2}{\beta}}, \frac{1}{2\sqrt{\beta}}\right)$	0	-
2.0	1	-	free	1	-	free	-
2.1	1	-	free	1	-	free	$\text{Ga}(2, \beta)$
2.2	1	-	free	1	$N\left(0.9\sqrt{\frac{2}{\beta}}, \frac{1}{2\sqrt{\beta}}\right)$	free	-
2.3	1	-	free	1	$N\left(0, \frac{1}{\sqrt{\beta}}\right)$	free	$\text{Ga}(1, \beta)$
2.4	1	-	free	2	$N\left(\frac{0.9}{\sqrt{\beta}}, \frac{1}{2\sqrt{\beta}}\right)$	0	-

### Prior heuristics for 1-task ICM:

- 1.0** In prior heuristic 1.0 both  $\mathbf{W}$  and  $\kappa$  are free and unpriorized. I expect my heuristics to at least to better than having no priors at all. When no priors are set, there distribution of parameter values should disperse, when an experiment is repeated many times.
- 1.1** In prior heuristic 1.1  $\mathbf{W}$  is fixed to 0, and  $\kappa$  is treated as it was the variance. This can be thought of as having  $\mathbf{W}$  with rank 0. The results should be equal to using a basic kernel alone.
- 1.2** In prior heuristic 1.2 both  $\mathbf{W}$  and  $\kappa$  were used to describe autocovariance. This equals having  $\mathbf{W}$  of rank 1 and letting  $\kappa$  describe independence of tasks. A good solution could have been to use  $\text{Ga}(2, \beta)/2$  prior for  $\kappa$  and  $\sqrt{\text{Ga}(2, \beta)/2}$  for  $w$ . However, Gaussian processes in BOSS are built around GPy (GPy 2012) which limited the selection of prior functions easily available to a few basic functions. In addition, only one prior function can be assigned to a hyperparameter, all instances of a hyperparameter must have the same prior, and values of a hyperparameter instance can not be tied to another. Implementing new prior functions to GPy was outside the scope and schedule of this thesis. Therefore, I approximated the priors with  $[\kappa] \sim \text{Ga}(1, \beta)$  and  $[w] \sim N\left(0, \frac{1}{\sqrt{\beta}}\right)$ . Previously

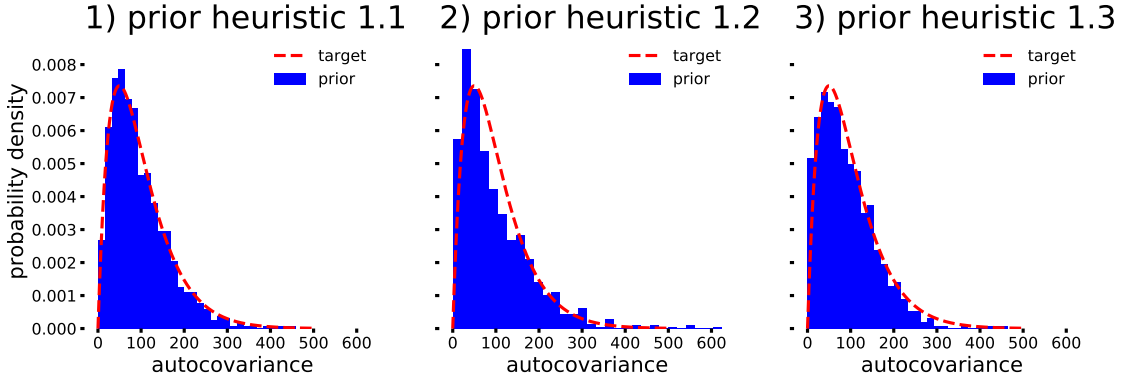


Figure 4: Example histograms from 1 task prior heuristics, excluding 1.0 as not applicable. In all plots sample size  $N = 1000$  and amplitude  $A = 10$ , from which we get  $B = \frac{2}{A^2}$ . The distributions appear very similar, except that 1.2 in plot 2 is slightly shifted to the left.

only gamma prior has been available in BOSS. In my implementation I also added support for Gaussian priors.

- 1.3** In prior heuristic 1.3  $\kappa$  is fixed to zero and  $\mathbf{W}$  is of full rank 1. Now a good solution for getting a  $Ga(2, \beta)$  prior for autocovariance would be  $w \sim \sqrt{Ga(2, \beta)}$ , but again this is not available. I approximated the prior with  $w \sim N\left(0.9\sqrt{\frac{2}{\beta}}, \frac{1}{2\sqrt{\beta}}\right)$ . Because  $w$  is taken to the second power, the possible negative values do not matter. I expect the results to be almost equal to 1.1.

In 2 task case the selection and treatment of hyperparameters gets more complicated. Having zero rank  $\mathbf{W}$  would be equal to modelling both tasks independently, but simultaneously. Thus, further evaluation and experimenting with it can be bypassed. In 3.6 I showed, that theoretically the best selection of hyperparameters would be  $\mathbf{W}$  of rank 1 with elements  $w$  tied equal to each other, and adding  $\kappa$  for modified independence. However, because parameter tying was not possible, other alternatives had to be considered. Because the coregionalization kernel must describe cross covariance, neither  $\mathbf{W}$  or  $\kappa$  can be fixed to zero, with the exception of having  $\mathbf{W}$  of full rank 2.

#### Prior heuristics for 2-task ICM:

- 2.0** Prior heuristic 2.0 is similar to 1.0, but for 2 tasks. Both  $\mathbf{W}$  or  $\kappa$  are free and have no priors set.  $\mathbf{W}$  has rank 1. This is a control case, and I expect the other heuristics to have less hyperparameter dispersion and converge faster than this.
- 2.1** Prior heuristic 2.1 was derived from 1.1. Now elements of  $\mathbf{W}$  can not be fixed to 0, because then cross covariance would be zero. Instead,  $\mathbf{W}$  is free and has no prior and elements of  $\kappa$  have  $Ga(2, \beta)$  prior.  $\mathbf{W}$  has rank 1. I expect the free unpriorized  $\mathbf{W}$  to cause dispersion to cross and autocovariances.

- 2.2** Prior heuristic 2.2 is expanded from 1.3 for two tasks.  $\mathbf{W}$  has rank 1 and  $N\left(0.9\sqrt{\frac{2}{\beta}}, \frac{1}{2\sqrt{\beta}}\right)$  prior, but  $\kappa$  is free and has no prior. As explained in 4.2,  $\kappa$  can not be fixed to 0, because that would heavily restrict the elements of the coregionalization matrix. I expect unpriorized  $\kappa$  to cause some dispersion in autocovariance.
- 2.3** Prior heuristic 2.3 follows strictly from 1.2.  $\mathbf{W}$  is of rank 1 and has prior  $N\left(0, \frac{1}{\sqrt{\beta}}\right)$ , and  $\kappa$  has prior  $Ga(1, \beta)$ . In this prior it is assumed that there is no strong cross covariance, unless the observations show evidence. Cross covariance is heavily weighted close to zero.
- 2.4** Prior heuristic 2.4 is analogous to 1 task full rank prior heuristic 1.3. Because  $\mathbf{W}$  has rank 2 (full rank),  $\kappa$  is no longer required and is fixed to zero. Elements of  $\mathbf{W}$  have prior  $N\left(\frac{0.9}{\sqrt{\beta}}, \frac{1}{2\sqrt{\beta}}\right)$ .

Example distributions of the priors introduced above are plotted in Figure 5. In plots 5.1a) and 5.2a) dispersion from unpriorized variables is visualized with a uniform distribution. This is a rough approximation, and the true dispersion may differ. The true theoretical dispersion is difficult to evaluate so I used an rough approximation instead. The true dispersion effect may be seen from the experimental results. Plot 5.1b) is a similar approximation of unpriorized cross covariance after MLE.

In addition, for each of the 2-task prior heuristics, excluding 2.0, I compared the mean, variance, skewness and kurtosis of the auto- and cross covariance distributions for amplitudes from  $10^{-3}$  to  $10^3$  in steps of powers of ten, 1000 samples at each step. There was little difference between the distributions except for heuristic 2.3, for which the cross covariance tails are heavier and the mean bifurcates after amplitudes  $A > 1$ . In addition, a dispersion effect from unfixed parameters can be assumed in heuristics 2.2 and 2.1, which however was hard to evaluate and thus left out from this comparison. Based on these findings I expect heuristic 2.4 to have be closest to heuristic 0 considering autocovariance. I also expect it to perform best of all the heuristics presented for transfer learning, because it should best model high cross covariance that can be expected when combining molecular simulations.

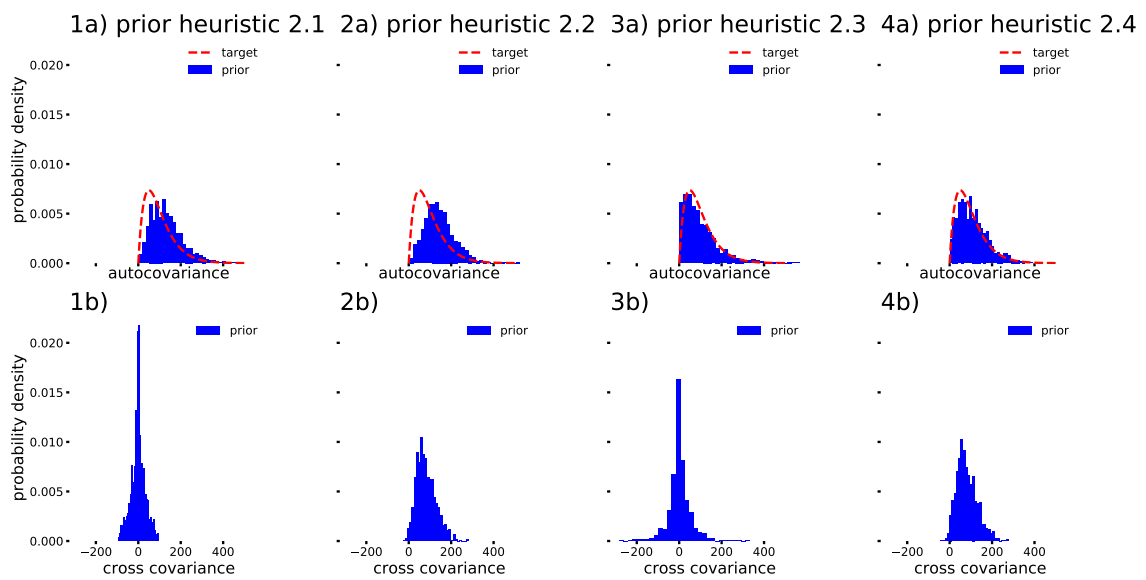


Figure 5: Example histograms from the example distributions of 2 task prior heuristics 2.1, 2.2, 2.3, 2.4. Row a) is autocovariances and row b) cross covariances. In all plots sample size  $N = 1000$  and amplitude  $A = 10$ , from which we get  $B = \frac{2}{A^2}$ . In row a) target distribution  $\text{Ga}(2, \beta)$  for autocovariance prior is plotted in red.

## 5 Computational Approach

In this section I describe the experimental part of the thesis. In the experiments I performed series of computational tests, where I evaluated the application and performance of ICM in BOSS. I begin by introducing the computing environment and the simulators used. Then describe the evaluation methods, the test subject and experiment design.

### 5.1 HPC Environment

Electronic structure simulations are usually run on high performance computing (HPC) clusters, commonly known as supercomputers. These are computers with many fast central processing units (CPU) with large fast access memory. I used the Finnish IT Center for Science (CSC) supercomputer Puhti. HPC is great for heavy scientific computing. Especially in matrix-heavy calculations the process can be speeded up by dividing the operations between multiple cores, or even multiple computer units. This allows scaling computational research into problems too large for an ordinary tabletop computer. I ran all my experiments on just a single core, because the 1-core CPU time is a good estimate of the resources used. Scaling up to multiple cores does not scale the resource demand in equal ratio.

### 5.2 Simulators

In my experiments used two simulators and three different fidelities. I call them low fidelity (LF), high fidelity (HF) and very high fidelity (VHF) simulations with the corresponding abbreviations. The fidelities are in order for fidelity and computational resource demand. I recommend a reader with further interest in the methods to see Lewars [2010](#), as well as Cornell et al. [1996](#) for AMBER and Parr and Weitao [1989](#); Jones [2015](#) for DFT.

**LF** denotes FF simulations using AMBERtools (Case et al. [2016](#)).

**HF** denotes FHI-aims simulations using PBE exchange-correlation functional with Hirshfield van der Waals correction  
(`xc pbe, vdw_correction_hirshfield`) (Blum et al. [2009](#)).

**VHF** denotes FHI-aims simulations using PBE0 exchange-correlation functional with many body dispersion  
(`xc pbe0, many_body_dispersion`).

### 5.3 Evaluation Methods

Since the objective of BOSS is to find the PES minimum with least resources, we are primarily interested in two things. Is the correct minimum found, and how fast? Because BOSS is a numerical method, we first need to determine what is a close enough objective value that can be accepted.

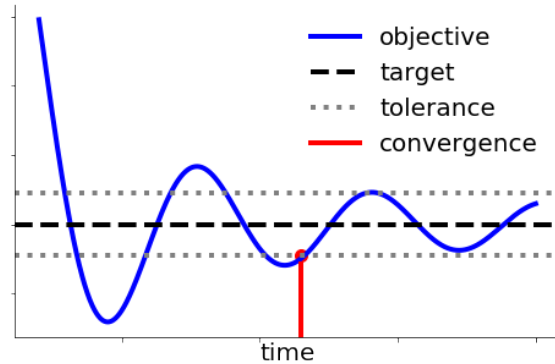


Figure 6: An objective function is converged to a convergence level at the point after which it remains within a tolerance level of a target value.

Let us define convergence. An objective function, here the PES global minimum prediction (GMP) is the value and location predicted by the GP surrogate model. GMP is converged after it remains within a tolerance level of the target value. The target value in this case is the predicted PES minimum taken from the very last iteration of a corresponding baseline experiment. It should be noted that I measure convergence as a postprocessing statistic. In practical use the target values might not be known and some other convergence criterion might have to be used. Convergence is visualized in Figure 6.

### 5.3.1 Convergence Speed

To approximate the resources used, I measured the time that it takes for the surrogate model GMP to converge. I did not account for memory requirements, because in BOSS memory is much cheaper and easier to scale compared to CPU time that is the dominant computational cost. There are two ways to measure time. First we can think the number of operations BOSS needs to perform. I approximate this with the number of BO iterations BOSS takes, because this is the smallest loop unit that we can control BOSS in. We get an abstract unit of time, BO iterations. We can think it as an abstract measure of the performance of the algorithms - how many iterations do we need to converge using TL compared to the baselines? Because in these experiments only one new data point was added per iteration, BO iterations also measure the number of data in the model.

Another measure is CPU time. It is a measure of how long time a BOSS run takes when everything is running on a single core. Even though in practical use it is recommended to run parallel on multiple cores, I used only one to measure the CPU time accurately. Even though parallel computing is faster in real time, it consumes more resources in total than calculating on single core only.

It may also be, that an experiment run does not converge at all. This is exceptionally bad performance, and I call this distraction. Distraction rate measures the proportion

of experiment runs in a setup that do not get within target distance of the GMP. This behaviour should be strongly avoided.

### 5.3.2 Detecting Outliers

Outliers are possible with all data, and their possibility should be acknowledged in research. Considering convergence speed, I had found two kinds of outliers in my earlier work. First, are the experiments that are initialized, or within first iterations hit the PES minimum by chance. These may give a false impression that the optimization was very efficient with underfitted model. However, there may be differences in 'how lucky' different active learning methods are depending on the problem, and a lucky shot may well happen in real research as well. On top of that these occasions rarely deviate far from the general trend because of the simple, fast converging test setups used in this thesis. Because of this, I did not attempt to exclude these results. However, any model can not be considered descriptive with very little data.

Another typical case is, that for some unexpected reason some iteration takes significant amount of time to converge. This may be because of the random processes involved in fitting hyperparameters, some underlying issue in the software that causes a lag, or other processes on stalling the optimization. This kind of behaviour has been observed with BOSS in previous research. In addition, these outliers typically deviate many orders of magnitude from the trend. Because I could only run a limited number of experiments, this kind of behaviour would distract any non-robust methods, and should be discarded. To automatically decide which observations to discard, I used the z-score, which measures how many standard deviations away a point is from the mean of a sample

$$Z = \frac{\text{value} - \text{mean}}{\text{standard deviation}}. \quad (38)$$

If the z-score was greater than 2.5 for an experiment run compared to identical ones, I discarded it as outlier.

Generally, the z-score should be used only when normality can be assumed. Even though I could not verify normality because of small sample sizes, I found that this was a good heuristic for detecting outliers based on visual inspection.

### 5.3.3 Loss Functions

In mathematical optimization, loss function describes the resources used for obtaining a result (Yang 2007). In my experiments, the result is a converged GMP, and the cost is the CPU time, normalized with respect to the baseline. Loss function allows me to compare results of all experiments in same unit and scale of performance. The definition of a loss function depends on the problem. Here I define two loss functions that I use in my analysis, an continuous loss function based on mean, i.e. the loss function, and an indicator loss function based on medians.

**Loss function** quantifies the performance of TL with regard the baselines

$$\text{loss} = \frac{\text{mean}(\text{experiment setup convergence time})}{\text{mean}(\text{baseline convergence time})}. \quad (39)$$

Loss can also be expressed as percentages of resource use. Instead of loss, gain is sometimes used

$$\text{gain} = 1 - \text{loss}. \quad (40)$$

To calculate loss, each BOSS run in each experiment was grouped by the number of secondary initialization points. For each group, the mean convergence time to 0.1 kcal/mol was divided by the mean convergence time of the corresponding baseline. This makes the baseline convergence time the unit of the loss function, meaning that the loss function for the baseline has value 1 or 100% of the resources. An experiment setup that on average takes 1.5 times the time to converge compared to the baseline, is assigned loss function value 1.5. All experiment setups with loss function less than 1 are declared faster than the baseline, and thus favourable.

The loss function definition above is only based on statistical representation and does not include any testing of significance. To do this, I define another loss function.

**Indicator loss function** gets the value 1 (True), if an experiment setup is statistically faster than the baseline, and 0 (False) otherwise

$$\text{indicator loss} = \begin{cases} 1, & \text{experiment setup converges faster than baseline} \\ 0, & \text{else} \end{cases} \quad (41)$$

Because my experiments do not provide enough data for making distributional assumptions, I used Mann Whitney U-test, a nonparametric test for comparing medians.

### 5.3.4 Comparing Distributions with Nonparametric Tests

Here I briefly introduce the selection of statistical tests that I use for my analysis. Because of small sample sizes and unknown distributions I used nonparametric tests. The tests are widely used in statistics.

**Mann-Whitney U-test or Wilcoxon rank sum test** is a nonparametric test for equality of distributions of i.i.d. samples that can be divided in two categories, but is not paired. The null hypothesis can be that the distributions are equal, or that one is shifted off the other.

**One-way analysis of variance ANOVA** is a technique for comparing distributions of two or more samples. Nonparametric ANOVA is based on comparison of medians. First, medians of all samples are compared using Kruskal-Wallis test. If the medians are not identical, one of the samples is removed and the test is repeated with the remaining. This is then repeated by excluding each sample at a turn. This is then repeated with recursion up to pairwise comparison of the distributions, to detect which samples differ from the others and which are identical. If a pairwise Kruskal-Wallis test is rejected, the order of the samples is determined with Wilcoxon Signed Rank test. Finally, for each sample we count the number of pairwise comparisons in which they were smaller than the other sample to obtain a rank of order.

**Kruskal-Wallis test or the one-way ANOVA on ranks** tests if two or more samples come from identical populations. The assumption is that the samples have the same shape of distribution, but no other distributional assumptions are required. The null hypothesis is that all samples come from identical populations.

**Wilcoxon Signed Rank Test** is a nonparametric test for comparing equality of paired samples. In the test it is assumed that the two samples are paired, i.e. they share the independent variables. This means that the samples are not independent, so Mann Whitney U-test can not be used. The null hypothesis can be that the sample distribution means are equal, or that one is larger than the other.

## 5.4 Alanine Molecule as a Test Subject

In the experiments I used the alanine molecule  $C_3H_7NO_2$  as an example structure. Alanine is one of the most simple amino acids. It has been widely studied and has PES with 13 known minima (Cao et al. 1995). Because of the small size, it is fast to simulate on both FF and DFT simulators. Alanine has also been used as test and experiment case in earlier BOSS development, so the ordinal behaviour of BOSS with alanine is well known. These factors make it a very good test subject for this work, too. The molecule has four independent dihedral angles that I use as structural variables. These are visualised in Figure 7. In addition, I use two variants of the molecule: one, **alanine 2D**, where the dihedrals d7 and d11 are fixed to angles 180 and 60, limiting the search space in two dimensions. This does not effect the simulation times, but makes optimization faster compared to the other one, **alanine 4D**, where all four independent independent dihedrals are free. The search space limits and periods are elaborated in Table 2.

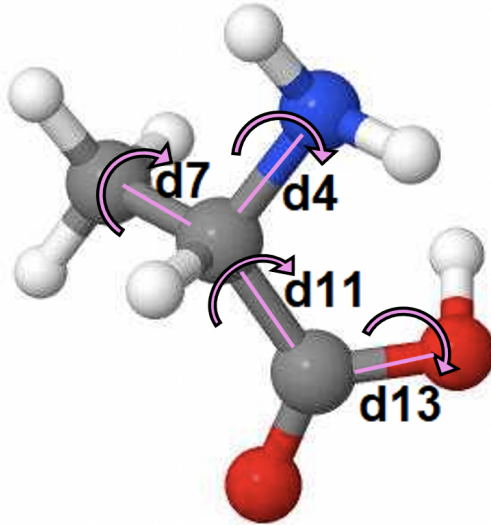


Figure 7: Dihedrals of the alanine molecule.

Table 2: Experiment structures. Experiments with their name beginning with 'a' are done on alanine 2D, and experiments that begin with 'b' are on alanine 4D.

exp.	Structure	$D$	$D$ limits	$\mathbf{x}$
axxx	Alanine	2	$-50 \leq x_d \leq 310, \forall d$	[d4 d13]
bxxx	Alanine	4	$\begin{cases} -50 \leq x_d \leq 310, d \in \{d4, d11, d13\} \\ -50 \leq x_d \leq 70, d = d7 \end{cases}$	[d4 d7 d11 d13]

## 5.5 Experiment Design

In this section I explain the experiment design. Let me begin by introducing the naming strategy I used to create identifiers for the experiments. To give unique and logical identifier for each experiment, I used a four character code to describe each. The code is explained in Figure 8. These unique names summarise most relevant aspects of the individual experiments, including the structure of interest, GP kernel used and the primary task fidelity, allowing me to also use these identifiers in visualization. In addition, this allowed me to add experiments to previous stage while preserving the logical order of the experiments.

All experiments are listed in Table 3, with the most relevant information of their setup. In the table, the experiments are listed in alphabetical order according to the name of the experiment in the first column. This also describes well the logical and chronological order of the experiments. There were three types of experiments, executed in three stages. The second column of the experiment table describes the type of the experiment, where 'Q' stands for query, 'B' for baseline, and 'C' for coregionalization. The third column denotes the stage of the experiment, where '1' denotes basic analysis, '2' is model refinement and '3' is evaluation.

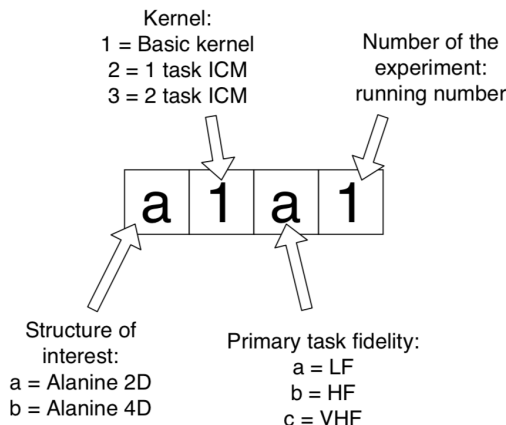


Figure 8: Naming of the experiments.

In query experiments the simulators were queried based on random or sobol locations, and no surrogate model was fitted. These experiments were used to evaluate the true correlation and covariance between simulators, and to provide secondary data for TL. Baselines were run using BOSS 0.9.17 release with no transfer learning methods enabled. They were used as a point of reference of the current best performance of BOSS, and provided secondary data for TL. Coregionalization experiments were run using a modification of BOSS 0.9.17 where ICM method was enabled. However, not all coregionalization experiments involved transfer learning. Those with only one task were used when testing some of the prior candidates.

Basic analysis stage consisted of query experiments and baselines. The purpose of the basic analysis experiments was to form a general overview of the simulators to form a basis for evaluating how the selection of simulators and the structure of interest may effect TL. In addition, they were used to evaluate current BOSS performance, and to see if the simulators have differences in how difficult it is to find the PES minimum. In the model refinement stage I tested the prior heuristics introduced in this thesis, to find out how ICM should be applied for BOSS. Then, in the evaluation stage I performed a series of TL experiments with varying setups, to see how the structure, search space dimensionality, simulator fidelity and correlation, and the amount and selection criteria for secondary data effect performance of ICM on BOSS. Based on these I also evaluated the potential benefit of TL in BOSS in different cases.

Table 3: Table of experiments. First three columns have the name, type and stage of the experiment. The fourth column denotes the hyperparameter prior heuristic used. Rest of the columns define the setup.  $N_{exp}$  is the number of BOSS runs in the experiment and 'BO iter.' denotes BO iterations per run.  $N_{tot}$  is the total number of observations used.  $task_1$  marks the fidelity of the primary task simulator and  $initpts_1$  is the number of initialization points from the primary task. The column  $inittype_1$  denotes the selection method of the primary initialization data. These are repeated for the secondary task accordingly, if there is one. The experiment where the secondary data was taken from is marked in parenthesis (xxxx) after the number of secondary initialization points. For a range of numbers the beginning, end and step size are marked as *begin - end : step*. If there is a range, the first BOSS run of the experiment has the value *begin*, the second run has *begin + step* and so on. When the range is full, the following run has again the value *begin*. A dash symbol '-' denotes *not applicable*.

exp.	Type	Stage	prior h.	$N_{exp}$	BO iter.	$N_{tot}$	$task_1$	$initpts_1$	$inittype_1$	$T_2$	$initpts_2$	$inittype_2$
a1a1	Q	1	-	1	0	100	LF	100	sobol	-	-	-
a1a2	B	1	0	30	100	102	LF	2	sobol	-	-	-
a1a3	B	1	0	30	100	102	LF	2	random	-	-	-
a1a4	Q	1	-	1	100	100	LF	100	random	-	-	-
a1b1	Q	1	-	1	0	100	HF	100	sobol	-	-	-
a1b2	B	1	0	30	100	102	HF	2	random	-	-	-

Table 3: Continuation from previous page.

exp.	Type	Stage	prior h.	$N_{exp}$	BO iter.	$N_{tot}$	task <sub>1</sub>	initpts <sub>1</sub>	inittype <sub>1</sub>	$T_2$	initpts <sub>2</sub>	inittype <sub>2</sub>
a1b3	Q	1	-	1	100	100	HF	100	random	-	-	-
a1c1	Q	1	-	1	0	100	VHF	100	sobol	-	-	-
a1c2	B	1	0	30	100	102	VHF	2	random	-	-	-
a2a1	C	2	1.0	30	100	102	LF	2	random	-	-	-
a2a2	C	2	1.1	30	100	102	LF	2	random	-	-	-
a2a3	C	2	1.2	30	100	102	LF	2	random	-	-	-
a2a4	C	2	1.3	30	100	102	LF	2	random	-	-	-
a3b1	C	2	2.0	30	50	52	HF	2	random	LF	30 (a1a3)	BO random
a3b2	C	2	2.1	30	50	52	HF	2	random	LF	30 (a1a3)	BO random
a3b3	C	2	2.2	30	50	52	HF	2	random	LF	30 (a1a3)	BO random
a3b4	C	2	2.3	30	50	52	HF	2	random	LF	30 (a1a3)	BO random
a3b5	C	2	2.4	30	50	52	HF	2	random	LF	30 (a1a3)	BO random
a3b6	C	3	2.2	100	50	57-102:5	HF	2	random	LF	5-50:5 (a1a3)	BO random
a3b7	C	3	2.2	100	50	57-102:5	HF	2	random	LF	5-50:5 (a1a3)	BO inoder
a3b8	C	3	2.2	100	50	57-102:5	HF	2	random	LF	5-50:5 (a1a1)	sobol
a3b9	C	3	2.2	100	50	57-102:5	HF	2	random	LF	5-50:5 (a1a4)	random
a3c1	C	3	2.2	100	50	57-102:5	VHF	2	random	LF	5-50:5 (a1a3)	BO random
a3c2	C	3	2.2	100	50	57-102:5	VHF	2	random	HF	5-50:5 (a1b2)	BO random
a3c3	C	3	2.2	100	50	57-102:5	VHF	2	random	LF	5-50:5 (a1a3)	BO inoder
a3c4	C	3	2.2	100	50	57-102:5	VHF	2	random	HF	5-50:5 (a1b2)	BO inoder
a3c5	C	3	2.2	100	50	57-102:5	VHF	2	random	LF	5-50:5 (a1a1)	sobol
a3c6	C	3	2.2	100	50	57-102:5	VHF	2	random	HF	5-50:5 (a1b1)	sobol
a3c7	C	3	2.2	100	50	57-102:5	VHF	2	random	LF	5-50:5 (a1a4)	random
a3c8	C	3	2.2	100	50	57-102:5	VHF	2	random	HF	5-50:5 (a1b3)	random
b1a1	Q	1	-	1	0	300	LF	300	sobol	-	-	-

Table 3: Continuation from previous page.

exp.	Type	Stage	prior h.	$N_{exp}$	BO iter.	$N_{tot}$	task <sub>1</sub>	initpts <sub>1</sub>	inittype <sub>1</sub>	$T_2$	initpts <sub>2</sub>	inittype <sub>2</sub>
b1a2	B	1	0	30	300	302	LF	2	random	-	-	-
b1b1	Q	1	-	1	0	300	HF	300	sobol	-	-	-
b1b2	B	1	0	30	300	302	HF	2	random	-	-	-
b1c1	Q	1	-	1	0	300	VHF	300	sobol	-	-	-
b1c2	B	1	0	30	300	302	VHF	2	random	-	-	-
b3b1	C	3	2.2	200	250	262-452:10	HF	2	random	LF	10-200:10 (b1a2)	BO in order
b3c1	C	3	2.2	200	250	262-452:10	VHF	2	random	LF	10-200:10 (b1a2)	BO in order
b3c2	C	3	2.2	200	250	262-452:10	VHF	2	random	HF	10-200:10 (b1b2)	BO in order
b3b2	C	3	2.2	200	250	262-452:10	HF	2	random	LF	10-200:10 (b1a2)	BO random
b3c3	C	3	2.2	200	250	262-452:10	VHF	2	random	LF	10-200:10 (b1a2)	BO random
b3c4	C	3	2.2	200	250	262-452:10	VHF	2	random	HF	10-200:10 (b1b2)	BO random

## 6 Experiments 1: Basic Analysis

The first stage of experiments was the basic analysis. The purpose of the basic analysis was to form understanding of the simulator outputs, create a point of reference on the single-output BOSS to compare TL with, and generate data to be utilized in the TL experiments. This stage does not directly answer to the research questions, but provides useful information for answering them with the rest of the experiments. I begin by introducing the basic analysis experiments, and then show and summarize the results.

Here, I did two types of experiments, queries and baseline experiments. In sobol queries I queried the simulators at locations defined by the sobol sequence. These include the experiments **a1a1**, **a1b1**, **a1c1**, **b1a1**, **b1b1** and **b1c1**. Sobol sequence is a quasi-random algorithm for sampling a hypercube (Sobol 1967). Sobol sequence covers the search space more evenly compared to uniform pseudo-random sampling. It is the default way of selecting initialization data in BOSS. No surrogate model was used for the query experiments. Because the acquisition locations are identical when using the sobol sequence, the true covariance and correlation of the PES landscapes could be evaluated. This was the main purpose of the sobol queries. Other query experiments included **a1a4** and **a1b3**, where the lower fidelity simulators were sampled exhaustively from random locations to form a pool of random initialization data to be used as secondary data in TL experiments.

Before the actual baselines I tested how much of the GP model variability is explained by the initial observations compared by comparing BOSS runs with random and sobol initialization. Then I could see if there are some differences that would make another method favourable for the experiments, and justify the initialization strategy for the baselines. Hyperparameter fitting involves random processes, which repeated over many iterations cause significant variability in the performance of the methods. The selection of initial data points causes additional variability. First iteration hyperparameters are fitted based on the first observations, and both are then used to determine the new acquisition locations. The differences then accumulate fast. I try to evaluate how big portion of the variability is explained by the hyperparameter fitting compared to the selection of the initial observations. Their effects are impossible to completely separate, because of the cumulative effect from both. This part consisted of the experiments **a1a2** with sobol and **a1a3** with random initialization.

Finally, I executed the baseline experiments. These were BOSS runs on both alanine 2D and alanine 4D, on all fidelity levels and with 30 BOSS runs in each for statistics. Their purpose was to present the best current performance of BOSS, to compare transfer learning with. In addition, if some of the baselines required more data than the others to converge, it might indicate that the PES minima is more difficult to find in that case. The output of these experiments was also used to initialize some of the TL experiments. The baselines include experiments **a1a3**, **a1b2**, **a1c2**, **b1a2**, **b1b2** and **b1c2**.

Table 4: Summary statistics of sobol experiments. Columns are experiment name, number of experiments, mean, variance, min, max and amplitude for observations.

exp	$m(f(\mathbf{x}))$	$var(f(\mathbf{x}))$	$min(f(\mathbf{x}))$	$max(f(\mathbf{x}))$	$A(f(\mathbf{x}))$
a1a1	11.523	28.717	0.269	20.527	10.129
a1b1	14.740	43.064	0.093	23.656	11.782
a1c1	14.068	41.040	0.094	22.554	11.230
b1a1	15.950	33.363	0.818	32.086	15.634
b1b1	16.852	37.775	2.289	29.953	13.832
b1c1	16.261	35.699	1.931	29.083	13.576

Table 5: Pearson’s correlation coefficient matrices based on sobol experiments.

alanine 2D	a1c1	a1b1	a1a1	alanine 4D	b1c1	b1b1	b1a1
a1c1	1.000	0.997	0.948	b1c1	1.000	0.998	0.924
a1b1	0.997	1.000	0.945	b1b1	0.998	1.000	0.915
a1a1	0.948	0.945	1.000	b1a1	0.924	0.915	1.000

## 6.1 Results of Basic Analysis

### 6.1.1 Sobol experiments

I used the sobol experiments to evaluate acquisition times, correlation and covariance of simulators and estimate expected amplitude of the simulators for hyperparameter selection. Table 4 Shows summary statistics from the sobol experiments, relocated based on the baselines so that output 0 equals the baseline minimum. In practical use this exact information of the simulator amplitudes is unlikely to be available, but I wanted to minimize the effect of expected amplitude error in the following experiments.

Figure 13 shows the mean acquisition times of the simulators. The acquisition time ratios are roughly 100:1 between VHF and LF, 2:1 between VHF and HF and 50:1 between HF and LF.

Figures 9 and 10 show trellis of potential energy scatter between sobol experiments. There appears to be surprisingly clear and high degree of linear correlation. Table 5 shows Pearson’s correlation coefficients and confirms the correlation. If linear correlation between tasks can be assumed, LMC should be able to utilize it. Table 6 shows the measured covariances between sobol experiment outputs. These serve as a sanity check for the TL experiments. LMC coregionalization matrix values should approach these values as new observations are added.

Table 6: Covariance matrices based on sobol experiments.

alanine 2D	a1c1	a1b1	a1a1	alanine 4D	b1c1	b1b1	b1a1
a1c1	41.040	42.052	34.878	b1c1	35.699	36.737	34.531
a1b1	42.052	43.064	35.890	b1b1	36.737	37.775	35.569
a1a1	34.878	35.890	28.717	b1a1	34.531	35.569	33.363

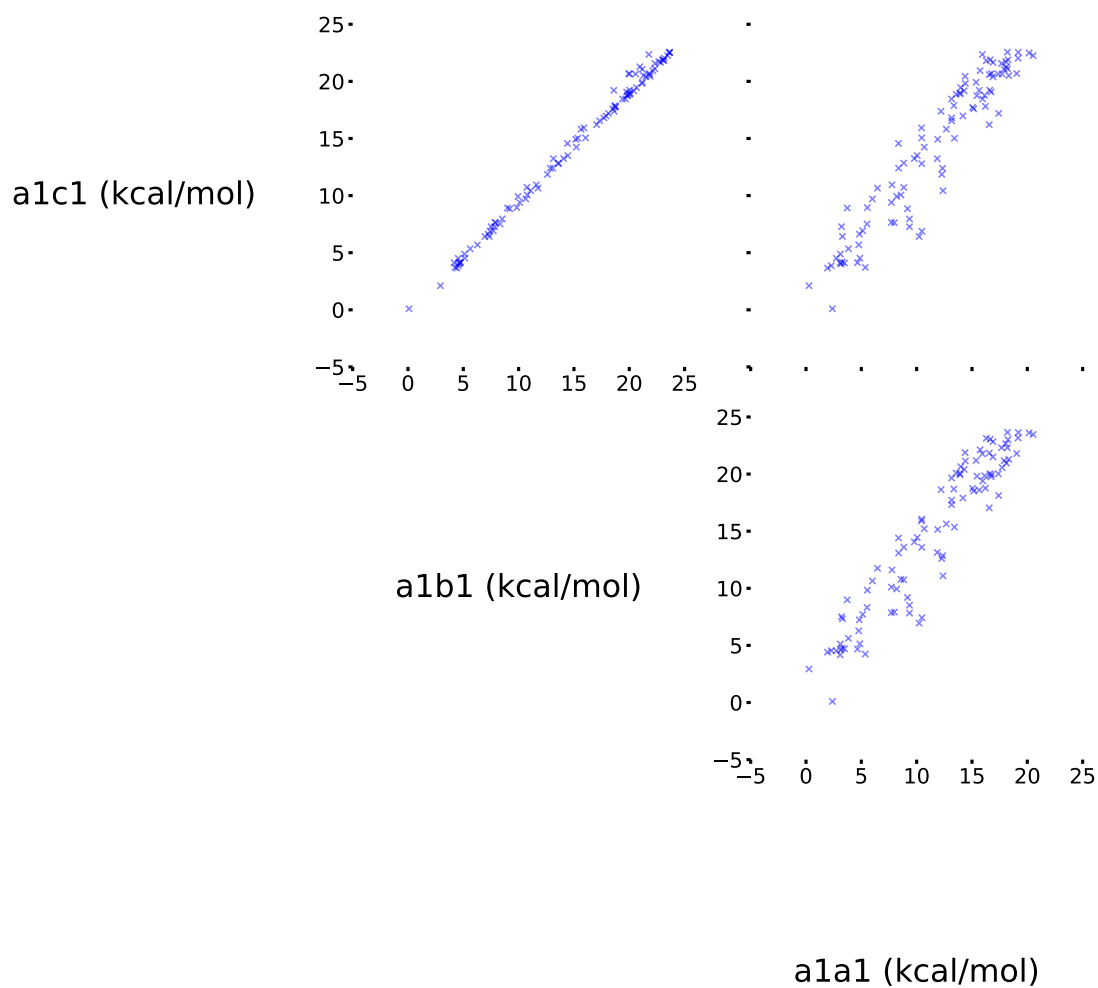


Figure 9: Scatter plots of alanine 2D potential energy sampled from identical sobol locations. The experiment names correspond to fidelities  $\mathbf{a1c1=VHF}$ ,  $\mathbf{a1b1=HF}$ ,  $\mathbf{a1a1=LF}$ .

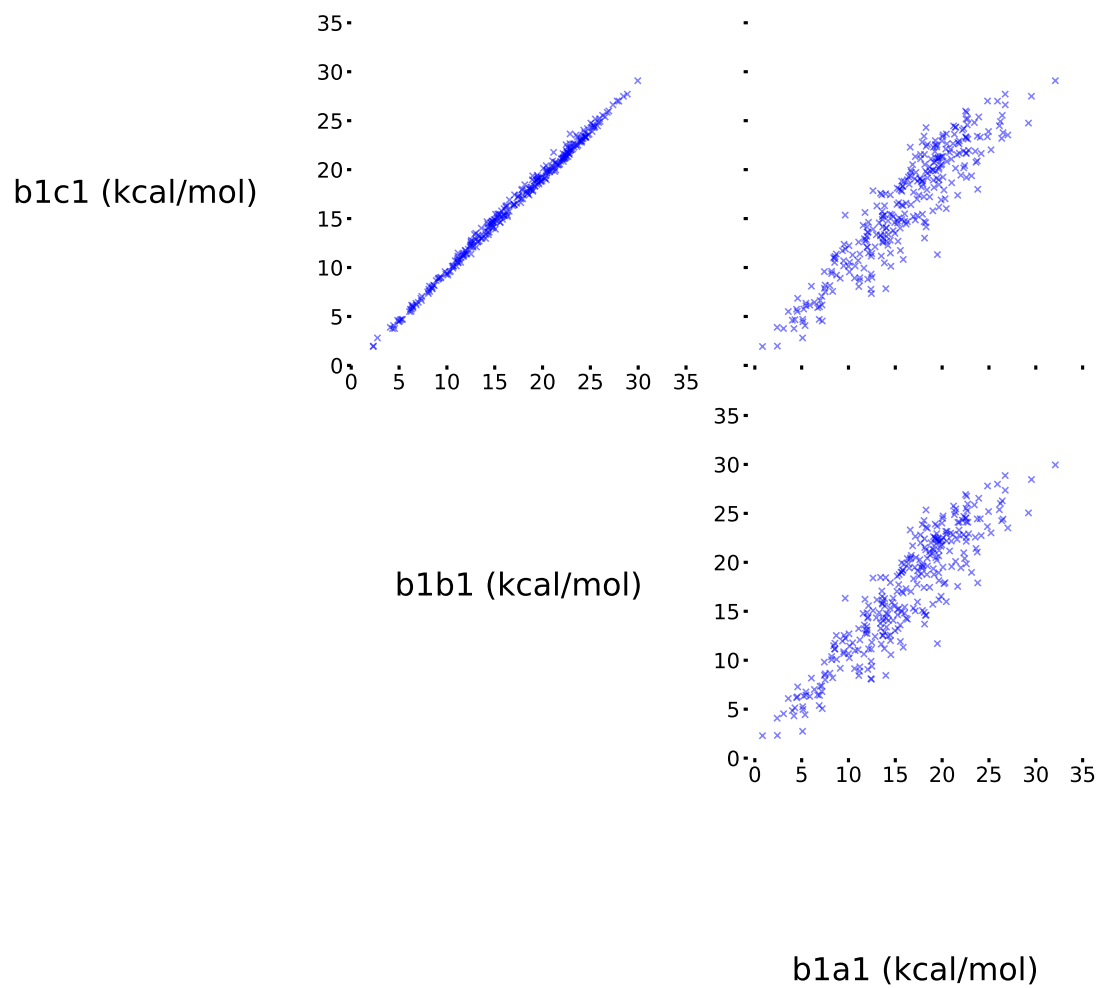


Figure 10: Scatter plots of alanine 4D potential energy sampled from identical sobol locations. The experiment names correspond to fidelities  $b1c1=VHF$ ,  $b1b1=HF$ ,  $b1a1=LF$ .

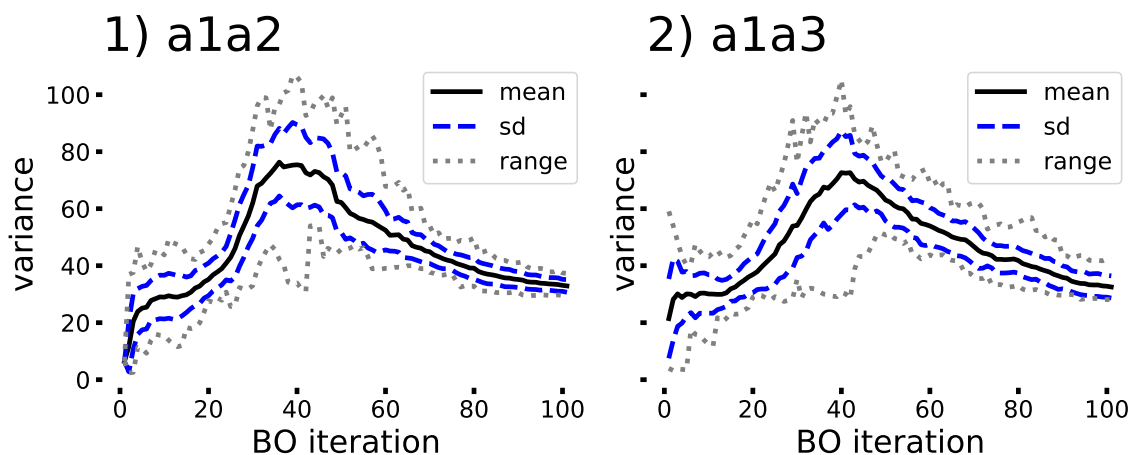


Figure 11: Basic kernel variance hyperparameter distribution over BO iterations for two initialization strategies. In plot 1) is experiment **a1a2** (sobol initialization) and in plot 2) experiment **a1a3** (random initialization).

### 6.1.2 Baselines

Prior to running the baseline experiments, I had to decide should I select the two initialization points from sobol or random locations. At least two initialization points are needed to initialize the GP model. The advantage of selecting the points from sobol were that then the initial data would be the same in all experiments sharing search space. Sobol is generally recommended for selecting the initialization points because it gives better coverage than random. However, there is a risk that these points would be trivial, i.e. especially good or bad choice for the model. Random selection would include statistical testing, but may increase model variability that could add noise to the results. Compared this by repeating BOSS of alanine 2D 30 times twice in experiments **a1a2** with sobol and **a1a3** with random initialization and by comparing the convergence speed and hyperparameter values.

Figure 11 shows the distribution of variance parameter over BO iterations. Variability of the variance is clearly smaller in the first iterations with sobol compared to random initialization, but after that there is little difference. Figure 12 shows BO iterations to convergence level and distraction rate by convergence level. Again, variability is smaller with sobol than with random. However, the mean convergence time is larger with sobol, which indicates that the choice of points from sobol happens to be poor in this case, compared to random. Distraction rate is 0 on all levels in both experiments, so they both find the same minimum. Based on this I decided to use random initialization the baselines and for initializing primary task in TL experiments.

The true minima and their locations for alanine 2D and 4D at different levels of fidelity are shown in Table 7. The potential energy value is raw simulator output converted to kcal/mol. Potential energies were relocated with regard to these values

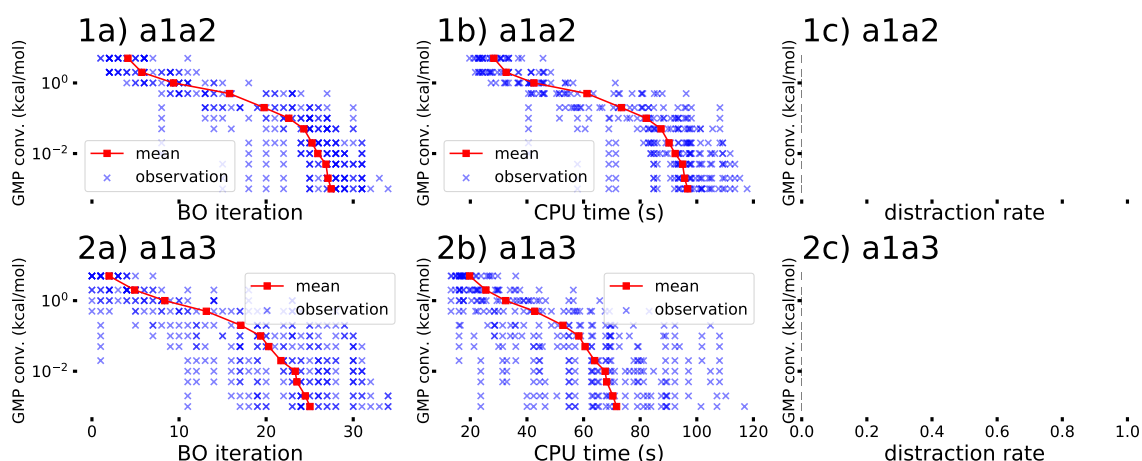


Figure 12: Convergence of different initialization strategies in BO iterations, CPU time and distraction rate. Row a) is for experiment a1a2 with sobol, and row b) is for experiment a1a3 with random initialization. The blue 'x'-glyphs denote a boss run at the current iteration, printed with 50% transparency to show overlapping points.

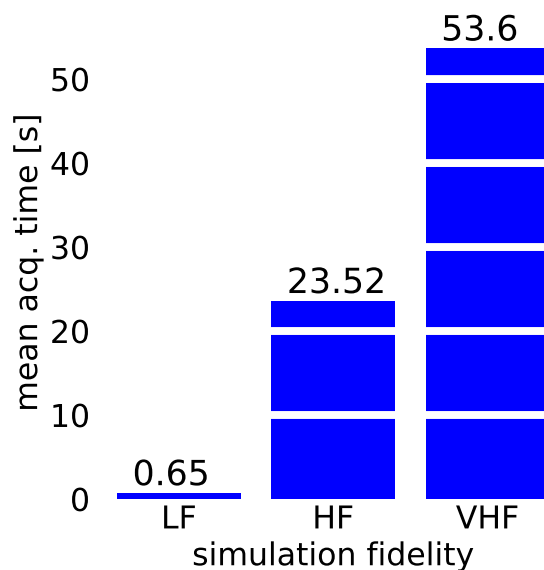


Figure 13: Mean acquisition times of the simulators.

for the analysis. LF simulation gives very different location compared to HF and VHF in both 2D and 4D simulations. This indicates that there are some differences in the PES landscapes. HF and VHF simulations have the minimum at roughly the same locations.

In addition to differences in acquisition times, the PES landscapes vary in shape. This can be seen from Figures 14 and 15 plotting the histograms of convergence times in BO iterations and CPU time.

Table 7: Alanine PES minima from different simulation fidelities and search spaces. Columns are baseline experiment, fidelity level, search space dimensions, the dihedral rotations at the PES minima, and the PES minima value as raw simulator output converted to kcal/mol. An asterisk (\*) denotes that the angle was fixed throughout the search.

exp.	fidelity	$D$	d4	d7	d11	d13	PES min (kcal/mol)
b1c2	VHF	4	232.64	55.24	63.78	-1.45	-203013.25
a1c2	VHF	2	237.42	180 *	60 *	0.75	-203017.32
b1b2	HF	4	242.03	52.01	60.40	-0.31	-203012.44
a1b2	HF	2	232.01	180 *	60 *	1.96	-203013.44
b1a2	LF	4	56.24	62.53	203.73	179.38	15.80
a1a3	LF	2	59.19	180 *	60 *	179.94	17.48

In Figure 14 we see, that there appears to be no clear difference in the BO iteration distributions, indicating that the GMP is equally difficult to find in all fidelity levels. Because of this, the CPU time distributions are scaled in accordance with the differences in the acquisition times.

Figure 15 shows, that the LF simulations are not only faster, but the GMP is found with fewer iterations. This suggests that the GMP of alanine4D LF PES landscape is easier to find than that of the higher fidelities. This further increases the differences in total computational cost for finding the GMP for these setups. The cost is not completely defined by the acquisition time.

In 2D the mean BO iterations to convergence are equal. In 4D, it appears that not only does fidelity correlate with increased acquisition time, but also the iterations required for finding the minimum. To support visual inspection and summary statistics, I compared the acquisition time distributions with Mann-Whitney U test. The results, shown in Table 8, agree with the visual inspection, with the exception that in 2D only HF and VHF simulations are likely to converge at same BO iteration speed. This indicates that the minimum is not as easy to find in all cases. The differences in CPU times to convergence are then not completely explained completely by the acquisition times, but rather also by differences in the PES landscape. No distraction was observed in the baselines.

## 6.2 Summary of Basic Analysis

From the basic analysis experiments we learned that there is a high degree of linear correlation between all simulator fidelities used. This means, that we have good basis for using linear coregionalization, that is based on linear relationship. I showed, that it is better to select the initialization points for single-output BOSS at random for better statistics compared to using identical pair of locations. Based on this, I decided to select the primary task initialization points at random in all experiments. In addition, the baseline experiments showed that simulators have

Table 8: Mann-Whitney U test results for testing the equality of the BO iteration convergence speed distributions of baseline experiments to tolerance level 0.1 kcal/mol. Columns in order are for experiment test pairs and their median iterations of convergence, the number of sampled pairs, the Wilcoxon test statistic and p-value. Experiment test pairs and the results are listed in the rows. Significance level is 0.05.

exp1	median1	exp2	median2	U statistic	p-value	critical value	equal medians
a1a3	20.0	a1b2	19.0	0.58	0.446	0.35	no
a1a3	20.0	a1c2	17.5	2.13	0.145	0.03	no
a1b2	19.0	a1c2	17.5	0.43	0.51	0.48	yes
b1a2	60.5	b1b2	135.0	38.21	0.0	0.0	no
b1a2	60.5	b1c2	142.5	44.28	0.0	0.0	no
b1b2	135.0	b1c2	142.5	3.1	0.078	0.01	no

differences in the PES landscapes. This can effect how difficult the minimum is to find, despite the structure of interest being the same.

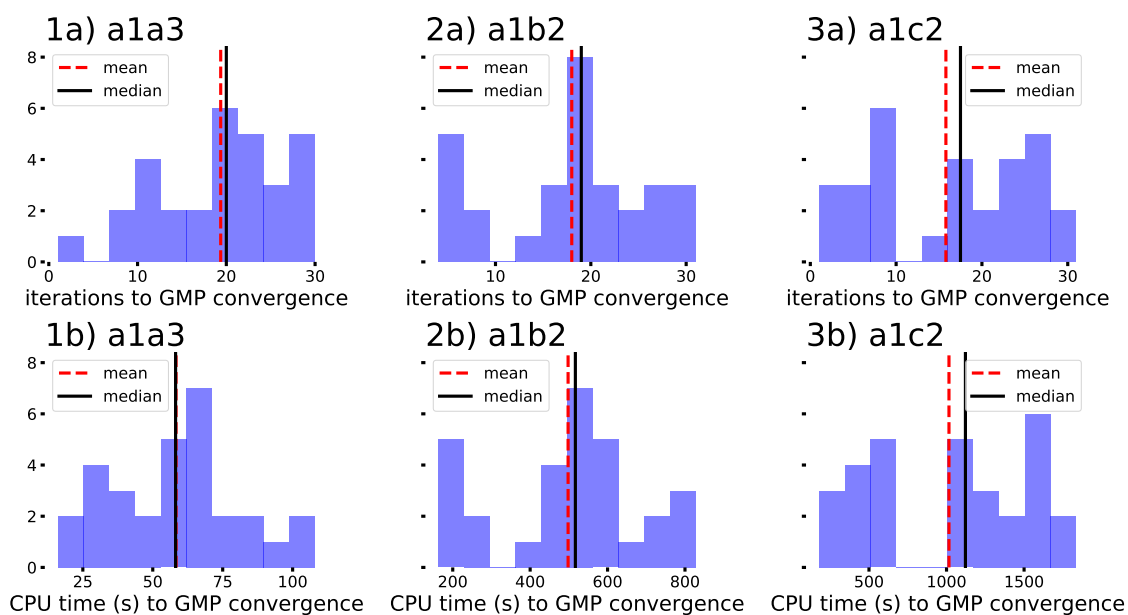


Figure 14: GMP Convergence speed histograms of 2D baselines to convergence level of 0.1 kcal/mol. Row a) is in BO iterations and row b) in 1-core CPU time. Columns 1, 2 and 3 correspond to simulator fidelities LF, HF and VHF.

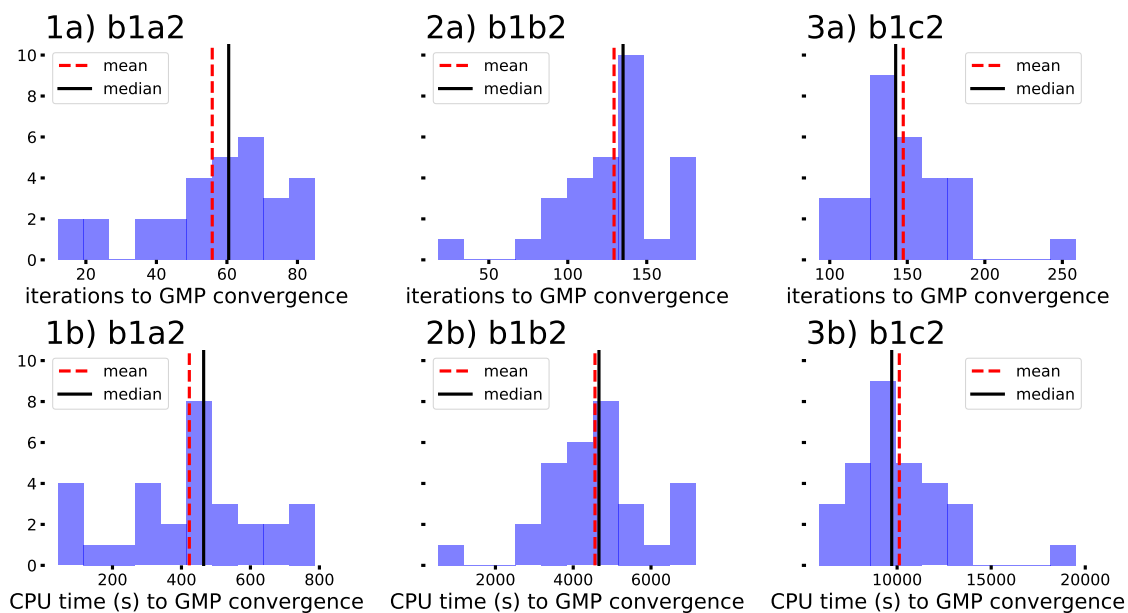


Figure 15: GMP Convergence speed histograms of 4D baselines to convergence level of 0.1 kcal/mol. Row a) is in BO iterations and row b) in 1-core CPU time. Columns 1, 2 and 3 correspond to simulator fidelities LF, HF and VHF.

## 7 Experiments 2: Model Refinement

The second stage was model refinement. These experiments answer to the research question on how ICM hyperparameters should be treated when applied to BOSS. In previous work I had not used priors or limitations for the hyperparameters in LMC. I introduced a collection of alternative prior heuristics, that I then tested in these experiments to find out, which one to use in the evaluation stage. In addition, I show that single-output BOSS performance can be replicated with the ICM kernel. The prior heuristics 1.0, 1.1, 1.2, 1.3, 2.0, 2.1, 2.2, 2.3, and 2.4 were tested in experiments **a2a1**, **a2a2**, **a2a3**, **a2a4**, **a3b1**, **a3b2**, **a3b3**, **a3b4** and **a3b5**, accordingly. These are compared to the baselines **a1a3** and **a1b2** with the default prior heuristic 0.

### 7.1 Results of Model Refinement

Figure 16 shows the distribution of baseline variance and autocovariances of 1-task ICM prior heuristic over BO iterations. In addition, the autocovariance is decomposed to  $\kappa$  and  $|w|$ . As expected, the autocovariance distribution spreads in **a2a1** compared to baseline in plot 1c) because no priors are set for the variables. Rest of the experiments are hard to distinguish from each other by looking at the autocovariance. They show a bit smaller dispersion than the baseline. From the autocovariance decomposition plots it is clear that most of the autocovariance is interpreted by **W** also when kappa has no prior. An exception is heuristic 2.1 where kappa is forced to describe variance alone. The autocovariances approach the measured values from Table 6.

Figure 17 shows convergence speed in BO iterations, CPU time and distraction rate of the 1 task ICM prior selection experiments. All experiments have distraction rate 0 for all convergence levels. This means that all experiment runs found the correct global minimum. Experiment **a2a1** with no LMC priors takes longer in both BO iterations and CPU time to converge to given level compared to the baseline and other heuristics. Setting priors may speed up the active learning process. For the experiments **a2a2**, **a2a3** and **a2a4** the expected number of BO iterations is about the same as in the baseline on all levels, but the variability of the speed appears to be smaller. The expected CPU time is larger. This is expected because ICM is not meant to be used with a single task. However, these experiments were just a sanity check before scaling. Experiment **a2a4** appears to converge fastest and with least variability compared to other ICM experiments. This indicates that logic in prior heuristic 1.3 replicates the baseline performance the best.

The next step was to evaluate hyperparameter priors when there's data from 2 simulators. This included experiments **a3b1**, **a3b2**, **a3b3**, **a3b4** and **a3b5**. To each, I randomly selected 30 observations from experiment **a1a3**. Here I assumed this data pre-existing, so no additional cost was assigned to it.

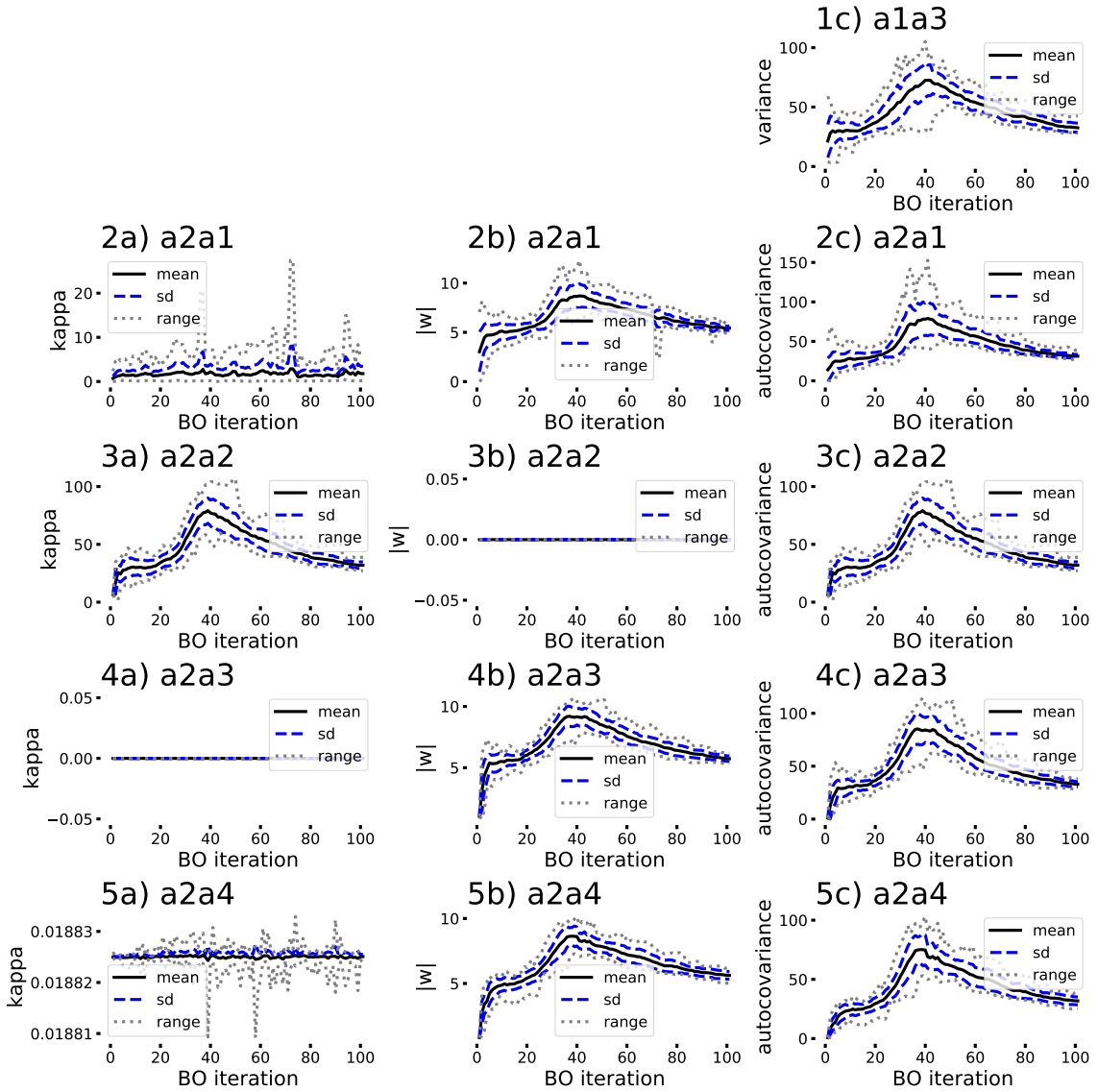


Figure 16: Distribution over BO iterations of basic kernel variance, and autocovariance and related hyperparameters in 1 task ICM. Rows from 1 to 5 are experiments **a1a3**, **a2a1**, **a2a2**, **a2a3** and **a2a4** that test the prior heuristics 0, 1.1, 1.2, 1.3 and 1.4. Columns are a)  $\kappa$  b)  $w$  and c) variance or autocovariance calculated from  $b = w^2 + \kappa$ .

Figure 18 shows the distributions of baseline variance hyperparameter and experiment auto- and cross covariance hyperparameters. Both autocovariances behave very similarly compared to cross covariance in all experiments. All values approach the measured covariances in Table 6. In experiments **a3b1** and **a3b2** the distributions disperse because there are many unpriorized hyperparameters. Experiment **a3b3** where only kappa has no prior is slightly less dispersed. The distribution is most condensed in **a3b4** where a prior is set for both hyperparameters. However, the general shape of the distribution is very different compared to the baseline and other experiments, which may be because of the different initial assumptions in the priors.

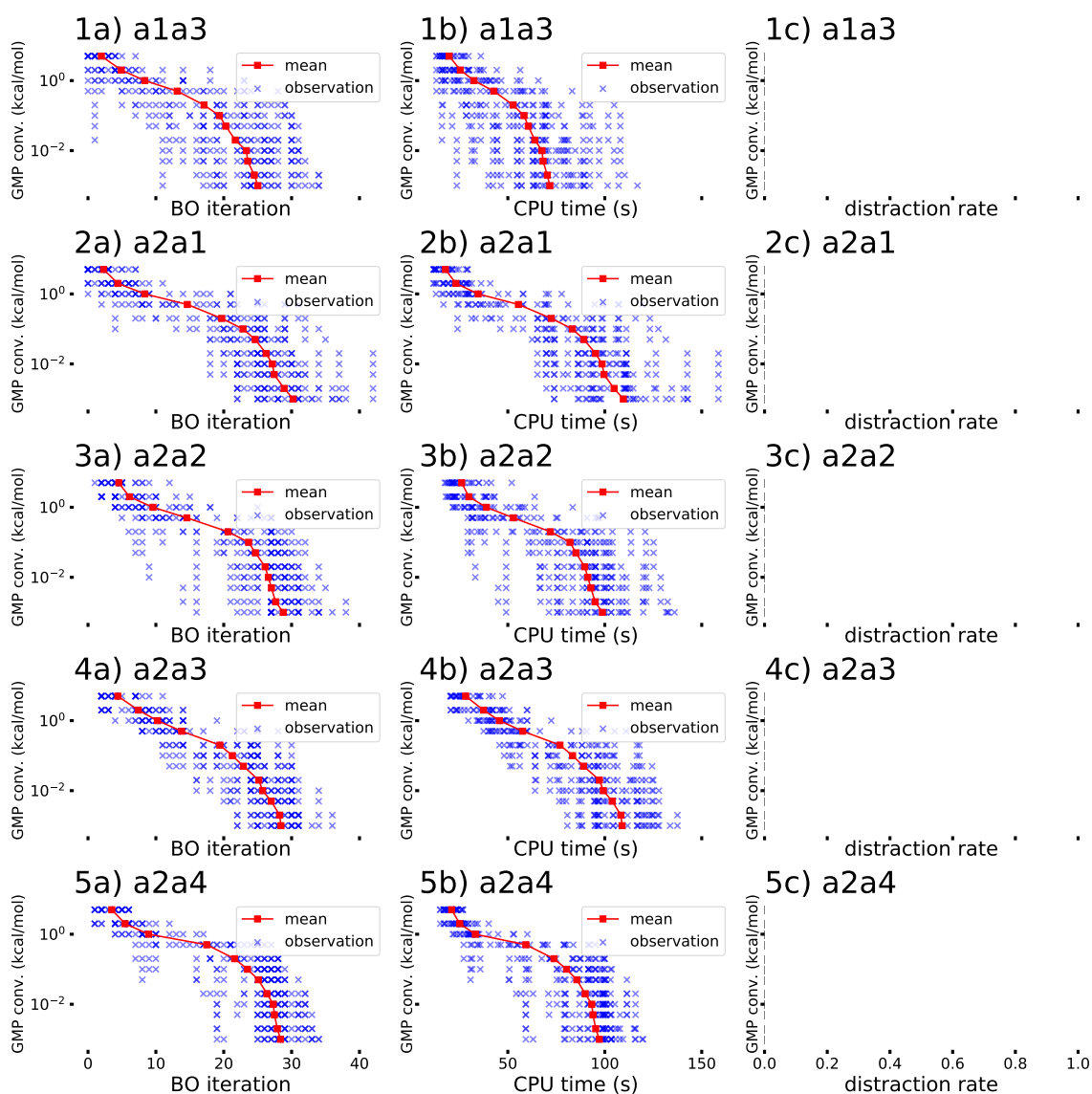


Figure 17: GMP convergence in BO iterations, CPU time, and distraction rate for 1 task prior heuristics. Rows are for experiment a1a3 (basic kernel) and for 1 task ICM experiments a2a1, a2a2, a2a3 and a2a4. Blue crosses are for individual runs and red lines are mean of the whole experiment, calculated for given convergence level.

It is also the only distribution that does not appear to approach but rather escapes the measured auto- and cross covariances. Experiment a3b5 appears to get closest to the baselines in shape, while maintaining small variability.

Figure 19 shows convergence in BO iterations, CPU time and distraction rate for the 2-task prior heuristic experiments. Mean of convergence time is smaller in all TL experiments compared to the baseline in both BO iterations and CPU time. The baseline has mean of convergence time to the lowest tolerance level at 28 iterations or

around 700 seconds, whereas the experiments take between 21 to 24 iterations or 550 to 600 seconds. The best results are from experiments **a3b3** and **a3b5**. There is not a clear difference between the two, but **a3b3** has a smoother mean curve, indicating more stable behaviour. In addition, it could further improve from parameter tying, i.e. fixing elements of  $\mathbf{B}$  to be defined by single scalar value, if it was to be implemented in the future. In experiment **a3b4** we observe nonzero distraction rate for the first time. Distraction should be strongly avoided, because it indicated incorrect results from the simulation. It also makes mean of converged experiments a bad estimate of expected convergence time, because convergence time in these experiments is large unknown. It appears that the prior heuristic 2.2 gives the best results with two tasks.

## 7.2 Summary of Model Refinement

In the model refinement experiments I tested the prior heuristics. I showed that the prior heuristic 1.3 best replicates baseline performance, and that prior heuristic 2.2 is the most promising for transfer learning with two tasks. In addition, I observed that a poor selection of priors can restrict the model too much, resulting in unacceptable distraction rates. I decided to apply the prior heuristic 2.2 in the evaluation stage.

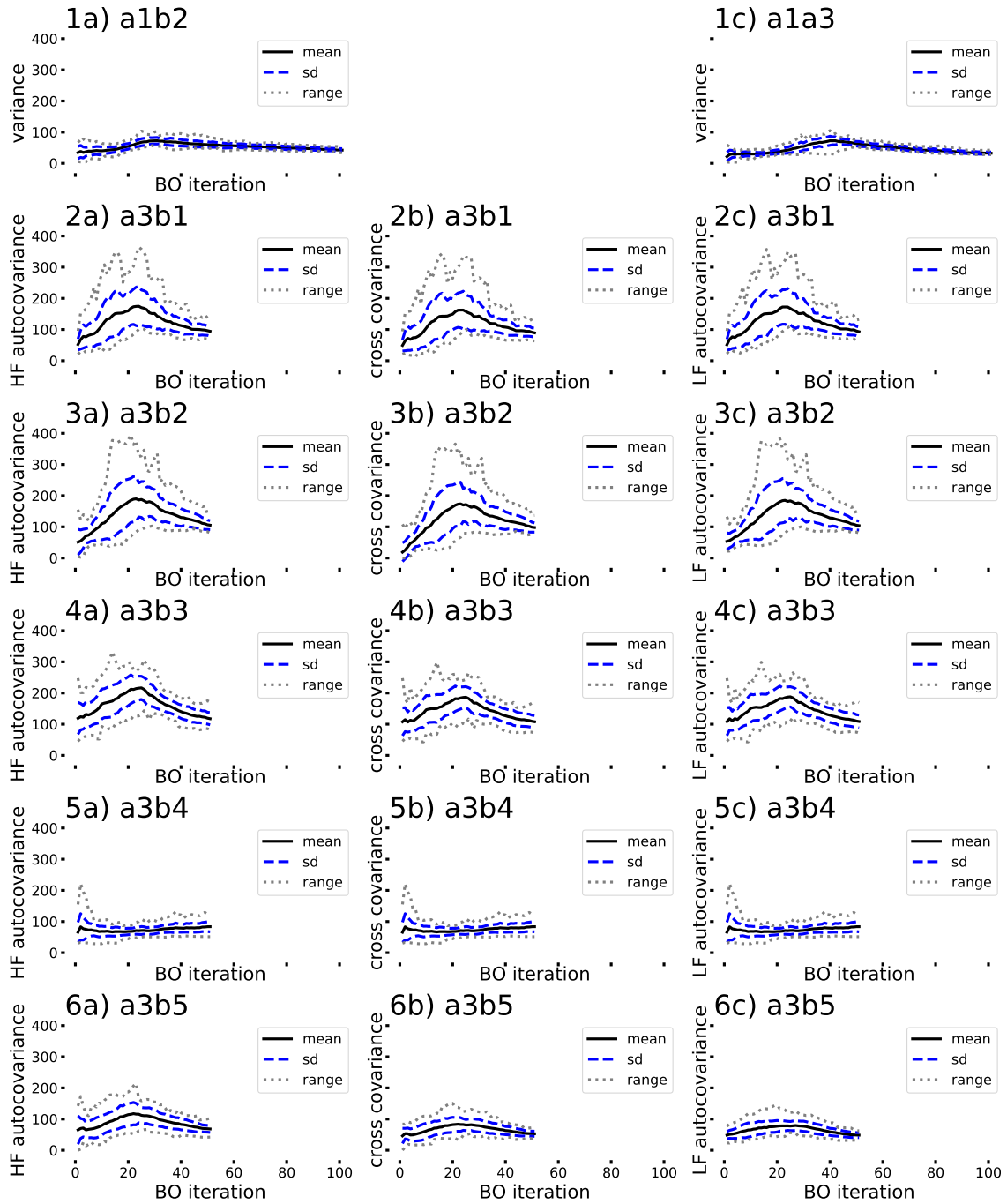


Figure 18: Variance or autocovariance and cross covariance distributions over BO iterations for 2 task prior hypothesis experiments. In first row are variances distributions of baselines as reference. Each row has in order primary task autocovariance, cross covariance and secondary task autocovariance for an experiment.

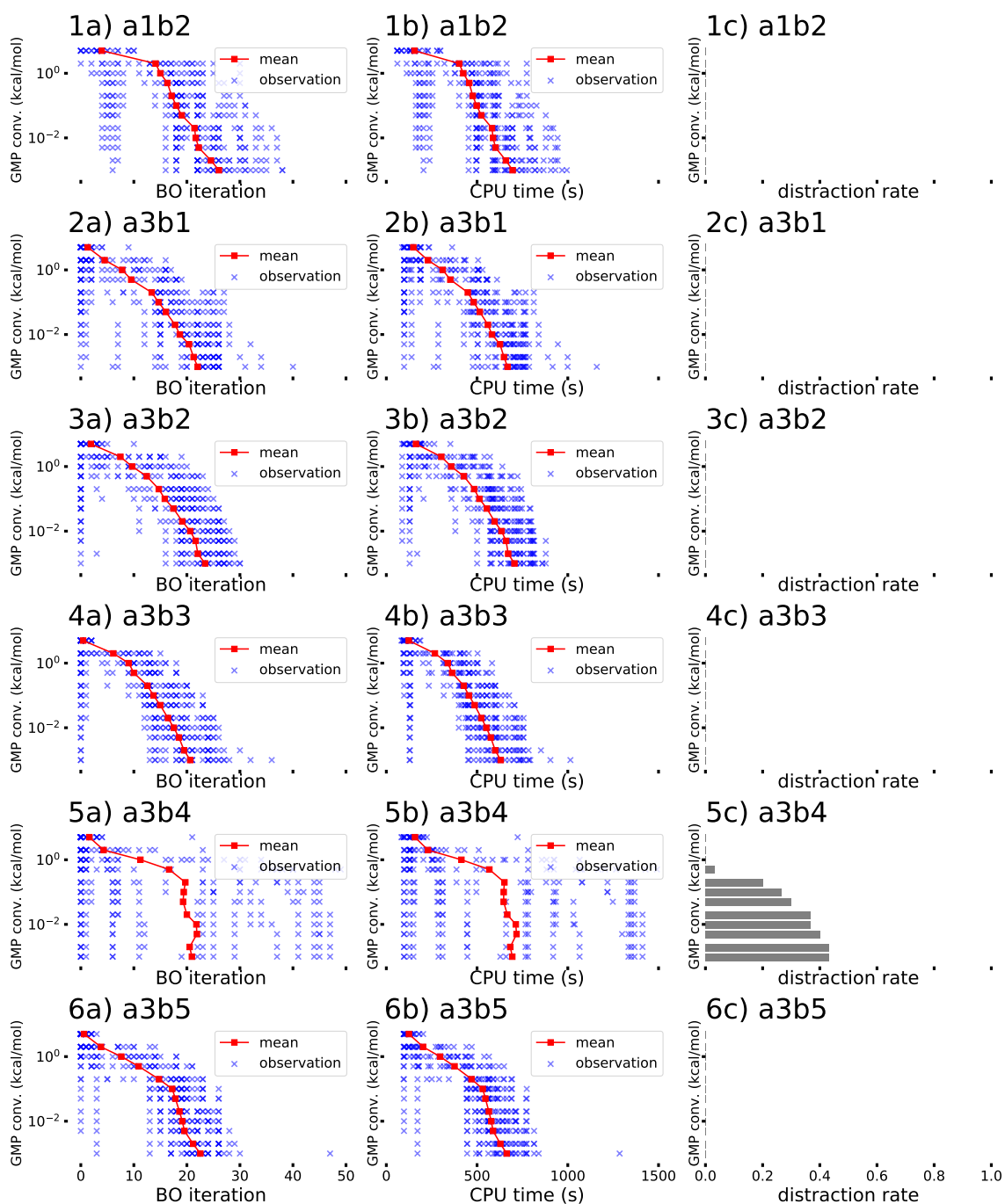


Figure 19: Convergence by BO iteration, CPU time and distraction rate in 2 task prior heuristic experiments. Rows are for experiments a1b2 (baseline), a3b1, a3b2, a3b3, a3b4 and a3b5 (TL heuristics).

## 8 Experiments 3: Evaluation

The last stage of the experiments was the evaluation. Here, I answer the research questions on how large computational cost reduction can be expected and on which conditions, and how the secondary data should be selected. I answer these by running a series of experiments with varying setups. I evaluate the potential benefit in two scenarios depending on if secondary data is readily available or not.

I tried to identify the dominant factors that effect the performance of ICM for TL in BOSS. I suspected that these would include the correlation and resource demand, i.e. acquisition time, ratio between simulators, the complexity and differences of the PES landscapes, the search space dimensionality, and most importantly the number of secondary initialization points, that is the data taken from previous experiments, and the method for selecting the secondary data, i.e. initialization strategy. Upper boundary for the amount of secondary data had to be set. I decided to limit the maximum number of secondary data to what I observed maximum number required for the model to converge in the baselines. Any data over that, I assumed, would be overfitting. I ran a series of experiments where I tried to vary the setups with regard to these assumptions, including a3b6, a3b7, a3b8, a3b9, a3c1, a3c2, a3c3, a3c4, a3c5, a3c6, a3c7, a3c8, b3b1, b3b2, b3c1, b3c2, b3c3 and b3c4.

In addition to the factor analysis I formulated two use case scenarios for evaluating the benefit of TL over single task optimization in different situations. Let us imagine that there was a researcher who wanted to do BOSS, and was wondering should they attempt transfer learning or not. We can assume two different scenarios.

In the first scenario the structure of interest has no pre-existing PES data available. This data should first be generated, before it can be used to initialize BOSS on a higher fidelity. I study three alternative initialization strategies: random, sobol and BO inorder. In random sampling the acquisition locations are selected by sampling random uniform distribution and in sobol following the sobol sequence. BO inorder means running BOSS with single task and reusing the run results as initialization data. In sobol and random the computational cost is limited to the acquisition times, where in BO inorder the cost also involves the active learning process.

In the second scenario it is assumed that there is PES data of the structure of interest readily available with no cost assigned to it. It does not matter whether or not the origin of this data is previous work of the researcher or someone else. The key difference to the first scenario is, that the data has already been obtained and there is no additional cost that would come from using it for future work. Let us also assume that the data is the result of some active learning experiment, i.e. that the information content of the data is high. However, the secondary data might not contain the points of interest for our researcher (PES minima), despite being

high quality. If the data is created for some other purpose than that of the current study, the focus of the active search might have been different. Is it better to use pre-existing active learning data over creating new BOSS data as in the first scenario?

These experiments included a3b6, a3b7, a3c1, a3c2, a3c3, a3c4, b3b1, b3b2, b3c1, b3c2, b3c3 and b3c4.

## 8.1 Results of Evaluation

### 8.1.1 Initialization Strategies for TL

Before evaluating the performance of TL experiments, I analyse the different methods for selecting secondary data. By comparing these initialization strategy alternatives I attempt to explain possible differences between them and how they may effect BOSS performance. In addition to hyperparameters I assume, that information content of the secondary data used will greatly effect the performance of TL. Here I use four different methods for selecting that data. First three strategies, random, sobol and BO in order are based on creating secondary data from scratch. Each strategy then increases the resource demand of the model, but also adds to the acquisition cost. In the fourth strategy I test a hypothetical case that there is pre-existing data from an active learning process. Because this data already exists, there is no acquisition cost.

The analysis results are visualised in Figure 20. In the first three the data points were taken in order from the first BOSS run in that experiment setup, except for BO random which was sampled from the experiment a1a3 in the analysis phase for demonstration. Let me go through the analysis of the strategies in Figure 20.

**Random** uniform sampling is visualized in the first column of Figure 20. There are no clear patterns in the scatter plot. Also, there are large uncovered areas, so called 'holes', in the scatter. From basic probability theory we also know, that when the dimensionality of the search space is increased, the sampling density increases towards the surface of the search space. There are also many points that are very close to each other. This is unfavourable, because there is little information gain in sampling almost the same location twice. We also see this in the histogram and potential energy scatter plot. The expected distance to closest other observation is smaller compared to other experiments.

**Sobol** sampling is visualized in the second column of Figure 20. There is a visible pattern in the scatter plot, although a very complex one. Sobol is designed for high-dimensional hypercubes, so it may be that this pattern weakens when dimensions are increased. There are large uncovered areas, but they appear smaller than with random uniform sampling. Looking at the histogram we can see that the expected nearest neighbour distance is larger than with random sampling, and that there are no very small distances. In the potential energy plot we see that now we do not find the lowest values of energy. From the striped pattern we can see that there are discrete

number of nearest neighbour distances. Based on the inspection I suspect that sobol sampled data would provide more information compared to random sampling.

**BO inorder** sampling strategy is plotted in the third column. Here the observations were taken in order from a BOSS run. In the scatter plot we see that there are no large uncovered areas, and that no location is sampled twice, except for a location of interest in the upper left region. This location is the global minimum location. From the histogram and potential energy scatter plot we can see that these closely sampled points are close to GMP. The search space is covered well, uninformative sampling is avoided and the search is focused in areas of interest. If we assume high correlation between tasks, it would be logical that this strategy provides the most information in the secondary data. However, the cost of random and sobol sampling equals to the acquisition cost of the observations in practice. With BO sampling, the surrogate model adds a significant cost to the data. It is interesting to see, if BO improves the quality of the data so much, that it covers this additional optimization cost.

**BO random** sampling is shown in the fourth column. The sample was resampled from experiment a1a3. Now, the clear difference in the scatter compared to BO in order is, that the focus on the minimum location is missing. With pre-existing data this might not be the case, but the choice was intentional. We can reasonably assume that taking data in order from previous experiment would have better information value and provide better results. The goal here was to rather simulate a situation where secondary data is free and high quality, but may be incomplete by some aspect. For example, if we use data from previous active learning experiment that has not converged, has fixed dimensions or otherwise is not guaranteed to have perfect information value, we may see something like this. No location is sampled twice, there are no large holes, but also the focus may be lost.

I performed two nonparametric ANOVA tests to compare loss function on the initialization strategies of the TL experiments (all stage 3 experiments). First, on 2D experiments and then on 4D experiments. I stacked the loss function values of each initialization strategy, sorted by the number of initialization points to get paired samples. Then I identified which strategies deviated from the mass using Kruskal-Wallis test. Finally, I did pairwise comparisons with Wilcoxon Signed Rank test to rank the methods on loss. I found no statistical difference in loss between random and sobol experiments. BO in order had lower loss than both random and sobol. Consequently, BO random had lower loss than BO in order.

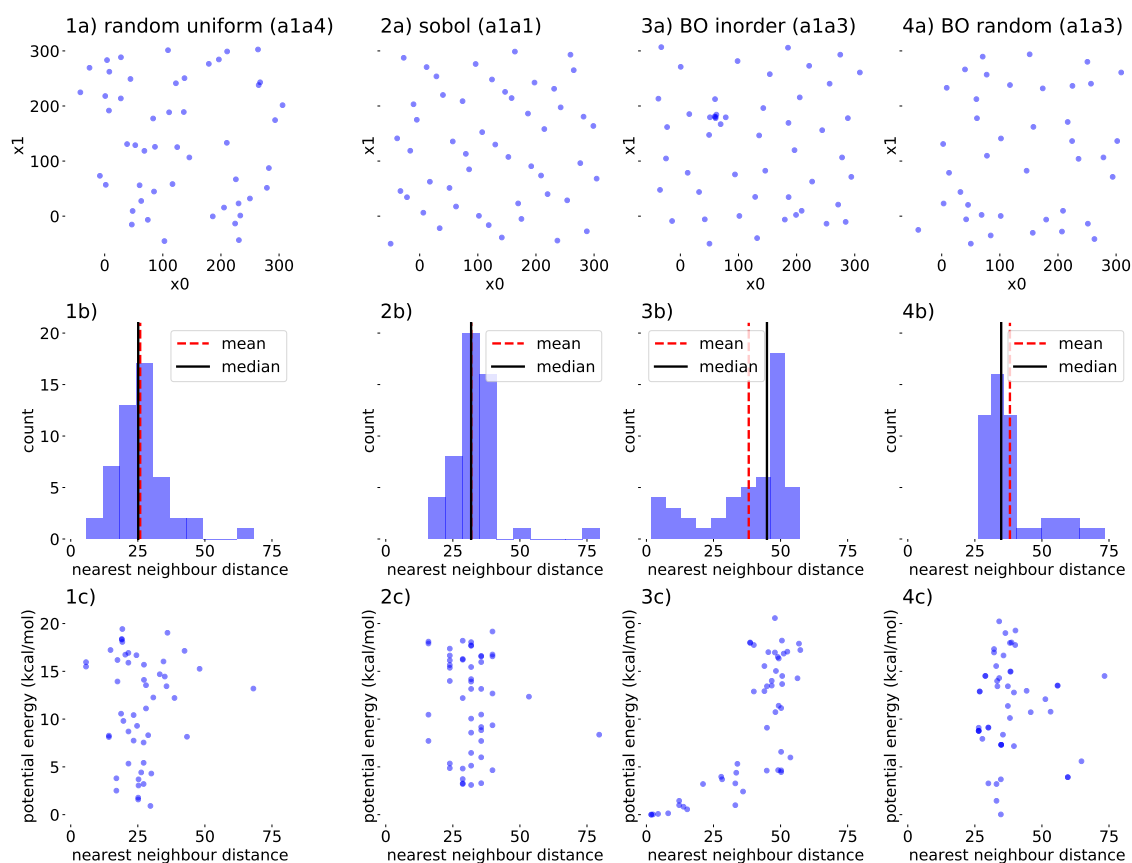


Figure 20: Scatter and coverage of different TL initialization strategies. In rows are first the scatter plot of values  $x$ , in the second row is the histogram of nearest neighbour distances and in the third row the potential energy plotted against the nearest neighbour distances. Nearest neighbour distance is the distance of a point to the next closest one. Each column has 50 points from an experiment representing an initialization strategy. This is more than what is used in most 2D experiments, but significant amount of observations are required to magnifying the differences.

### 8.1.2 TL Experiments

To evaluate TL performance with different setups, I plotted the convergence time to tolerance level of 0.1 kcal/mol as a function of secondary initialization points for all TL experiments. These plots are shown in Figures 21 - 26. In all figures the vertical axis is speed of convergence. On the first row the convergence speed is in BO iterations, and on the second row in CPU time. In the horizontal axis is the number of secondary data used in the experiment. Again, blue 'x'-glyphs denote a single BOSS run at given state, and the printout is in 50% transparency to show overlap. At the location zero is the results from a baseline experiment. The experiment runs were grouped by the number of secondary data, and for each of these sets the mean and a linear regression model was calculated for evaluating the trend and expected performance. Points with z-score more than 2.5 were excluded. There were only few instances of distraction in the transfer learning experiments, which may indicate greater risk for distraction compared to baselines, where it was not observed. Apparently distraction is possible but extremely rare.

### 8.1.3 Experiments on Alanine 2D

In Figure 21 are plotted the results for experiments a3b9, a3c7 and a3c8. The initialization strategy is random uniform. There appears to be a declining trend in number of BO iterations for convergence in all experiments. This indicates, that there is transfer of information and that the secondary data is theoretically useful. With more data, more information is transferred and the GP model is improved. The means deviate from the trend line, indicating that best results would be obtained with 20-30 secondary data points. However, there is large scatter in the results. Looking at the second row we see, that no benefit in CPU time is obtained, aside from few setups that deviate from the trend in away that we can discard them. The explanation can be, that the baseline model is already light and converges fast. The additional cost from the secondary data outweighs the benefits. Increasing the acquisition time ratio between simulators decreases the number of primary data required, but increases the total computational time.

In Figure 22 are plotted the results for experiments a3b8, a3c5 and a3c6. The initialization strategy is sobol. The declining trend is not as clear as in random initialized experiments. Scatter is surprisingly larger compared to random initialization. Again there appears to be no benefit compared to the baselines, only few trivial cases. In experiments a3b8 and a3c5 it is not clear that the lower fidelity data even reduces the number of primary data required, compared to a3c6 with lowest fidelity difference.

In Figure 23 are plotted the results for experiments a3b7, a3c3 and a3c4. The initialization strategy is BO in order. There is a slightly declining trend in BO iterations as number of secondary data is increased, and in fact the number of BO iterations to convergence drops close to zero in a3c4. This also indicates that low

fidelity difference between simulators may provide better results, probably because of the higher correlation. The benefit are not that drastic measured in CPU time. However, in all plots it seems that between 20 to 30 secondary data points the mean CPU time is below the baseline. After that, the additional cost from the observations devours the benefit. Reasonable number of points should be required from the primary task to construct a reliable model. Comparing to the convergence speeds of baselines, it appears that the fastest convergence is observed when as much secondary data is used as BOSS requires to converge the secondary task. The fact, that only few iterations were required in all runs, as in **a3c4**, may indicate that the model is overfitted. Of the three initialization strategies where new secondary data is created, BO inorder appears to give the best results. There is least scatter and visible region where TL is faster than the baseline.

In Figure 24 are plotted the results for experiments **a3b6**, **a3c1** and **a3c2**. The initialization strategy is BO random, and now the secondary data is assumed free. The results are similar to BO inorder initialization, but the scatter is greater. The declining trend is not as strong as with BO inorder. However, because the secondary data is free, the benefit in CPU time is stronger compared to any of the experiments where new secondary data is created.

#### 8.1.4 Experiments on Alanine 4D

BO inorder gave best results in the 2D experiments when secondary data was created from scratch. Because of this, I dropped random and sobol strategies when moving on to experimenting with alanine 4D, and focused on evaluating BO inorder and BO random strategies.

In Figure 25 are plotted the results for experiments **b3b1**, **b3c1** and **b3c2**. The initialization strategy is BO inorder. With the increased search space dimension there are clear benefits from the secondary data. The trend line in BO iterations is strongly decreasing in all cases. The decline of the trend increases the closer the fidelities of the simulators are, in both BO iterations and CPU time. There are visible ranges of secondary initialization point where benefit in CPU time can be expected. Again best results are observed around 160 secondary points, when the secondary task is assumed converged. After that the number of iterations required for convergence drops to very few, or the CPU time starts to incline anyway.

In Figure 26 are plotted the results for experiments **b3b2**, **b3c3** and **b3c4**. The initialization strategy is BO random. The scatter is increased compared to BO inorder, and the decline in BO iterations is not as strong. This indicates that information value of the secondary data is not as great. It also appears that the model does not get overconfident as fast. Because no cost is assigned to the secondary data, significant benefit can be expected compared to baseline and BO inorder initialization. Again, lowest fidelity difference provides the steepest decline in both BO iterations

and CPU time. However, it appears that more secondary data could have been provided in experiments b3b2 and b3c3 without overfitting.

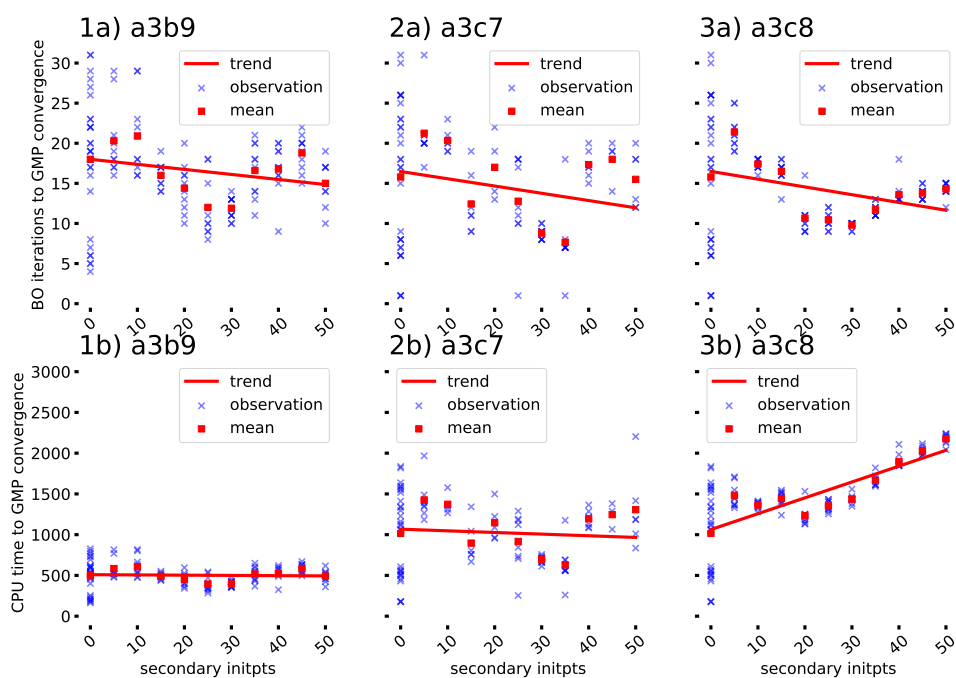


Figure 21: GMP convergence speeds to 0.1 kcal/mol of alanine 2D with random uniform sampling initialization strategy.

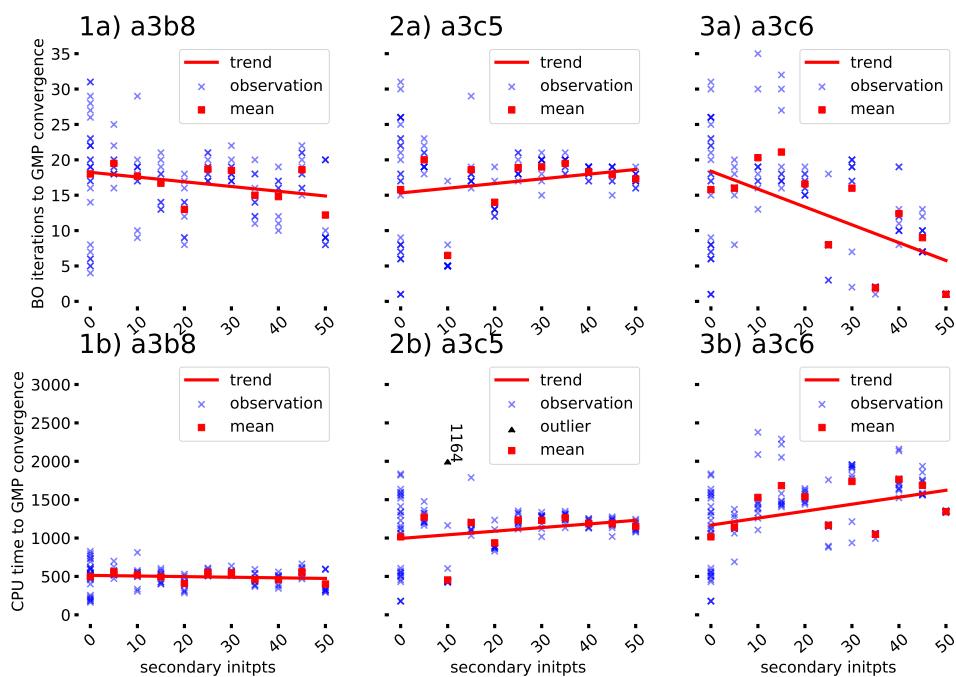


Figure 22: GMP convergence speeds to 0.1 kcal/mol of alanine 2D with sobol sampling initialization strategy.

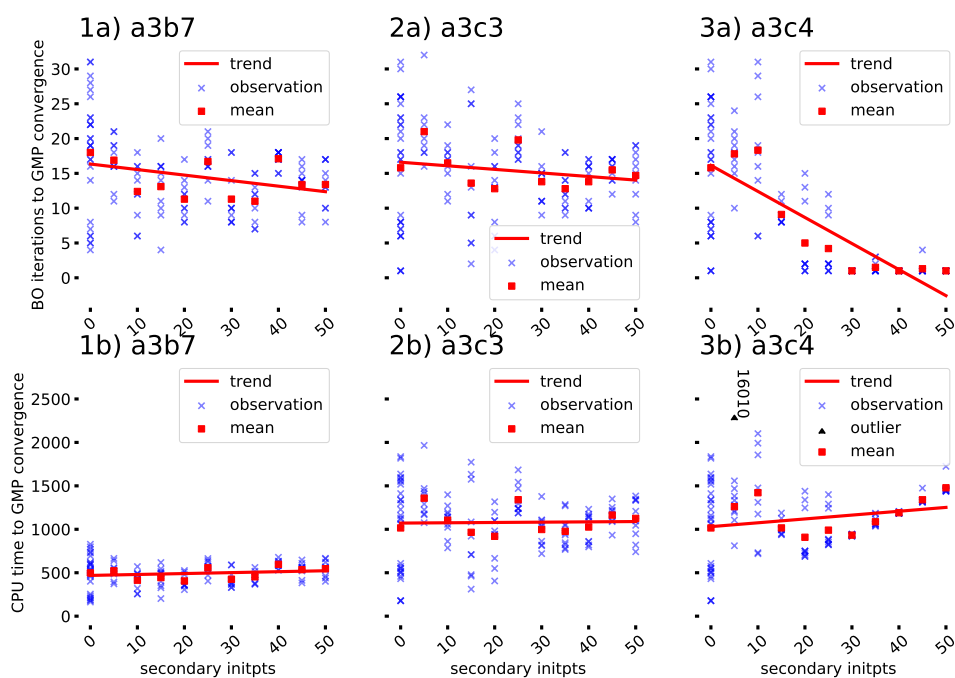


Figure 23: GMP convergence speeds to 0.1 kcal/mol of alanine 2D with BO in-order sampling initialization strategy.

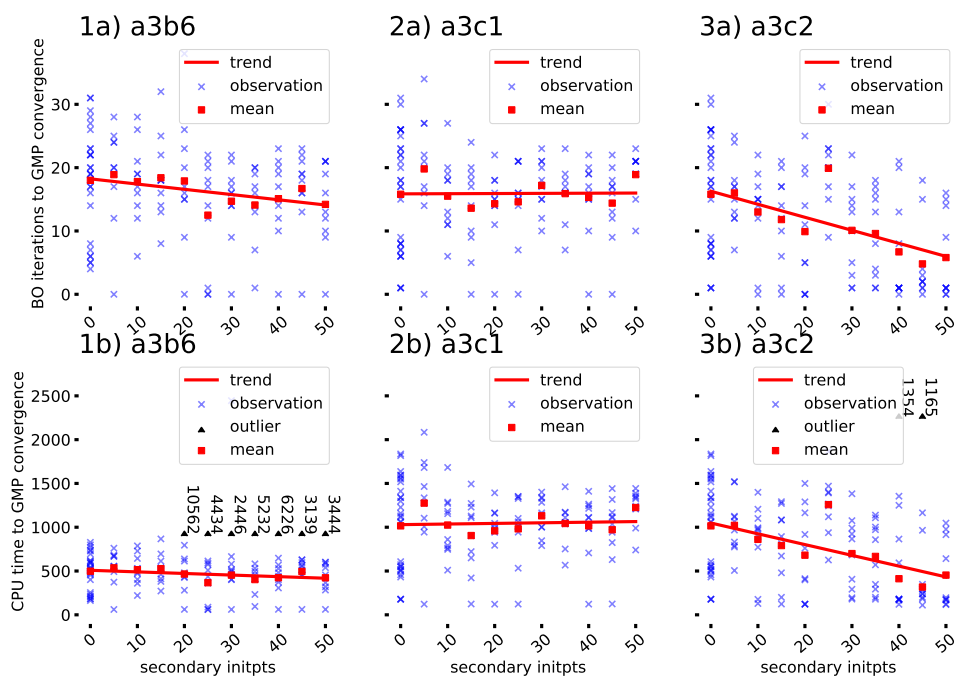


Figure 24: GMP convergence speeds to 0.1 kcal/mol of alanine 2D with BO random sampling initialization strategy.

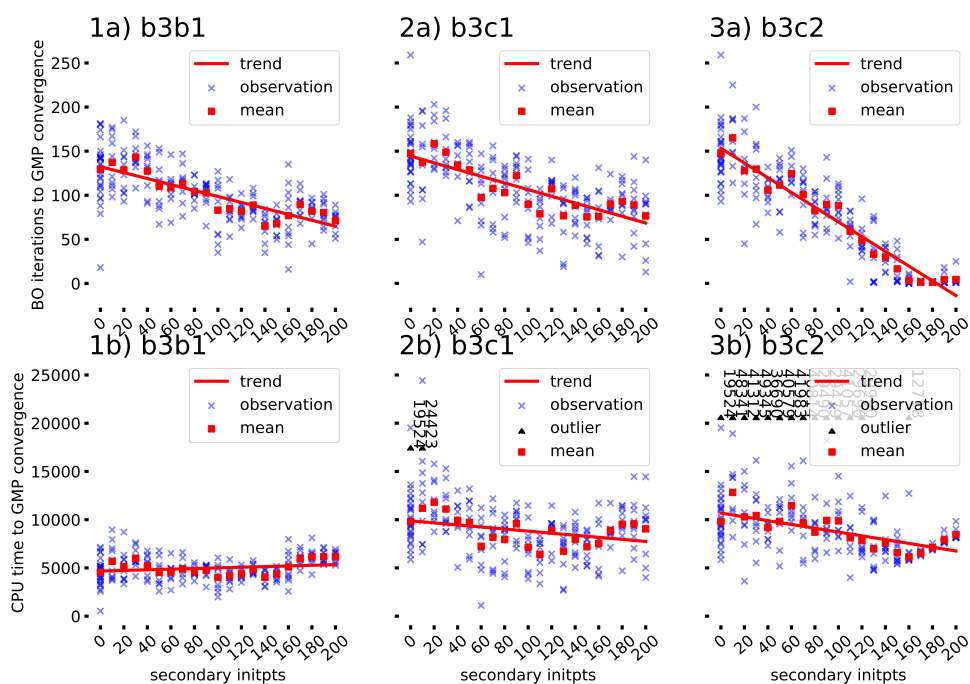


Figure 25: GMP convergence speeds to 0.1 kcal/mol of alanine 4D with BO in-order sampling initialization strategy.

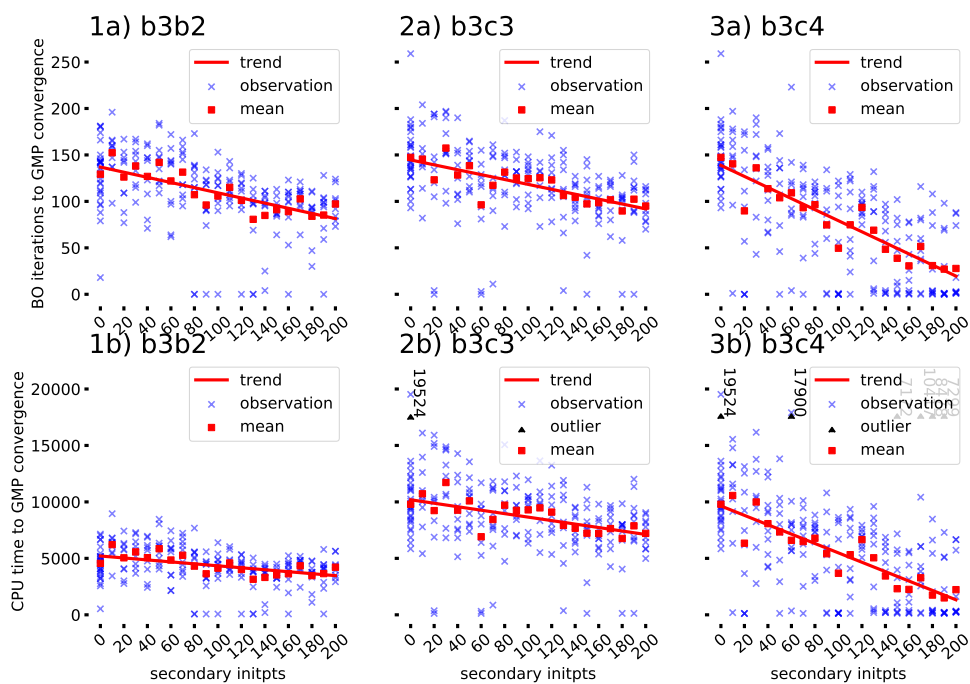


Figure 26: GMP convergence speeds to 0.1 kcal/mol of alanine 4D with BO random sampling initialization strategy.

### 8.1.5 Loss Functions

I used the loss functions to compare TL experiment results in an equal scale. Loss function evaluates the mean resource demand (CPU time) of an experiment setup compared to the baseline. Indicator loss function evaluates, is the setup statistically preferable compared to the baseline. Loss function values for all setups are listed in Appendix A1.

Figure 27 shows the results for the indicator loss function. The results agree with the visual inspection of the convergence speeds. In most cases, TL appears to converge faster compared to the baselines. Too little or too much data slow the GMP convergence. However, an indicator loss function does not reveal the magnitude of the benefit or disadvantage.

In Figure 28 are plotted the loss function minima as function of secondary initpts for all experiments. The same values are presented in Table 9. These points equal to the minimum mean CPU time points in Figures 21 - 26. Loss function value describes the resources used. The loss function values appear to decrease with search space dimension, whereas the number of secondary data required increases.

In 2D experiments most of the minimum loss function values are around 0.8. The lowest values are a3c7 with value 0.62 (BO inorder) and a3c2 with value 0.31 (BO random). The general trend appears to be between 0.65 and 0.95. The a3c5 loss minimum oddly deviated from the general trend, and I consider it trivial case that would be unlikely to be observed in any other setup. We also see a conflict between the indicator and continuous loss functions in experiment a3c6. Minimum loss setup of it is indicated to be faster than the baseline, even though it has loss value over 1. This is because the loss function is based on mean, and the indicator loss on median. In 4D experiments the loss minima are focused around 0.65 in both BO inorder and BO random experiments, with the exception of experiment b3c4 with surprisingly low value of 0.15 at 190 secondary initialization points.

## 8.2 Summary of Evaluation

In the TL experiments I evaluated different factors that could effect performance, how to select the secondary data, and the potential benefit of TL depending on if secondary data is readily available or not. I observed that high search space dimension, low fidelity difference between simulators, high linear correlation between simulators, and difficulty of finding the primary task minimum predict improved cost reduction. If secondary data is readily available, it should be used up to reasonable amount. I suspect that reasonable amount is the number of data that BOSS would require on that task alone. With this setup, loss of around 0.15-0.65 was observed. If secondary data is not available, it should be created with BOSS for good information content. The best results were observed when the secondary task is converged. Then,

Table 9: Loss function minimas per TL experiment

experiment	secondary initpts	min loss
a3b6	25	0.74
a3b7	20	0.81
a3b8	50	0.81
a3b9	30	0.79
a3b9	25	0.79
a3c1	15	0.89
a3c2	45	0.31
a3c3	20	0.91
a3c4	20	0.89
a3c5	10	0.45
a3c6	35	1.03
a3c7	35	0.62
a3c8	20	1.21
b3b1	100	0.88
b3b2	130	0.69
b3c1	110	0.65
b3c2	160	0.62
b3c3	180	0.69
b3c4	190	0.15

loss of around 0.65-0.95 was observed. Too much secondary data may result in overfitting.

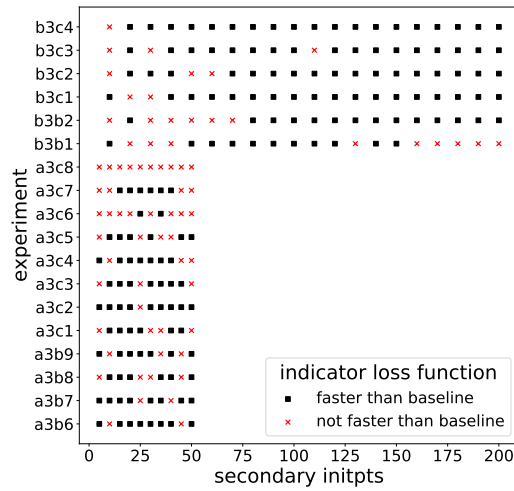


Figure 27: Indicator loss function results for all TL experiments. On vertical axis is the experiment and on horizontal axis the number of secondary initialization points. A black square denotes that the specific experiment setup converged statistically faster than the baseline. A red x means the opposite.

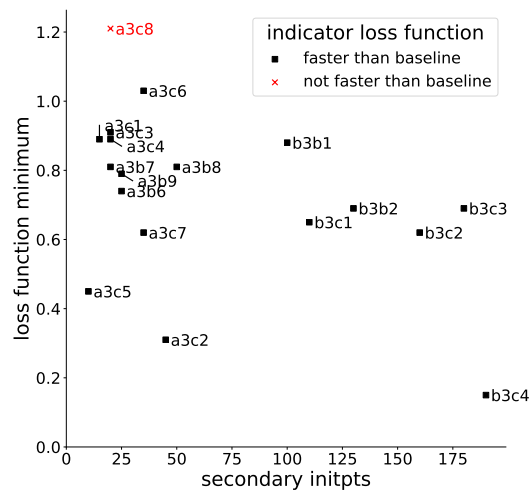


Figure 28: Loss function minima for all experiments. On vertical axis is the loss function value, where 1 is the mean convergence time of the baselines. On horizontal axis is the number of secondary initialization points. The glyphs denote the indicator loss function value. This is to check for possible conflicts between the two loss functions, as in experiment a3c6.

## 9 Conclusions

### 9.1 Research Summary

In this thesis I studied the use of TL in resource optimization of computational electronic structure search. I tried to utilize faster low fidelity simulations to initialize active learning of PES with BOSS on a slower, higher fidelity simulator. I chose to study a particular method, LMC, and its restricted version ICM because of the narrow scope of the thesis work, and the wide applicability and scalability of the method. The ICM method was convenient to adopt to the GP framework of BOSS. The thesis was continuation on my previous work of implementing the method to BOSS. In the previous work I had established a proof of concept, that reduction of computational resource demand in active learning is possible with TL. In this thesis I focused on refining and evaluating the application of the ICM method in BOSS.

The thesis work consisted of a literature review, a mathematical study, and experimental work. In the literature review I introduced the background and basic concepts of computational materials science, structure search, BOSS and transfer learning through LMC. In the mathematical study I evaluated rules for treating the hyperparameters of LMC. In addition I formulated heuristic priors for LMC hyperparameters in BOSS. The experimental work consisted of three stages. First was basic analysis, where I created baseline knowledge of the current best performance of BOSS and studied the simulators. In the model refinement stage I evaluated the LMC priors and hyperparameter treatment. In the evaluation stage I studied performance of TL in BOSS.

The experimental results show that TL can reduce computational cost of BOSS significantly. The lowest value of loss observed with TL was around 0.65-0.95. The best results were obtained when there was high correlation between simulators and secondary data is taken from converged BOSS experiment. With data from 2 simulators, hyperparameter prior heuristic 2.2 gave the best results and was used for the evaluation.

If secondary data from active learning is readily available, the experiments predict resource demand of around 0.15-0.65 compared baselines. The results suggest that if active learning data is readily available for transfer learning, it should be used. If it is not available, BOSS should be used to obtain that data.

High dimension of search space, complexity of PES, low fidelity difference and high linear correlation between tasks predict successful and beneficial TL. If the minima of the primary task PES is difficult to find, it may increase the benefit from transfer learning. Because few instances where correct PES minimum was not found with TL were observed, I assume the risk of distraction is greater when using TL compared to single-output BOSS. However, this risk very small with correct setup.

Avoiding too restrictive hyperparameter priors reduces the risk. Too much secondary data may overfit the GP model.

## 9.2 Practical Implications on Using TL with BOSS

Based on the results of my thesis work, I suggest using LMC for TL in BOSS when possible. To reduce the computational cost of transfer learning I propose the following.

First and foremost the researcher should be familiar and comfortable with the basic BOSS use before attempting TL. Active learning is a powerful tool, but only when used correctly. One should study and understand the the basic mathematical concepts behind the methods before using them. Hasting into methods beyond comprehension risks wasting time and resources.

When moving on to doing TL, the problem at hand should be considered carefully. If there is active learning data available, it should be used instead of generating new. If the data is plenty, it should be sampled to a smaller subset. My research suggests that too little or too much data may be disadvantage. I suggest using as much secondary data as you would expect BOSS would require on a single output setup of that task alone.

If secondary active learning data is not readily available, it should be created with BOSS. To improve resource savings, BOSS should be run only up to a point of convergence. The secondary data can also be used to evaluate the structure and PES, to evaluate amplitude for setting the priors. I suggest using the ICM prior heuristic 2.2, where the variance hyperparameter is fixed to 1,  $\kappa$  is free and unpriorized and the hyperparameters  $W$  have rank 1 and Gaussian prior

$$\text{prior}_W = N\left(0.9\sqrt{\frac{2}{\beta}}, \frac{1}{2\sqrt{\beta}}\right), \beta = \frac{2}{\hat{A}^2}$$

where  $\hat{A}$  is the expected amplitude.

Based on the findings I expect that best results are obtained when the fidelities of primary and secondary task are close. Then, highest degree of correlation between the tasks can be expected. Potential energies from different simulators should be converted to same unit before giving them to BOSS input.

## 9.3 Limitations

The results of the thesis may not apply for problems not presented here, such as new structures and other levels of fidelity. For example the linear correlation assumption of simulators may not apply for high dimensional structures. It should also be noted, that no true resource savings were observed in the thesis work, because all experiments

were run for a predetermined number of iterations and convergence was measured in postprocessing. Overfitting, underfitting and too restrictive hyperparameter treatment may slow the active learning process with transfer learning, or even completely corrupt it. The hyperparameter priors were set based on knowledge of the amplitudes from sobol query experiments. Estimates this accurate are unlikely to be available.

## 9.4 Suggestions and Future Work

Based on the thesis work my suggestions for future work include:

**Studying more complex structures.** TL with BOSS should be tested with structures of higher dimension for more generalized proof and evaluation of the benefits.

**Implementing and testing full LMC capability.** Only single basic kernel was applied in the thesis work. However, it might be for example that different basic kernels would describe different tasks better than the other, and adding multiple kernels could improve the learning.

**Evaluating potential risks of TL.** The results suggest, that TL may involve a risk of distraction that was not observed in the baselines. With the suggested hyperparameter treatment these incidences were rare. However, since distraction is a sever error, this should be studied more.

**Implementing new priors and parameter tying to GPy.** I showed that the selection of available priors in GPy was limited for our purposes. In addition, I believe that enabling hyperparameter tying could further improve the LMC performance by simplifying the hyperparameter fitting process.

**Studying active learning with multiple targets.** In this thesis I focused on transfer learning, a limited setup of multi-task learning. Here, observations were made first from one source and then the other. However, this does not have to be the case. With a multi-task acquisition function BOSS could select at each iteration, which simulator to acquire from based on the observations and cost of acquisitions. This would allow optimization of multiple properties simultaneously.

**Studying other TL methods.** In this thesis I studied linear coregionalization. However, for example nonlinear coregionalization methods for transfer learning could provide good results, especially if high linear correlation can not be assumed.

## 9.5 Reproducibility

The analysis of the thesis is completely reproducible. This includes analysis figures, tables and statistical testing. The analysis workflow and intermediate result dependencies were automated using a scientific workflow management tool for reproducible and scalable data analysis. Code dependencies were managed using a virtual environment. The Git repository with preprocessed (cleaned) data, analysis scripts and virtual environment requirements is available at Zenodo (Sten 2020b). The repository contains instructions for reproducing the analysis. Raw data can be requested from the *BOSS project n.d.*

The BOSS project moved under new version control system during the thesis work, and unfortunately there is not yet a stable release with the ICM capability available to the public. Before then, this source code can be requested by contacting the BOSS project.

Seed control was not enabled in the versions of BOSS used in the thesis, so experimental results are only statistically reproducible. This has been fixed in the later releases, and further algorithmic development work of BOSS can be seeded for complete reproducibility.

## References

- Blum, V. et al. (2009). *The Fritz Haber Institute ab initio molecular simulations package (FHI-aims)*. <http://www.fhi-berlin.mpg.de/aims>.
- Bonilla, Edwin V, Kian M. Chai, and Christopher K. I. Williams (2008). “Multi-task Gaussian Process Prediction”. In: *Advances in Neural Information Processing Systems 20*. Ed. by J. C. Platt et al. Curran Associates, Inc., pp. 153–160. Retrieved 18/06/2020. Available at: <http://papers.nips.cc/paper/3189-multi-task-gaussian-process-prediction.pdf>.
- BOSS project (n.d.). <https://cest.aalto.fi/boss/>.
- BOSS python module (n.d.). <https://cest-group.gitlab.io/boss>.
- Brochu, Eric, Vlad M. Cora, and Nando de Freitas (2010). “A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning”. In: arXiv: [1012.2599](https://arxiv.org/abs/1012.2599) [cs.LG].
- Cao, Ming et al. (1995). “Ab initio conformational analysis of alanine”. In: *Journal of Molecular Structure: THEOCHEM* 332.3, pp. 251–267. ISSN: 0166-1280. DOI: [https://doi.org/10.1016/0166-1280\(94\)03943-F](https://doi.org/10.1016/0166-1280(94)03943-F).
- Case, D.A. et al. (2016). *AMBER 2016*. <https://ambermd.org/>.
- Cornell, W.D. et al. (1996). “A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules”. In: *J. Am. Chem. Soc.* 117 (19), pp. 5179–5197. DOI: [10.1021/ja00124a002](https://doi.org/10.1021/ja00124a002).
- Curtarolo, S. et al. (2013). “The high-throughput highway to computational materials design”. In: *Nature Materials* 12.3, pp. 191–201. DOI: [10.1038/nmat3568](https://doi.org/10.1038/nmat3568).
- Egger, A. T. et al. (2020). “Charge Transfer into Organic Thin Films: A Deeper Insight through Machine-Learning-Assisted Structure Search”. In: *Adv. Sci* 7 (15). DOI: [10.1002/advs.202000992](https://doi.org/10.1002/advs.202000992).
- Fang, L. et al. (2020). “Efficient Cysteine Conformer Search with Bayesian Optimization”. In: arXiv: [2006.15006](https://arxiv.org/abs/2006.15006).
- Goovaerts, Pierre (1997). *Geostatistics for Natural Resources Evaluation*. Applied Geostatistics Series. Oxford University Press, Inc., pp. 1–123. ISBN: 0-19-511538-4.
- GPy (2012). *GPy: A Gaussian process framework in python*. <http://github.com/SheffieldML/GPy>.
- Harville, David A. (1997). *Matrix algebra from a statistician’s perspective*. Springer-Verlag New York, Inc., pp. 135, 333–375. ISBN: 0-387-94978-X.
- Järvi, J., B. Alldritt, et al. (2020). “Integrating Bayesian Inference with Scanning Probe Experiments for Robust Identification of Surface Adsorbate Configurations”. In: *under revision at Nat. Commun.* DOI: [10.21203/rs.3.rs-50783/v1](https://doi.org/10.21203/rs.3.rs-50783/v1).
- Järvi, J., P. Rinke, and M. Todorović (2020). “Detecting stable adsorbates of (1S)-camphor on Cu(111) with Bayesian optimization”. In: arXiv: [2002.05598](https://arxiv.org/abs/2002.05598).
- Jones, R.O. (2015). “Density functional theory: Its origins, rise to prominence, and future”. In: *Reviews of Modern Physics* 87.3. cited By 286. DOI: [10.1103/RevModPhys.87.897](https://doi.org/10.1103/RevModPhys.87.897).

- Journel, A. G. and CH. J. Huijbregts (1978). *Mining Geostatistics*. ACADEMIC PRESS INC. (LONDON) LTD, pp. 1–91, 171–176, 303–326. ISBN: 0-12-391050-1.
- Kale, D. and Y. Liu (2013). “Accelerating Active Learning with Transfer Learning”. In: *2013 IEEE 13th International Conference on Data Mining*, pp. 1085–1090. DOI: [10.1109/ICDM.2013.160](https://doi.org/10.1109/ICDM.2013.160).
- Lee, Joohwi and Ryoji Asahi (2020). *Transfer learning for materials informatics using crystal graph convolutional neural network*. arXiv: [2007.09932](https://arxiv.org/abs/2007.09932).
- Lewars, Errol (2010). *Computational chemistry : introduction to the theory and applications of molecular and quantum mechanics*. 2nd ed. Springer. ISBN: 9789048138623.
- Morgan, Dane and Ryan Jacobs (2020). “Opportunities and Challenges for Machine Learning in Materials Science”. In: *Annual Review of Materials Research* 50, pp. 71–103. DOI: [10.1146/annurev-matsci-070218-010015](https://doi.org/10.1146/annurev-matsci-070218-010015).
- Murphy, Kevin P. (2013). *Machine learning : a probabilistic perspective*. Cambridge, Mass. [u.a.]: MIT Press. ISBN: 9780262018029 0262018020.
- Parr, Robert G. and Yang Yang Weitao (1989). *Density-Functional Theory of Atoms and Molecules*. International Series of Monographs on Chemistry. Oxford University Press, Incorporated. ISBN: 0-19-504279-4.
- Rasmussen, C. E. and Christopher K. I. Williams (2006). *Gaussian processes for machine learning*. the MIT Press. ISBN: 026218253X. Available at: [www.GaussianProcess.org/gpml](http://www.GaussianProcess.org/gpml).
- Schmidt, Jonathan et al. (2019). “Recent advances and applications of machine learning in solid-state materials science”. In: *npj Computational Materials* 5 (83). DOI: [10.1038/s41524-019-0221-0](https://doi.org/10.1038/s41524-019-0221-0).
- Sobol, I.M (1967). “On the distribution of points in a cube and the approximate evaluation of integrals”. In: *USSR Computational Mathematics and Mathematical Physics* 7.4, pp. 86–112. ISSN: 0041-5553. DOI: [10.1016/0041-5553\(67\)90144-9](https://doi.org/10.1016/0041-5553(67)90144-9).
- Sten, Nuutti A. (2020a). *Implementing Multi-Task Learning for Bayesian Optimization Structure Search*. Seminar Work. URL: <http://urn.fi/URN:NBN:fi:aalto-202005043002>.
- (2020b). *Preprocessed data and analysis scripts of the thesis*. DOI: [10.5281/zenodo.4291627](https://doi.org/10.5281/zenodo.4291627).
- Todorović, M. et al. (2019). “Bayesian inference of atomistic structure in functional materials”. In: *npj Computational Materials* 5.1. DOI: [10.1038/s41524-019-0175-2](https://doi.org/10.1038/s41524-019-0175-2).
- Yamada, Hironao et al. (2019). “Predicting Materials Properties with Little Data Using Shotgun Transfer Learning”. In: *ACS Central Science* 5 (10), pp. 1717–1730. DOI: [10.1021/acscentsci.9b00804](https://doi.org/10.1021/acscentsci.9b00804).
- Yang, Xin-She (2007). *Introduction to Mathematical Optimization: From Linear Programming to Metaheuristics*. Cambridge International Science Publishing. ISBN: 9781907343667.
- Zhan, D. and H. Xing (2020). “Expected improvement for expensive optimization: a review.” In: *J Glob Optim* 78, pp. 507–544. DOI: [10.1007/s10898-020-00923-x](https://doi.org/10.1007/s10898-020-00923-x).

Zheng, Xiaolong, Peng Zheng, and Ruizhi Zhang (2018). “Machine learning material properties from the periodic table using convolutional neural networks”. In: *Chemical Science* 9. DOI: [10.1039/C8SC02648C](https://doi.org/10.1039/C8SC02648C).

## A Table of Loss Function Values

Table A1: Table of loss function values for TL experiments. Loss function 1 equals as fast as the baseline. Indicator loss function value 1 indicates faster than baseline and 0 not faster than baseline.

experiment	secondary initpts	loss	indicator loss
a3b6	5	1.08	1
a3b6	10	1.03	1
a3b6	15	1.06	0
a3b6	20	0.94	1
a3b6	25	0.74	1
a3b6	30	0.91	1
a3b6	35	0.81	1
a3b6	40	0.85	1
a3b6	45	0.99	1
a3b6	50	0.85	1
a3c1	5	1.26	0
a3c1	10	1.01	1
a3c1	15	0.89	1
a3c1	20	0.94	1
a3c1	25	0.97	1
a3c1	30	1.11	1
a3c1	35	1.03	1
a3c1	40	1.0	1
a3c1	45	0.96	1
a3c1	50	1.21	1
a3c2	5	1.01	1
a3c2	10	0.85	1
a3c2	15	0.78	1
a3c2	20	0.67	1
a3c2	25	1.24	0
a3c2	30	0.69	1
a3c2	35	0.66	1
a3c2	40	0.41	1
a3c2	45	0.31	1
a3c2	50	0.45	1
a3b7	5	1.05	0
a3b7	10	0.83	1
a3b7	15	0.89	1
a3b7	20	0.81	1
a3b7	25	1.12	0
a3b7	30	0.85	1

Table A1: Continuation from previous page.

experiment	secondary initpts	loss	indicator loss
a3b7	35	0.91	1
a3b7	40	1.2	0
a3b7	45	1.08	0
a3b7	50	1.1	1
a3c3	5	1.34	0
a3c3	10	1.09	1
a3c3	15	0.95	0
a3c3	20	0.91	1
a3c3	25	1.32	0
a3c3	30	0.98	1
a3c3	35	0.96	1
a3c3	40	1.01	1
a3c3	45	1.15	1
a3c3	50	1.11	0
a3c4	5	1.24	1
a3c4	10	1.4	0
a3c4	15	1.0	1
a3c4	20	0.89	1
a3c4	25	0.97	1
a3c4	30	0.92	1
a3c4	35	1.07	1
a3c4	40	1.17	0
a3c4	45	1.32	0
a3c4	50	1.45	0
a3b8	5	1.12	1
a3b8	10	1.05	1
a3b8	15	0.99	1
a3b8	20	0.82	1
a3b8	25	1.11	1
a3b8	30	1.1	0
a3b8	35	0.93	1
a3b8	40	0.92	1
a3b8	45	1.12	1
a3b8	50	0.81	1
a3c5	5	1.25	0
a3c5	10	0.45	1
a3c5	15	1.18	1
a3c5	20	0.92	1
a3c5	25	1.21	0
a3c5	30	1.21	1
a3c5	35	1.24	0
a3c5	40	1.17	0

Table A1: Continuation from previous page.

experiment	secondary initpts	loss	indicator loss
a3c5	45	1.17	0
a3c5	50	1.13	1
a3c6	5	1.12	0
a3c6	10	1.51	0
a3c6	15	1.66	0
a3c6	20	1.51	0
a3c6	25	1.15	1
a3c6	30	1.71	0
a3c6	35	1.03	1
a3c6	40	1.74	0
a3c6	45	1.66	0
a3c6	50	1.32	0
a3b9	5	1.17	1
a3b9	10	1.22	0
a3b9	15	0.98	0
a3b9	20	0.91	1
a3b9	25	0.79	1
a3b9	30	0.79	1
a3b9	35	1.04	1
a3b9	40	1.05	0
a3b9	45	1.17	0
a3b9	50	0.99	1
a3c7	5	1.4	0
a3c7	10	1.35	0
a3c7	15	0.88	0
a3c7	20	1.13	1
a3c7	25	0.9	0
a3c7	30	0.68	1
a3c7	35	0.62	1
a3c7	40	1.18	1
a3c7	45	1.23	1
a3c7	50	1.29	0
a3c8	5	1.45	0
a3c8	10	1.34	0
a3c8	15	1.42	0
a3c8	20	1.21	0
a3c8	25	1.33	1
a3c8	30	1.41	0
a3c8	35	1.64	0
a3c8	40	1.87	0
a3c8	45	2.0	0
a3c8	50	2.14	0

Table A1: Continuation from previous page.

experiment	secondary initpts	loss	indicator loss
b3b1	10	1.25	0
b3b1	20	1.13	1
b3b1	30	1.32	0
b3b1	40	1.15	0
b3b1	50	0.99	1
b3b1	60	1.0	1
b3b1	70	1.07	1
b3b1	80	1.01	1
b3b1	90	1.05	0
b3b1	100	0.88	1
b3b1	110	0.92	1
b3b1	120	0.96	1
b3b1	130	1.06	0
b3b1	140	0.89	1
b3b1	150	0.96	1
b3b1	160	1.12	1
b3b1	170	1.31	0
b3b1	180	1.35	0
b3b1	190	1.35	0
b3b1	200	1.35	0
b3c1	10	1.14	0
b3c1	20	1.21	0
b3c1	30	1.13	0
b3c1	40	1.02	0
b3c1	50	0.99	1
b3c1	60	0.74	1
b3c1	70	0.83	1
b3c1	80	0.81	1
b3c1	90	0.98	1
b3c1	100	0.73	1
b3c1	110	0.65	1
b3c1	120	0.91	1
b3c1	130	0.69	1
b3c1	140	0.81	1
b3c1	150	0.73	1
b3c1	160	0.77	1
b3c1	170	0.91	1
b3c1	180	0.97	1
b3c1	190	0.98	1
b3c1	200	0.92	1
b3c2	10	1.31	0
b3c2	20	1.05	1

Table A1: Continuation from previous page.

experiment	secondary initpts	loss	indicator loss
b3c2	30	1.07	1
b3c2	40	0.94	1
b3c2	50	1.0	1
b3c2	60	1.17	0
b3c2	70	0.99	1
b3c2	80	0.89	1
b3c2	90	1.01	1
b3c2	100	1.01	1
b3c2	110	0.83	1
b3c2	120	0.8	1
b3c2	130	0.71	1
b3c2	140	0.78	1
b3c2	150	0.67	1
b3c2	160	0.62	1
b3c2	170	0.67	1
b3c2	180	0.73	1
b3c2	190	0.81	1
b3c2	200	0.86	1
b3b2	10	1.36	0
b3b2	20	1.11	0
b3b2	30	1.22	0
b3b2	40	1.11	1
b3b2	50	1.28	0
b3b2	60	1.06	1
b3b2	70	1.16	1
b3b2	80	0.95	1
b3b2	90	0.8	1
b3b2	100	0.91	1
b3b2	110	1.01	1
b3b2	120	0.88	1
b3b2	130	0.69	1
b3b2	140	0.73	1
b3b2	150	0.78	1
b3b2	160	0.8	1
b3b2	170	0.95	1
b3b2	180	0.76	1
b3b2	190	0.8	1
b3b2	200	0.92	1
b3c3	10	1.09	1
b3c3	20	0.94	1
b3c3	30	1.2	0
b3c3	40	0.95	1

Table A1: Continuation from previous page.

experiment	secondary initpts	loss	indicator loss
b3c3	50	1.03	1
b3c3	60	0.71	1
b3c3	70	0.86	1
b3c3	80	0.99	1
b3c3	90	0.95	1
b3c3	100	0.95	1
b3c3	110	0.97	1
b3c3	120	0.93	1
b3c3	130	0.81	1
b3c3	140	0.78	1
b3c3	150	0.74	1
b3c3	160	0.73	1
b3c3	170	0.78	1
b3c3	180	0.69	1
b3c3	190	0.8	1
b3c3	200	0.74	1
b3c4	10	1.08	0
b3c4	20	0.65	1
b3c4	30	1.02	1
b3c4	40	0.82	1
b3c4	50	0.75	1
b3c4	60	0.67	1
b3c4	70	0.66	1
b3c4	80	0.69	1
b3c4	90	0.55	1
b3c4	100	0.38	1
b3c4	110	0.54	1
b3c4	120	0.68	1
b3c4	130	0.52	1
b3c4	140	0.35	1
b3c4	150	0.24	1
b3c4	160	0.23	1
b3c4	170	0.33	1
b3c4	180	0.18	1
b3c4	190	0.15	1
b3c4	200	0.23	1