CHALMERS
UNIVERSITY OF TECHNOLOGY

Optimization of Hierarchical Decisions in Airline Manpower Planning

Modifying job vacancy announcements using Kriging-based optimization

Master's thesis in Engineering Mathematics and Computational Science

FRIDA ERIKSSON

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2020

Master's thesis 2020

Optimization of Hierarchical Decisions in Airline Manpower Planning

Modifying job vacancy announcements using Kriging-based optimization

FRIDA ERIKSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathecathical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2020

Optimization of Hierarchical Decisions in Airline Manpower Planning
Modifying job vacancy announcements using Kriging-based optimization
FRIDA ERIKSSON

Supervisor: Pontus Ekh, Jeppesen Systems AB

Supervisor and examiner: Ann-Brith Strömberg, Mathematical Sciences, Chalmers University of Technology

Master's Thesis 2020
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Abstract

This thesis work was performed at Jeppesen in Gothenburg, a company which develops and sells optimization software for scheduling to airlines. Among their different products, Manpower Planning has been the focus in this project.

Airlines typically operate between many different airports, with several different aircraft types and a large number of crews. The major part of the manpower planning problem is that of deciding which and how many crews should be positioned at which base airport, operating which aircraft type. The promotion process differs between airlines. Typically, on the North American market, job vacancies are announced, whereafter crews make bids on the different vacancies. The allocation is then carried out according to several rules, such that, e.g., the most senior crews' bids are prioritized. The allocation decisions may lead to deficiencies, meaning that for some fleets the crew demand exceeds the supply. By modifying the announced vacancies and by allowing staffing deviation on certain fleets, such deficiencies can be reduced, as well as the cost of simulator training and other tutoring needed for crews which are promoted. Such modifications are currently done manually.

In this thesis, an optimization algorithm is developed for modifying vacancies and allowing staffing deviations. The problem itself is a black-box optimization problem, meaning that it is computationally expensive to evaluate, and that no analytical derivatives of the objective function exist. To solve such problems, a surrogate model can be built which approximates the true objective. The surrogate model implemented in this thesis is based on so-called Kriging, in which functions are modelled as realizations of Gaussian processes. In addition, due to the problem being high-dimensional, dimensionality reduction techniques are employed.

The optimization algorithm is implemented in Python and communicates with the optimizer used at Jeppesen. Its performance and quality is tested on a benchmark problem as well as on real airline data. The allocation solutions found by the algorithm are associated with lower costs compared to manually constructed reference solutions.

Acknowledgements

During the work with this thesis, many people have supported me in different ways and I would like to thank them all.
First of all, I would like to thank the people at the manpower optimization department at Jeppesen for a few great months at the office, which continued remotely, due to the Corona virus situation. Especially I would like to thank my supervisor Pontus Ekh and his team members for all support and engagement throughout the thesis work.

I would also like to thank Ann-Brith Strömberg, my supervisor and examiner at the Mathematical Sciences department at Chalmers, for providing support in ideas, software licences and giving valuable feedback.

Contents

1 Introduction and problem description 2
- 1.1 Jeppesen . . . 2
- 1.2 Manpower Planning . . . 2
- 1.3 Key performance indicators . . . 4
- 1.4 Problem description . . . 4

2 Theory and background 5
- 2.1 Surrogate modeling . . . 5
  - 2.1.1 Interpolation of scattered data in $\mathbb{R}$ . . . 6
  - 2.1.2 Radial basis function models . . . 7
  - 2.1.3 Kriging . . . 8
- 2.2 Dimensionality reduction . . . 12
  - 2.2.1 Kriging partial least squares (KPLS) . . . 12
  - 2.2.2 Improved parameter estimation (KPLS+K) . . . 15
  - 2.2.3 Dropout . . . 15
- 2.3 Experimental design . . . 16
  - 2.3.1 Random sampling . . . 16
  - 2.3.2 Stratified sampling . . . 17
  - 2.3.3 Latin hypercube design . . . 18
- 2.4 Infill sampling criteria . . . 20
  - 2.4.1 Maximizing probability of improvement . . . 20
  - 2.4.2 Maximizing expected improvement . . . 21
- 2.5 Numerical optimization algorithms . . . 24
  - 2.5.1 COBYLA . . . 24
  - 2.5.2 L-BFGS-B . . . 24

3 Explicit problem formulation 26
- 3.1 A simplified problem example . . . 26
- 3.2 Model formulation . . . 28

4 The optimization algorithm 32
- 4.1 Creating a surrogate model . . . 33
- 4.2 Design of experiments . . . 34
- 4.3 Infill sample criteria . . . 34
- 4.4 Termination criterion . . . 35

5 Implementation 36
- 5.1 General algorithm pseudocode . . . 36
  - 5.1.1 Design of experiments . . . 37
  - 5.1.2 Parameter estimation . . . 38
  - 5.1.3 Infill sampling criteria . . . 38
  - 5.1.4 Termination criterion . . . 39
- 5.2 Implementation into the Jeppesen Manpower Planning framework . . . 39
- 5.3 Problem settings . . . 40

6 Algorithm evaluation 41
- 6.1 Evaluation of the KPLS algorithm using a benchmark function . . . 41
- 6.2 Evaluation of the KPLS+K algorithm using a benchmark function . . . 44

7 Results **45**
7.1 Evaluating the KPLS algorithm on airline data with setting S1 46
7.2 Evaluating the KPLS algorithm on airline data with setting S2 51
7.3 Evaluating the KPLS+K algorithm on airline data with setting S1 55
7.4 Evaluating the KPLS+K algorithm on airline data with setting S2 59
7.5 General observations 63

8 Discussion and future research **67**
8.1 Method selection 68
8.2 Algorithm evaluation 70

9 Conclusions **71**

Symbols and Notation

Symbol	Meaning
det	determinant of a matrix
$\lvert\cdot\rvert$	absolute value
$\mathbb{R}$	set of real numbers
$\mathbb{R}_+$	set of positive real numbers (including 0)
$\mathbb{R}_+^*$	set of positive real numbers (excluding 0)
$(\cdot)_+$	$\max(\cdot, 0)$
$.^{\mathrm{T}}$	transpose operation
n	number of sampling points
d	number of variables
h	number of principal components
$\boldsymbol{x}$	$1 \times d$ vector containing one sample point
x_j	j^{th} element of $\boldsymbol{x}$
$x_{i,j}$	j^{th} element of i^{th} sample point $\boldsymbol{x}_i$
$\boldsymbol{X}$	$n \times d$ matrix of n sample points
$\boldsymbol{y}$	$n \times 1$ vector of sample point evaluations
$y(\boldsymbol{x})$	evaluation of sample point $\boldsymbol{x}$
$\hat{Y}(\boldsymbol{x})$	prediction of $y(\boldsymbol{x})$
$k(\cdot, \cdot)$	covariance function
$\mathcal{N}(0, k(\cdot, \cdot))$	Gaussian distribution of a process with mean 0 and covariance $k(\cdot, \cdot)$
$\boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}}$	$n \times 1$ correlation vector
$\boldsymbol{R}$	$n \times n$ correlation matrix

1 Introduction and problem description

This Master's Thesis was performed at Jeppesen and Mathematical Sciences at Chalmers University of Technology in Gothenburg during the spring of 2020. Jeppesen build solutions for crew scheduling and management problems and is a part of the Boeing Company. Their products consist of both long-term schedules, as well as schedule recovery in case of disruption on the day of operation.

This project focuses on the airline Manpower Planning problem. The goal when solving the airline Manpower Planning problem is to make sure that the crew demand in airline fleets does not exceed the corresponding supply of crews. A major part of finding such solutions is to assign crew transitions between the different airports and different aircraft types for which the airline operates. In this project, much emphasis is put on the crew transitions.

Since airlines typically operate between many different airports, have several different fleets and thousands of crew personnel, the Manpower Planning problem is impossible to solve by an exhaustive search, due to its combinatorial nature. In order to find near-to-optimal solutions, a combination of heuristics and exact methods are implemented within the Jeppesen Manpower Planning framework. The evaluations are costly and the quality of model parameters reflects the solution quality. Generally speaking, this thesis concerns improving certain parameter's quality, by optimizing a substep within the Manpower Planning framework, which is currently performed manually.

The thesis outline is as follows. In this section, Jeppesen is presented, followed by Manpower Planning , key performance indicators and lastly the problem description is formulated. In Section 2, relevant theory and background is presented for the reader. Topics as surrogate modeling, dimensionality reduction, experimental design and infill sampling criteria are presented. In Section 3, the dynamics behind the problem is described, as well as an explicit model formulation of a simplified problem. Thereafter, in Section 4, the optimization algorithm is presented, including the parts from Section 2 which are implemented as a part of the thesis. The implementation of the optimization algorithm then follows in Section 5. This includes pseudocode of the different parts of the algorithm, how implementation into the Jeppesen Manpower Planning framework is carried out, as well as the specific problem settings which are considered. In Section 6, the optimization algorithm is evaluated using a benchmark problem, in order to validate its performance quality. Lastly, in Section 7 and Section 8, numerical results obtained by running the optimization algorithm on the specific problem are presented and discussed, as well as possible future research.

1.1 Jeppesen

Jeppesen in Gothenburg origins from a planning project at Volvo in the mid 1980s. In connection with the aquisition by Boeing in 2006, the current name Jeppesen was established [1]. Jeppesen in Gothenburg is, since then, a part of the Boeing Company and have at this date approximately 350 employees.

Jeppesen develops and sells solutions for crew planning and management problems to airlines. Among their products, optimization software for tail assignment, i.e., the problem of assigning flights to aircrafts, is found. Other products are related to crew pairing, i.e., finding sequences of flight connections which starts and ends at the same base airport, and crew rostering, which means assigning anonymous crew pairings to crew members. However, the product in focus within this project is their optimization software for Manpower Planning, which is described in detail below.

1.2 Manpower Planning

The goal in airline Manpower Planning is to find solutions where the crew demand does not exceed the crew supply within airline fleets. The planning is done long before the day of operation and considers a number of hard and soft constraints, such as government regulations, union rules and crew preferences. Building an optimal crew resource plan includes the decision of how many crews should be allocated to the different aircraft types and the base airports, from which the airline operate. A

major part of finding such solutions is to assign crew transitions between the different airports and the different aircraft types. In this project, we focus on such transitions, which are associated with crew promotions.

In the airline promotion process, crews are typically asked to make bids on where and how they prefer to continue working. The bids include desired base airport, type of aircraft, and rank, where the rank can be either captain or first officer. Each crew should make several bids, since they are not guaranteed to obtain their first choice. When solving the resource planning problem, these bids are taken into consideration to decide where all crews should be positioned, in order to meet a given demand. One thing that makes the problem practically complicated is that crews need to go to several weeks of training in order to advance to higher positions, which means that they will be unavailable (for regular duty) during this period. Moreover, airlines have a strict seniority ranking among their crews, meaning that pilots with higher seniority should have their preferences fulfilled ahead of less senior pilots.

The business process differs in different parts of the world. For most airlines on the European Manpower Planning market, the crew bids are first collected, whereafter the airline announces job vacancies, i.e., which jobs are vacant and need to be filled by a crew. The crews are then allocated to these vacancies according to their bids (and other rules). The corresponding process related to the North American market is more complicated. Since the data used in this project originates from a North American airline, the business process applied to the North American Manpower Planning market is to be described in more detail.

The North American business process can be divided into four parts; see Figure 1. In the first step, vacancies are announced. Since crew preferences are not known in advance, uncertainty will be present in this part of the process. In the second step, the crew bids are collected, which makes the subsequent problems deterministic. As a third step in the process, crews are allocated to vacancies. This problem can be seen as a network flow with dependencies, where a net demand must be fulfilled. However, this problem becomes considerably more complicated than a pure network flow problem, due to the crew seniority ranking. Lastly, the problem of when in time crew training will take place is solved, given the allocations from step three. The problem solved in the last step is the most complex; it includes constraints associated with, e.g., government regulations and union rules. Other factors which make this step complicated is its high dimensionality and the requirements on integrality of its solutions. The input-output response is not at all smooth and can not be represented by an analytical function.

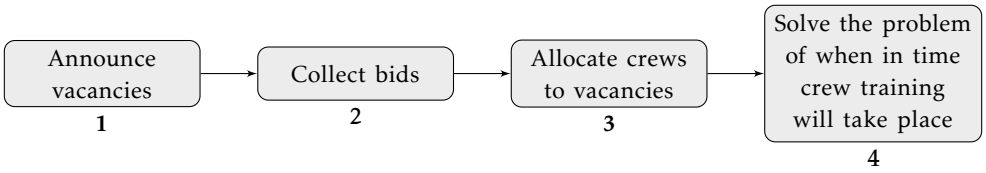


Figure 1: Flowchart of the Manpower Planning process for the North American market.

The solution process applied by Jeppesen is presented in Figure 2. First of all, decisions have to be made on how to announce the vacancies. Typically, historical data is used to generate several different scenarios within the first step of the process. After the bid collection in step two, some modifications of the originally posted vacancies are done, by comparing with the vacancies posted in the first step. Moreover, feedback is sent to the initial step, in order to improve future vacancy announcements. Currently, the modifications between steps two and three are done manually. Such modifications may also be to allow some staffing deviation within different groups, where a group is referred to as a triple; (base airport, aircraft type, rank).

When solving the crew-to-vacancy allocation problems, backfilling among groups is a recurrent phenomenon. In other words, if a vacancy appears in a certain group, it tends to be filled by a person

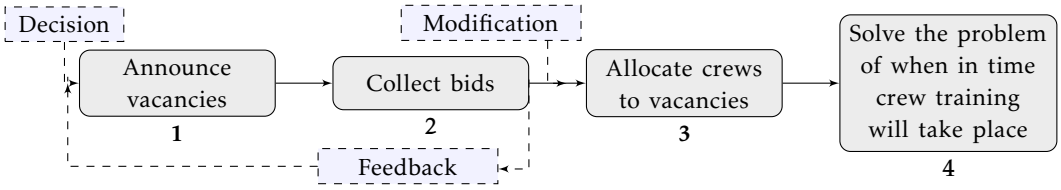


Figure 2: Flowchart for the Jeppesen Manpower Planning solution process.

from a less desirable[1] group. In the worst case scenario, this behavior is propagating into a long chain of group re-distributions, which leads to unnecessary crew training. By allowing some staffing deviation in groups far up in the chain, many group re-distributions and thus crew training can be avoided.

1.3 Key performance indicators

To evaluate solutions of the Manpower Planning problem, so called *Key Performance Indicators* (KPIs) are used. They consist of different performance measurements, based on, e.g., cost and quality. Some of the KPIs that are generated by Jeppesen's optimizer are

- total cost,
- number of courses given, and
- group deficiency costs.

The total cost is the usual minimization aim, and represents the sum of several different costs; for instance, the cost of rosters, stand-ins, courses, and deficiencies. However, we must not restrict ourselves to one of the KPIs. Several of them can be considered simultaneously, which is the idea of multiobjective optimization, in comparison with single objective optimization where only one objective function is to be minimized. In this thesis work, the focus is mainly on single objective optimization, but the method is possible to extend to multiobjective optimization. Nevertheless, all of the KPIs mentioned are utilized in some sense, either as the minimization objective or within the optimization algorithm.

1.4 Problem description

The goal of the thesis project is to optimize the decisions taken between steps 2 and 3 in the Manpower Planning process illustrated in Figure 2. As previously mentioned, such decisions are mainly addressing how much staffing deviation to allow within different groups. The final optimization procedure developed shall be theoretically substantiated and ideally possible to integrate into Jeppesen's optimizer in order to improve the product.

Currently, the problem of allocating crews to vacancies is solved using an iterative process. At each iteration, one crew is reallocated between groups according to the crew's bids, the current vacancies, and the crew seniority ranking. The procedure continues until no crew can alter between groups to obtain a more preferable bid, without making it worse for any other crew with higher seniority. This is one of the rules stated within Jeppesen's Manpower Planning framework. The seniority attribute makes it non-trivial to formulate and solve the problem explicitly, e.g., as a mixed integer linear program (MILP). However, this is investigated and such a formulation is included in the report, to give a deeper understanding of the problem in question. In any case, since the final

[1] In general, larger aircraft types and higher rank are more desirable than smaller aircraft types and lower rank, due to salary differences

evaluation of the model can not be done before the whole Manpower Planning process is completed, a different, black-box based model will be requisite in order to utilize solution feedback.

In general, if an explicit formulation can not be found, or if function evaluations are computationally too expensive, a surrogate model which approximates input-output (I/O) responses can be utilized. To design a surrogate model which approximates the true model properly will be the main objective of this project. This can be broken down into several different issues, mainly the following.

- How to reduce the problem into fewer dimensions. Manpower Planning problems typically involve many variables, which is problematic due to the resulting heavy computations.
- Design of experiment, i.e., how to choose initial sample points for the surrogate model.
- A strategy for choosing new evaluation points.
- How to modify surrogate modelling techniques in order to utilize the properties of this specific problem.

When an adequate model has been developed, it is implemented in Python and its performance will be examined using Jeppesen's real data sets, as well as benchmark functions, in order to justify the model and its performance quality. However, mimicking the behavior of Jeppesen's optimizer is complicated, which means that the benchmark function typically will be much smoother than the true objective and thus may produce too promising results.

2 Theory and background

In this section, relevant theory and background is presented, which is used when constructing the optimization algorithm. The subjects are presented with an emphasis to the mathematics, in order to give insight into the model and method choices made. For the casual reader, this section does not have to be read thoroughly; it can rather be used as a reference for the subsequent sections when required.

In Section 2.1 surrogate modeling is presented, with emphasis on Kriging. Thereafter, dimensionality reduction techniques are described in Section 2.2, followed by experimental design in Section 2.3, infill sampling criteria in Section 2.4 and finally numerical optimization algorithms in Section 2.5.

2.1 Surrogate modeling

A *surrogate model* (also known as approximation model, metamodel or response surface) acts as a model of a model [2]. Surrogate models are advantageous when the objective is to optimize black-box functions (no analytical derivatives) or models which are expensive to evaluate. By interpolating I/O data, the surrogate model approximates the underlying function for inputs not yet evaluated. There are different types of surrogate models; *Radial Basis Functions* and *Kriging* (see [3]) being two widely used methods. References to other types of surrogate models can be found in [4].

There are also non-interpolating methods, which are instead minimizing the sum of squared errors from a predetermined function. An example of a non-interpolating method is the *fitted quadratic surfaces*. However, as discussed in [5], such methods are unreliable since the surface may not capture the function's shape sufficiently. Another problem with non-interpolating methods is that the addition of new sample points will necessarily not lead to a more accurate surface fit. In contrast, interpolating methods always become more accurate as new points are added to the sample, and will eventually converge to the true function. However, interpolation can lead to overfitting, meaning that the model approximated from a limited set of data points becomes overly complex.

2.1.1 Interpolation of scattered data in $\mathbb{R}$

Data interpolation is the problem of constructing new data points within the range of a set of known data points $\boldsymbol{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$. Formally, we seek a function $P \in C^0$ such that

$$P(\boldsymbol{x}_i) = y_i, \qquad i = 1, ..., n, \tag{1}$$

given the points in $\boldsymbol{x}_i \in \boldsymbol{X}$ and the corresponding data $\boldsymbol{y} = (y_1, ..., y_n)$. The interpolation function P is not unique, since C^0 is an infinite dimensional function space. However, for some specific linear finite dimensional spaces, e.g., Π_{n-1} (the space of polynomials of degree less than n), uniqueness holds. For Π_{n-1}, the problem can be stated as finding the coefficients α_i, $i = 1, ..., n$, such that

$$P(\boldsymbol{x}_i) = \sum_{j=1}^{n} \alpha_i x_{i,j-1}, \qquad i = 1, ..., n, \tag{2}$$

which is equivalent to finding the solution $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_n)$ to the linear system

$$\boldsymbol{A}\boldsymbol{\alpha} = \boldsymbol{y}, \tag{3}$$

where

$$A_{ij} = x_{i,j-1}, \qquad i, j = 1, ..., n.$$

If the matrix $\boldsymbol{A}$ is invertible, then the problem (3) (and equivalently (2)) has a unique solution. Moreover, Π_{n-1} is independent of the data information; it depends only on the data quantity. Despite of these properties, the polynomial function space is not very practical for implementations, since the polynomial degree increases in proportion to the number of data points evaluated. This usually leads to highly oscillating interpolations; see Figure 3.

To avoid the oscillations, *splines* can be used. Splines are piecewise polynomials, which are used to interpolate the data between evaluated points. The most common choice is cubic splines (of third degree), but polynomials of any order can be used. The problem in $\mathbb{R}$ is, given one-dimensional points $a < x_1 < ... < x_n < b$ and the function space

$$\mathcal{S}_3(\boldsymbol{X}) = \left\{ S \in C^2((a, b)) : S|_{[x_i, x_{i+1}]} \in \Pi_3(\mathbb{R}),\ i = 1, ..., n-1 \right\}$$

to find

$$S \in \mathcal{S}_3(\boldsymbol{X}): \quad S(x_i) = y_i, \qquad i = 1, ..., n. \tag{4}$$

Unlike Π_{n-1}, $\mathcal{S}_3$ is dependent of the data points $\boldsymbol{X}$. Moreover, a unique solution to (4) can not be found. With n points and hence $n-1$ splines, each with four[2] degrees of freedom, there are $(n-1)+3 = n+2$ degrees of freedom in total, compared to the data which only has n degrees of freedom.

The function space can be modified in order to ensure a unique solution. For example, setting the second derivative to zero at the end points reduces the dimensionality. This approach ensures a unique solution to (4), and is obtained by replacing $\mathcal{S}_3$ with the natural spline space:

$$\mathcal{NS}_3(\boldsymbol{X}) = \left\{ S \in \mathcal{S}_3(\boldsymbol{X}) : S''(x_1) = S''(x_n) = 0 \right\}.$$

However, even though high oscillations are avoided, this approach still has the drawback of being strongly location dependent, in comparison to Π_{n-1}.

A general drawback with the mentioned interpolation methods is that it is difficult to extend them to higher dimensions. In order to do this, we have to reformulate the problems. According to [6], all

[2] Four degrees of freedom at each interval and three degrees of freedom fixed at each of the inner points, since the function and its first and second order derivatives have to be continuous

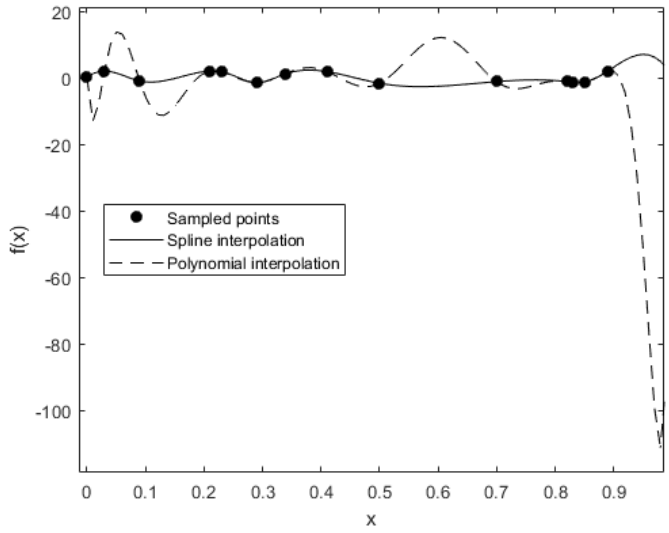


Figure 3: Interpolation of sampled data points using polynomial (dashed) and spline (solid) interpolation.

natural cubic splines $S_{\mathcal{N}}$ can be represented on the form:

$$S_{\mathcal{N}}(x) = \sum_{i=1}^{n} \alpha_i \phi(|x - x_i|) + p(x), \quad x \in \mathbb{R}$$
$$\sum_{i=1}^{n} \alpha_i = \sum_{i=1}^{n} \alpha_i x_i = 0,$$

where $\phi(r) = r^3$, $r \geq 0$, is a basis function and p is a first order polynomial. This motivates us to consider multivariate splines and therefore we are introducing *radial basis functions* (RBF).

2.1.2 Radial basis function models

A *radial function* is a function $f : \mathbb{R}^d \to \mathbb{R}$ such that there exists a univariate function $\phi : \mathbb{R}_+ \to \mathbb{R}$ for which $f(\boldsymbol{x}) = \phi(\|\boldsymbol{x}\|)$, $\boldsymbol{x} \in \mathbb{R}^d$, i.e. the function value depends only on the Euclidean norm. A RBF model is in turn a model constructed by linear superposition of radial functions as basis functions. Radial basis functions are widely used in different applications, such as image processing and discrete data interpolation [2].

The radial basis function model idea (for polynomials of order 0) is as follows. Let $\boldsymbol{x}_i = \{x_{i1}, ..., x_{id}\}$, $i = 1, ..., n$, be n d-dimensional sample points, selected through experimental design, see Section 2.3. Let $y_1, ..., y_n \in \mathbb{R}$ be the corresponding data. We seek a prediction model $f(\boldsymbol{x})$ which interpolates the data $\{(\boldsymbol{x}_i, y_i) : \ i = 1, ..., n\}$, and thus satisfies

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \beta_i \phi(\|\boldsymbol{x} - \boldsymbol{x}_i\|) = \boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{\Phi}, \tag{5}$$

where $\boldsymbol{\beta} = \begin{pmatrix} \beta_1 \ ... \ \beta_n \end{pmatrix}^{\mathrm{T}}$, $\Phi_{ij} = \phi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|)$, the $\beta_i \in \mathbb{R}$ are weight coefficients and ϕ is a radial function of the Euclidean distance between the point $\boldsymbol{x}$ to be evaluated and the sample point $\boldsymbol{x}_i$. Thus, $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^d$. Moreover, according to the interpolation condition (1), the prediction

model must satisfy

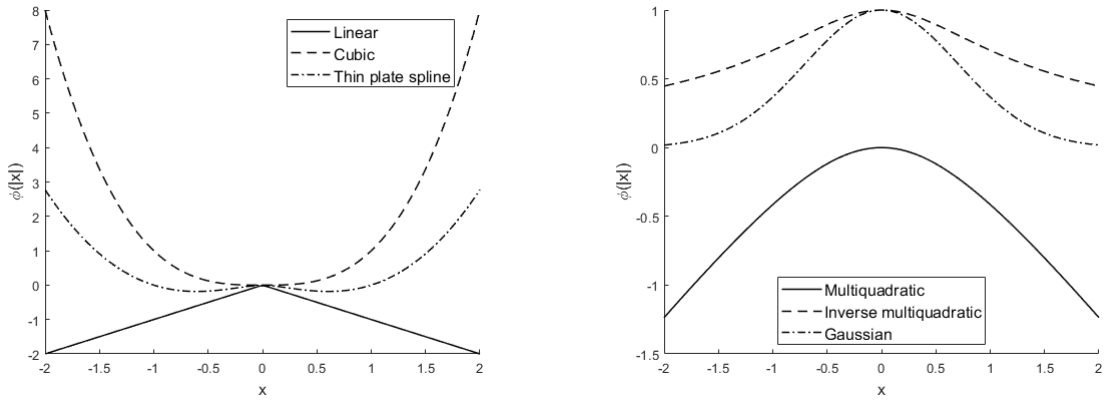
$$f(\boldsymbol{x}_i) = y_i,\ i = 1,...,n, \tag{6}$$

ensuring that predictions are exact at the sampled points $\boldsymbol{x}_i$. To obtain the coefficients β_i, the matrix equation $f = \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{\Phi}$ can be uniquely solved if and only if $\boldsymbol{\Phi}$ is invertible. Since Φ is defined by the radial function ϕ, this have to be chosen in such a way that invertibility is guaranteed. For details on how to uniquely solve (5), as well as how the RBF model is extended to involve higher order polynomials, see [7].

Some of the most commonly used RBFs are presented in Table 1. All of them ensure unique solutions to the interpolation problem. The RBFs are plotted in 1D in Figure 4, but all of them can be used for interpolation of scattered data points in arbitrary dimensions.

Table 1: Common RBFs

Name	$\phi(r)$
Linear	$-r$
Cubic	r^3
Thin plate spline	$r^r \log r$
Multiquadratic	$-\sqrt{r^2+1}$
Inverse multiquadratic	$1/\sqrt{r^2+1}$
Gaussian	$\exp(-r^2)$



(a) Linear, cubic and thin plate spline.

(b) Multiquadratic, inverse multiquadratic and Gaussian.

Figure 4: The RBFs from Table 1.

2.1.3 Kriging

Another classic type of surrogate model which invokes Gaussian processes is the Kriging method, also known as precisely *Gaussian process model* or *Design and Analysis of Computer Experiments* (DACE). The Kriging method was first presented in geostatistics, see [3], but has been extended to computer experiments as well as machine learning.

Kriging is often described as 'modeling the function as a realization of a stochastic process' [5]. However, there is nothing random about the function values of unsampled points, we just do not know them yet. Just like in other surrogate model techniques, e.g., RBF models, a set of d-dimensional sample points $\boldsymbol{x}_i = \{x_{i1},...,x_{id}\}$, $i = 1,...,n$, and corresponding data y_i, $i = 1,...,n$, are

given. Before sampling any new points, there will be an uncertainty about the function values at these points. The uncertainty is modelled by considering the deterministic response $f(\boldsymbol{x})$ as a realization of the stochastic process

$$Y(\boldsymbol{x}) = \sum_{j=1}^{d} \beta_j \varphi_j(\boldsymbol{x}) + Z(\boldsymbol{x}), \tag{7}$$

where, for $j = 1,...,d$, $\beta_j \in \mathbb{R}$ is an unknown parameter, φ_j is a known independent basis function, and $Z(\boldsymbol{x})$ is a normally distributed random variable with $\mathbb{E}[Z(\boldsymbol{x})] = 0$ and a covariance function (also known as kernel function) given by

$$\mathrm{Cov}(Z(\boldsymbol{x}), Z(\boldsymbol{x}')) = k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_z^2 r(\boldsymbol{x}, \boldsymbol{x}') = \sigma_z^2 r_{\boldsymbol{x}\boldsymbol{x}'} \ \forall \boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d,$$

where σ_z^2 denotes the process variance and $r_{\boldsymbol{x}\boldsymbol{x}'}$ denotes the correlation function between points $\boldsymbol{x}$ and $\boldsymbol{x}'$. The correlation function depends on some hyper-parameters, which we will denote by θ and p, and is given by

$$\mathrm{Corr}(\boldsymbol{x}, \boldsymbol{x}') = r_{\boldsymbol{x}\boldsymbol{x}'} = \exp\Big(- \sum_{j=1}^{d} \theta_j |x_j - x_j'|^{p_j} \Big), \tag{8}$$

where it is assumed that $\theta_j \geq 0$ and $0 < p_j \leq 2$, $j = 1,...,d$. Moreover, we denote the $n \times 1$ correlation vector as $\boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}} = [r_{\boldsymbol{x}\boldsymbol{x}_1}, ..., r_{\boldsymbol{x}\boldsymbol{x}_n}]$ and the $n \times n$ correlation matrix as

$$\boldsymbol{R} = [\boldsymbol{r}_{\boldsymbol{x}_1\boldsymbol{X}}, ..., \boldsymbol{r}_{\boldsymbol{x}_n\boldsymbol{X}}]^{\mathrm{T}} = \begin{bmatrix} r_{\boldsymbol{x}_1\boldsymbol{x}_1} & \cdots & r_{\boldsymbol{x}_1\boldsymbol{x}_n} \\ \vdots & \ddots & \vdots \\ r_{\boldsymbol{x}_n\boldsymbol{x}_1} & \cdots & r_{\boldsymbol{x}_n\boldsymbol{x}_n} \end{bmatrix},$$

where the $(\boldsymbol{x}, \boldsymbol{x}')$ entry in $\boldsymbol{R}$ is usually given by (8).

When constructing the Kriging model, the hyper-parameters are considered to be known. The θ_j parameter determines how fast the correlation decreases when moving in the j^{th} coordinate direction. A large value of θ_j means that the function is strongly active in the j^{th} variable and may cause the function value to change rapidly, even when the distance between two points is small. The p_j variable determines the function's smoothness in the j^{th} coordinate direction. While values of p_j close to 2 serve to model smooth functions, values of p_j close to 0 are more suitable when modeling less smooth, non-differentiable functions [5]. With a correlation function given by (8), the covariance kernel $k(\boldsymbol{x}, \boldsymbol{x}')$ is called a Gaussian exponential kernel, which is the most widely used kernel function. Alternatives exist, e.g., the Matérn kernel, see [8].

Derivation of the Kriging predictor

The Kriging predictor is the predictor which minimizes the expected squared prediction error. It is unbiased and modelled as a linear function of the observed data. Thus, the goal of this derivation is to determine the *Best Linear Unbiased Predictor* (BLUP), which is done in accordance with [9]. In this context, unbiasedness means that the expected value of the linear predictor should equal the expected value of (7). A linear predictor of $Y(\boldsymbol{x})$ is on the form $\boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}} \boldsymbol{y}$, where $\boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}$ is a linear weighting vector and $\boldsymbol{y} = (y_1, ..., y_n)^{\mathrm{T}}$ is the sampled data output. Since $\mathbb{E}[Z(\boldsymbol{x})] = 0$, an unbiased linear predictor is such that

$$\mathbb{E}[\boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}} \boldsymbol{y}] = \boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}} \boldsymbol{\varphi} \beta, \tag{9}$$

where $\boldsymbol{\varphi} = [\boldsymbol{\varphi}^{\mathrm{T}}(\boldsymbol{x}_1), ..., \boldsymbol{\varphi}^{\mathrm{T}}(\boldsymbol{x}_n)]^{\mathrm{T}}$, $\boldsymbol{\varphi}(\boldsymbol{x}) = [\varphi_1(\boldsymbol{x}), ..., \varphi_d(\boldsymbol{x})]^{\mathrm{T}}$ and $\beta = [\beta_1, ..., \beta_d]^{\mathrm{T}}$. By a similar reasoning, the expected value of (7) is given by

$$\mathbb{E}[Y(\boldsymbol{x})] = \boldsymbol{\varphi}_{\boldsymbol{x}}^{\mathrm{T}} \beta. \tag{10}$$

Equating (9) and (10) for all β, we obtain a set of d unbiasedness constraints:

$$\boldsymbol{\varphi}^{\mathrm{T}} \boldsymbol{c}_{\boldsymbol{x}} = \boldsymbol{\varphi}_{\boldsymbol{x}}. \tag{11}$$

Now, for any linear predictor $\boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{y}$ of $Y(\boldsymbol{x})$, the mean squared error (MSE) of the prediction $\hat{Y}$ equals

$$\begin{aligned}\mathrm{MSE}[\hat{Y}(\boldsymbol{x})] &= \mathbb{E}\Big[\boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{y} - Y(\boldsymbol{x})\Big]^2 = \mathbb{E}\Big[\boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{y}\boldsymbol{y}^{\mathrm{T}}\boldsymbol{c}_{\boldsymbol{x}} + Y^2(\boldsymbol{x}) - 2\boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{y}Y(\boldsymbol{x})\Big] \\ &= \mathbb{E}\Big[\boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}(\boldsymbol{\varphi}\beta+\boldsymbol{z})(\boldsymbol{\varphi}\beta+\boldsymbol{z})^{\mathrm{T}}\boldsymbol{c}_{\boldsymbol{x}} + (\boldsymbol{\varphi}_{\boldsymbol{x}}^{\mathrm{T}}\beta + Z(\boldsymbol{x}))^2 - 2\boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}(\boldsymbol{\varphi}\beta+\boldsymbol{z})(\boldsymbol{\varphi}_{\boldsymbol{x}}^{\mathrm{T}}\beta + Z(\boldsymbol{x}))\Big] \\ &= (\boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{\varphi}\beta - \boldsymbol{\varphi}_{\boldsymbol{x}}^{\mathrm{T}}\beta)^2 + \boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}\sigma_z^2\boldsymbol{R}\boldsymbol{c}_{\boldsymbol{x}} + \sigma_z^2 - 2\boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}\sigma_z^2\boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}} \\ &= \boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}\sigma_z^2\boldsymbol{R}\boldsymbol{c}_{\boldsymbol{x}} + \sigma_z^2 - 2\boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}\sigma_z^2\boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}} = \sigma_z^2\Big(1 + \boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{c}_{\boldsymbol{x}} - 2\boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}}\Big),\end{aligned} \tag{12}$$

where $\boldsymbol{z} = [Z(\boldsymbol{x}_1), ..., Z(\boldsymbol{x}_n)]^{\mathrm{T}}$ and the other variables are as previously defined. The term $(\boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{\varphi}\beta - \boldsymbol{\varphi}_{\boldsymbol{x}}^{\mathrm{T}}\beta)^2$ in the second last row vanishes because of the unbiasedness constraints (11).

For the constrained minimization of the MSE, we introduce d Lagrange multipliers $\boldsymbol{\lambda}$ for the d unbiasedness constraints in (11) and we obtain the following minimization problem.

$$\begin{aligned}\min_{\boldsymbol{c}_{\boldsymbol{x}}} \quad & \sigma_z^2\Big(1 + \boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{c}_{\boldsymbol{x}} - 2\boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}}\Big) - \boldsymbol{\lambda}\Big(\boldsymbol{\varphi}^{\mathrm{T}}\boldsymbol{c}_{\boldsymbol{x}} - \boldsymbol{\varphi}_{\boldsymbol{x}}\Big) \\ \text{s.t.} \quad & \boldsymbol{\varphi}^{\mathrm{T}}\boldsymbol{c}_{\boldsymbol{x}} - \boldsymbol{\varphi}_{\boldsymbol{x}} = 0.\end{aligned} \tag{13}$$

Taking the partial derivative with respect to $\boldsymbol{c}_{\boldsymbol{x}}$ of the objective function in (13), we have

$$\sigma_z^2\boldsymbol{R}\boldsymbol{c}_{\boldsymbol{x}} - \sigma_z^2\boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}} - \frac{\boldsymbol{\lambda}}{2}\boldsymbol{\varphi} = 0.$$

Since it is permissible to define a new choice of Lagrange multipliers $\boldsymbol{\lambda} := 2\boldsymbol{\lambda}$, the slightly more dense equation

$$\sigma_z^2\boldsymbol{R}\boldsymbol{c}_{\boldsymbol{x}} - \sigma_z^2\boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}} - \boldsymbol{\lambda}\boldsymbol{\varphi} = 0 \tag{14}$$

is used instead. Combining (14) with the set of d unbiasedness constraints in (11) yields the system

$$\begin{bmatrix}\boldsymbol{0} & \boldsymbol{\varphi}^{\mathrm{T}} \\ \boldsymbol{\varphi} & \sigma_z^2\boldsymbol{R}\end{bmatrix}\begin{bmatrix}-\boldsymbol{\lambda} \\ \boldsymbol{c}_{\boldsymbol{x}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{\varphi}_{\boldsymbol{x}} \\ \sigma_z^2\boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}}\end{bmatrix}. \tag{15}$$

By inverting the matrix, the BLUP is obtained as

$$\hat{Y}(\boldsymbol{x}) = \boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{y} = \begin{bmatrix}-\boldsymbol{\lambda}^{\mathrm{T}} & \boldsymbol{c}_{\boldsymbol{x}}^{\mathrm{T}}\end{bmatrix}\begin{bmatrix}\boldsymbol{0} \\ \boldsymbol{y}\end{bmatrix} = \begin{bmatrix}\boldsymbol{\varphi}_{\boldsymbol{x}}^{\mathrm{T}} & \boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}}^{\mathrm{T}}\end{bmatrix}\begin{bmatrix}\boldsymbol{0} & \boldsymbol{\varphi}^{\mathrm{T}} \\ \boldsymbol{\varphi} & \boldsymbol{R}\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{0} \\ \boldsymbol{y}\end{bmatrix}.$$

Equivalently, this can be written as

$$\hat{Y}(\boldsymbol{x}) = \boldsymbol{\varphi}_{\boldsymbol{x}}^{\mathrm{T}}\hat{\beta} + \boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{\varphi}\hat{\beta}), \tag{16}$$

where

$$\hat{\beta} = (\boldsymbol{\varphi}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{\varphi})^{-1}\boldsymbol{\varphi}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{y} \tag{17}$$

is the usual generalized least squares estimator of β. It happens to be the maxmimum likelihood estimate (MLE) as well, which will be derived later on.

In addition, the MSE of the estimate can be expressed by substituting in the matrices from (15) into (12):

$$\mathrm{MSE}[\hat{Y}(\boldsymbol{x})] = \hat{\sigma}^2(\boldsymbol{x}) = \sigma_z^2\left[1 - \begin{bmatrix}\boldsymbol{\varphi}_{\boldsymbol{x}}^{\mathrm{T}} & \boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}}^{\mathrm{T}}\end{bmatrix}\begin{bmatrix}\boldsymbol{0} & \boldsymbol{\varphi}^{\mathrm{T}} \\ \boldsymbol{\varphi} & \boldsymbol{R}\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{\varphi}_{\boldsymbol{x}} \\ \boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}}\end{bmatrix}\right], \tag{18}$$

where $\hat{\sigma}^2$ denotes the Kriging variance, which is different from the z-subscripted process variance. A less complicated formula for the MSE of the predictor can be obtained by considering $\boldsymbol{y}$ and $Y(\boldsymbol{x})$ together. Assuming that they are jointly normally distributed, i.e.,

$$\begin{bmatrix}\boldsymbol{y} \\ Y(\boldsymbol{x})\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\varphi} \\ \boldsymbol{\varphi}_{\boldsymbol{x}}^{\mathrm{T}}\end{bmatrix}\beta, \sigma_z^2\begin{bmatrix}\boldsymbol{R} & \boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}} \\ \boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}}^{\mathrm{T}} & 1\end{bmatrix}\right),$$

we can see $\hat{Y}(\boldsymbol{x})$ as the conditional expectation of $\hat{Y}(\boldsymbol{x}) = Y(\boldsymbol{x})|\boldsymbol{y}$. Thus,

$$\hat{Y}(\boldsymbol{x}) = Y(\boldsymbol{x})|\boldsymbol{y} \sim \mathcal{N}\Big(\mu(\boldsymbol{x}), \sigma^2(\boldsymbol{x})\Big),$$

where

$$\mu(\boldsymbol{x}) = \boldsymbol{\varphi}_{\boldsymbol{x}}^{\mathrm{T}}\beta + \boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{\varphi}\beta), \tag{19}$$

$$\sigma^2(\boldsymbol{x}) = \sigma_z^2(1 - \boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}}). \tag{20}$$

Comparing with previous estimates, equations (16) and (19) are equivalent, while (18) and (20) differ. The reason why they differ is because the estimation of β is ignored in (20), which means that (18) is slightly more accurate. However, both formulas ensure that the measure will be large for predictions far away from sampled data, and become smaller when approaching the sampled data points.

To summarize, the Kriging mean and variance estimates are given by

$$\hat{\mu}(\boldsymbol{x}) = \boldsymbol{\varphi}_{\boldsymbol{x}}^{\mathrm{T}}\beta + \boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{\varphi}\beta)$$

$$\hat{\sigma}^2(\boldsymbol{x}) = \sigma_z^2\left[1 - \begin{bmatrix}\boldsymbol{\varphi}_{\boldsymbol{x}}^{\mathrm{T}} & \boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}}^{\mathrm{T}}\end{bmatrix}\begin{bmatrix}\boldsymbol{0} & \boldsymbol{\varphi}^{\mathrm{T}} \\ \boldsymbol{\varphi} & \boldsymbol{R}\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{\varphi}_{\boldsymbol{x}} \\ \boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}}\end{bmatrix}\right] \approx \sigma_z^2(1 - \boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}}).$$

Maximum likelihood estimation

Yet, the model is not complete since the parameters β, σ_z^2, $\theta = (\theta_1, ..., \theta_d)$, and $p = (p_1, ..., p_d)$ are not specified. Within the Kriging framework, these parameters are usually obtained by applying MLE. Cross Validation (CV) techniques can be used, but in most of the cases studied the MLE variance is lower [10], which motivates the use of this technique. In practice, it is rather the log-likelihood function which is maximized, since it is generally much easier to find its derivative. Since the logarithm is a monotonically increasing function, maximizing the log-likelihood function is equivalent to maximizing the likelihood function.

Given a Gaussian random process, the log-MLE is (up to a constant) given by

$$-\frac{1}{2}\Big(n\ln\sigma_z^2 + \ln\det\boldsymbol{R} + (\boldsymbol{y} - \boldsymbol{\varphi}\beta)^{\mathrm{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{\varphi}\beta)/\sigma_z^2\Big), \tag{21}$$

where $\det(\cdot)$ denotes the determinant. If the hyper-parameters θ and p are assumed to be known, the MLE of β and σ_z^2 are given by

$$\hat{\beta} = (\boldsymbol{\varphi}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{\varphi})^{-1}\boldsymbol{\varphi}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{y}$$

and

$$\hat{\sigma}_z^2 = \frac{1}{n}(\boldsymbol{y} - \boldsymbol{\varphi}\hat{\beta})^{\mathrm{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{\varphi}\hat{\beta})$$

respectively. Observe that the MLE estimate of β is equivalent to its generalized least squares estimate given in (17). Now, substituting the expressions for $\hat{\beta}$ and $\hat{\sigma}_z^2$ back into (21) yields the so-called concentrated likelihood function:

$$-\frac{1}{2}\Big(n\ln\hat{\sigma}_z^2 + \ln\det\boldsymbol{R}\Big). \tag{22}$$

This is an equation of the hyper-parameters θ and p only, which should be chosen properly to maximize the function. Direct MLE is computationally expensive, since it requires the inversion of the $n \times n$ correlation matrix $\boldsymbol{R}$ for each evaluated parameter setting. Thus, numerical optimization techniques are used, with some simplifications. For example, the parameters p_j are often fixed to the value 2 for all $j = 1, ..., d$. Most often the correlation function is robust enough to fit data adequately using only the hyper-parameter θ [11]. Moreover, e.g., [12] suggests that without a significant loss in model fidelity, a simple constant term model can be used, i.e., with $\boldsymbol{\varphi} = \boldsymbol{1}^n$, $\boldsymbol{\varphi} = \boldsymbol{1}^{n\times n}$. This simplification is known as *ordinary Kriging* in geostatistics and yields the predictors

$$\hat{Y}_{\mathrm{OK}}(\boldsymbol{x}) = \hat{\beta} + \boldsymbol{r}_{\boldsymbol{x}\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{1}\hat{\beta}), \tag{23}$$

with

$$\hat{\beta} = (\mathbf{1}^{\mathrm{T}}\boldsymbol{R}^{-1}\mathbf{1})^{-1}\mathbf{1}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{y}$$

and

$$\hat{\sigma}^2_{z,\mathrm{OK}} = \frac{1}{n}(\boldsymbol{y} - \mathbf{1}\hat{\beta})^{\mathrm{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - \mathbf{1}\hat{\beta}).$$

Lastly, for $\boldsymbol{\chi} = (\chi_1, ..., \chi_d)$, denoting $g(\boldsymbol{\chi}) = \exp\Big(-\sum_{j=1}^{d}\theta_j|\chi_j|^{p_j}\Big)$, then the i^{th} element of $\boldsymbol{r}_{\boldsymbol{xX}}$ equals $g(\boldsymbol{x} - \boldsymbol{x}_i)$ and thus it holds that

$$\hat{Y}(x) = \hat{\beta} + \sum_{i=1}^{n}\alpha_i g(\boldsymbol{x} - \boldsymbol{x}_i), \tag{24}$$

where α_i denotes the i^{th} element of $\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{\varphi}\hat{\beta})$. Comparing (24) with (5), we observe that the Kriging predictor has a similar form as the RBF surrogate. However, generally $g(\boldsymbol{v}) \neq g(\|\boldsymbol{v}\|)$ and hence $g(\cdot)$ is not a RBF.

2.2 Dimensionality reduction

High dimensional optimization problems are generally computationally heavy to solve. In order to reduce computation times, dimensionality reduction techniques are introduced. There are numerous of ways to tackle high-dimensionality, since the dimensionality challenge is more or less universal within the fields of science and engineering. Here, the focus will be on how to reduce dimensionality within the development of a Kriging-based surrogate model. In Section 2.2.1, the Kriging partial least squares (KPLS) method is introduced, followed by its extension KPLS+K in Section 2.2.2. Lastly, the dropout method is described in Section 2.2.3.

2.2.1 Kriging partial least squares (KPLS)

The Kriging model is advantageous due to its ability to accurately imitate the behavior of computationally expensive simulations and in addition providing an estimate for the prediction error. However, Kriging has drawbacks which become evident as the number of dimensions increases. As mentioned in Section 2.1.3, as the sample size n increases, the inversion of the $n \times n$ correlation matrix $\boldsymbol{R}$ becomes computationally expensive. As a consequence, the estimation of the hyper-parameters becomes complex, since it requires several inversions of the correlation matrix. To prevent such problems and hence obtain a fast predictor, [13] presents a method which combines Kriging with the technique of Partial Least Squares (PLS) which is referred to as the *Kriging Partial Least Squares* (KPLS) method.

Partial least squares

Partial Least Squares is a technique for predicting trends in data. It is particularly useful when the original data set consists of highly collinear variables and when the sample size is small. PLS finds a linear relationship between I/O variables, by forming a new space spanned by so called *principal components* (PC) and projecting the input variables onto this new space. The PCs are constructed as linear combinations of the input variables. Below, a brief description of the PLS method follows. For more details, see e.g., [14]. Another popular technique for dimensionality reduction is principal component analysis (PCA), which exposes dependencies among the input variables. Since surrogate modeling considers I/O responses, PLS is chosen rather than PCA, as PLS in contrast to PCA considers I/O relationships.

The idea of the PLS method is to find the multidimensional direction in the input space which best explains the characteristics of the output. First, the $n \times d$ dimensional initial sample matrix $\boldsymbol{X}$ and the corresponding data $\boldsymbol{y}$ are centered and scaled. Thereafter, the principal components are constructed.

Let h denote the number of PCs retained[3], with $h \ll d$. The PCs are constructed sequentially. The l^{th} PC, denoted as $\boldsymbol{t}^{(l)}$, is obtained by finding the direction $\boldsymbol{w}^{(l)}$ which maximizes the squared covariance between $\boldsymbol{t}^{(l)} = \boldsymbol{X}^{(l-1)}\boldsymbol{w}^{(l)}$ and $\boldsymbol{y}^{(l-1)}$, i.e.,

$$\boldsymbol{w}^{(l)} = \begin{cases} \arg\max\limits_{\boldsymbol{w}^{(l)}} \; \boldsymbol{w}^{(l)\mathrm{T}}\boldsymbol{X}^{(l-1)\mathrm{T}}\boldsymbol{y}^{(l-1)}\boldsymbol{y}^{(l-1)\mathrm{T}}\boldsymbol{X}^{(l-1)}\boldsymbol{w}^{(l)} \\ \text{s.t.} \;\; \boldsymbol{w}^{(l)\mathrm{T}}\boldsymbol{w}^{(l)} = 1, \end{cases} \tag{25}$$

for $l = 1,...,h$. The maximum is found when $w^{(l)}$ is the eigenvector of the matrix $\boldsymbol{X}^{(l-1)\mathrm{T}}\boldsymbol{y}^{(l-1)}\boldsymbol{y}^{(l-1)\mathrm{T}}\boldsymbol{X}^{(l-1)}$ corresponding to the largest eigenvalue (in absolute terms). The vector $w^{(l)}$ thus contains the l^{th} principal component's weights. To find an estimate of the solution to the problem (25), the power iteration method introduced by [16] can be used.

To obtain the first PC, we use $\boldsymbol{X}^{(0)} = \boldsymbol{X}$ and $\boldsymbol{y}^{(0)} = \boldsymbol{y}$. The matrices $\boldsymbol{X}^{(l)}$ and $\boldsymbol{y}^{(l)}$ are called residual matrices from the regression of $\boldsymbol{X}^{(l-1)}$ and $\boldsymbol{y}^{(l-1)}$ onto $\boldsymbol{t}_l$, respectively, and are given by

$$\begin{aligned} \boldsymbol{X}^{(l)} &= \boldsymbol{X}^{(l-1)} - \boldsymbol{t}^{(l)}\boldsymbol{p}^{(l)}, \\ \boldsymbol{y}^{(l)} &= \boldsymbol{y}^{(l-1)} - \boldsymbol{t}^{(l)}c_l, \end{aligned} \tag{26}$$

where

$$\begin{aligned} \boldsymbol{p}^{(l)} &= (\boldsymbol{t}^{(l)\mathrm{T}}\boldsymbol{t}^{(l)})^{-1}\boldsymbol{t}^{(l)\mathrm{T}}\boldsymbol{X}^{(l-1)} \in \mathbb{R}^{1\times d} \;\; \forall l = 1,...,h, \\ c_l &= (\boldsymbol{t}^{(l)\mathrm{T}}\boldsymbol{t}^{(l)})^{-1}\boldsymbol{t}^{(l)\mathrm{T}}\boldsymbol{y}^{(l-1)} \in \mathbb{R} \;\; \forall l = 1,...,h, \end{aligned} \tag{27}$$

contain the regression coefficients. An illustration of how the method works in a case with $d = 3$ and $h = 2$ is given in Figure 5.

Now, by rotating the original coordinate system with axes $x_1,...,x_d$, the principal components form a new coordinate system [14], which can be written as

$$\boldsymbol{t}^{(l)} = \boldsymbol{X}^{(l-1)}\boldsymbol{w}^{(l)} = \boldsymbol{X}\boldsymbol{w}_*^{(l)}, \quad l = 1,...,h \tag{28}$$

In accordance with [17], we introduce the new weights $\boldsymbol{w}_*^{(l)}$ as the entries of the weight matrix $\boldsymbol{W}_*$, given by

$$\boldsymbol{W}_* = \boldsymbol{W}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{W})^{-1},$$

where $\boldsymbol{W} = (\boldsymbol{w}^{(1)},...,\boldsymbol{w}^{(h)})$ and $\boldsymbol{P} = (\boldsymbol{p}^{(1)\mathrm{T}},...,\boldsymbol{p}^{(h)\mathrm{T}})$. When $h = d$, $\boldsymbol{W}_*$ rotates the original coordinate system $(x_1,...,x_d)$ into the new system $(t^{(1)},...,t^{(d)})$, following the principal directions $\boldsymbol{w}^{(1)},...,\boldsymbol{w}^{(d)}$. As a last step of the KPLS approach, the principal components in (28) are utilized to estimate the hyper-parameters of the correlation function $\boldsymbol{R}$.

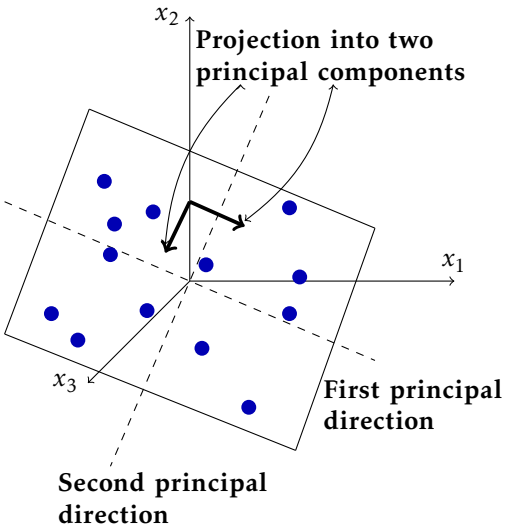
Construction of the KPLS model

In the original Kriging model, the hyper-parameters to be estimated are $\theta = (\theta_1,...,\theta_d)$ and $p = (p_1,...,p_d)$. Within the KPLS framework, p_j is fixed to 2 for all $j = 1,...,d$. As discussed in Section 2.1.3, this is a common simplification, which assumes the function to be smooth in all coordinate directions. The parameters θ are measuring how strongly each of the input variables $x_1,...,x_d$ are affecting the output y. The weight coefficients $\boldsymbol{w}_*^{(l)}$, $l = 1,...,h$, can in turn be interpreted as measuring the influence of the input variables on the output. In the KPLS model, the number of hyper-parameters θ is equal to the number of principal components. Since $h \ll d$, the estimation of the new hyper-parameters $\theta_1,...,\theta_h$ takes considerably less time than estimating the original d-dimensional parameter vector θ.

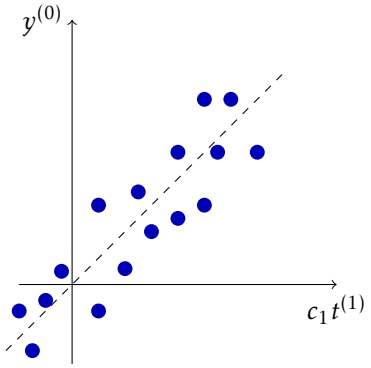
To construct the KPLS model, we first define a kernel $k_1^{\text{KPLS}} : B \times B \to \mathbb{R}$ given by $k_1(\varphi_1(\boldsymbol{x}), \varphi_1(\boldsymbol{x}'))$, where $k_1 : B \times B \to \mathbb{R}$ is an isotropic[4] stationary kernel, $\varphi_1 : B \to B$ is the matrix of basis functions,

[3] in practice, h does not generally exceed 4 [15]

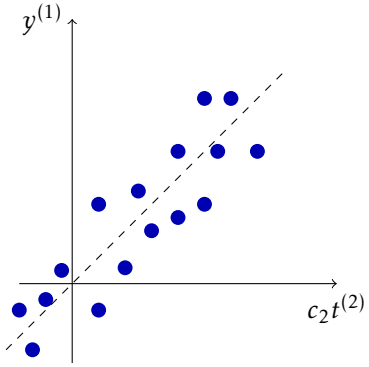
[4] the same hyper-parameter is used for all directions



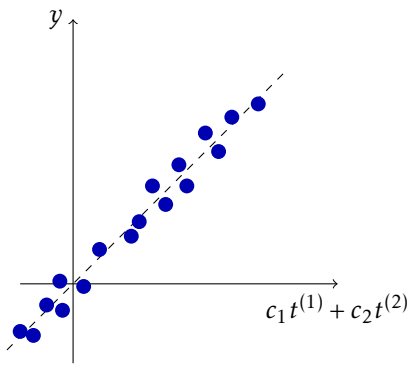
(a) Construction of two principal components in $\boldsymbol{X} \subseteq \mathbb{R}^3$ space



(b) Prediction of $y^{(0)}$



(c) Prediction of $y^{(1)}$



(d) Final prediction of y

Figure 5: A 3D example where two principal components are constructed, inspired by [13].

$\boldsymbol{x} \mapsto \left[w_{*1}^{(1)} x_1, ..., w_{*d}^{(1)} x_d\right]^{\mathrm{T}}$ and B is a hypercube inluded in $\mathbb{R}^d$. The weights $w_{*1}^{(1)}, ..., w_{*d}^{(1)}$ can be interpreted as measuring the importance of the corresponding variables $x_1, ..., x_d$ for constructing the first principal component $\boldsymbol{t}^{(1)}$. However, the information in $\boldsymbol{w}_*^{(1)}$ is generally insufficient, and is therefore supplemented by the information in the remaining PCs $\boldsymbol{t}^{(2)}, ..., \boldsymbol{t}^{(h)}$. To incorporate this information, a new kernel is constructed sequentially using the tensor product of the kernels k_l^{KPLS}, $l = 1, ..., h$:

$$k_{1:h}^{\mathrm{KPLS}}(\boldsymbol{x}, \boldsymbol{x}') = \prod_{l=1}^{h} k_l^{\mathrm{KPLS}}(\varphi_l(\boldsymbol{x}), \varphi_l(\boldsymbol{x}')).$$

The $k_{1:h}^{\mathrm{KPLS}}$ kernel thus accounts for all PC information in one single covariance kernel.

Using the Gaussian exponential kernel corresponding to (8) with $p \equiv 2$, the KPLS kernel becomes

$$
\begin{aligned}
k_{1:h}^{\mathrm{KPLS}}(\boldsymbol{x}, \boldsymbol{x}') &= \sigma_z^2 \prod_{l=1}^{h} \prod_{j=1}^{d} \exp\Big(-\theta_l \Big(w_{*j}^{(l)} x_j - w_{*j}^{(l)} x_j' \Big)^2 \Big) \\
&= \sigma_z^2 \exp\Big(\sum_{l=1}^{h} \sum_{j=1}^{d} -\theta_l w_{*j}^{(l)2} \Big(x_j - x_j' \Big)^2 \Big) \\
&= \sigma_z^2 \exp\Big(\sum_{j=1}^{d} -\eta_j \Big(x_j - x_j' \Big)^2 \Big) \\
&= \sigma_z^2 \prod_{j=1}^{d} \exp\Big(-\eta_j \Big(x_j - x_j' \Big)^2 \Big),
\end{aligned}
$$

where $\eta_j = \sum_{l=1}^{h} \theta_l w_{*j}^{(l)2} \ \forall j = 1, ..., d$. Since the KPLS kernel and hence the corresponding correlation matrix involves far less hyper-parameters, the MLE computation time will decrease significantly. However, the improved computation time might be at the cost of a less accurate estimation.

2.2.2 Improved parameter estimation (KPLS+K)

To compromise between computation time and accuracy, [18] suggests an additional step, where the complete hyper-parameter vector $\theta = (\theta_1, ..., \theta_d)$ is estimated using η_j, $j = 1, ..., d$, as a starting point. This approach is referred to as KPLS+K (KPLS to initialize the Kriging hyper-parameters). After the θ_l-MLE, for $l = 1, ..., h$, a local optimization of the log-likelihood function (22) is performed in the complete $\mathbb{R}^d$ space, with $\eta = \{\eta_j\} \in \mathbb{R}^d$ used as the start vector. This step allows to correct the estimation in many of the directions by optimizing η, although with a slight increase in computational time compared to the KPLS model.

2.2.3 Dropout

Dropout is a dimensionality reduction technique applied to Bayesian optimization[5] proposed by [19]. The idea is inspired by the dropout algorithm in neural networks and is fairly simple. At each iteration, only a subset of magnitude $d' < d$ of the input variables is optimized. The rest of the $d - d'$ variables are chosen according to an alternative strategy; either using random numbers, by retrieving the result of the best solution found, or a combination of these two methods.

Let $I_{d'}$ denote the d' indices corresponding to the variables $\boldsymbol{x}^{d'} = \boldsymbol{x}^{I_{d'}}$ which are to be optimized and equivalently let $I_{d-d'}$ be the indices of the left-out $d - d'$ dimensions, with variables $\boldsymbol{x}^{d-d'} = \boldsymbol{x}^{I_{d-d'}}$. Clearly, $I_{d'} \cup I_{d-d'} = \{1, ..., d\}$ and $I_{d-d'} \cap I_d = 0$. We will denote the set of observations as $\boldsymbol{x}_{1:t} = [\boldsymbol{x}_{1:t}^{d'}, \boldsymbol{x}_{1:t}^{d-d'}]$, where at each iteration t, $\boldsymbol{x}_t^{d'}$ is optimized using a desired algorithm, while $\boldsymbol{x}_t^{d-d'}$ is selected according to one of the following "fill-in" strategies:

Dropout-Random

The variables are randomly generated from a uniform distribution within the domain:

$$
\boldsymbol{x}_t^{d'-d} \sim \mathcal{U}(\boldsymbol{x}^{d'-d}). \tag{29}
$$

[5] yet another synonym for black-box optimization

Dropout-Copy

The values corresponding to the best solution found so far is copied:

$$\begin{aligned} \boldsymbol{x}_t^+ &= \arg\max_{t' \leq t} f(\boldsymbol{x}_{t'}) \\ \boldsymbol{x}_t^{d-d'} &= (\boldsymbol{x}_t^+)^{d-d'}, \end{aligned} \tag{30}$$

where $f(\boldsymbol{x}_{t'}) = y_{t'}$, $t' = 1,...,t$ is the previously evaluated data.

Dropout-Mix

A mixture of Dropout-Random and Dropout-Copy. With probability p, $\boldsymbol{x}_t^{d-d'}$ is obtained from (29) and consequently from (30) with probability $1-p$.

2.3 Experimental design

Experimental design or *design of experiments* (DoE) is the process of constructing an initial sample to provide to the surrogate model. The initial sample consists of a number of sample points $\boldsymbol{x}_i = (x_{i1},...,x_{id})$, $i = 1,...,n$, and the data $\boldsymbol{y} = (y_1,...,y_n)$ obtained by evaluating the sample points. There are many different DoE techniques; see, e.g., [2]. Here the methods of random sampling, stratified sampling and Latin hypercube design (LHD) are presented, with focus on LHD.

Considering DoE methods designed for computer experiments, the primary target is to obtain a space filling sample, in order to provide as much information as possible of the domain under investigation. Further on, this general domain is denoted by Ω. Secondly, the experimental design should be *non-collapsing*. In case of the black-box function value being (almost) non-influenced by a certain design parameter, two potential sample points which differ only in the direction corresponding to this parameter will "collapse", meaning that they can be seen as the same sample point evaluated twice. This situation is not desirable for deterministic black-box functions, and thus two DoE sample points should not share any coordinate values [20].

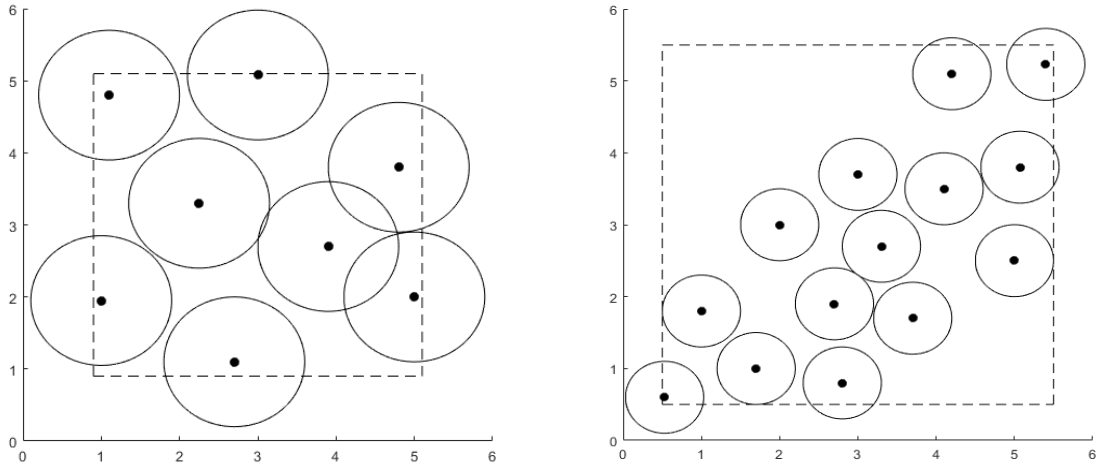
2.3.1 Random sampling

A simple strategy for constructing an initial sample $\boldsymbol{x}_i = (x_{i1},...,x_{id})$, $i = 1,...,n$, is to pick n points randomly from a uniform probability distribution in Ω. This approach is however unable to ensure the space-fillingness criterion, which emphasizes the need for a method to keep the sampled points apart.

A suggestion of such a method is to pick points randomly from Ω and add them to the initial sample $\boldsymbol{X}$ if they are at least a certain distance δ away from any other point in $\boldsymbol{X}$. Equivalently, in d dimensions, they should not lie within a d-dimensional hyper-sphere of radius δ centered around the points in $\boldsymbol{X}$. The performance of this method depends on the choices of δ and n. If they are too large, it might not be possible to create such a sample, due to overlapping hyper-spheres. In Figure 6 this method is illustrated in two dimensions, with δ being 0.9 and 0.5, respectively. In order to obtain a well spread sample, the radius δ should be estimated depending on n and Ω. To estimate δ, a relationship between the volume of the hyper-spheres and the volume of the domain Ω is required. The volume of a d-dimensional hyper-sphere is

$$V_d(\delta) = \frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)}\delta^d = \begin{cases} \frac{\pi^{d/2}}{(\frac{d}{2})!}\delta^d, & d \text{ even}, \\ 2^{(d+1)/2}\frac{\pi^{(d-1)/2}}{d!!}\delta^d, & d \text{ odd}, \end{cases} \tag{31}$$

where $d!! = \prod_{i=0}^{\lceil \frac{d}{2} \rceil - 1}(d-2i)$ denotes the double factorial. To ensure that points are well spread out, the radius δ and thus the volume $V_d(\delta)$ must increase as d increases. Considering (31), it is clear



(a) $\delta = 0.9$ (b) $\delta = 0.5$

Figure 6: Random sampling using different radii δ.

that δ must increase as d increases, since the fraction coefficients are vanishing as $d \to \infty$ due to the factorials in the denominators. The main drawback of the random sampling method is the fact that the radius δ has to be estimated, which can be computationally demanding in higher dimensions.

2.3.2 Stratified sampling

With the stratified sampling method, the sample space Ω is divided into I disjoint subspaces S_i called *strata*. The size of a strata is given by $p_i = \mathbb{P}\{\boldsymbol{X} \in S_i\}$, with $\sum_{i=1}^{I} p_i = 1$. In this way, all subareas of Ω are represented by input variables [21]. The case when $I = 1$ corresponds to random sampling in the whole sampling space Ω.

In each of the stratus, we obtain a random sample X_{ij}, $j = 1,...,\nu_i$, from S_i, where ν_i is the sample size of stratus $i \in \{1,...,I\}$. Given the probability density $f(\boldsymbol{x})$, we have

$$X_{ij} \overset{iid}{\sim} \begin{cases} f(\boldsymbol{x})/p_i, j = 1,...,\nu_i, & \boldsymbol{x} \in S_i, \\ 0, & \text{elsewhere.} \end{cases} \tag{32}$$

As shown in [21], stratified sampling (with proportional allocation, to be described in short) improves over random sampling. To see this, we first introduce some notation. Let T denote the class of estimators, given by

$$T(u_1,...,u_n) = \frac{1}{n}\sum_{i=1}^{n} g(u_i),$$

where $g(\cdot)$ is an arbitrary function. The choice $g(u) = u^m$ is used to estimate the m^{th} sample moment[6]. Further, let τ denote the expected value of T obtained using a random sample design of size n, and let T_{R} be the estimate of τ. The estimate obtained via stratified sampling is denoted T_{S}, with the first

[6] first moment: mean, second moment: variance

two moments given by

$$\mu_i = \mathbb{E}[g(Y_{ij})] = \int_{S_i} g(\boldsymbol{y})(f(\boldsymbol{x})/p_i)\mathrm{d}\boldsymbol{x},$$
$$\sigma_i^2 = \mathrm{Var}(g(Y_{ij})) = \int_{S_i} (g(\boldsymbol{y}) - \mu_i)^2 (f(\boldsymbol{x})/p_i)\mathrm{d}\boldsymbol{x},$$

where $Y = h(\boldsymbol{X})$ is an unknown, but observable univariate transformation of $\boldsymbol{X}$. Considering the general form

$$T_{\mathrm{S}} = \sum_{i=1}^{I} (p_i/\nu_i) \sum_{j=1}^{\nu_i} g(Y_{ij})$$

it is easy to see that T_{S} is an unbiased estimator of τ, with variance

$$\mathrm{Var}(T_{\mathrm{S}}) = \sum_{i=1}^{I} (p_i^2/\nu_i)\sigma_i^2. \tag{33}$$

From [22] we obtain the following results. If we choose p_i and ν_i such that $\nu_i = p_i n$, $\forall i = 1,...,I$, we achieve a so called proportional allocation. Then, equation (33) can be written as

$$\mathrm{Var}(T_{\mathrm{S}}) = \mathrm{Var}(T_{\mathrm{R}}) - \frac{1}{n}\sum_{i=1}^{I} p_i(\mu_i - \tau)^2,$$

which shows that $\mathrm{Var}(T_{\mathrm{S}}) \leq \mathrm{Var}(T_{\mathrm{R}})$ and hence stratified sampling with proportional allocation improves upon random sampling.

2.3.3 Latin hypercube design

Latin Hypercube design (LHD) is a widely used sampling method, first introduced in [21]. Compared to random sampling and stratified sampling, it is found to be more accurate in estimating mean, variance, and distribution of function outputs. Except for fulfilling the criteria of being non-collapsing, it has the advantage of being computationally cheap to generate. Space-fillingness is not guaranteed by LHD itself, but procedures for finding a well spread sample using LHD exist and will be described in short. Moreover, LHD can be seen as the extension of Latin square sampling (see, e.g., [23]) into d dimensions. Just like stratified sampling, LHD ensures that all areas of the domain Ω are represented in the initial sample, but in addition all input variables also have all portions of their distribution represented.

To obtain a LHD sample, the range of each of the d input variables are divided into n intervals of equal marginal probability, where n denotes the desired initial sample size. The numbers n and d are independent, i.e., a higher dimension does not require more sample points, which is one of the main advantages of the LHD sampling method. In each of the n intervals (which can be different for different variables), one observation is made using random sampling.

To exemplify, assuming a uniform probability distribution and setting $d = 1$, $n = 10$ and $\Omega = [0, k]$, the first data point should be selected randomly from the interval $[0, k/10]$, the second from $[k/10, 2k/10]$ and so on until 10 sample points are obtained. Extending this to higher dimensions, we simply follow the 1D approach in each of the d dimensions, yielding d n-tuples $\boldsymbol{x}_j = (x_{1j},...,x_{nj})$, $j = 1,...,d$, or equivalently the matrix

$$M = \begin{bmatrix} x_{11} & \dots & x_{1d} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nd} \end{bmatrix}.$$

To obtain the LHD sample, each column in the matrix is randomly perturbed. This means that each new column is obtained as a permutation of the sequence $\{1,...,n\}$, without replacement. After the pertubation, n new d-tuples $\boldsymbol{x}_i' = (x_{i1}', ..., x_{id}')$, $i = 1, ..., n$ are obtained. Thus our LHD sample matrix is

$$M' = \begin{bmatrix} x_{11}' & \dots & x_{1d}' \\ \vdots & \ddots & \vdots \\ x_{n1}' & \dots & x_{nd}' \end{bmatrix},$$

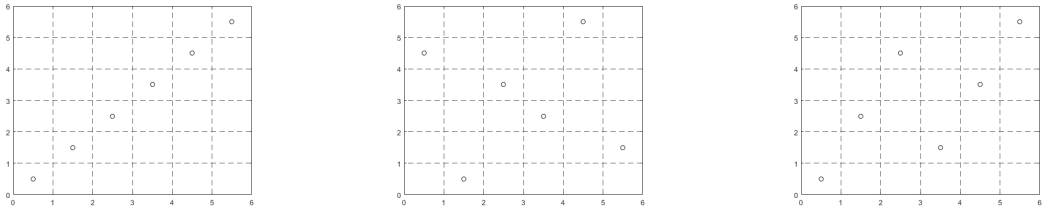
where each row in M' represents one sample point.

An example of a conversion from M to M' in two dimensions with $\Omega = [0, 9]^2$ and $n = 5$ is:

$$M = \begin{bmatrix} 1 & 0 \\ 2 & 3 \\ 5 & 4 \\ 6 & 7 \\ 9 & 8 \end{bmatrix} \longrightarrow M' = \begin{bmatrix} 6 & 0 \\ 5 & 4 \\ 1 & 8 \\ 9 & 3 \\ 2 & 7 \end{bmatrix}.$$

In this example the entries are integral for ease, but that is not the case in practice.

Generating the LHD sample using the procedure just described may result in points which are not well spread out, if the correlation ρ between sample points is too high. In Figure 7a the worst case scenario $\rho = 1$ is plotted. In order to get a more well spread sample, e.g., [24] proposes a method which minimizes the correlation between the sample points. In Figure 7b a LHD with small correlation ($\rho = -0.03$) obtained using the method in [24] is plotted; it is more scattered in the space compared to the LHD with correlation $\rho = 1$.



(a) LHD with $\rho = 1$

(b) LHD with minimized correlations, $\rho = -0.03$

(c) Maximin LHD

Figure 7: Different LHDs, all with $d = 2$ and $n = 6$.

However, the two middle points are still close. To avoid this, the so called maximin LHD method (see [20]) can be applied, which maximizes the minimum distance between the n sample points. In Figure 7c a maximin LHD is plotted[7]. The corresponding correlation $\rho = 0.2$, which is not very small. This suggests that there could be a trade-off between small correlation and maximum minimal distance between points, which turns out to be true. In [25] it is shown that there is no one-to-one relationship between the two objectives, and that LHD samples obtained by these two criteria can differ quite much. This motivates us to consider a multi-objective approach, which both minimizes correlation between points and maximizes the minimum distance between them. However, restricting ourselves to only minimizing correlations provides results which are good enough, especially if more points are to be sampled later on.

The number of possible LHD samples equals $(n!)^d$ and thus becomes huge as n and d increase. Due to this, the problem of finding the optimal LHD becomes very difficult. Several algorithms for finding an optimal LHD are proposed in the literature, such as, e.g., simulated annealing ([26]) and columnwise-pairwise algorithms ([27]).

[7] found at www.spacefillingdesigns.nl

2.4 Infill sampling criteria

After having specified the surrogate model and experimental design, a method for finding new sample points should be selected, which is done by numerical optimization of an auxiliary function. In geostatistics literature, this criterion is called the *infill sampling criterion* (ISC). This approach is advantageous for problems with computationally expensive function evaluations, since all information available is used in order to determine where to evaluate the function next. Obviously, if function evaluations are expensive, the number of function evaluations should ideally be few. There are other methods for determining new sample points, such as genetic algorithms and gradient-based algorithms. Both of them are computationally inexpensive, but require many function evaluations for converging to a good solution. Therefore, ISC is the best suited method for modeling expensive black-box functions.

The ISC can be divided into two different categories — one-stage and two-stage methods, where the latter is the more common one. Considering two-stage methods, the surrogate model is first fitted to the data by estimating the relevant parameters. When the parameters have been estimated they are considered as true, whereafter the surrogate model is used in order to find new points to evaluate. One-stage methods, on the other hand, do not create a surrogate model explicitly, but rather "merge" the two stages. For example, response surface models can be used to set up and evaluate a hypothesis of where the optimum is located. To determine the credibility of the hypothesis that $(\boldsymbol{x}^*, f(\boldsymbol{x}^*))$ constitutes an optimum, the properties of the best-fitting response surface which passes through $(\boldsymbol{x}^*, f(\boldsymbol{x}^*))$ and the observed data can be examined. Intuitively, the smoother the response surface is, the more probable is the hypothesis, but it depends on the problem application [5].

Below follows several different ISCs, all of them being two-stage methods. Even though two-stage methods are widely used, the property of considering the estimated parameter values as true is deceiving and may lead to a response surface which is unable to model the underlying function adequately. This may happen if the initial sample is too small, which can give rise to undesired consequences, such as premature convergence or a too local search. A good ISC should balance local and global search (also denoted as exploitation and exploration, respectively), so that information in the surrogate model is utilized, and simultaneously not leaving any part of the domain completely unexplored.

2.4.1 Maximizing probability of improvement

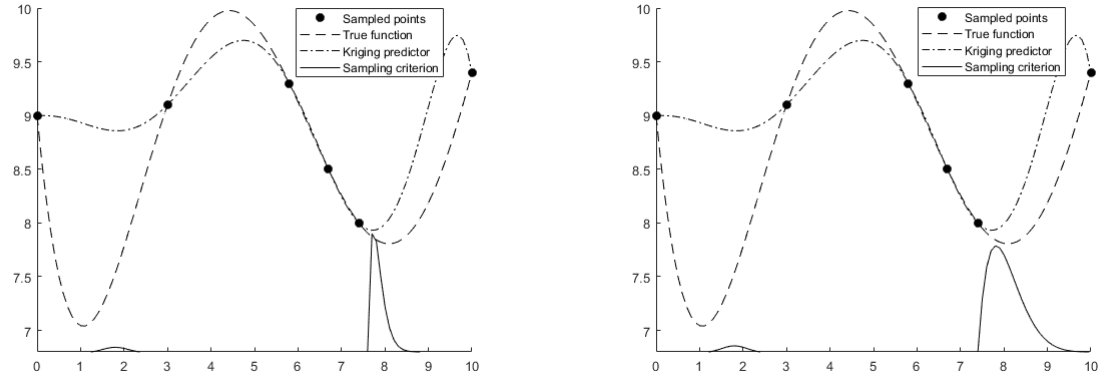
Some of the most popular infill sampling criteria are based on the idea of finding the points with highest probability of improving the function beyond some target T. Given the Normally distributed Kriging predictor $Y(\boldsymbol{x})$ with mean $\hat{Y}(\boldsymbol{x})$ and standard deviation $\hat{\sigma}(\boldsymbol{x})$, the probability of improving the best function value $f_{\min}$ at least to a number $T < f_{\min}$ is simply the probability that $Y(\boldsymbol{x}) \leq T$, given by

$$\mathbb{P}\{Y(\boldsymbol{x}) \leq T\} = \Phi\left(\frac{T - \hat{Y}(\boldsymbol{x})}{\hat{\sigma}(\boldsymbol{x})}\right), \tag{34}$$

where $\Phi(\cdot)$ denotes the standard Normal cumulative distribution function. The idea was introduced in [28] (in one dimension), and have been widely extended in the literature.

The target T can either be chosen by the user or set to the default value of $0.999 f_{\min}$ [11]. The closer T is to $f_{\min}$, the more local is the search. As T decreases, the numerator of the argument in (34) decreases, which tends to produce less sharp peaks of the probability metric in promising areas. This trend is significant in Figure 8, where the probability of improvement criterion is used with different values of T.

The criterion in (34) can be used directly, but this method is found to be extremely sensitive to the value of the target T. A too high target leads to an excessively global search, which means that the algorithm will fine-tune promising results very slowly. On the other hand, a small target may lead to a local search which searches exhaustively nearby the current best point and does not consider unexplored regions as desired. To handle this, [5] proposes two options; an enhanced method which



(a) $T = 99.9\%$ of $f_{\min}$ (b) $T = 99\%$ of $f_{\min}$

Figure 8: Finding new points to evaluate using the probability of improvement criterion with different targets.

uses several targets simultaneously and the idea of maximizing the *expected improvement* (EI).

Enhanced probability of improvement

By selecting several values of the target T, we are able to simultaneously search locally and globally. The procedure of selecting a finite set of target values should be rather arbitrary. The approach proposed in [5] starts by finding the minimum $s_{\min}$ of the surrogate model, as well as the minimum and maximum function value of the sampled points, called $f_{\min}$ and $f_{\max}$, respectively. The targets are thereafter constructed as

$$T = s_{\min} - \alpha(f_{\max} - f_{\min}),$$

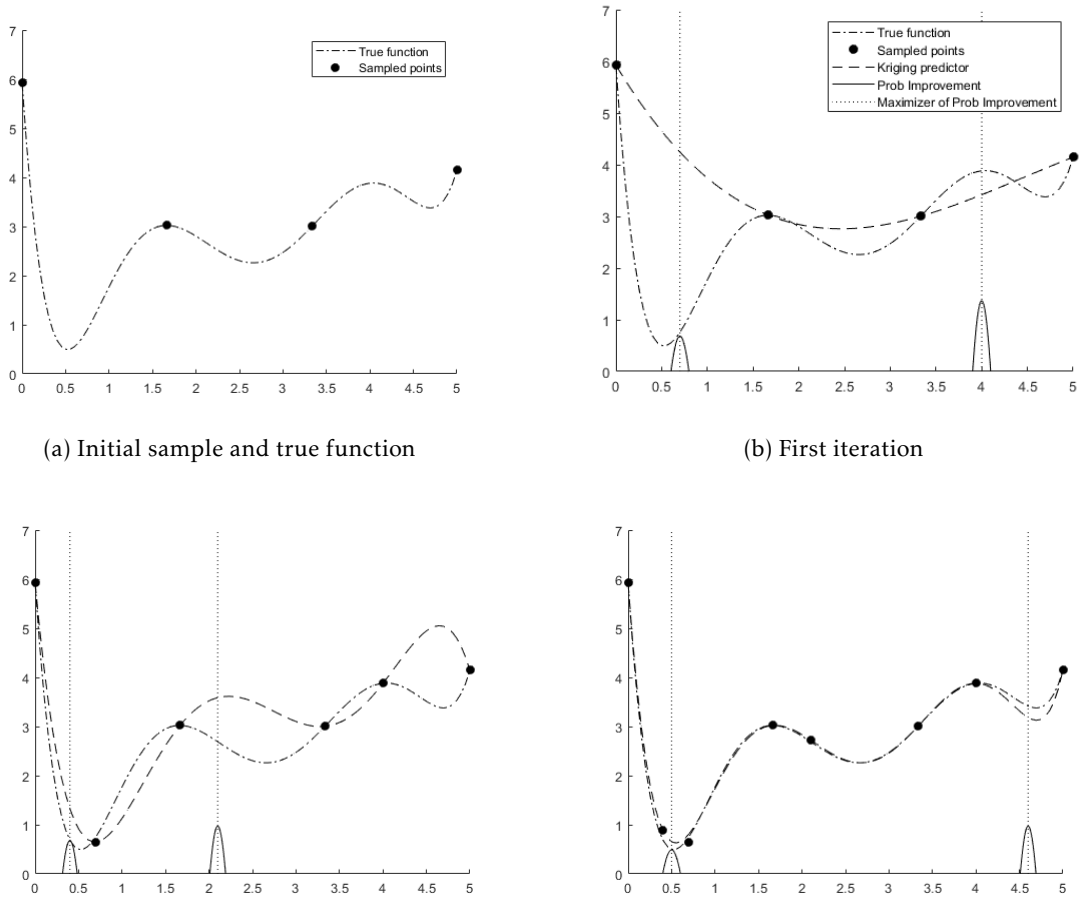
where $\alpha \in [0, 3]$. The case $\alpha = 0$ corresponds to finding the point which minimizes the response surface.

When a number of different values of α are used, the points $\boldsymbol{x}$ tend to cluster in different parts of the solution space. Solutions corresponding to small values of α tend to cluster around the minimum of the response surface, while points corresponding to larger values of α cluster in an unexplored area. In practice, it is reasonable to sample only one point from each of the clusters, as a suggestion the point associated with the smallest target value. An illustration of how this method proceeds is found in Figure 9.

Since the method involves both local and global search, the area where the global optimum is located can be found early. Moreover, by sampling several points per iteration, it is possible to speed up the process, if parallel computations are possible. These are the main reasons why the enhanced probability of improvement approach is considered as a promising one.

2.4.2 Maximizing expected improvement

The idea of maximizing the expected improvement was first suggested by [29] and is another way of balancing exploitation and exploration. Complete exploitation simply minimizes the surrogate model, while complete exploration maximizes the uncertainty, which is measured as the standard error of the predictor. A balance is obtained by considering function improvement. Seeing the function value at a point $\boldsymbol{x}$ as the realization of a random variable $Y(\boldsymbol{x})$, there is a positive probability of improving the current best function value $f_{\min}$ if the tail of the density function of $Y(\boldsymbol{x})$ extends below the line $y = f_{\min}$; see Figure 10.



(a) Initial sample and true function

(b) First iteration

(c) Second iteration

(d) Third iteration

Figure 9: Illustration of the enhanced probability of improvement method.

Different distances below the $y = f_{\min}$ line correspond to different density values (i.e., probabilities), and by weighting all the possible improvements by their respective density values, the so called expected improvement is obtained. The improvement at a point $\boldsymbol{x}$ is given by $(f_{\min} - Y(\boldsymbol{x}))_+$, which is a random variable. Taking expectations, we obtain the expected improvement as

$$\mathrm{EI}(\boldsymbol{x}) \equiv \mathbb{E}[(f_{\min} - Y(\boldsymbol{x}))_+] = \int_{-\infty}^{f_{\min}} (f_{\min} - Y(\boldsymbol{x})) \frac{1}{\sqrt{2\pi}\hat{\sigma}(\boldsymbol{x})} \exp\left(- \frac{\hat{Y}(\boldsymbol{x})^2}{2\hat{\sigma}(\boldsymbol{x})^2} \right) \mathrm{d}Y. \tag{35}$$

By applying integration by parts to (35), we obtain the closed form

$$\mathrm{EI}(\boldsymbol{x}) = \begin{cases} (f_{\min} - \hat{Y}(\boldsymbol{x}))\Phi\left(\frac{f_{\min} - \hat{Y}(\boldsymbol{x})}{\hat{\sigma}(x)}\right) + \hat{\sigma}(\boldsymbol{x})\phi\left(\frac{f_{\min} - \hat{Y}(\boldsymbol{x})}{\hat{\sigma}(\boldsymbol{x})}\right), & \text{if } \hat{\sigma} > 0, \\ 0, & \text{if } \hat{\sigma} = 0, \end{cases} \tag{36}$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ denote the standard Normal cumulative distribution and probability density function, respectively. As usual, $\hat{Y}$ is the Kriging estimator, and $\hat{\sigma}$ denotes its standard error. Naturally, the expected improvement at the sampled points equals zero, since they are known and therefore treated as deterministic.

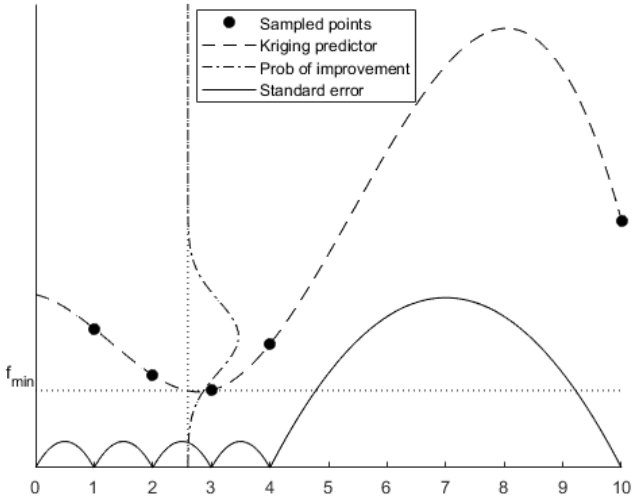


Figure 10: Probability of improvement at a 1D point.

By inspecting (36), two important trends can be revealed. The first term represents the difference between the best function value found so far and the prediction, multiplied by the probability that $\hat{Y} < f_{\min}$. This term becomes large for points where $\hat{Y}$ is likely to be smaller than $f_{\min}$. On the other hand, the second term is increasing with increasing $\hat{\sigma}$, thus yielding large values when the uncertainty about whether or not $\hat{Y}$ improves upon $f_{\min}$ is high. Hence, the expected improvement is large both for regions with high uncertainty and high probability of improvement. This can also be observed by taking the partial derivatives of $\mathrm{EI}(x)$ with respect to $\hat{Y}(x)$:

$$\frac{\partial \mathrm{EI}(\boldsymbol{x})}{\partial \hat{Y}(\boldsymbol{x})} = -\Phi\left(\frac{f_{\min} - \hat{Y}(\boldsymbol{x})}{\hat{\sigma}(\boldsymbol{x})}\right) < 0,$$

and $\hat{\sigma}(\boldsymbol{x})$:

$$\frac{\partial \mathrm{EI}(\boldsymbol{x})}{\partial \hat{\sigma}(\boldsymbol{x})} = \phi\left(\frac{f_{\min} - \hat{Y}(\boldsymbol{x})}{\hat{\sigma}(\boldsymbol{x})}\right) > 0,$$

showing that $\mathrm{EI}(\boldsymbol{x})$ is monotonically decreasing in $\hat{Y}(\boldsymbol{x})$ and monotonically increasing in $\hat{\sigma}(\boldsymbol{x})$.

The EI criterion is not very efficient for high-dimensional problems, due to the model uncertainty being high using many input variables. The EI criterion emphasizes exploration rather than exploitation, since it is more or less impossible to fill all areas of the domain in high-dimensional problems.

Locating the regional extreme

A criterion which exploits the surrogate model more than the EI criterion and hence is more suited for high-dimensional problems is the so-called Watson and Barnes (WB_2) criterion, presented in [11]. The WB_2 criterion is given by

$$\mathrm{WB}_2(\boldsymbol{x}) = \begin{cases} -\hat{Y}(\boldsymbol{x}) + (f_{\min} - \hat{Y}(\boldsymbol{x}))\Phi\left(\frac{f_{\min} - \hat{Y}(\boldsymbol{x})}{\hat{\sigma}(\boldsymbol{x})}\right) + \hat{\sigma}(\boldsymbol{x})\phi\left(\frac{f_{\min} - \hat{Y}(\boldsymbol{x})}{\hat{\sigma}(\boldsymbol{x})}\right), & \text{if } \hat{\sigma} > 0, \\ -\hat{Y}(\boldsymbol{x}), & \text{if } \hat{\sigma} = 0, \end{cases} \tag{37}$$

which is almost identical to the EI criterion in (36), except for the additional $-\hat{Y}$-term, yielding more focus on exploitation. Moreover, it does not return a zero value at sampled points, giving a smoother behavior which may help in locating the maximum.

2.5 Numerical optimization algorithms

In order to find the optima of a continuous function without known derivative or which is in some sense computationally expensive to optimize explicitly, numerical optimization algorithms can be used. Various such algorithms exists, each with its own specific feature. Below, the algorithms COBYLA and L-BFGS-B are presented, which both incorporate the possibility of handling box constraints, i.e., constraints on the form $a_i \leq x_i \leq b_i$, $\forall i = 1, ..., n$, where x_i denotes the i^{th} variable.

2.5.1 COBYLA

The constrained optimization by linear approximation (COBYLA) algorithm was introduced in [30] and implemented in Fortran. The method is numerical and can be adopted to constrained problems where the objective function derivative is unknown. Formally, it finds the point $\boldsymbol{x} \in \Omega$ which has optimal objective function value $f(\boldsymbol{x})$, given inequality constraints on the form $g(\boldsymbol{x}) \geq 0$, while not knowing the gradient of f. The functions f and g are real valued and can be calculated for each $\boldsymbol{x}$, but there are no assumptions regarding smoothness.

The algorithm is a trust-region method, which, as suggested by its name, employs linear approximations of the objective function and the constraints. At each iteration, the approximate linear problem is solved, yielding a candidate solution. The candidate solution is in turn evaluated using the original objective and constraint functions, which results in a new point within the sampling space. This information is then used for improving the linear approximation in the next iteration of the algorithm. The procedure continues until the solution can not be improved anymore, whereafter the step size is reduced in order to refine the search. When a sufficiently small step size has been reached, the algorithm terminates [18]. The pseudocode of the COBYLA algorithm is presented in Algorithm 1.

Algorithm 1 COBYLA

$\mathcal{S} := \{(x_{\text{start}}, f(x_{\text{start}}))\}$
$\delta :=$ initial step size
while δ **not** sufficiently small **do**
  repeat
    Approximate linear problem $\mathcal{LP}(\mathcal{S})$
    Solve $\mathcal{LP}(\mathcal{S}) \rightarrow x_{\text{candidate}}$
    Evaluate $f(x_{\text{candidate}})$
    $\mathcal{S} := \mathcal{S} \cup \{(x_{\text{candidate}}, f(x_{\text{candidate}}))\}$
  until $f(x_{\text{candidate}})$ **not** improved
  Reduce δ
end while
return $x_{\text{candidate}}$

2.5.2 L-BFGS-B

The Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm solves unconstrained non-linear optimization problems by an iterative approach. It belongs to the family of quasi-Newton methods, which approximate the objective function's Jacobian or Hessian. The BFGS method is widely used in the literature, especially the extension L-BFGS, where L is an abbreviation for *Limited memory*. Rather than storing a dense approximation of the $n \times n$ inverse Hessian matrix, as in BFGS, the L-BFGS algorithm stores just a few vectors which implicitly represent the approximation. Hence, L-BFGS requires less memory than BFGS, implying that it is a more suitable method when high-dimensional data sets are considered. An additional exension to L-BFGS is the so called L-BFGS-B algorithm, which, in addition, can handle box constraints.

Starting from an initial feasible point $\boldsymbol{x}_0$, the algorithm proceeds by finding new, better estimates $\boldsymbol{x}_1, \boldsymbol{x}_2, ...$ The objective function is denoted by f, $g_k := \nabla f(\boldsymbol{x}_k)$, and H_k^0 denotes the initial approximation of the Hessian matrix of $f(\boldsymbol{x})$. In each iteration of the algorithm we discard the oldest information

contained in the matrix, and replace it by new information. In this way, the model of the function will be more appropriate. Therefore, we store the last l updates of the form

$$s_k = x_{k+1} - x_k$$
$$t_k = g_{k+1} - g_k,$$

where x_k is the position at the k:th iteration, $g_k = \nabla f(x_k)$, and all vectors are column vectors. The BFGS formula for updating the Hessian approximation is given by

$$H_{k+1} = H_k + \underbrace{\frac{s_k s_k^{\mathrm{T}}}{t_k^{\mathrm{T}} s_k}\left(\frac{t_k^{\mathrm{T}} H_k t_k}{t_k^{\mathrm{T}}} + 1\right) - \frac{1}{t_k^{\mathrm{T}} s}\left(s_k t_k^{\mathrm{T}} H_k + H_k t_k s_k^{\mathrm{T}}\right)}_{=:U(s_k, t_k, H_k)}$$

and the l:th update is given by

$$H_l = H_0 + U(s_o, t_0, H_0) + \cdots + U(s_{l-1}, t_{l-1}, H_{l-1}),$$

In the next iteration, we want to replace the term $U(s_0, t_0, H_0)$ by one involving s_l and t_l. However, all terms $U(s_1, t_1, H_1), ..., U(s_{l-1}, t_{l-1}, H_{l-1})$ depend on $U(s_o, t_0, H_0)$ and hence if this term is discarded, all the other terms will change as well. To avoid this problem, the inverse BFGS formula (38) can be used:

$$H_{k+1} = V_k^{\mathrm{T}} H_k V_k + \rho_k s_k s_k^{\mathrm{T}}, \tag{38}$$

where $\rho_k = 1/t_k^{\mathrm{T}} s_k$ and $V_k = I - \rho_k t_k s_k^{\mathrm{T}}$.

The Hessian matrix should be a diagonal, positive definite matrix. It can be easily verified that if H_k is positive definite and $t_k^{\mathrm{T}} s_k > 0$, then H_{k+1} is positive definite. For details, see [31]. In Algorithm 2, pseudocode (from [32]) for the L-BFGS-B algorithm is presented.

Algorithm 2 L-BFGS-B

1. Choose $\boldsymbol{x}_0$, H_0, l and $0 < \beta' < 1/2$, $\beta' < \beta < 1$
Set box constraints b_{low} and b_{up}
2. Compute approximate Newton's direction $d_k = -H_k g_k$
Compute update $x_{k+1} = x_k + \alpha_k d_k$, where the step length α satisfies the Wolfe conditions:

$$\begin{cases} f(x_k + \alpha_k d_k) & \leq f(x_k) + \beta' \alpha_k g_k^{\mathrm{T}} d_k \\ g(x_k + \alpha_k d_k)^{\mathrm{T}} d_k & \leq \beta g_k^{\mathrm{T}} d_k \end{cases}$$

and fulfills the box constraints: $b_{low} \leq x_{k+1} \leq b_{up}$
($\alpha = 1$ is always the first try)
3. Let $\hat{l} = \min\{k, l-1\}$
Update H_0 $\hat{l}+1$ times using the pairs $\{s_j, t_j\}_{j=k-\hat{l}}^{k}$:

$$\begin{aligned} H_{k+1} =& \left(V_k^{\mathrm{T}} \cdots V_{k-\hat{l}}^{\mathrm{T}}\right) H_0 (V_{k-\hat{l}} \cdots V_k) \\ &+ \rho_{k-\hat{l}} \left(V_k^{\mathrm{T}} \cdots V_{k-\hat{l}+1}^{\mathrm{T}}\right) s_{k-\hat{l}} s_{k-\hat{l}}^{T} (V_{k-\hat{l}+1} \cdots V_k) \\ &+ \rho_{k-\hat{l}+1} \left(V_k^{\mathrm{T}} \cdots V_{k-\hat{l}+2}^{\mathrm{T}}\right) s_{k-\hat{l}+1} s_{k-\hat{l}+1}^{T} (V_{k-\hat{l}+2} \cdots V_k) \\ &\vdots \\ &+ \rho_k s_k s_k^{\mathrm{T}} \end{aligned}$$

4. Set $k := k + 1$ and go to 2

3 Explicit problem formulation

In order to get a deeper understanding of the problem of allocating crews to vacancies while allowing staffing deviation among groups, an explicit model formulation is presented. However, the problem modeled in this section is a simplification of the original problem. The problem statement of the simplified version is to *optimize staffing deviation among groups in such a way that crew training is minimized*.

One of the main difficulties in constructing an explicit problem formulation is related to the objective function, which is complicated to construct properly, since numerous kinds of costs have to be considered and which take a lot of time to evaluate. For instance, Jeppesen's optimizer takes into account monthly deficiencies, vacations, flight demand, etc., which will not be covered in our simplified model. Hence, in order to solve the problem of modifying vacancies and staffing deviations, an iteration based scheme is necessary, such as the one implemented in Jeppesen's optimizer (step 4 in Figure 2). However, the simplified problem presented in this section is similar in terms of problem dynamics, and is mainly included to improve the reader's comprehension of the problem.

3.1 A simplified problem example

To exemplify, assume that there are two different groups (1,2) and five crews (A,B,C,D,E) to be distributed among the groups. All five crews are initially positioned in any of the groups. Crew A is the most senior one, followed by B, and so on. Each crew is asked to make bids on how to be positioned in the future; in this simple case their preference list can be either $[1,2]$ or $[2,1]$. Consider the initial state in Figure 11. The number inside the circle represents the group, while the number inside the square represents the number of vacancies announced. All crews are placed in their current group, represented by the blocks, and their preference lists are superscripted.

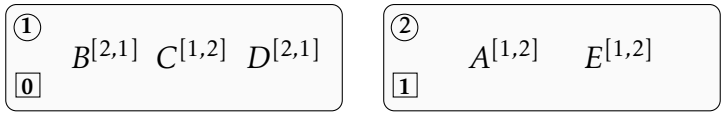


Figure 11: Crew's initial positions in groups.

Crew A is ranked as the most senior crew, and should hence have its preference fulfilled ahead of all the other crews. However, since no vacancy is announced in group 1 where crew A prefers to be, A is not allowed to transfer to group 1. Considering the second most senior crew B, group 2 is preferred rather than group 1. Since there is a vacancy in group 2, B is allowed to transfer to group 2. When crew B moves, a vacancy appears in group 1, while the vacancy in group 2 disappears.

In the next iteration, the most senior crew A is again the first to be considered. Since a vacancy has appeared in group 1, crew A is allowed to switch group. The vacancies are updated and the procedure continues until no crew can move to a more preferred group (which is associated with a positive number of vacancies). The complete process is presented in Figure 12. The number of crew trainings needed (i.e., the number of crews which have changed group) is four.

Now, we investigate whether the number of crew training can be decreased if we are allowed to have some staffing deviation within the groups. Consider the same initial setting as in Figure 11, but with the additional allowed staffing deviation printed in the rightmost squares, see the first iteration in Figure 13. This number indicates how much staffing deviation we allow within the certain group and as long as this lower limit is not reached, the corresponding vacancy value does not increase when crews are leaving the group. Once again, in the first iteration crew B transfers to group 2. However, since we allow one unit of deviation in group 1, the vacancy number is not increased in this group. This leads to a situation where there are no vacancy in any group, and the procedure terminates, resulting in one crew training solely.

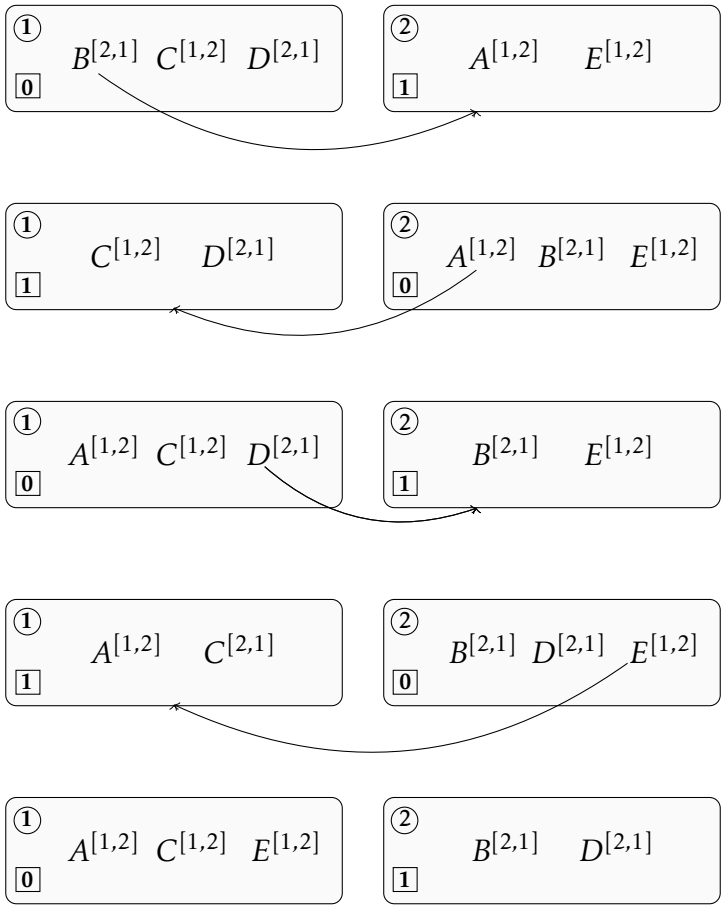


Figure 12: Crew's movements during the complete process. Arrows are representing the movement of a crew from one group to the other.

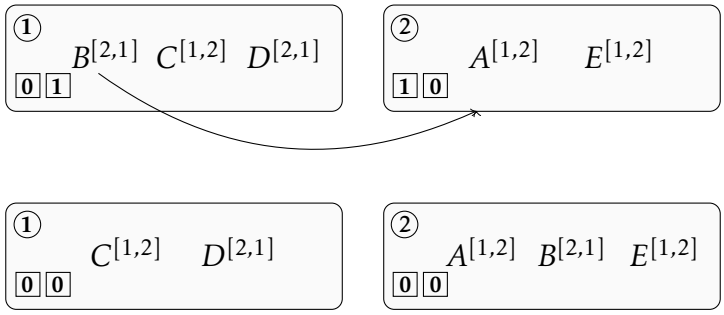


Figure 13: Crew's movements in the case where one unit of staffing deviation is allowed.

In order to get a fair result, the allowed staffing deviation in groups should be penalized, so that they are not implicitly maximized. However, the example presented above is very simple, and hence does not capture all behaviors of a larger problem instance. One of the main reasons for allowing staffing deviation is to avoid so called backfilling, meaning that one crew movement needs to be filled by another crew, which can propagate to a chain of movements. Moreover, the fact that the most senior crew A did not get its most preferred bid in this last example will not happen in general,

when considering larger problem instances.

3.2 Model formulation

In the example presented in Section 3.1, the allowed staffing deviation is fixed. Then, based on that value, the number of crew re-allocations is evaluated. Instead of fixing the allowed staffing deviation within the different groups, we want to optimize it. In the remainder of this section, an explicit optimization model for this problem is built. The resulting model is a MILP (Mixed Integer Linear Program), which can be implemented in a linear optimization software, e.g., AMPL. However, the purpose with this explicit model is not to solve large problem instances, but rather to get a deeper understanding of the problem and its features.

We start by introducing all the sets needed:

- $\mathcal{I} = \{1, ..., n_1\}$ set of crews;
- $\mathcal{J} = \{1, ..., n_2\}$ set of groups;
- $\mathcal{T} = \{1, ..., T\}$ set of iteration indices.

The maximum number of iterations needed is denoted by T, and should be set to a sufficiently large number. Secondly, we introduce the model parameters:

- $p_{i,j} = \begin{cases} 1, & \text{if crew } i \in \mathcal{I} \text{ is initially in group } j \in \mathcal{J}, \\ 0, & \text{otherwise;} \end{cases}$
- $v_j \in \mathbb{Z}_+$ = number of vacancies announced in group $j \in \mathcal{J}$;
- u_j = the binary equivalent of v_j;
- $b_{i,j,k} = \begin{cases} 1, & \text{if crew } i \in \mathcal{I} \text{ prefers group } j \in \mathcal{J} \text{ rather than group } k \in \mathcal{J}, j \neq k, \\ 0, & \text{otherwise;} \end{cases}$
- M_1, M_2, M_3 = large positive constants;
- $\omega \in \{0, 1\}$ = training cost weight.

The large constants M_1, M_2 and M_3 are needed for certain logical constraints, such as if-then-else constraints and constraints for obtaining binary equivalents of integer variables. The purpose of the parameter ω is to weight the training cost term in the objective function, which enables different balancing between the training cost and the allowed staffing deviation.

Next, the model variables are introduced:

- $y_{i,j,s} = \begin{cases} 1, & \text{if crew } i \in \mathcal{I} \text{ is in group } j \in \mathcal{J} \text{ at iteration } s \in \mathcal{T}, \\ 0, & \text{otherwise;} \end{cases}$
- $\bar{y}_{i,j} = |y_{i,j,T} - y_{i,j,0}| = \begin{cases} 1, & \text{if crew } i \in \mathcal{I} \text{ does not start and end in the same group } j \in \mathcal{J}, \\ & \text{i.e., needs training,} \\ 0, & \text{otherwise;} \end{cases}$
- $w_{j,s} \in \mathbb{Z}_+$ = number of vacancies in group $j \in \mathcal{J}$ at iteration $s \in \mathcal{T}$;
- $h_{j,s}$ = the binary equivalent of $w_{j,s}$;
- $d_j \in \mathbb{Z}_+$ = allowed initial staffing deviation in group $j \in \mathcal{J}$;
- $z_{j,s} \in \mathbb{Z}_+$ = allowed staffing deviation in group $j \in \mathcal{J}$ at iteration $s \in \mathcal{T}$;

- $m_{i,j,k,s} = \begin{cases} 1, & \text{if crew } i \in \mathcal{I} \text{ moves to group } j \in \mathcal{J} \text{ from group } k \in \mathcal{J}, \text{ where } j \neq k, \\ & \text{at iteration } s \in \mathcal{T}, \\ 0, & \text{otherwise;} \end{cases}$
- $q_{i,j,k,s} = \begin{cases} 1, & \text{if crew } i \in \mathcal{I} \text{ prefers group } j \in \mathcal{J} \text{ rather than group } k \in \mathcal{J}, \text{ where } j \neq k, \text{ and has} \\ & \text{not moved to another, more preferred group at iteration } s \in \mathcal{T}, \\ 0, & \text{otherwise;} \end{cases}$
- $r_{j,s} \in \{0,1\}$ = binary variables for if-else constraints,

where the sought variables are d_j, $j \in \mathcal{J}$. In order to obtain a linear problem, the absolute values $|y_{i,j,T} - y_{i,j,0}|$, $i \in \mathcal{I}$, $j \in \mathcal{J}$, can not be used explicitly. Instead, the variables $\bar{y}_{i,j}$, $i \in \mathcal{I}$, $j \in \mathcal{J}$, are used, which possess the absolute value properties.

Since the variables $\bar{y}_{i,j}$, $i \in \mathcal{I}$, $j \in \mathcal{J}$, indicate whether or not crews need training, they should be included in the objective function, as well as the initial allowed staffing deviation. Summing the $\bar{y}_{i,j}$ variables, the training is counted twice[8], and hence this term is halved. The two terms are balanced by the weight ω, completing the minimization statement:

$$\min \quad \frac{\omega}{2} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} \bar{y}_{i,j} - \sum_{j \in \mathcal{J}} d_j.$$

In order for the variables $\bar{y}_{i,j}$, $i \in \mathcal{I}$, $j \in \mathcal{J}$ to have the absolute value properties, the constraints (C1) and (C2) are included in the model:

$$\bar{y}_{i,j} \geq y_{i,j,T} - y_{i,j,0} \qquad \forall i \in \mathcal{I}, j \in \mathcal{J} \tag{C1}$$
$$\bar{y}_{i,j} \geq -(y_{i,j,T} - y_{i,j,0}) \qquad \forall i \in \mathcal{I}, j \in \mathcal{J} \tag{C2}$$

Next, the constraints (C3)–(C6) are incorporated, to initialize positions, vacancies and allowed staffing deviation:

$$y_{i,j,0} = p_{i,j} \qquad \forall i \in \mathcal{I}, j \in \mathcal{J} \tag{C3}$$
$$w_{j,0} = v_j \qquad \forall j \in \mathcal{J} \tag{C4}$$
$$h_{j,0} = u_j \qquad \forall j \in \mathcal{J} \tag{C5}$$
$$z_{j,0} = d_j \qquad \forall j \in \mathcal{J} \tag{C6}$$

Moreover, considering the initial allowed staffing deviation in particular, two additional set of constraints are needed:

$$z_{j,s} \leq d_j, \qquad \forall j \in \mathcal{J}, s \in \mathcal{T}; \tag{C7}$$
$$d_j \leq M_1(1 - u_j), \qquad \forall j \in \mathcal{J}. \tag{C8}$$

The constraints (C7) make sure that the staffing deviation never reaches above the initial allowed staffing deviation, while the constraints (C8) ensures that staffing deviation is allowed only in groups where no vacancy is announced. In fact, the constraints (C8) are optional, but we decide to keep them in this version of the problem. The number M_1 can be chosen arbitrary, and represents the maximum allowed staffing deviation in any group. To obtain the binary equivalents $h_{j,s}$ of $w_{j,s}$, the constraints (C9)–(C10) are added to the model:

$$h_{j,s} \leq w_{j,s}, \qquad \forall j \in \mathcal{J}, s \in \mathcal{T}; \tag{C9}$$
$$M_2 h_{j,s} \geq w_{j,s}, \qquad \forall j \in \mathcal{J}, s \in \mathcal{T}. \tag{C10}$$

[8] e.g., if crew $i \in \mathcal{I}$ moves from group $j \in \mathcal{J}$ to group $k \in \mathcal{J}, j \neq k$, then both $\bar{y}_{i,j} = 1$ and $\bar{y}_{i,k} = 1$

The large number M_2 should be at least $\max_j\{v_j\} + n_1$.

Next, several different types of constraints should be added to model the crew movements. First of all, no crew is allowed to make a move in the initial state, which is ensured by the constraints

$$m_{i,j,k,0} = 0, \quad \forall i \in \mathcal{I}, j \in \mathcal{J}, k \in \mathcal{J}. \tag{C11}$$

Moreover, no more than one movement is allowed at each iteration and each crew is only allowed to be in one group at each iteration. These attributes are presented by the constraints (C12) and (C13), respectively:

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{J}\setminus\{j\}} m_{i,j,k,s} \le 1 \qquad \forall s \in \mathcal{T}\setminus\{0\} \tag{C12}$$

$$\sum_{j\in\mathcal{J}} y_{i,j,s} = 1 \qquad \forall i \in \mathcal{I}, s \in \mathcal{T}. \tag{C13}$$

The constraint corresponding to $s = 0$ in (C12) is omitted, since it is redundant due to the constraints (C11). To obtain the desired features of the movement variables m, the constraints (C14) are used:

$$m_{i,j,k,s+1} \ge y_{i,j,s+1} + y_{i,k,s} - 1, \quad \forall i \in \mathcal{I}, j,k \in \mathcal{J} : j \ne k, s \in \mathcal{T}\setminus\{T\}. \tag{C14}$$

These constraints guarantee that crew $i \in \mathcal{I}$ moves to group $j \in \mathcal{J}$ from group $k \in \mathcal{J}, where j \ne k$, at iteration $s+1 \in \mathcal{T}\setminus\{T\}$. In addition, we want to ensure that crew $i \in \mathcal{I}$ moves to group $j \in \mathcal{J}$ from group $k \in \mathcal{J}$ if and only if all of the following four criteria are fulfilled:

- crew i is currently in group k;
- group j has a non-zero number of vacancies;
- none of the more senior crews than i have already made a move at the current iteration;
- crew i prefers group j rather than group k and has not moved to another, more preferred group at the current iteration.

These criteria are ensured by the constraints (C15)-(C17):

$$y_{i,j,s+1} \ge q_{i,j,k,s+1} + h_{j,s} + y_{i,k,s} + \Big(1 - \sum_{\substack{l\in\mathcal{I}:\\ l<i}}\sum_{g_1\in\mathcal{J}}\sum_{g_2\in\mathcal{J}\setminus\{g_1\}} m_{l,g_1,g_2,s+1}\Big) - 3 \tag{C15}$$

$$\forall i \in \mathcal{I}, j,k \in \mathcal{J}, j \ne k, s \in \mathcal{T}\setminus\{T\}$$

$$m_{i,j,k,s+1} \ge q_{i,j,k,s+1} + h_{j,s} + y_{i,k,s} + \Big(1 - \sum_{\substack{l\in\mathcal{I}:\\ l<i}}\sum_{g_1\in\mathcal{J}}\sum_{g_2\in\mathcal{J}\setminus\{g_1\}} m_{l,g_1,g_2,s+1}\Big) - 3 \tag{C16}$$

$$\forall i \in \mathcal{I}, j,k \in \mathcal{J}, j \ne k, s \in \mathcal{T}\setminus\{T\}$$

$$m_{i,j,k,s+1} \le \frac{1}{4}\Big(q_{i,j,k,s+1} + h_{j,s} + y_{i,k,s} + \Big(1 - \sum_{\substack{l\in\mathcal{I}:\\ l<i}}\sum_{g_1\in\mathcal{J}}\sum_{g_2\in\mathcal{J}\setminus\{g_1\}} m_{l,g_1,g_2,s+1}\Big)\Big) \tag{C17}$$

$$\forall i \in \mathcal{I}, j,k \in \mathcal{J}, j \ne k, s \in \mathcal{T}\setminus\{T\}$$

Additionally, we want to obtain the proper behavior of the variables q, which is ensured by the

constraints (C18)–(C19):

$$q_{i,j,k,s} \geq \Big(1 - \sum_{g_1 \in \mathcal{J}: \left(\sum_{g_2 \in \mathcal{J}\setminus\{j,g_1\}} b_{i,g_1,g_2} \geq \sum_{g_2 \in \mathcal{J}\setminus\{j\}} b_{i,j,g_2}\right)} m_{i,g_1,k,s}\Big) + b_{i,j,k} - 1 \quad \forall i \in \mathcal{I}, j,k \in \mathcal{J}, j \neq k, s \in \mathcal{T} \tag{C18}$$

$$q_{i,j,k,s} \leq \frac{1}{2}\left(\Big(1 - \sum_{g_1 \in \mathcal{J}: \left(\sum_{g_2 \in \mathcal{J}\setminus\{j,g_1\}} b_{i,g_1,g_2} \geq \sum_{g_2 \in \mathcal{J}\setminus\{j\}} b_{i,j,g_2}\right)} m_{i,g_1,k,s}\Big) + b_{i,j,k}\right) \quad \forall i \in \mathcal{I}, j,k \in \mathcal{J}, j \neq k, s \in \mathcal{T}. \tag{C19}$$

In this way, $q_{i,j,k,s}$ will equal one if and only if

- crew i prefers group j rather than the current group k, and
- crew i has not moved to another, more preferred group than group j at the current iteration.

One way to see which groups are more preferred than group j, is to add the corresponding preference parameters $b_{i,g,k}$. Crew i's most preferred group will have the largest value of $\sum_{g \in \mathcal{J}\setminus\{k\}} b_{i,g,k}$ (maximum $n_1 - 1$, if k is the least preferred group), and for all groups g_1 which crew i prefers more than j, it will hold that

$$\sum_{g_2 \in \mathcal{J}\setminus\{j,g_1\}} b_{i,g_1,g_2} \geq \sum_{g_2 \in \mathcal{J}\setminus\{j\}} b_{i,j,g_2}.$$

Thus, to see whether or not crew i has moved to a more preferred group than j, we sum all m variables corresponding to such groups g_1. If this sum equals one, it means that i has moved to a more preferred group than j at the current iteration s, and hence i should not move to j as well.

Finally, vacancies and staffing deviations shall be updated as crews move between groups. When a crew enters a group, this group's vacancy number is decreased by one unit. Considering the group which the crew is leaving, the vacancy update depends on the staffing deviation. If there is a non-negative staffing deviation, the vacancy number is not increased. Instead, the staffing deviation is decreased by one unit. However, if the group under consideration does not allow staffing deviation, the vacancy is increased by one unit. To model this, we introduce the binary help variables $r_{j,s}$, $j \in \mathcal{J}$, $s \in \mathcal{T}$, which have the following property:

$$r_{j,s} = 1 \Leftrightarrow z_{j,s} > 0 \quad \forall j \in \mathcal{J}, s \in \mathcal{T}.$$

This can be modelled using the equations (C20)–(C21):

$$z_{j,s} \geq r_{j,s} \qquad \forall j \in \mathcal{J}, s \in \mathcal{T} \tag{C20}$$

$$z_{j,s} \leq M_1 r_{j,s} \qquad \forall j \in \mathcal{J}, s \in \mathcal{T}. \tag{C21}$$

Now, the r variables can be used to model the if-then-else behavior of the vacancy and staffing deviation updates:

$$r_{j,s} = 1 \;\Rightarrow\; \sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{J}\setminus\{j\}} (m_{i,j,k,s+1} - m_{i,k,j,s+1}) = z_{j,s+1} - z_{j,s}, \qquad \forall j \in \mathcal{J}, s \in \mathcal{T} \setminus \{T\}; \tag{39}$$

$$r_{j,s} = 0 \;\Rightarrow\; \sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{J}\setminus\{j\}} (m_{i,j,k,s+1} - m_{i,k,j,s+1}) = w_{j,s} - w_{j,s+1}, \qquad \forall j \in \mathcal{J}, s \in \mathcal{T} \setminus \{T\}. \tag{40}$$

Observe that the sum

$$\sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{J}\setminus\{j\}} (m_{i,j,k,s+1} - m_{i,k,j,s+1}) = \begin{cases} 1, & \text{if any crew moves } \textit{to} \text{ group } j \text{ (at time } s+1), \\ -1, & \text{if any crew moves } \textit{from} \text{ group } j \text{ (at time } s+1), \end{cases}$$

which means that vacancies z and staffing deviations w are updated accordingly. Observe also that the left hand sides of relations (39)–(40) are reversed.

The four last constraints (C22)–(C25) of the model ensure the implications in the relations (39)–(40):

$$z_{j,s+1} - z_{j,s} - M_3(1 - r_{j,s}) \leq \sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{J} \setminus \{j\}} m_{i,j,k,s+1} - m_{i,k,j,s+1}, \qquad \forall j \in \mathcal{J}, s \in \mathcal{T} \setminus \{T\}; \tag{C22}$$

$$\sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{J} \setminus \{j\}} (m_{i,j,k,s+1} - m_{i,k,j,s+1}) \leq z_{j,s+1} - z_{j,s} + M_3(1 - r_{j,s}), \qquad \forall j \in \mathcal{J}, s \in \mathcal{T} \setminus \{T\}; \tag{C23}$$

$$w_{j,s} - w_{j,s+1} - M_3(1 - r_{j,s}) \leq \sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{J} \setminus \{j\}} m_{i,j,k,s+1} - m_{i,k,j,s+1}, \qquad \forall j \in \mathcal{J}, s \in \mathcal{T} \setminus \{T\}; \tag{C24}$$

$$\sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{J} \setminus \{j\}} (m_{i,j,k,s+1} - m_{i,k,j,s+1}) \leq w_{j,s} - w_{j,s+1} + M_3(1 - r_{j,s}), \qquad \forall j \in \mathcal{J}, s \in \mathcal{T} \setminus \{T\}; \tag{C25}$$

this completes the model. Since all variables in the constraints (C22)–(C25) are binary and the sum $\sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{J} \setminus \{j\}} (m_{i,j,k,s+1} - m_{i,k,j,s+1}) \in \{-1, 0, 1\}$ $\forall j \in \mathcal{J}, s \in \mathcal{T} \setminus \{T\}$, it is sufficient to set $M_3 \geq 1$.

4 The optimization algorithm

The black-box optimization algorithm presented in this thesis is a two-stage method based on surrogate modeling, in particular Kriging, described in Section 2.1.3. Dimensionality reduction techniques are utilized, such as Kriging partial least squares and dropout; see Sections 2.2.1 and 2.2.3, respectively. In this section, the different parts of the optimization algorithm are presented and described based on the theory in Section 2. There, different approaches for surrogate modeling, experimental design and infill sampling is described, and here the best suited methods are selected and presented in Sections 4.1, 4.2, and 4.3, respectively, in order to build the algorithm. Lastly, in Section 4.4, the algorithm's termination criterion is declared.

In Figure 14, a flowchart of the two-stage approach is presented. The procedure is as follows. First, an initial sample $\boldsymbol{X}$, including sample points $\boldsymbol{x}$ and their evaluated true function values y, is created by employing a DoE technique. Using the initial sample, a surrogate model is built, by estimating the model parameters. The resulting response surface is then optimized based on given m infill sampling criteria, yielding m new points. These points are then evaluated by invoking the expensive black-box function. If the termination criterion is fulfilled, the algorithm finishes and the point $\boldsymbol{x}_* \in \boldsymbol{X}$ corresponding to the minimum in $\boldsymbol{y}$ is considered as the minimum of the problem. Yet, if the termination criterion is not fulfilled, the newly evaluated points are added to the sample and the algorithm restarts by re-building the surrogate model. The surrogate model is re-built by re-estimating the model parameters. The procedure continues until the termination criterion is fulfilled.

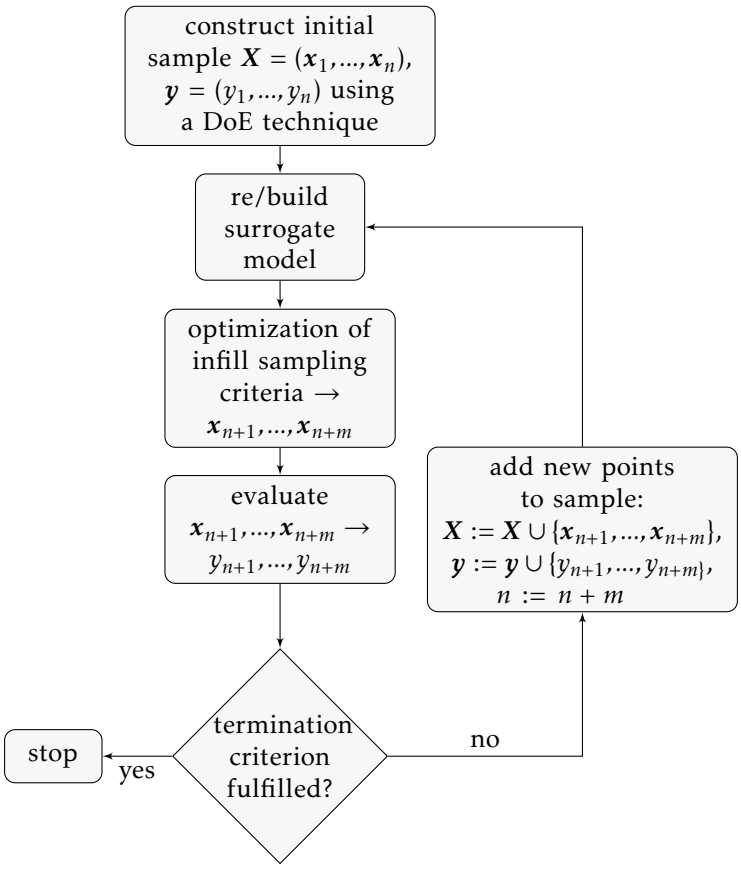


Figure 14: Flowchart of the two-stage optimization algorithm, inspired by [18].

4.1 Creating a surrogate model

As discussed in Section 2.1, interpolating surrogate models are preferred rather than non-interpolating models in most cases, since they are more accurate. This becomes even more evident as the number of sample points increases. The interpolating surrogate model type chosen for this application is Kriging, due to its ability to measure the uncertainty in the predictor itself. Moreover, the fact that the hyper-parameters can be "tuned" using MLE is the main reason why Kriging often outperforms other basis-function methods when it comes to prediction accuracy [5]. However, the parameter estimation is expensive compared to, e.g., radial basis function models, which only requires the solution of a linear system.

Due to the high dimensional nature of the problem, parameter estimations become computationally heavy. Thus, the Kriging-based dimensionality reduction technique KPLS is utilized. The KPLS approach starts off by constructing principal components. To construct the $h \ll d$ principal components, we first have to solve the optimization problem (25). We want to find the eigenvector $\boldsymbol{w}^{(l)}$ of the matrix $\boldsymbol{X}^{(l-1)\mathrm{T}}\boldsymbol{y}^{(l-1)}\boldsymbol{y}^{(l-1)\mathrm{T}}\boldsymbol{X}^{(l-1)}$, $l = 1, ..., h$, corresponding to the dominant eigenvalue, i.e., the eigenvalue with largest absolute value.

Following the power iteration method, we denote $A = \boldsymbol{X}^{(l-1)\mathrm{T}}\boldsymbol{y}^{(l-1)}\boldsymbol{y}^{(l-1)\mathrm{T}}\boldsymbol{X}^{(l-1)}$ and drop the principal component index l for ease, since the procedure is independent of the iteration number. Then we proceed by choosing an initial approximation $\boldsymbol{w}^0 \neq \boldsymbol{0}^d$ of the sought eigenvector $\boldsymbol{w}$. Moreover, $\|\boldsymbol{w}^{(0)}\|_\infty = \max_{0 \leq i \leq d} |w_i^{(0)}| = 1$, i.e., the vector is rescaled so that all its components are within the interval $[-1, 1]$. Note that, here, the superscipt indicates iteration number, not PC index. Now, we form

the successive approximations as:

$$\begin{aligned} \boldsymbol{w}^{(1)} &= A\boldsymbol{w}^{(0)} \\ \boldsymbol{w}^{(2)} = A\boldsymbol{w}^{(1)} = A(A\boldsymbol{w}^{(0)}) &= A^2\boldsymbol{w}^{(0)} \\ &\vdots \\ \boldsymbol{w}^{(k)} = A\boldsymbol{w}^{(k-1)} = A^2\boldsymbol{w}^{(k-2)} = \cdots &= A^k\boldsymbol{w}^{(0)}, \end{aligned}$$

where the vector $\boldsymbol{w}^{(k)}$ yields an approximation for the eigenvector corresponding to the dominating eigenvalue. The approximation will be good for large powers k and with properly rescaled sequences. In order to get h such vectors, the procedure is repeated h times, with A updated according to the iteration number $l = 1,...,h$ and the equations (26)–(27). The initial approximation $\boldsymbol{w}^{(0)}$ can be kept the same for all iterations.

When the vectors $\boldsymbol{w}_*^{(l)}$, $l = 1,...,h$, are constructed, they are used to construct the KPLS kernel function. As suggested by [12], we use ordinary Kriging, i.e., choosing basis function $\boldsymbol{\varphi} = \mathbf{1}^n$, $\boldsymbol{\varphi} = \mathbf{1}^{n\times n}$, without a significant loss in model fidelity. Thus, we use the kernel function

$$k(\boldsymbol{x},\boldsymbol{x}') = \sigma_z^2 \prod_{l=1}^{h}\prod_{j=1}^{d} \exp\Big(-\theta_l\Big(w_{*j}^{(l)}x_j - w_{*j}^{(l)}x_j'\Big)^2\Big) \tag{41}$$

$$= \sigma_z^2 \exp\Bigg(\sum_{j=1}^{d} -\eta_j\Big(x_j - x_j'\Big)^2\Bigg), \tag{42}$$

where $\eta_j = \sum_{l=1}^{h}\theta_l w_{*j}^{(l)2}$. For KPLS, the kernel function in (41) is considered, where the parameters θ_l, $l = 1,...,h$, are to be estimated. The extension KPLS+K is to be examined as well, and for that the kernel function (42) is used. Both the KPLS and the KPLS+K optimization algorithm are to be evaluated and compared in the sense of performance.

To estimate the parameters θ and η, the concentrated likelihood function (22) should be maximized. Such calculations are computationally expensive, especially when the sample size n is large, due to the inversion of the $n \times n$ correlation matrix. In this work, the derivative-free algorithm L-BFGS-B is used to find approximate solutions to the maximization problems. The COBYLA algorithm is discarded, due to poor performance during testing.

4.2 Design of experiments

In order to build a surrogate model an initial sample consisting of points $\boldsymbol{X} = (\boldsymbol{x}_1,...,\boldsymbol{x}_n)$ and corresponding data $\boldsymbol{y} = (y_1,...,y_n)$ needs to be constructed, as indicated in Figure 14. In this work, the Design of Experiment technique Latin Hypercube design, described in Section 2.3.3, is used to create the initial sample. It is chosen because of its good properties, such as guaranteeing non-collapsingness. Moreover, by minimizing correlations, a sufficiently spread set of points is obtained. When using a two-stage method the initial configuration is less important compared to if a one-stage method was to be used, since additional points will be added to $\boldsymbol{X}$. However, a good initial sample is necessary in order for the surrogate model to approximate the black box function adequately. In addition, two points corresponding to $\min_{j=1,...,d}\{\Omega_j\}$ and $\max_{j=1,...,d}\{\Omega_j\}$ are added to the initial sample, providing extra information to the algorithm .

4.3 Infill sample criteria

We can, using the initial sample points $\boldsymbol{X}$ and corresponding data $\boldsymbol{y}$ obtained via Latin Hypercube design, approximate the response surface $f(\boldsymbol{x})$, $\boldsymbol{x} \in \boldsymbol{X}$. By adding new sample points to $\boldsymbol{X}$ which are

well spread, the uncertainty about the model will decrease and the surrogate function will approach the true function shape. However, only focusing on decreasing uncertainty to get a more reliable model will not provide the sought after minima per se. Therefore, we need a trade-off when it comes to space-fillingness and explicit surrogate function minimization when choosing new points to evaluate.

Due to the possibility of parallel computations within the software at hand, several points will be added to the sample at each iteration, in order to decrease the total computation time of the algorithm. Since the evaluation of the black-box function is time consuming, the number of evaluation iterations should be kept low. However, if too many points are added to the sample at each iteration, the sample size will increase rapidly, which increases the MLE and ISC optimization times. Moreover, many of the evaluations will probably be "wasted", since the initial function evaluations are few and thus may provide bad approximations in yet unexplored regions.

To utilize each evaluation iteration to the fullest, we focus both on model exploitation and exploration. Therefore, at each iteration a complete local optimization (minimizing the surrogate model) and a complete global search (maximizing model uncertainty) is performed. Moreover, the expected improvement (EI) criterion (36) and the WB_2 criterion (37), both presented in Section 2.4.2, are maximized to obtain additional sample points, as well as modifications of them. The enhanced probability of improvement method presented in Section 2.4.1 is left out, due to the large number of function evaluations needed to build the clusters. However, the maximizing probability of improvement criterion is used, but with one choice of target only. All infill sampling criteria to be used are found in Table 2, where

$$\mathrm{ISC}(\boldsymbol{x}, A, B, C) = \begin{cases} -A\hat{Y}(\boldsymbol{x}) + B(f_{\min} - \hat{Y}(\boldsymbol{x}))\Phi\left(\frac{f_{\min}-\hat{Y}(\boldsymbol{x})}{\hat{\sigma}(\boldsymbol{x})}\right) + C\hat{\sigma}(\boldsymbol{x})\phi\left(\frac{f_{\min}-\hat{Y}(\boldsymbol{x})}{\hat{\sigma}(\boldsymbol{x})}\right), & \text{if } \hat{\sigma} > 0, \\ -A\hat{Y}(\boldsymbol{x}), & \text{if } \hat{\sigma} = 0. \end{cases}$$

Table 2: Infill sampling criteria used for finding new points to evaluate.

	Name	Infill sampling criterion	Objective
1	LCL	Minimize surrogate model	min $\hat{Y}(\boldsymbol{x})$
2	GBL	Maximize model uncertainty	max $\hat{\sigma}(\boldsymbol{x})$
3	EI	Maximize expected improvement	max $\mathrm{ISC}(\boldsymbol{x}, 0, 1, 1)$
4	WB_2	Maximize WB_2	max $\mathrm{ISC}(\boldsymbol{x}, 1, 1, 1)$
5	$\mathrm{WB}_2^{\mathrm{A}}$	Maximize modified WB_2	max $\mathrm{ISC}(\boldsymbol{x}, 1, 2, 1)$
6	$\mathrm{WB}_2^{\mathrm{B}}$	Maximize modified WB_2	max $\mathrm{ISC}(\boldsymbol{x}, 1, 0, 5)$
7	$\mathrm{WB}_2^{\mathrm{C}}$	Maximize modified WB_2	max $\mathrm{ISC}(\boldsymbol{x}, 1, 5, 0)$
8	POI	Maximize probability of improvement	max $\phi\left(\frac{0.9 f_{\min}-\hat{Y}(\boldsymbol{x})}{\hat{\sigma}(\boldsymbol{x})}\right)$

4.4 Termination criterion

In order to decide when to stop the algorithm, a termination criterion is specified. For this algorithm, there are two main reasons for termination; convergence and attaining a maximum number of iterations. The maximum number of iterations is specified by the user, and should be sufficiently large to make sure that the algorithm does not terminate prematurely.

Considering the convergence criterion, we are looking for a point in time when the progress of the algorithm is very small. Since it is not certain that the algorithm will make progress in each single iteration, it is a bad idea to consider the progress made between the latest two consecutive iterations solely. Instead, the n_T most recent iterations are considered, where $n_T > 2$. Thus, the algorithm will terminate if the progress in the last n_T iterations is sufficiently small. Assuming that m points are

added to the sample in each iteration of the algorithm, resulting in $n+mi$ points in total after iteration i is completed. The progress p_i made in iteration i is calculated as

$$p_i = \frac{\min\limits_{j=1,\dots,m} \{\boldsymbol{y}_{n+m(i-1)+j}\} - \min\limits_{j<n+m(i-1)} \{\boldsymbol{y}_j\}}{\min\limits_{j<n+m(i-1)} \{\boldsymbol{y}_j\}}. \tag{43}$$

Consequently, the algorithm terminates after iteration i if

$$p_k - p_{k-1} > -\epsilon, \ \forall k \in \{i - n_T + 1, \dots, i\},$$

where $0 < \epsilon \ll 1$.

5 Implementation

In this section, the methods presented in Section 4 are used to construct an algorithm for solving the black-box optimization problem. The general algorithm is described in pseudocode in Section 5.1, followed by more detailed descriptions of the algorithms used for creating an initial sample in Section 5.1.1, parameter estimation in Section 5.1.2, to find new points for evaluation in Section 5.1.3, and termination criteria in Section 5.1.4. The pseudocode for the L-BFGS-B algorithm, which is used both for the parameter estimation and the infill sampling criteria, was presented earlier, in Section 2.5.2. In Section 5.2, the implementation into Jeppesen's Manpower Planning framework is described. Lastly, in Section 5.3, the different problem settings to be evaluated are specified.

5.1 General algorithm pseudocode

In Algorithm 3, the pseudocode of the main program `Optimize(modelType)` is presented, where the argument `modelType` can be either `KPLS` or `KPLS+K`, which were introduced in Section 2.2.1 and 2.2.2, respectively.

Algorithm 3 `Optimize(modelType)`

Set number of variables, d
Set variable domain, $\Omega \subseteq \mathbb{Z}^d$
Set initial sample size, n
Create initial sample $(\boldsymbol{X}, \boldsymbol{y}) =$ `CreateInitialSample`(Ω, n, d)
Normalize $(\boldsymbol{X}, \boldsymbol{y}) \rightarrow (\boldsymbol{X}^{\mathcal{N}}, \boldsymbol{y}^{\mathcal{N}})$
Set number of infill sampling criteria, m
Set number of principal components, h
Set initial parameters, $\theta^{(0)} = \theta_1^{(0)}, \dots, \theta_h^{(0)}$
Set termination criterion, $\mathcal{T} =$ `false`
while $\mathcal{T} ==$ `false` **do**
  $\theta =$ `EstimateParameters`$(\theta^{(0)}, \boldsymbol{X}^{\mathcal{N}}, \boldsymbol{y}^{\mathcal{N}}, \texttt{global})$
  $\theta^{(0)} = \theta$
  $\eta^{(0)} = \{\eta_j^{(0)}\}_{j=1,\dots,d} = \sum_{l=1}^{h} \theta_l w_{*j}^{(l)2}$
  if `modeltype ==` `KPLS+K` **then**
    $\eta =$ `EstimateParameters`$(\eta^{(0)}, \boldsymbol{X}^{\mathcal{N}}, \boldsymbol{y}^{\mathcal{N}}, \texttt{local})$
  else
    $\eta = \eta^{(0)}$
  end if
  for $i = 1 : m$ **do**
    $(\boldsymbol{x}_{n+i}^{\mathcal{N}}, \hat{Y}(\boldsymbol{x}_{n+i}^{\mathcal{N}})) =$ `GetNewPoint`$(i, \boldsymbol{X}^{\mathcal{N}}, \boldsymbol{y}^{\mathcal{N}}, \eta, \Omega, d)$
  end for
  $(y_{n+1}, \dots, y_{n+m}) =$ `Evaluate`$(\boldsymbol{x}_{n+1}^{\mathcal{N}}, \dots, \boldsymbol{x}_{n+m}^{\mathcal{N}})$
  $\boldsymbol{X}^{\mathcal{N}} = \boldsymbol{X}^{\mathcal{N}} \cup \{\boldsymbol{x}_{n+1}^{\mathcal{N}}, \dots, \boldsymbol{x}_{n+m}^{\mathcal{N}}\}, \ \ \boldsymbol{y}^{\mathcal{N}} = \boldsymbol{y}^{\mathcal{N}} \cup \{y_{n+1}^{\mathcal{N}}, \dots, y_{n+m}^{\mathcal{N}}\}$
  $n \mathrel{+}= m$
  $\mathcal{T} =$ `ShouldTerminate`$(\boldsymbol{y}^{\mathcal{N}}, n, m)$
end while

Initially, the number of variables is set, as well as a variable domain Ω, which for each variable specifies its allowed (integer valued) range. The initial sample size is chosen and an initial sample is created using the `CreateInitialSample` function, described in Section 5.1.1. The initial sample $\boldsymbol{X}$, consisting of sample points $\boldsymbol{x}$, and the corresponding data $\boldsymbol{y}$ are then normalized. The number of infill sampling criteria m and principal components h are set, as well as an initial parameter vector θ, to be provided for the parameter estimation procedure. The termination criterion T is initially not fulfilled, and as long as it is not, the same procedure is repeated for each iteration.

First, the parameter vector θ of length h is estimated using the previously specified start vector $\theta^{(0)}$ and the function `EstimateParameters`, described in Section 5.1.2. Next, the parameter vector $\eta^{(0)}$ is obtained from θ as $\{\eta_j^{(0)}\}_{j=1,\dots,d} = \sum_{l=1}^{h} \theta_l w_{*j}^{(l)2}$, where the $w_{*j}^{(l)2}$ are constructed as described in Section 2.2.1. In the KPLS case, $\eta = \eta^{(0)}$ constitutes our d-dimensional model parameter vector, which completes the surrogate model. In the KPLS+K case, $\eta^{(0)}$ is locally optimized with the goal of improving the surrogate model. This is, once again, done using the `EstimateParameters` function.

When the surrogate model is specified, new points are to be added to the sample. By calling the function `GetNewPoint`, described in Section 5.1.3, m times, the new points $(\boldsymbol{x}_{n+1}^{\mathcal{N}}, \dots, \boldsymbol{x}_{n+m}^{\mathcal{N}})$ are generated using m different infill sampling criteria. Thereafter, these points are evaluated in Jeppesen's optimizer, yielding the data $(y_{n+1}, \dots, y_{n+m})$. The sample matrix $\boldsymbol{X}^{\mathcal{N}}$ and the cost vector $\boldsymbol{y}^{\mathcal{N}}$ are updated with the new data and the current sample size is increased. Lastly, the termination criterion is (possibly) updated by calling the function `ShouldTerminate`, described in Section 5.1.4. Once it is fulfilled, the algorithm terminates.

5.1.1 Design of experiments

In Algorithm 4, the pseudocode for generating a design of experiments by Latin Hypercube design is presented. To obtain a sufficiently well spread-out set of sample points, we focus on minimizing correlations.

Algorithm 4 `CreateInitialSample`(Ω, n, d)

```
Set lower and upper bounds, l_j = min Ω_j, u_j = max Ω_j ∀j ∈ d
Set ρ_max = 1
Set number of retries, n_R
Set X_best = ∅
for loopIndex = 1 : n_R do
    Set X = ∅
    for j = 1 : d do
        p = random perturbation of 1,...,n
        for i = 1 : n do
            s = GetRandomIntegerInRange(0,(u_j − l_j + 1)/n) + l_j + i(u_j − l_j + 1)/n)
            X_{j,p_{n−i+1}} = s
        end for
        R = {r_{i,k}}_{i,k=1,...,n} = GetCorrelationMatrix(X)
        if max_{i≠k} |r_{i,k}| < ρ_max then
            ρ_max = max_{i≠k} |r_{i,k}|
            X_best = X
        end if
    end for
end for
return [X_best, {l_j}_{j=1,...,d}, {u_j}_{j=1,...,d}]
```

First, we denote by l and u the lower and upper limits, respectively, of each variable. Now, the idea is to generate a large number n_R of Latin Hypercube samples, and then choose the sample which corresponds to the smallest correlation. The current smallest correlation is denoted by $\rho_{\max}$ and set to 1, while the current best sample X_{best} is initialized as the empty set.

In each of the retries a new matrix X is constructed, and each variable is then considered one at the time. We want to divide each variable's domain into n parts of equal size. Since the variables take

on integer values only, the variable domain Ω have to be constructed carefully, so that it is possible to make such a division. In fact, for each variable j it must hold that $(u_j - l_j + 1)/n \in \mathbb{N}$. When the domain is divided into n equally sized parts, a value s is randomly sampled in each part i as

$$s = \texttt{GetRandomIntegerInRange}(0, (u_j - l_j + 1)/n) + l_j + i(u_j - l_j + 1)/n).$$

This yields an n-dimensional vector, which shall be randomly perturbed. Practically, a random perturbation p of $1,...,n$ is created. When a value s is sampled in part i of variable j, we set $X_{j,p_{n-i+1}} = s$. After all entries of X have been filled, its correlation matrix is computed, and the largest non-diagonal value is considered. If this value is smaller than $\rho_{\max}$, $\rho_{\max}$ and X_{best} are updated, which means that X_{best} will correspond to the Latin Hypercube sample with the smallest maximal off-diagonal element. Finally, the X_{best} matrix is concatenated with the vectors containing the lower and upper values, respectively.

5.1.2 Parameter estimation

The function `EstimateParameters` in Algorithm 5 can be used for estimating both θ and η. The difference lies in the number of dimensions d and the type of search, where the θ estimate is found by a global optimization, while η is estimated by a local optimization. In both cases, the objective function to optimize is the concentrated likelihood function (22), but with different covariance kernels. When estimating θ, the covariance kernel (41) is used, while (42) is used in case of estimating η. Neither θ nor η are allowed to be non-positive, and hence bounds $b: \ p_l > 0 \ \forall l = 1,...,d$ are specified. The bounds, together with the initial guess $p^{(0)}$, the sample matrix $\boldsymbol{X}^{\mathcal{N}}$, and data $\boldsymbol{y}^{\mathcal{N}}$ are sent as arguments to the `L-BFGS-B` minimization function. The `minimize` function in the Python package `scipy.optimize` is used, with `L-BFGS-B` specified as *method*. In the case of local optimization, we simply set 'maxiter'=1, meaning that the L-BFGS-B algorithm finishes after one iteration.

Algorithm 5 `EstimateParameters`$(p^{(0)}, \boldsymbol{X}^{\mathcal{N}}, \boldsymbol{y}^{\mathcal{N}}$, `typeOfSearch`$)$

Set d =length$(p^{(0)})$
Set objective function: `LL` = concentrated likelihood function (22)
if `typeOfSearch == global` **then**
  Set kernel function $k = k_{1:h}^{\text{KPLS}}$ (41)
else
  Set kernel function $k = k^{\text{KPLS+K}}$ (42)
end if
Set bounds, $b: \ p_l > 0 \ \forall l = 1,...,d$
p = `L-BFGS-B`(`LL`$(k), p^{(0)}, \boldsymbol{X}^{\mathcal{N}}, \boldsymbol{y}^{\mathcal{N}}, b,$ `typeOfSearch`)
return p

5.1.3 Infill sampling criteria

In Algorithm 6 the pseudocode for the `GetNewPoint` function is presented, which considers the surrogate model and provides new sample points according to different infill sampling criteria. The criteria to be used are found in Table 2. As for parameter estimation, the `minimize` function in the package `scipy.optimize` in Python is used, with `L-BFGS-B` as the method choice. After having specified an initial guess $\boldsymbol{x}^{(0)} \in \Omega$ and an objective function, the `L-BFGS-B` algorithm is run to find a sample point $\boldsymbol{x}$. In comparison with the parameter estimation, both lower and upper bounds (given by the end points of the variable domain in each dimension; Ω_j, $j = 1,...,d$) $l_j = \min \Omega_j$ and $u_j = \max \Omega_j$, need to be provided to the `L-BFGS-B` function.

Algorithm 6 `GetNewPoint`$(i, \boldsymbol{X}^{\mathcal{N}}, \boldsymbol{y}^{\mathcal{N}}, \eta, \Omega, d)$

Set lower and upper bounds, $l_j = \min \Omega_j, u_j = \max \Omega_j,\ j = 1,...,d$
Choose an initial guess: $\boldsymbol{x}^{(0)} \in \Omega$
Set objective function: `ISC`(i, η), given on the i:th row in Table 2
$\boldsymbol{x}$ = `L-BFGS-B`(`ISC`$(i, \eta), \boldsymbol{x}^{(0)}, \boldsymbol{X}^{\mathcal{N}}, \boldsymbol{y}^{\mathcal{N}}, \{l_j\}_{j=1,...,d}, \{u_j\}_{j=1,...,d}$)
return $\boldsymbol{x}$

5.1.4 Termination criterion

The pseudocode for the function `ShouldTerminate` is presented in Algorithm 7, which determines when to terminate the complete optimization procedure. As mentioned in Section 4.4, there are two criteria for termination; convergence or if a maximum number of iterations have been reached.

Algorithm 7 `ShouldTerminate`$(\boldsymbol{y}^{\mathcal{N}}, n, m)$

Choose maximum number of iterations, `maxIter`
Set `currentIter` = (length($\boldsymbol{y}^{\mathcal{N}}$) − n)/m
if `currentIter` ≥ `maxIter` **then**
  return `True`
end if
Set number of points for convergence verification, n_T
Set tolerance ϵ
Set `counter` = 0
for k = `currentIter` − n_T : `currentIter` **do**
  Set p_k and p_{k-1} according to (43)
  if $p_k - p_{k-1} > -\epsilon$ **then**
    `counter+` = 1
  end if
end for
if `counter` == n_T **then**
  return `True`
end if
return `False`

The first thing to be done in `ShouldTerminate` is to choose the maximum number of iterations `maxIter` (which was denoted by $\mathcal{T}$ in Section 3.2). This number is then compared with the current iteration number `currentIter` = (length($\boldsymbol{y}^{\mathcal{N}}$) − n)/m. If `maxIter` exceeds `currentIter`, the function `ShouldTerminate` returns `True`.

To verify whether or not the algorithm has converged, the number of points for convergence verification n_T is set, as well as the tolerance ϵ. Then, for all points k in the range (`currentIter`-n_T,`currentIter`), the number p_k is computed according to (43). If all differences between consecutive p_k are greater than the (negative) tolerance, i.e., if $p_k - p_{k-1} > -\epsilon,\ \forall k \in \{\texttt{currentIter} - n_T + 1, ..., \texttt{currentIter}\}$, `ShouldTerminate` returns `True`. If none of the termination criteria are fulfilled, `ShouldTerminate` returns `False`.

5.2 Implementation into the Jeppesen Manpower Planning framework

All algorithms and functions presented in this section are implemented in Python, except for the function `Evaluate`, which corresponds to running Jeppesen's optimizer. When a starting sample $\boldsymbol{X}$ is created using the function `CreateInitialSample`, all points $\boldsymbol{x}$ are evaluated in the optimizer, resulting in several KPIs. The KPIs, especially the total cost y, are then utilized in order to find the next set of sample points. The complete optimization procedure is launched from a Linux terminal and proceeds automatically until any of the termination criteria are fulfilled. The implementation into a GUI is not within the scope of this thesis, but should be possible t accomplish.

5.3 Problem settings

There are several different interesting problem instances to be investigated. The original data provided by Jeppesen have $d = 76$ groups and more than 10,000 crews, which means that the problems to solve are 76-dimensional. Due to heavy computation times, mainly in the L-BFGS-B optimization procedure, we prefer to consider instances where the number of dimensions is reduced. The following problem settings are considered which are all associated with lower dimensions:

- Optimize only on groups where no vacancy is announced ($d = 37$)
  Referred to as S1
- Optimize only on a subset $\mathcal{G}$ of the groups. For consistency, we choose $d = 37$ groups, but the number can be arbitrary in the range $[1, 76]$. Out of the 37 groups, the ten groups with highest deficiency cost are selected, while the remaining groups are picked randomly. The dimensionality reduction technique Dropout-Copy (30) is used for the 39 groups not being considered for optimization, meaning that those variables are copied from the best-so-far solution
  Referred to as S2. The variable o_j denotes the original number of vacancies announced in group $j \in \mathcal{G}$, and this data is provided by Jeppesen

Both problem settings are presented in Table 3 and will be solved using both the KPLS and the KPLS+K approach. Moreover, as several points are evaluated in each iteration, it may happen that different infill sampling criteria yield the same optimal point. In the worst case scenario, very few points are added in each iteration, which becomes inefficient in the sense of total computation time. To get around this problem, we make sure that at least a number n_L of points are added in each iteration. It is more beneficial to include points corresponding to the most exploiting infill sampling criteria, i.e., the ISC named LCL, WB_2, WB_2^{A} and WB_2^{B} in Table 2. Thus, if a point found by any of these versions of ISC is non-unique, noise is applied to change the point slightly. In practice, a few of the variables are changed by one unit. This procedure ensures that at least $n_L = 4$ points are added to the sample in each iteration. Moreover, the initial sample size is 32 in both the S1 and S2 settings, which constitutes of 30 LHD points and two points corresponding to the lower and upper variable values, i.e. $\min_{j=1,\dots,d}\{\Omega_j\}$ and $\max_{j=1,\dots,d}\{\Omega_j\}$.

Table 3: Parameter values of problem setting S1 and S2. The variable o_j denotes the original number of vacancies announced in group j.

Parameter	Value (S1)	Value (S2)
Number of groups, d	37	37
Initial sample size, n	32	32
Variable domain Ω_j for groups j with vacancies announced	$[-29, 0]^d$	$[o_j - 4, o_j + 5]$
Variable domain Ω_j for groups j with no vacancies announced	$[-29, 0]^d$	$[-29, 0]$
Number of retries when building an initial sample matrix, n_R	1000	100
Maximum number of algorithm iterations, `maxIter` ($\mathcal{T}$)	20	20
Number of principal components, h	3	3
Initial paramater, $\theta_j^{(0)}$	$0.1\ \forall j = 1, \dots, h$	$0.1\ \forall j = 1, \dots, h$
Number of points for convergence verification, n_T	3	3
Tolerance, ϵ	10^{-3}	10^{-3}

6 Algorithm evaluation

Before running the algorithm with data provided by Jeppesen, the performance quality of the algorithm is investigated. Since the true minimum of Jeppesen's problem instance is not known, it is impossible (in reasonable time) to know whether a near-optimum solution has been reached. Therefore, by evaluating the algorithm using a function with a known minimum value, it is possible to see how well the algorithm performs. However, it is not certain that the performance quality of the algorithm is the same considering different problem instances, but it indicates whether the algorithm is able to produce promising results or not.

To obtain a reliable result, the benchmark function and the black-box function should produce as similar results as possible. After some trial, the following benchmark function was chosen:

$$y(\boldsymbol{x}) = 900 \sum_{j=1}^{d} (x_j - r_j)^2 + 2 \cdot 10^7. \tag{44}$$

The minimum is $r = \{r_1, ..., r_d\}$, where each r_j is randomly generated in the integer valued range defined by the variable domain in the j^{th} dimension; Ω_j, i.e., in the range $[\min \Omega_j, \max \Omega_j]$ $\forall j = 1, ..., d$. The goal of the optimization algorithm is to find r, or at least a point close to r.

What should be noticed is that the benchmark function (44) is convex and hence it has a very smooth behavior, which is not the case when it comes to the black-box function representing the total cost obtained from Jeppesen's optimizer. The black-box function is very complex and contains many irregularities. Thus, we can not expect the algorithm to perform as well when using real airline data as when using the benchmark function. This is taken into consideration later on when the results obtained from the algorithms on real airline data is presented and discussed.

In the remainder of the section, the KPLS and the KPLS+K algorithms, introduced in Sections 2.2.1 and 2.2.2, are evaluated on the problem setting S1 found in Table 3. All simulations presented in this thesis are done on an Intel® Core™ i7-6600U CPU @ 2.60 GHz 2.81 GHz PC.

6.1 Evaluation of the KPLS algorithm using a benchmark function

To evaluate the performance of the KPLS algorithm, we begin by generating a start sample $\boldsymbol{X}$ using the `CreateInitialSample` function, and via (44) the corresponding data $\boldsymbol{y}$ is obtained. The best point in the start sample has a cost of 23,205,800 and is located 59.68 distance units away from the true minimum. The distance considered is the Euclidean distance in $d = 37$ dimensions, given by

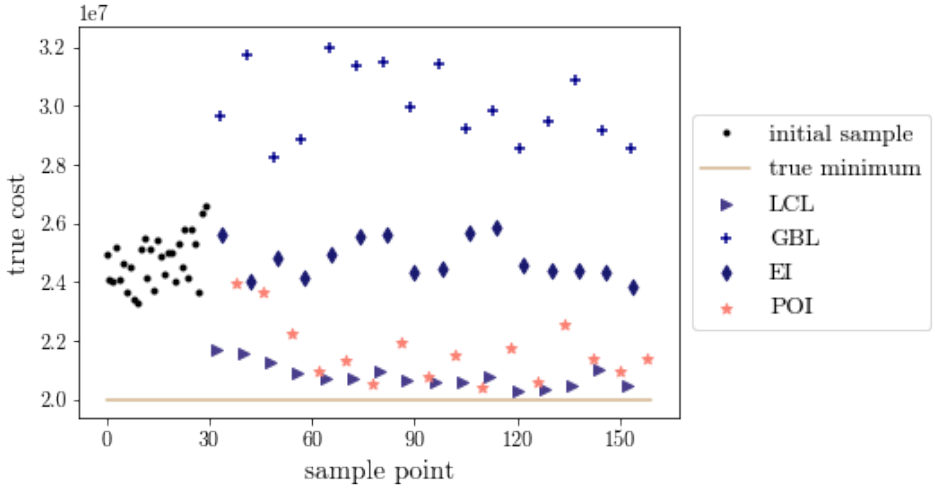
$$D(\boldsymbol{x}, \boldsymbol{x}^*) = \sqrt{\sum_{j=1}^{d} (x_i - x_i^*)^2}$$

Running the algorithm on the start sample results in the points visualized in Figure 15. In Figure 15a the initial sample is plotted as small dots, followed by most of the ISC points. However, some are suppressed and plotted only in Figure 15b for clarity, showing the evolution of the points with exploiting ISC. The lines at the bottom of the plots represent the true minimum at $2 \cdot 10^7$, which the new points are approaching.

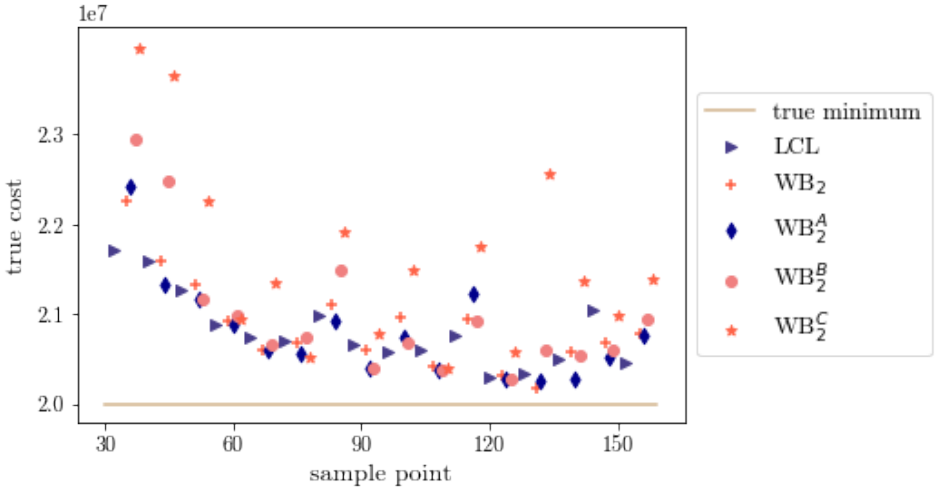
The best point found before the algorithm terminates has a cost of 20,175,500 and a distance of 13.96 length units to the optimum $\boldsymbol{x}^*$. It corresponds to infill sampling criterion WB_2 in the 13th iteration.

Even before the new points are evaluated, their predicted value can be calculated using the Kriging predictor (23). These predictions are plotted in Figure 16. We observe that the predictions are generally smaller than the true values, but the overall trend is similar. To be able to see the differences in predictions and true values more clearly, the error given by

$$\text{Error}(\boldsymbol{x}) = \frac{|\hat{Y}(\boldsymbol{x}) - Y(\boldsymbol{x})|}{Y(\boldsymbol{x})} 100, \tag{45}$$



(a) Initial sample and evolution of most of the ISC points; some local criteria are excluded for clarity.



(b) Evolution of points associated with exploiting ISC.

Figure 15: KPLS algorithm performance using the benchmark function (44).

is plotted in Figure 17. It can be observed that most errors are below 5%. The largest errors correspond to infill sampling criterion GBL. This is natural, since the criterion of GBL is to *maximize model uncertainty*.

Another way of investigating whether the ISC characteristics seem reasonable or not is to look at distances between points. Once again, it is the Euclidean distance in $d = 37$ dimensions that is considered and for each new point we pay attention to the *smallest* distance to any of the points in the sample. We expect that global/space filling ISC should have a larger smallest distance to any point as compared with the more local ISC. From Figure 18 this behavior is confirmed. The criterion GBL corresponds to the largest values, while points associated with local optimization of the surrogate model, e.g., criteria LCL, WB_2, $\mathrm{WB}_2^{\mathrm{A}}$, $\mathrm{WB}_2^{\mathrm{B}}$, and $\mathrm{WB}_2^{\mathrm{C}}$ have small values. Moreover, the distances stay more or less the same for each of the different ISC as the sample size increases, meaning that the space is not significantly filled by the addition of approximately 120 new points.

The computation times for estimating θ and optimizing the different ISC are plotted in Figure

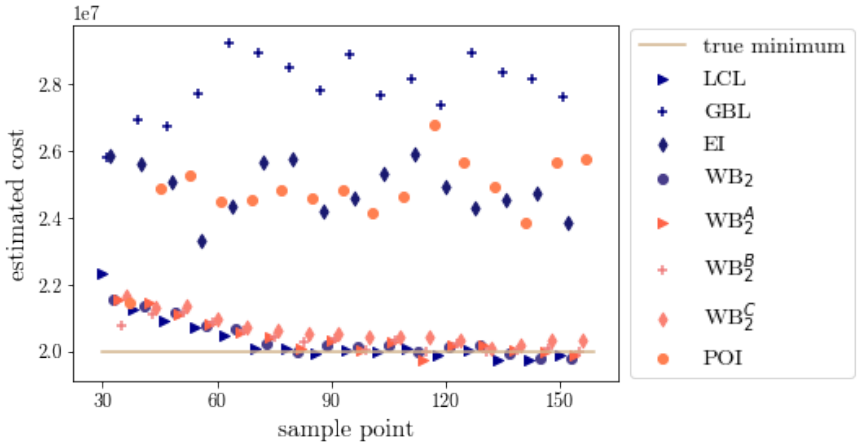


Figure 16: Kriging predictions of the benchmark function (44) within the KPLS algorithm.

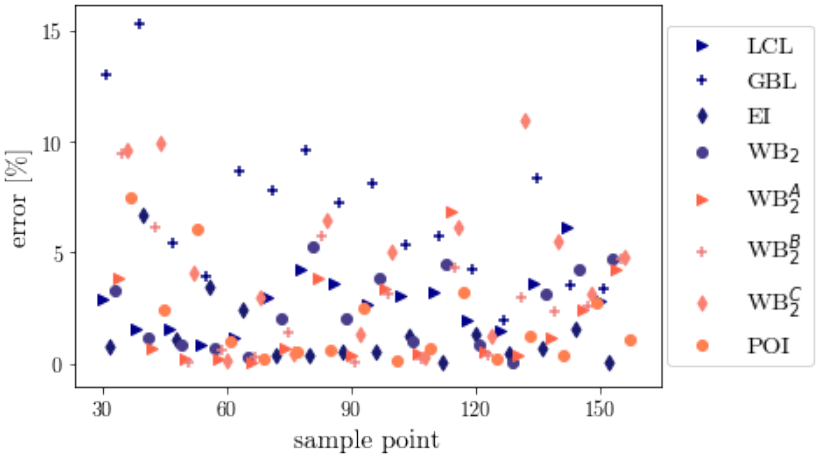


Figure 17: Kriging predictor errors given by (45) within the KPLS algorithm.

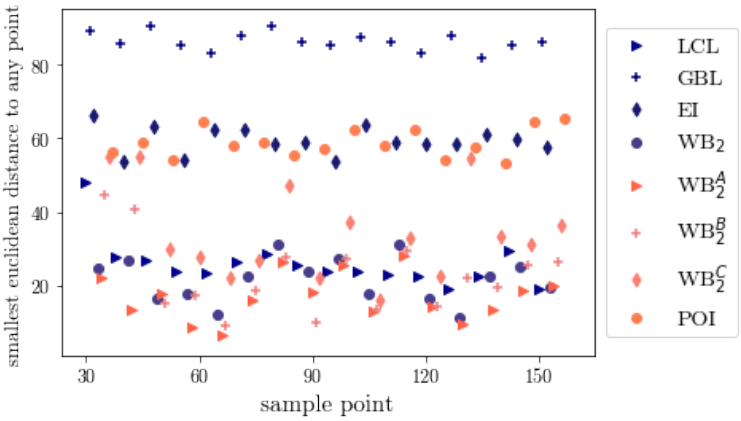


Figure 18: The smallest distance to any point in the sample, for each new point.

19. The dashed line represents the complete time spent on each iteration of the algorithm, which is generally increased with an increased sample size. The total time, from start to termination, is 4 hours and 8 minutes, while the minimum is found after 3 hours and 32 minutes.

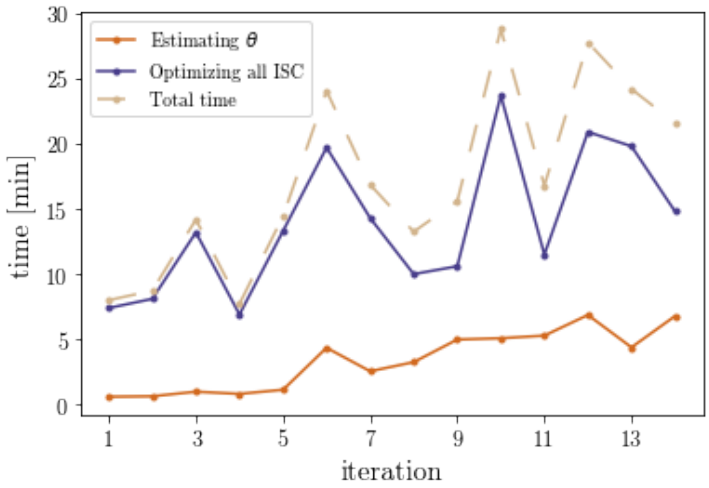


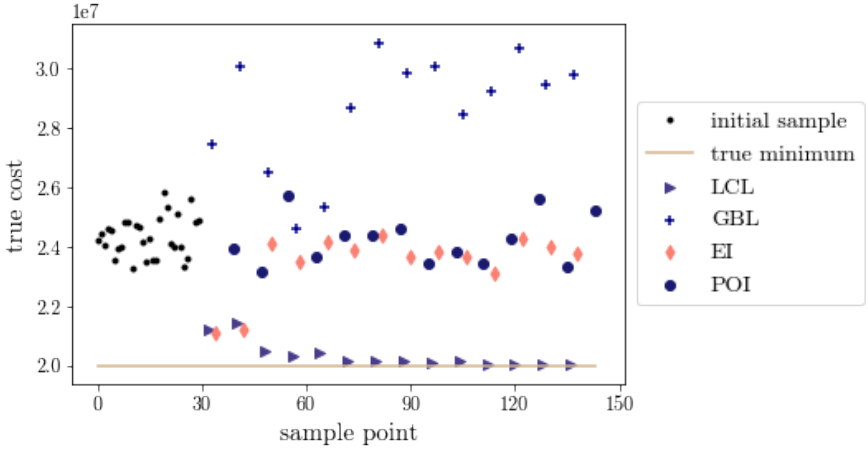
Figure 19: Computation times for each iteration of the KPLS algorithm when the benchmark function (44) is used as the black-box function

6.2 Evaluation of the KPLS+K algorithm using a benchmark function

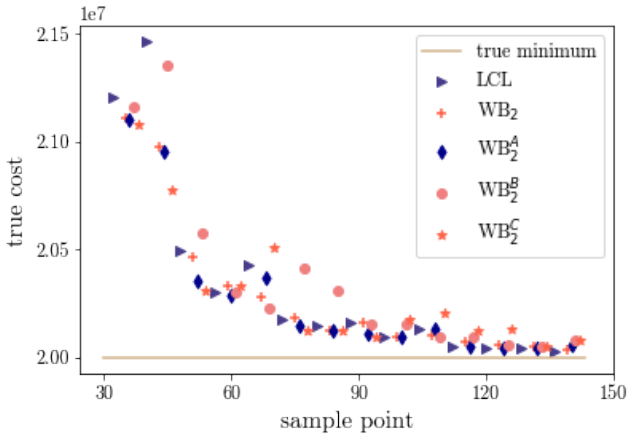
In this section, the KPLS+K algorithm is evaluated using the same problem setting as when evaluating the KPLS algorithm. This time, the minimum found by the algorithm has a cost of 20,030,600 and a distance of 5.83 length units to the optimum. The sample is visualized in Figure 20, with the points corresponding to different behaving ISCs are plotted in Figure 20a and the points corresponding to local optimizations are plotted in Figure 20b for clarity. The minimum is observed in the 14:th iteration, associated with infill sampling criterion LCL. Compared to the KPLS algorithm, a lower cost than the minimum of 20,175,500 is found at earliest in iteration 6 for criterion WB_2^{A}. This means that, if the total time of reaching the point corresponding to WB_2^{A} in iteration 6 is less than 3 hours and 32 minutes (which is the time it takes to reach criterion WB_2^{A} in iteration 6), the KPLS+K algorithm is preferred rather than the KPLS algorithm for the problem instance considered. The time of estimating the parameters θ and η, as well as the time for optimizing the ISC and the total time spent in each iteration are plotted in Figure 22.

It is clear that the time of estimating the parameter η is increasing with the iteration number, i.e., with the sample size. The total time from start to termination for the KPLS+K algorithm is 6 hours and 46 minutes, which is obviously longer than the running time of the KPLS algorithm. However, it takes 1 hour and 9 minutes until the KPLS+K algorithm reaches a lower cost than the minimum found by the KPLS algorithm, which indicates that the KPLS+K algorithm is more beneficial. The fact that fewer iterations are needed is even more important when running the algorithms on real airline data, since evaluations of the black-box function (represented by Jeppesen's optimizer) are time consuming.

Lastly, in Figure 22 the errors of the KPLS+K algorithm computed as in (45) are presented. These errors are generally smaller than the errors of the KPLS algorithm plotted in Figure 17. Once again, the largest errors correspond to criterion GBL, i.e., maximizing model uncertainty. It is natural that errors become smaller using the KPLS+K algorithm, since the parameter estimation is more accurate, meaning that the concentrated likelihood function (22) is increased.



(a) Initial sample and evolution of most of the ISC points; some local criteria are excluded for clarity.



(b) Evolution of points associated with exploiting ISC.

Figure 20: KPLS+K algorithm performance using the benchmark function (44).

The fact that the algorithms find near-optimal points using the benchmark function (44) is promising and we thus proceed by evaluating the KPLS and KPLS+K algorithms on real airline data, using both parameter settings S1 and S2. However, as mentioned in the beginning of this section, the black-box function representing the total cost obtained from Jeppesen's optimizer is less smooth, which means that the algorithm may differ in performance when applied on real airline data.

7 Results

When evaluating the KPLS and the KPLS+K algorithm using the real airline data provided by Jeppesen, both parameter settings S1 and S2 are considered. The results obtained by running the KPLS algorithm with parameter settings S1 and S2 are presented in Section 7.1 and 7.2, respectively, while the corresponding KPLS+K results are presented in Sections 7.3 and 7.4. Two different initial samples are generated, one to use with S1 and one to use with S2. Lastly, some general observations are presented in Section 7.5

In comparison with the benchmark problem, the minimum black-box function value is unknown,

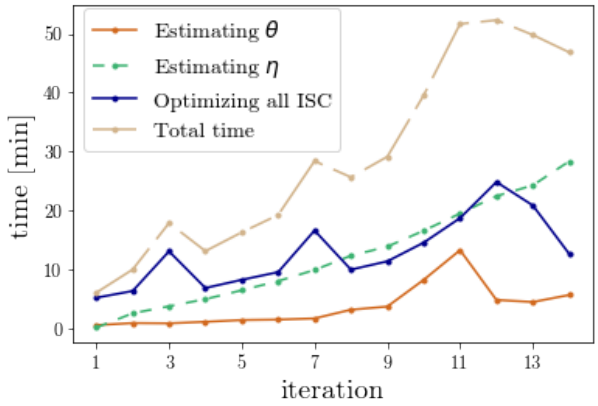


Figure 21: Computation times for each iteration of the KPLS+K algorithm when the benchmark function (44) is used as the black-box function.

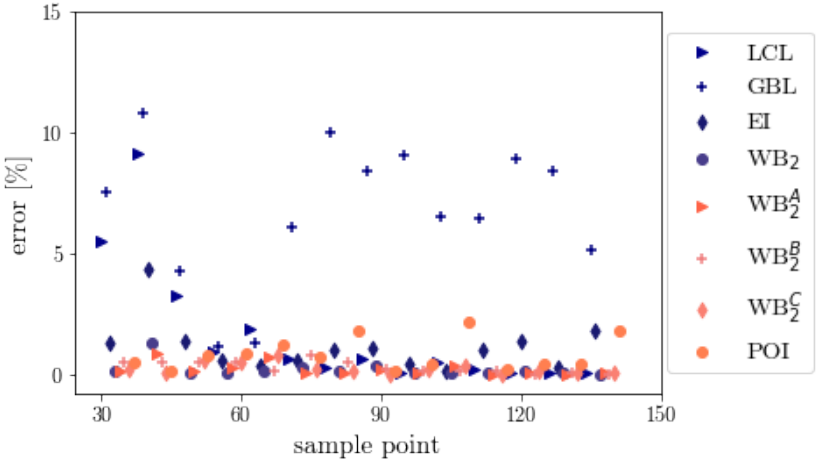


Figure 22: Kriging predictor errors given by (45) when running the KPLS+K algorithm and using the benchmark function (44).

but we still want to be able to assess the performance quality of the algorithms. Hence, a reference solution is constructed manually by the manpower optimization team at Jeppesen, who have much insight in the problem and its dynamics. The reference solution is associated with a cost of 21,866,261 and does not represent the black-box function minimum. Hence, the algorithms should ideally find solutions with costs lower than this reference value. The total cost is a sum of different costs, e.g., deficiency costs and cost of courses. Such values are to be investigated as well. The deficiency cost corresponding to the reference solution is 7,666,282, while the share of crews being transitioned is 0.213.

7.1 Evaluating the KPLS algorithm on airline data with setting S1

In Figure 23 the result of running the KPLS algorithm on real airline data with parameter setting S1 is visualized. The best solution found by the algorithm has a cost of 20,875,584 and it was found in the 7^{th} iteration, corresponding to criterion WB_2^B. The algorithm terminated after ten iterations and found many points related to a lower cost than the manually constructed reference solution, which is

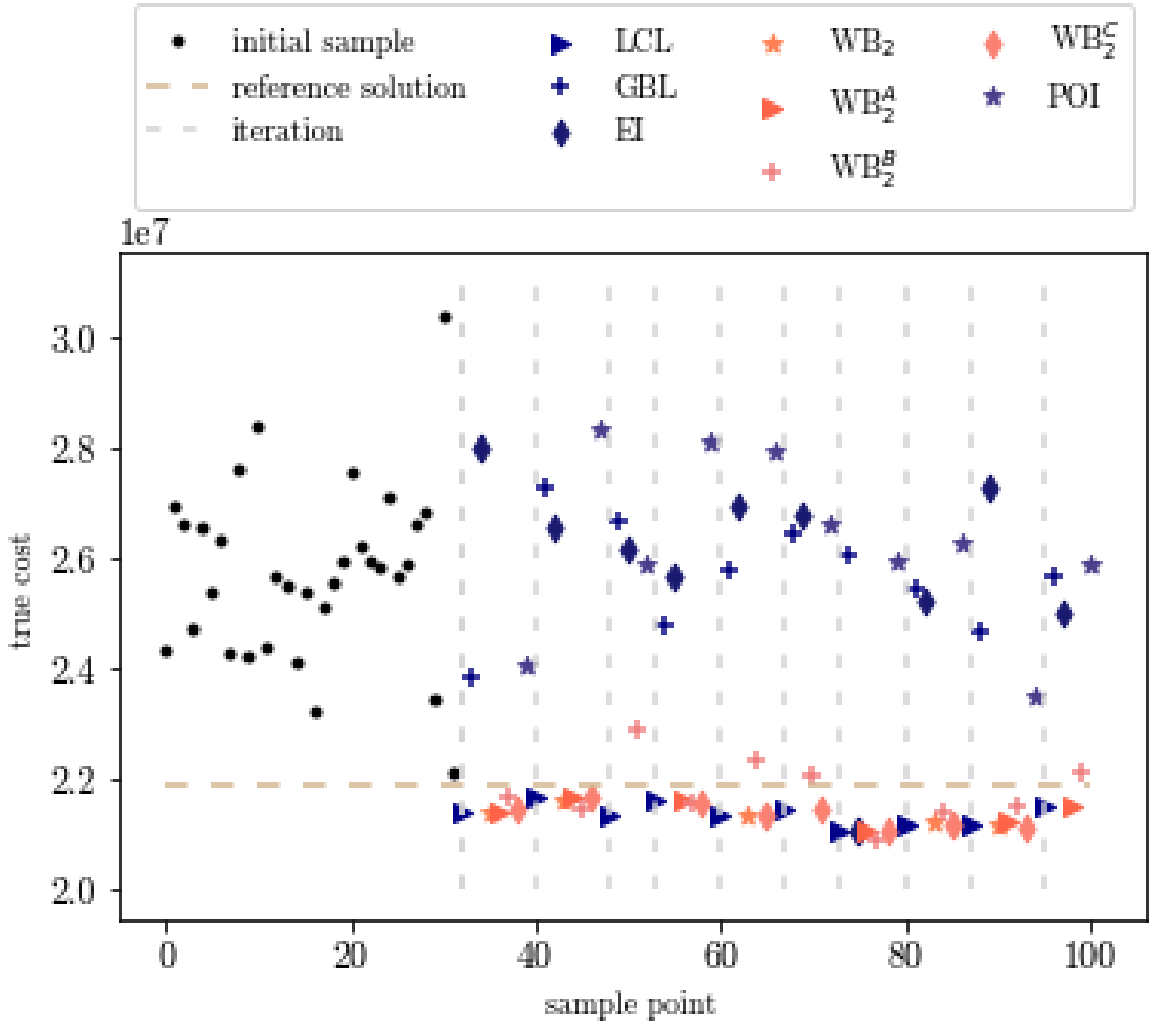


Figure 23: KPLS algorithm performance using real airline data and parameter setting S1. The horizontal dashed line represents the reference solution constructed manually, while the vertical dashed lines indicate the iterations.

represented by pale dashed lines in Figure 23.

The sample points plotted in Figure 23 corresponds to the best solution found. To find out whether the algorithm is robust, we would like to run it several times. However, since computation times are heavy, we restrict our computations to three runs per case. In Table 4, the lowest cost among three evaluations are presented, together with the corresponding ISC, iteration number and time to reach that point, as well as the total time and number of iterations before algorithm termination.

In contrast to the benchmark problem, points close to each other (in Euclidean distance) does not necessarily have similar costs. For example, the point corresponding to the minimum cost in this evaluation is very close to three other points; the points corresponding to criteria GBL, EI, and POI in the 7^{th} iteration. The fact that the points are close means that they differ with a small number in few of the dimensions. The Euclidean distances are 3.61, 1.73 and 2.45, respectively, while the costs are 26,032,238, 21,044,766 and 25,949,687. This means that, e.g., the points corresponding to criteria WB_2^{A} and POI in iteration 7 differs by one unit in six of the groups, and this difference gives rise to a

Table 4: Three evaluations of the KPLS algorithm using problem setting S1.

Lowest cost	ISC	Iteration	Time reaching best solution [h:min]	Time including black-box evaluation	Total # iterations	Total time [h:min]	Total time including black-box evaluation
20,875,584	WB_2^B	7	1:32	$\approx$ 7:30	10	2:36	$\approx$ 11:00
20,882,789	WB_2	2	0:05	$\approx$ 2:00	5	0:38	$\approx$ 5:00
20,941,931	WB_2^B	5	0:52	$\approx$ 5:00	8	1:34	$\approx$ 8:30

24% cost increment.

As mentioned in the beginning of this section, the total cost is a sum of several different costs, deficiency costs and cost of courses being two such terms. Ideally, both the deficiency cost and the number of courses given, i.e., crew transitions, should be small. The deficiency costs and the share of crews being transitioned, corresponding to the total costs in Figure 23, are plotted in Figures 24 and 25, respectively. We observe that the deficiency costs are generally decreasing with the iteration number, while the share of crew transfers is slightly increasing. The best solution found, i.e., the sample point corresponding to the lowest total cost is associated with both a relatively low deficiency cost (8,259,762) and a low share of crews being transitioned (0.155). There is a general trade-off between deficiency cost and the share of crews being transitioned, which is realized when observing that the sample points which give rise to high deficiency costs generally are associated with low shares of crews being transitioned. If desired, more importance can be given to either of these KPIs, by weighting the different costs in the objective function.

The prediction errors, calculated as in (45), are plotted in Figure 26. The errors are generally small, most errors are below 5%. It is neither evident that errors are increasing nor decreasing as the sample grows. Also, the size of the error does not have the same correspondence to the ISC. It is not the case that the more space-filling criteria (GBL, EI and POI) generally have larger errors than the criteria which are focusing on the surrogate model minimum (LCL, WB_2, WB_2^A, WB_2^B and WB_2^C).

The computation time for each of the iterations are plotted in Figure 27. The time for optimizing all the ISC is generally increasing with a larger sample size, while the computation time for estimating the parameter θ does increase at first, but then decreases again. However, since optimizing the ISC makes up the majority of the total computation time, the total computation time is still increasing with the number of iterations in general. The total time from start to termination of the algorithm (excluding cost evaluations performed in Jeppesen's optimizer) is 2 hours and 51 minutes, and the best solution was found after 1 hour and 32 minutes. The points to be evaluated in the optimizer are sent in parallel, and hence the evaluation wall clock time is independent of the number of points evaluated in each iteration. Nevertheless, the optimizer's evaluation time is approximately 50 minutes, which means that the complete computation time is roughly 11 hours, while the best point was found after roughly 7.5 hours.

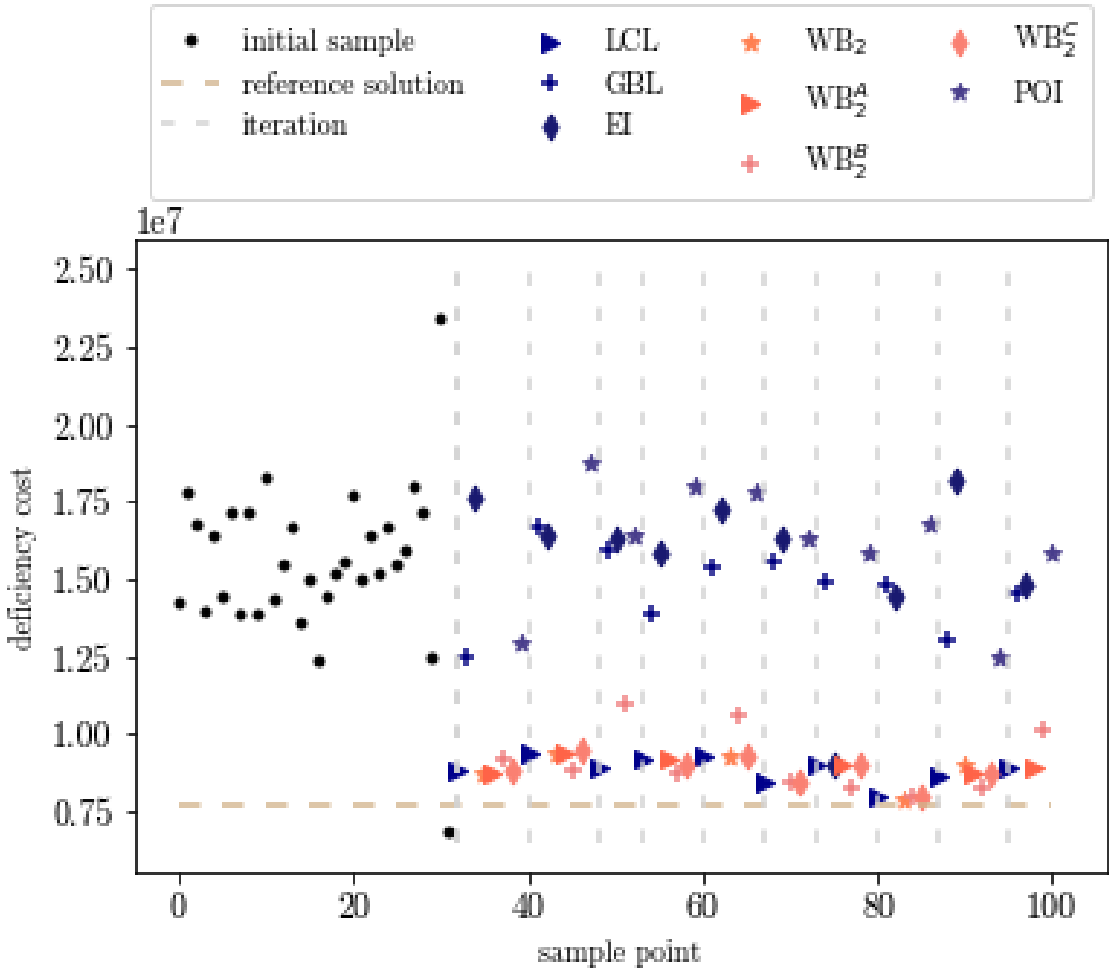


Figure 24: Deficiency costs for each of the sample points evaluated using the KPLS algorithm on problem setting S1 and real airline data.

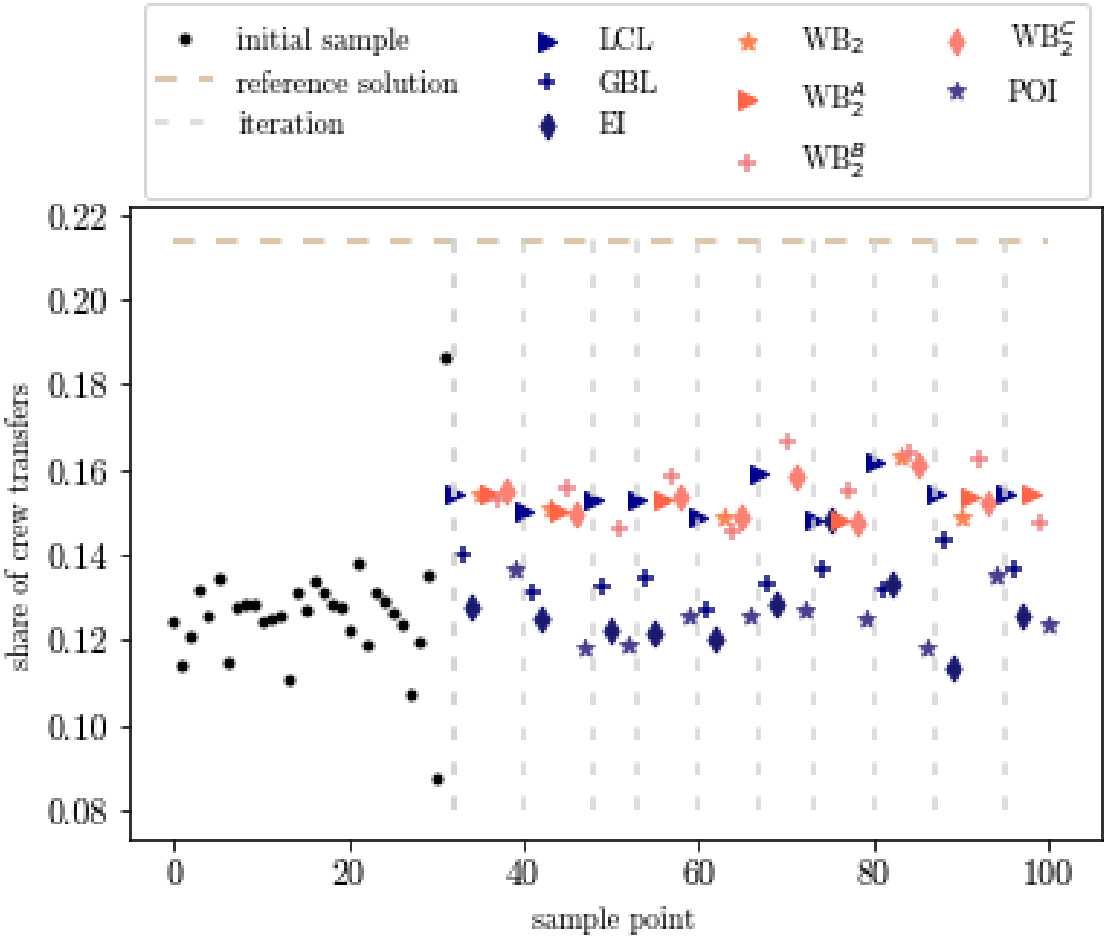


Figure 25: Share of crew transfers for each of the sample points evaluated using the KPLS algorithm on problem setting S1 and real airline data.

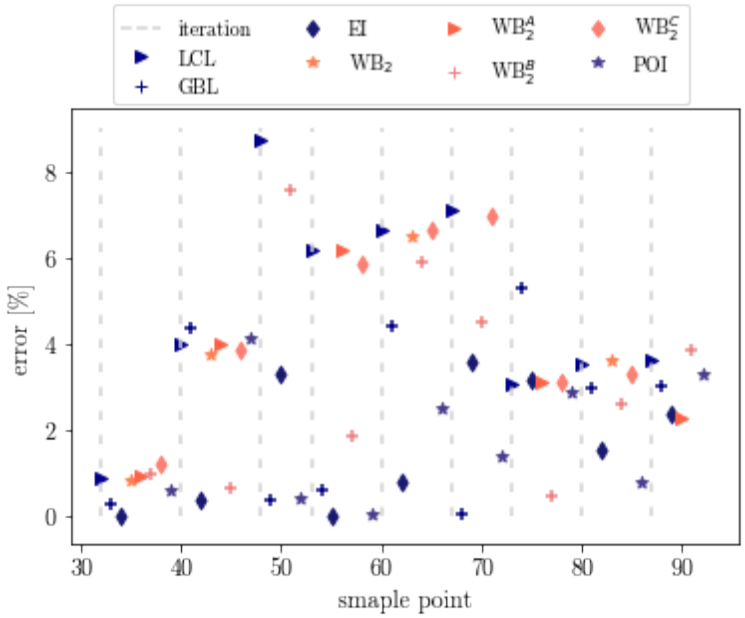


Figure 26: Kriging predictor errors given by (45) when running the KPLS algorithm with parameter setting S1 on real airline data.

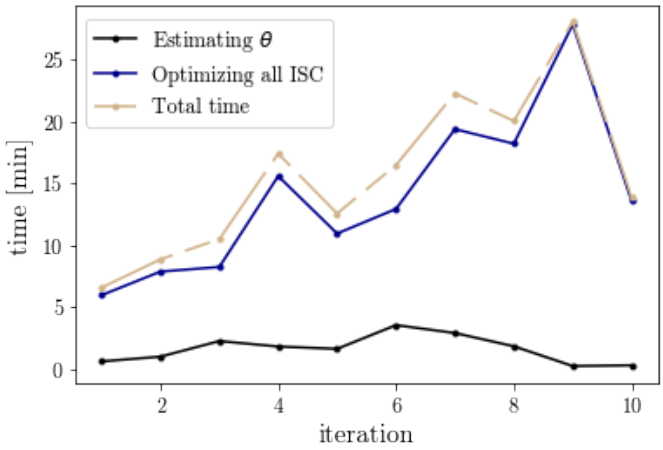


Figure 27: Computation times for each iteration of the KPLS algorithm using real airline data and parameter setting S1.

7.2 Evaluating the KPLS algorithm on airline data with setting S2

Here, the KPLS algorithm is evaluated on a new problem setting, namely S2. Three different evaluations are presented in Table 5, where the evaluation corresponding to the lowest cost found is to be looked into in short. A different initial sample is constructed which, as compared to the initial sample used with S1, is related to higher costs. The costs associated with the new initial sample are plotted together with the evolution of the costs of the new sample points, in Figure 28.

The best point found is related to a cost of 21,965,782, which corresponds to criterion WB_2 in the fourth iteration. This cost is much higher than the lowest cost found using parameter setting S1, and

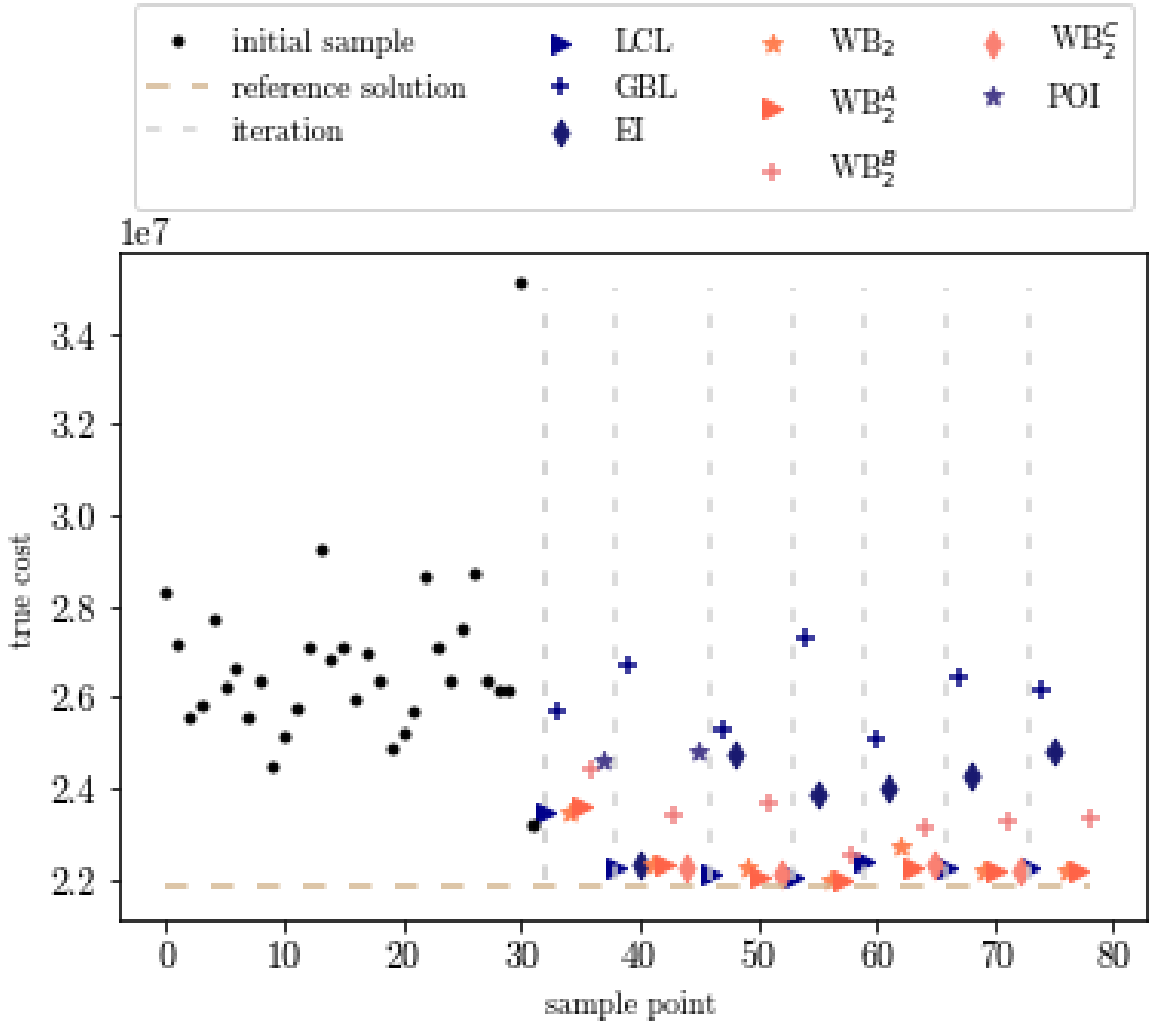


Figure 28: KPLS algorithm performance using real airline data and parameter setting S2. The horizontal dashed line represents the reference solution constructed manually, while the vertical dashed lines indicate the iterations.

slightly larger than the reference solution. The algorithm terminates after seven iterations, since it can not find an improved solution within three iterations after the fourth iteration, at which the point with the lowest cost point was found.

Table 5: Three evaluations of the KPLS algorithm using problem setting S2.

Lowest cost	ISC	Iteration	Time reaching best solution [h:min]	Time including black-box evaluation	Total # iterations	Total time [h:min]	Total time including black-box evaluation
21,965,782	WB_2	4	0:46	≈ 4:00	7	1:40	≈ 7:30
21,996,566	WB_2^C	3	0:24	≈ 3:00	6	1:02	≈ 6:00
22,260,378	WB_2^A	4	0:43	≈ 4:00	7	2:11	≈ 8:00

The corresponding deficiency costs and shares of crew transitions are plotted in Figures 29 and 30, respectively. In contrast to the S1 case, the deficiency costs are much lower than the deficiency cost of the reference solution. However, due to the trade-off between a low deficiency cost and a low share of crews being transitioned, the share of crews being transitioned is much higher as compared to the S1 case. For the S2 problem instance, almost all sample points correspond to solutions were the share of crew transitions are approaching 20 %.

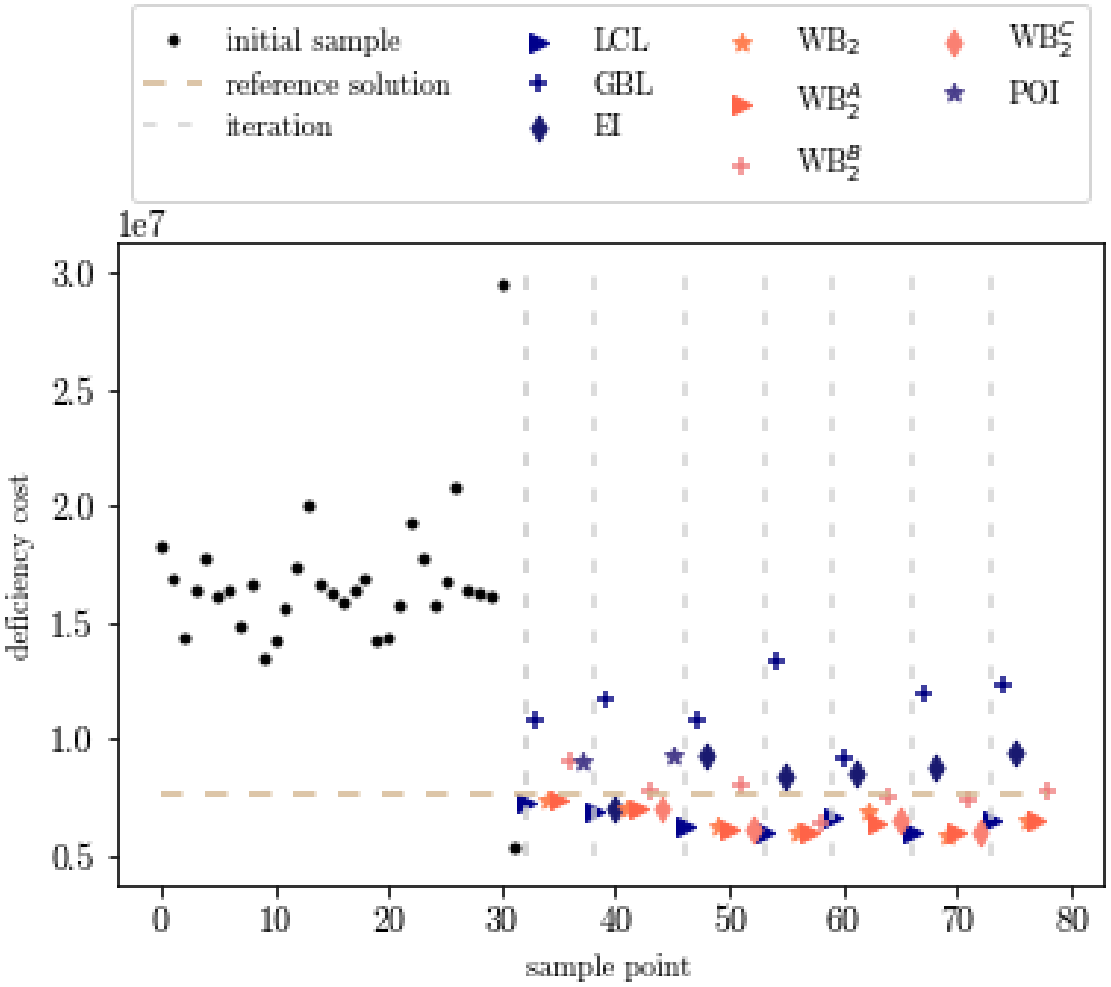


Figure 29: Deficiency costs for each of the sample points evaluated using the KPLS algorithm on problem setting S2 and real airline data.

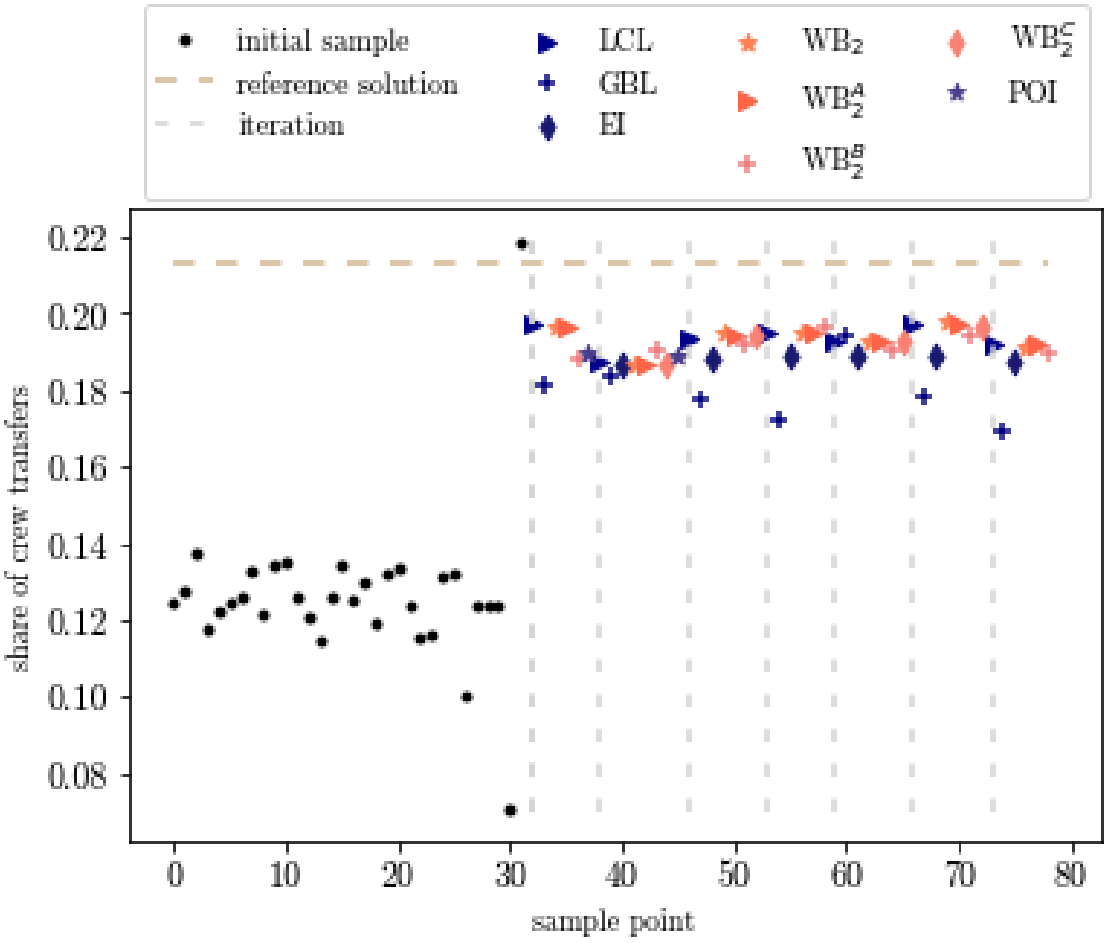


Figure 30: Share of crew transfers for each of the sample points evaluated using the KPLS algorithm on problem setting S2 and real airline data.

The prediction errors for each sample point, calculated as in (45), are plotted in Figure 31. The errors are much larger as compared with the S1 case and once again it is not the case that points corresponding to space-filling ISC generally have larger errors than points corresponding to exploiting ISC. However, there is a trend of errors decreasing as the number of sample points is increased.

The resulting computation times are visualized in Figure 32. We observe that, in general, the time of estimating the parameter vector θ, as well as optimizing all the ISCs, is increased with the number of iterations, even though there is a high peak in the second iteration considering the time of estimating all ISCs.

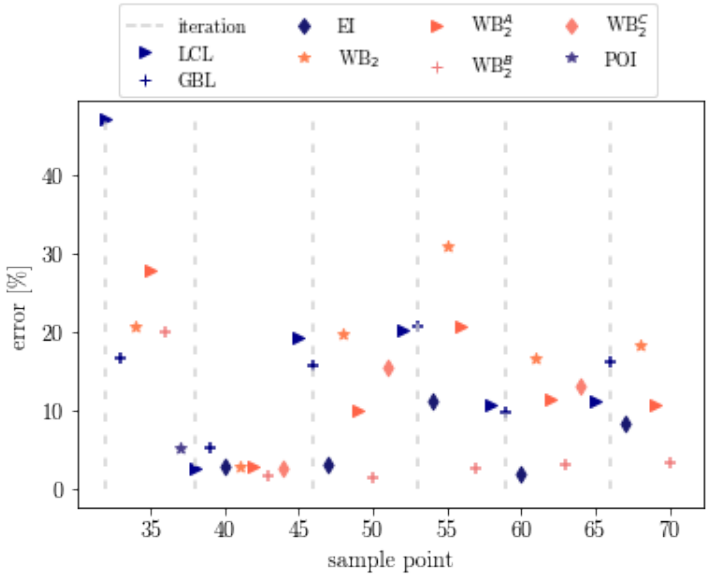


Figure 31: Kriging predictor errors given by (45) when running the KPLS algorithm with parameter setting S2 on real airline data.

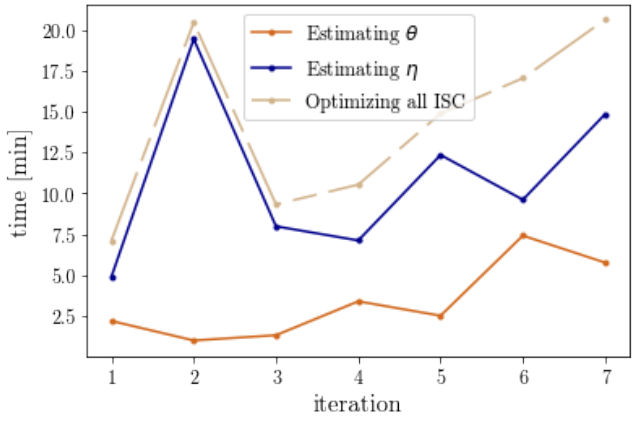


Figure 32: Computation times for each iteration of the KPLS algorithm using real airline data and parameter setting S2.

7.3 Evaluating the KPLS+K algorithm on airline data with setting S1

The next thing to do is to evaluate our algorithm which includes an improved parameter estimation, referred to as KPLS+K. Three evaluations are presented in Table 6, and the best evaluation is to be examined further.

In Figure 33, the initial sample created for problem setting S1 is plotted, together with the evolution of the new sample points. The lowest cost found is 20,894,365 and it corresponds to criterion LCL in the 9^{th} iteration. The algorithm then terminates three iterations later, after the 12^{th} iteration. Many of the points found are associated with lower costs than the reference solution.

The corresponding deficiency costs and shares of crew transitions are plotted in Figures 34 and 35, respectively. The behavior is similar to the case when the KPLS algorithm was evaluated on the same problem instance (S1), where the deficiency cost was generally decreasing with the number

Table 6: Three evaluations of the KPLS+K algorithm using problem setting S1.

Lowest cost	ISC	Iteration	Time reaching best solution [h:min]	Time including black-box evaluation	Total # iterations	Total time [h:min]	Total time including black-box evaluation
20,894,365	LCL	9	2:46	≈ 10:00	12	4:18	≈ 14:30
20,945,693	WB_2	4	1:17	≈ 4:30	7	3:04	≈ 9:00
20,947,911	WB_2^C	6	1:27	≈ 6:30	9	4:04	≈ 11:30

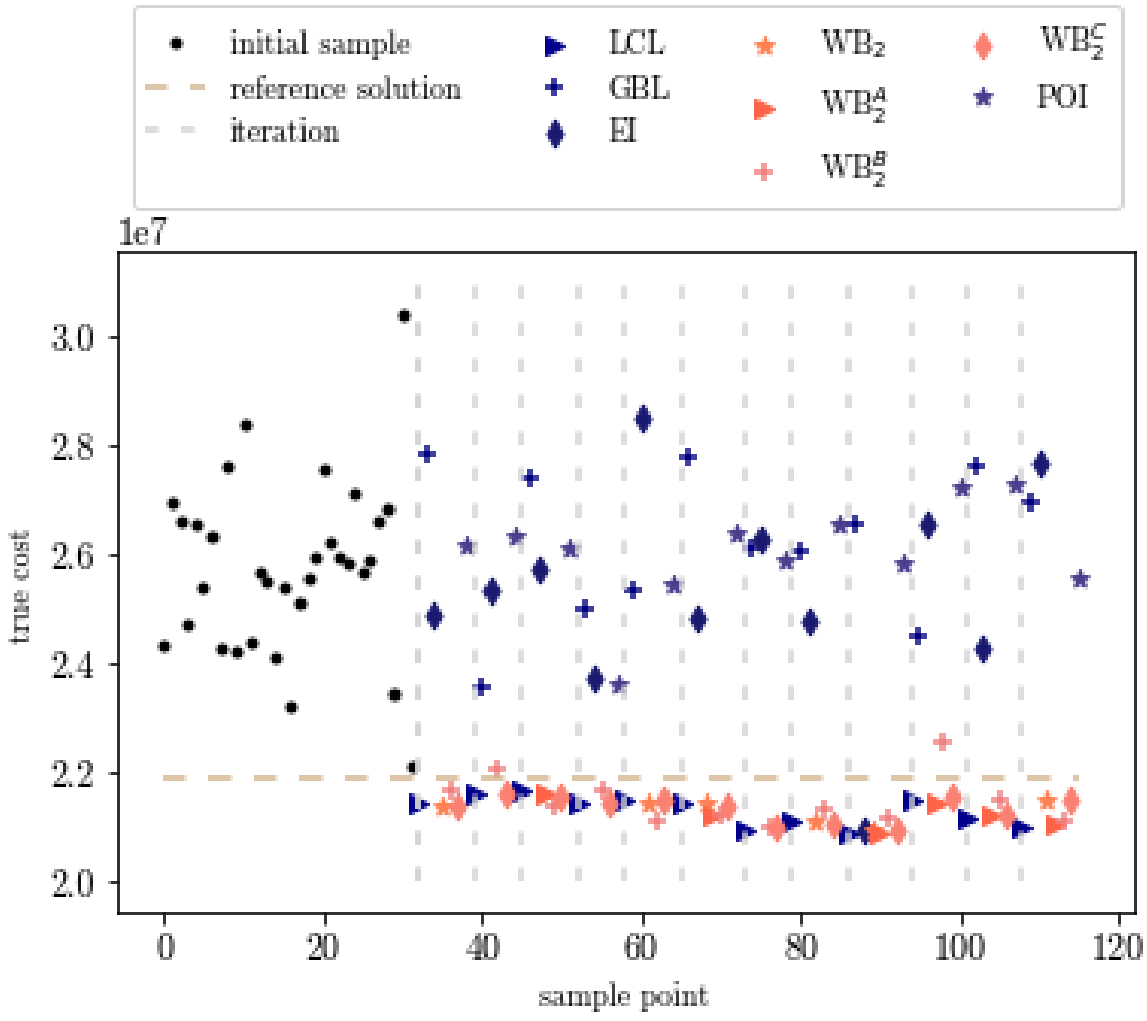


Figure 33: KPLS+K algorithm performance using real airline data and parameter setting S1. The horizontal dashed line represents the reference solution constructed manually, while the vertical dashed lines indicate the iterations.

of iterations, while the share of crews being transitioned are rather increasing with the iteration number. The best solution (i.e., the solution corresponding to the lowest total cost) has a deficiency cost of 7,896,336, which is slightly higher as compared to the deficiency cost of the reference solution,

but lower compared to the KPLS case. However, 15.9 % of the crews are being transitioned, which is more than the corresponding value of the best solution found evaluating the KPLS algorithm on problem setting S1.

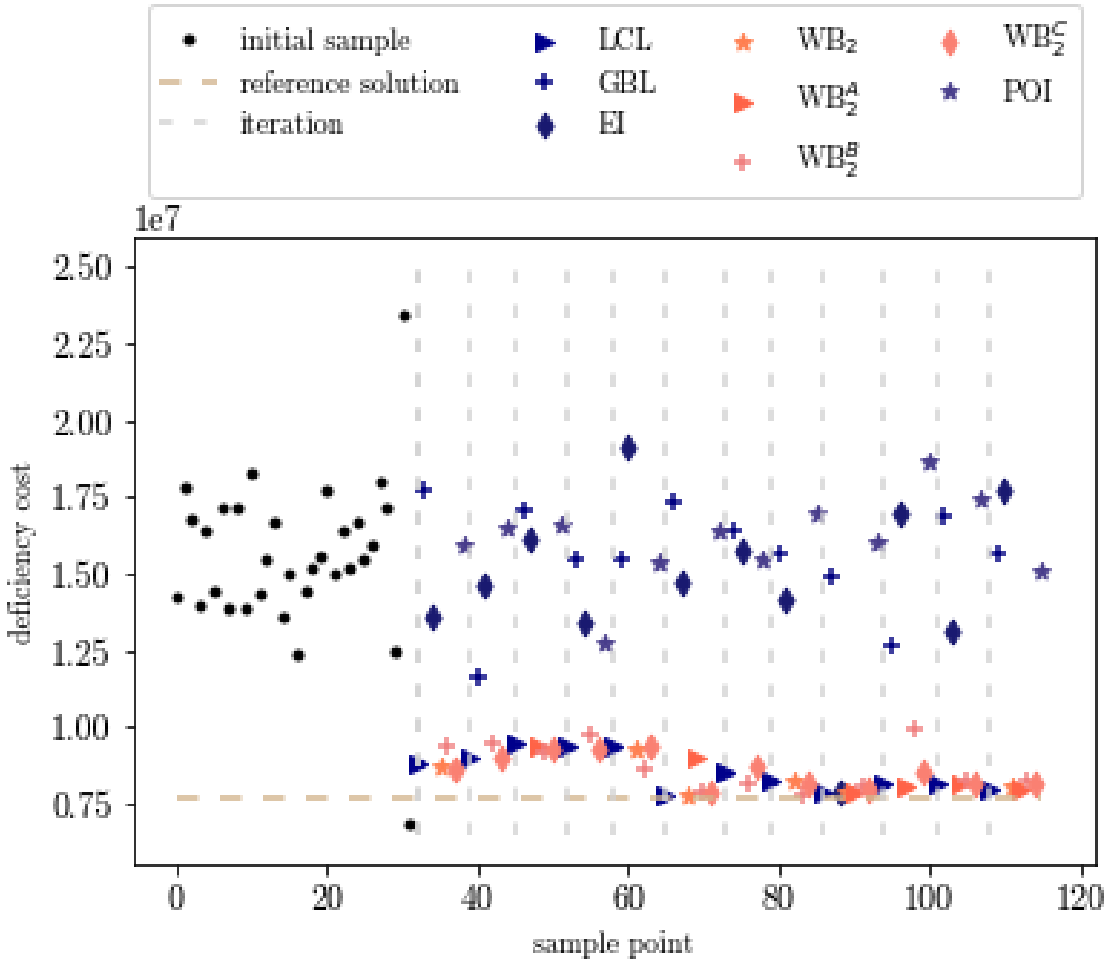


Figure 34: Deficiency costs for each of the sample points evaluated using the KPLS+K algorithm on problem setting S1 and real airline data.

The prediction errors are visualized in Figure 36. A trend of slightly increasing errors can be observed, but they all stay below 8%. There is no trend that points corresponding to space-filling ISC are associated with larger errors compared to the exploiting ISC.

As presented in Table 6, the computation time for reaching the minimum cost point is 2 hours and 46 minutes, or approximately 10 hours including the black-box evaluations in Jeppesen's optimizer. The complete evaluation time is approximately 14.5 hours, where 4 hours and 18 minutes of those are spent on estimating θ and η and optimizing the different ISCs. The computation times for estimating both parameters θ and η are unexpectedly decreased in the last two iterations.

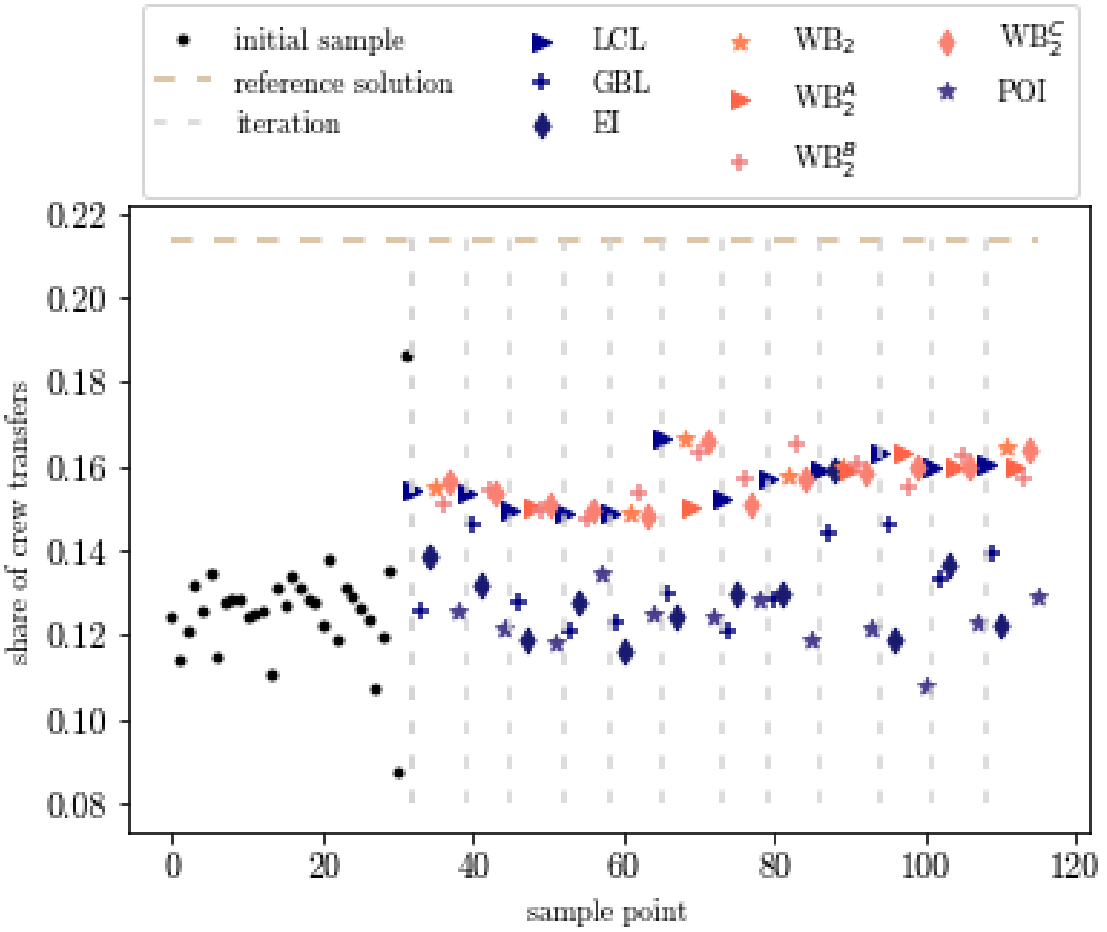


Figure 35: Share of crew transfers for each of the sample points evaluated using the KPLS+K algorithm on problem setting S1 and real airline data.

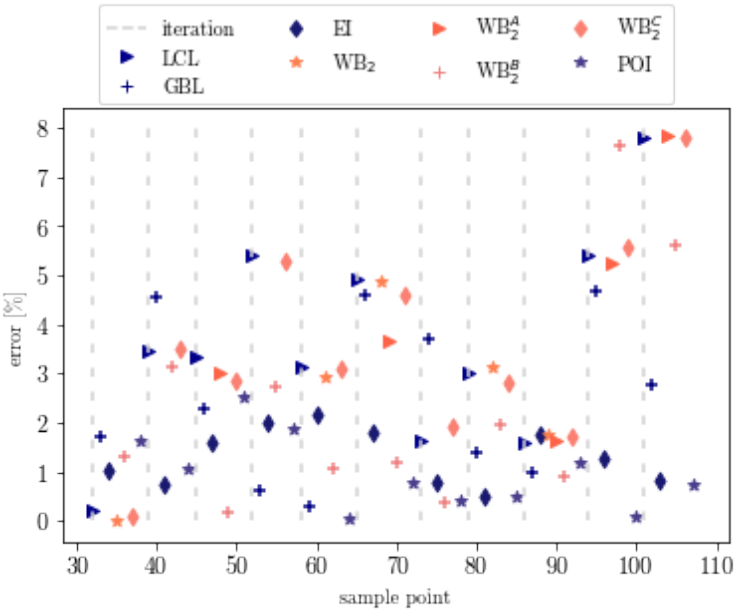


Figure 36: Kriging predictor errors given by (45) when running the KPLS+K algorithm with parameter setting S1 on real airline data.

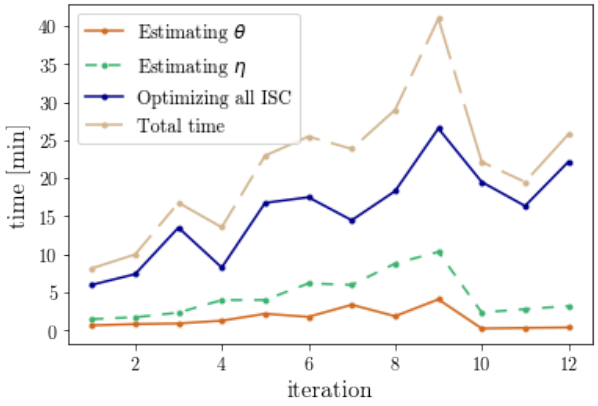


Figure 37: Computation times for each iteration of the KPLS+K algorithm using real airline data and the parameter setting S1.

7.4 Evaluating the KPLS+K algorithm on airline data with setting S2

Lastly, the KPLS+K algorithm is evaluated when problem setting S2 is used. Three different evaluations are presented in Table 7.

For the evaluation corresponding to the best solution, the resulting sample points are plotted in Figure 38. The lowest cost evaluated is 21,719,179, which is found in the fifth iteration, with criterion WB_2^C. The algorithm was unable to find a better solution within three iterations and hence terminated after seven iterations. The cost attained is slightly lower than the cost of the reference solution.

The corresponding deficiency costs and shares of crew transitions are plotted in Figures 39 and 40, respectively. Just like when evaluating the KPLS algorithm using problem setting S2, the deficiency costs are generally smaller than that of the reference solution. However, this time the percentage

Table 7: Three evaluations of the KPLS+K algorithm using problem setting S1.

Lowest cost	ISC	Iteration	Time reaching best solution [h:min]	Time including black-box evaluation	Total # iterations	Total time [h:min]	Total time including black-box evaluation
21,719,179	WB_2^C	5	1:00	≈ 5:00	8	2:02	≈ 8:30
21,807,441	WB_2^A	3	0:28	≈ 3:00	6	1:15	≈ 6:00
22,309,445	WB_2	3	0:31	≈ 3:00	6	1:21	≈ 6:00

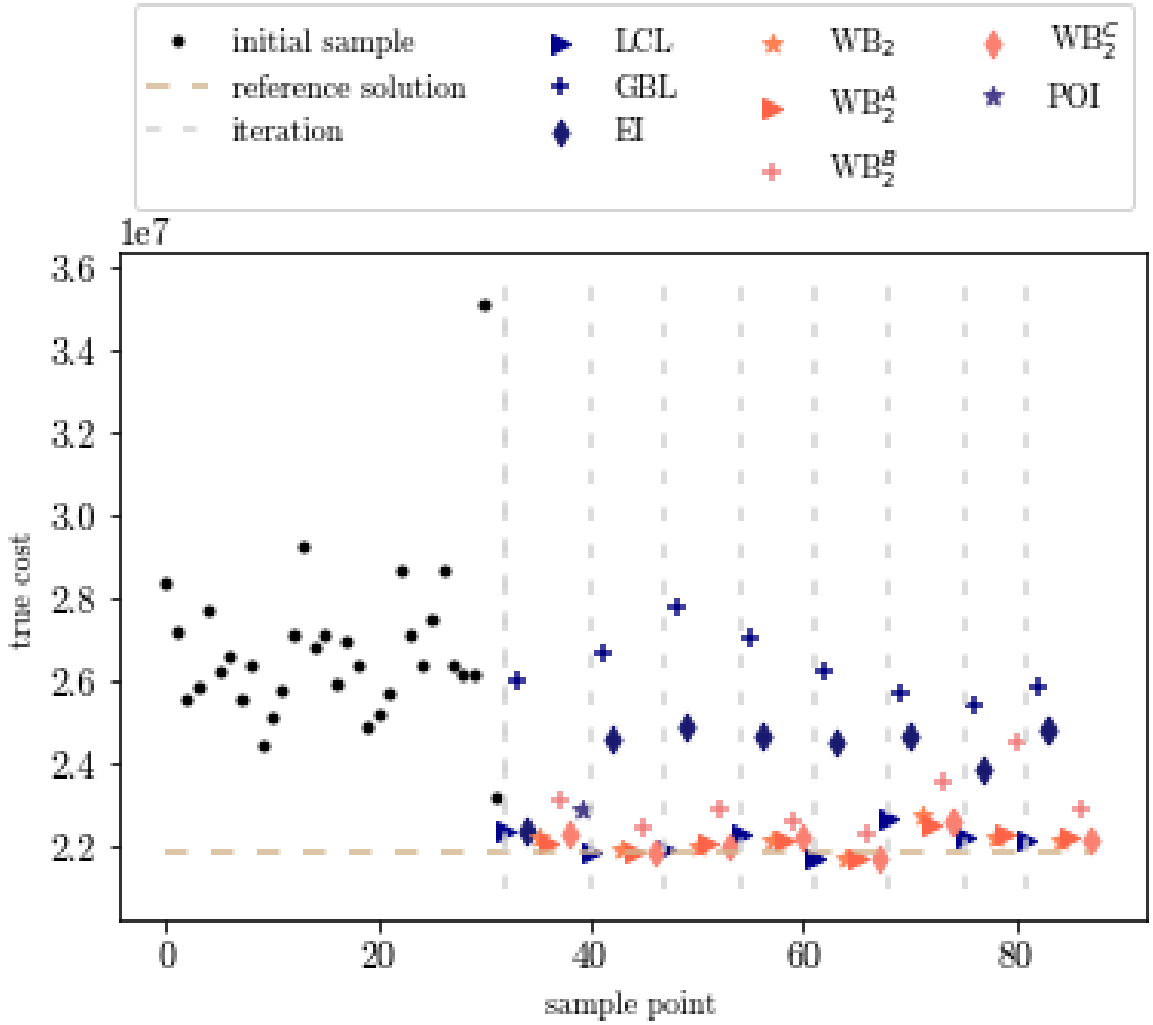


Figure 38: KPLS+K algorithm performance using real airline data and parameter setting S2. The horizontal dashed line represents the reference solution constructed manually, while the vertical dashed lines indicate the iterations.

of crews being transitioned is not as high, which may be a reason why the total costs where lower running the KPLS+K algorithm as compared to running the KPLS algorithm, using the S2 problem setting.

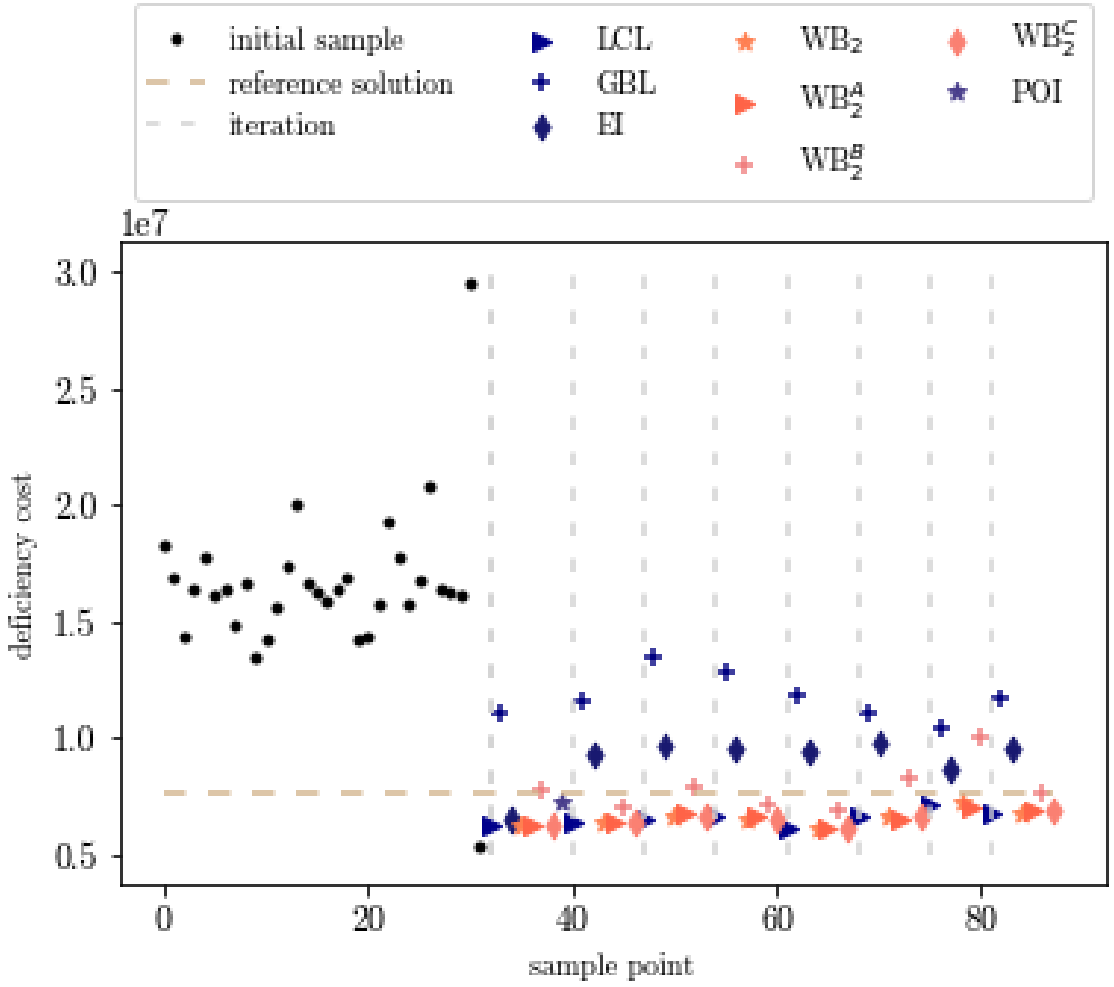


Figure 39: Deficiency costs for each of the sample points evaluated using the KPLS+K algorithm on problem setting S2 and real airline data.

The Kriging prediction errors calculated as (45) are presented in Figure 41. The errors are relatively large compared to the other cases studied, and they tend to increase with the sample size.

All relevant computation times are plotted in Figure 42. As in all previous cases, the computation time of optimizing each of the ISCs is generally increasing with the sample size. In this case, so are the computation times of estimating the parameter vectors θ and η. The time measured from start to termination of the KPLS+K algorithm with parameter setting S2 is 2 hours and 2 minutes (excluding the evaluations in Jeppesen's optimizer), while it takes roughly 8.5 hours including the time spent by Jeppesen's optimizer. Moreover, the time it takes to reach the point with the lowest cost is roughly 5 hours, or 1 hour excluding the evaluation performed by the optimizer. The largest share of the computation time in each iteration is, once again, related to optimizing all ISCs. The time of estimating θ is larger than that of estimating η.

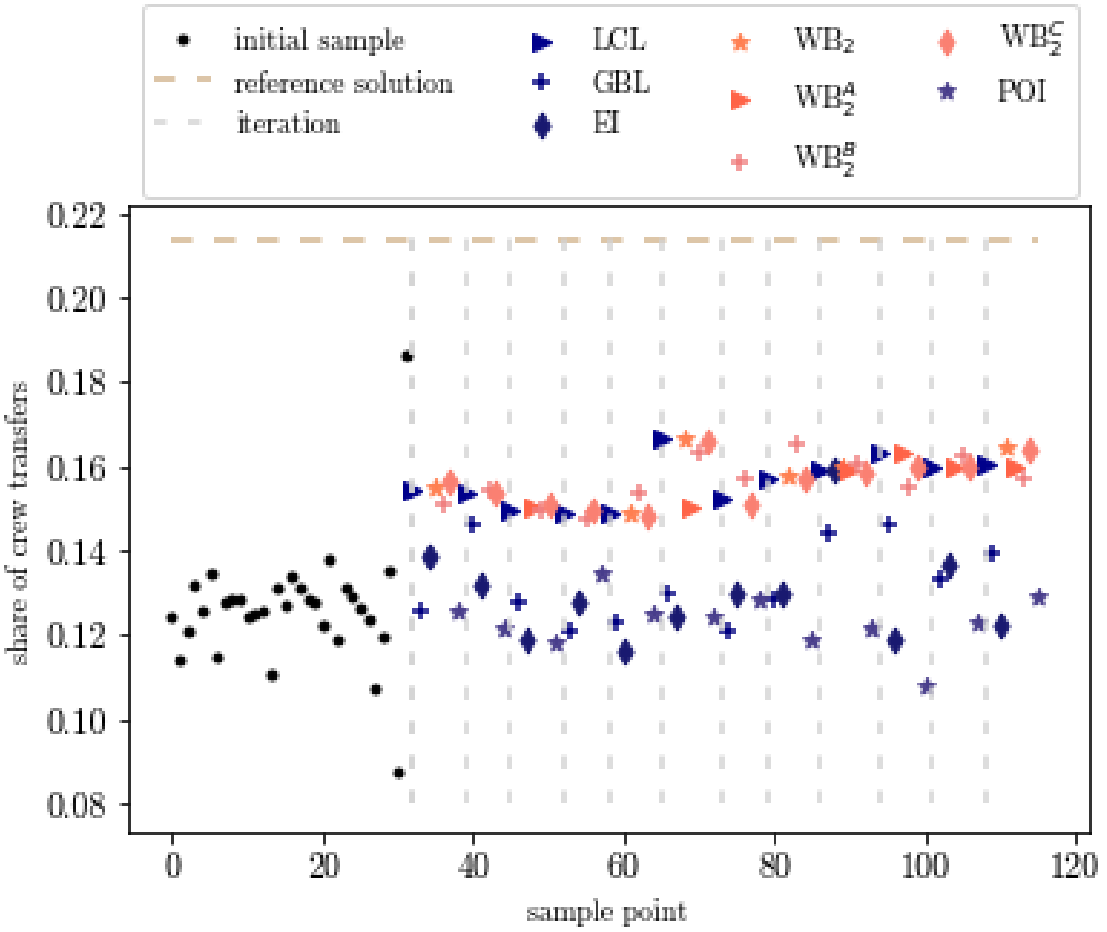


Figure 40: Share of crew transfers for each of the sample points evaluated using the KPLS+K algorithm on problem setting S2 and real airline data.

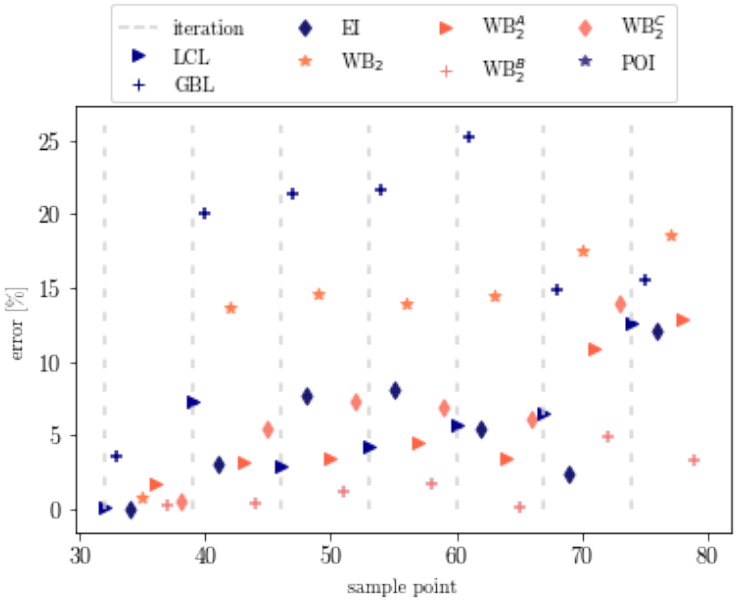


Figure 41: Kriging predictor errors given by (45) when running the KPLS+K algorithm with parameter setting S2 on real airline data.

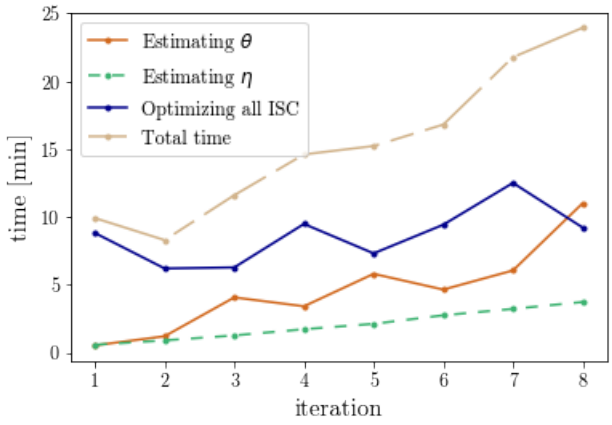


Figure 42: Computation times for each iteration of the KPLS+K algorithm using real airline data and parameter setting S2.

7.5 General observations

Lastly, some general observations are presented. What is common for all evaluations is that the most space-filling ISCs (i.e., GBL, EI, and POI) are associated with high costs, and barely ever reaches below the reference solution's cost (except the EI criterion, once in each of Figures 23 and 33). By investigating the points corresponding to space-filling ISC, it can be observed that the variables to a great extent equal the lower and upper values of the variable domain. In the case of S1, this means that many of the variables equals either −29 or 0, see Table 3. In practice, the number −29 represent the maximum allowed staffing deviation, while the number 0 means that there is no staffing deviation in the corresponding group. The behavior that variables tend to equal the lower and upper values of the variable domain is especially evident for points corresponding to the complete global search

criterion GBL. However, even though these points do not explicitly result in low costs, their presence in the sample can be useful when building the surrogate model and may lead to the exploration of new, promising areas. To investigate the usefulness of the space-filling ISC, we consider a new setting, where only criteria LCL, WB_2, WB_2^A, and WB_2^B are used for finding new points to evaluate. The rest of the parameter values are set according to S1 in Table 3. The algorithm performance is presented in Figure 43. The lowest cost found is 20,717,286, corresponding to WB_2^B in iteration 8. The algorithm terminates after 11 iterations and finds many solutions with lower costs than that of the reference solution.

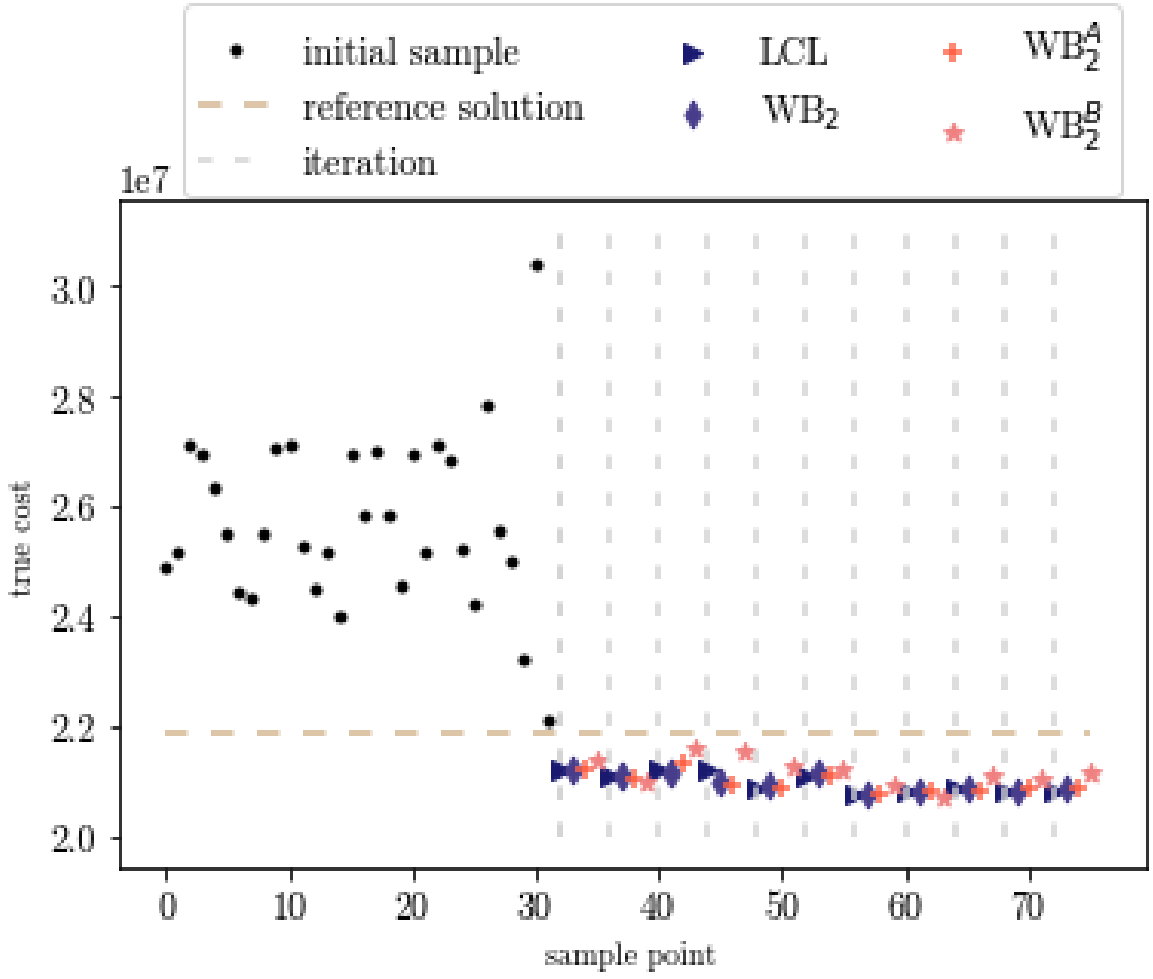


Figure 43: Performance of the KPLS algorithm on problem setting S1 using real airline data, when only exploiting ISCs are considered for finding new evaluation points. The horizontal dashed line represents the reference solution constructed manually, while the vertical dashed lines indicate the iterations.

The corresponding deficiency costs and shares of crew transitions are plotted in Figures 44 and 45, respectively. Similar results as in Section 7.1 are observed when it comes to deficiency costs, i.e., they are generally decreasing. However, this time there is no evident trend of the share of crew transitions increasing with the number of iterations, which may be a reason why such low total costs were found using this problem instance. The best solution found has a deficiency cost of 8,099,209 and the share of crews being transitioned is 0.154, which are both lower as compared to the values

obtained when running the KPLS algorithm on the original S1 problem setting.

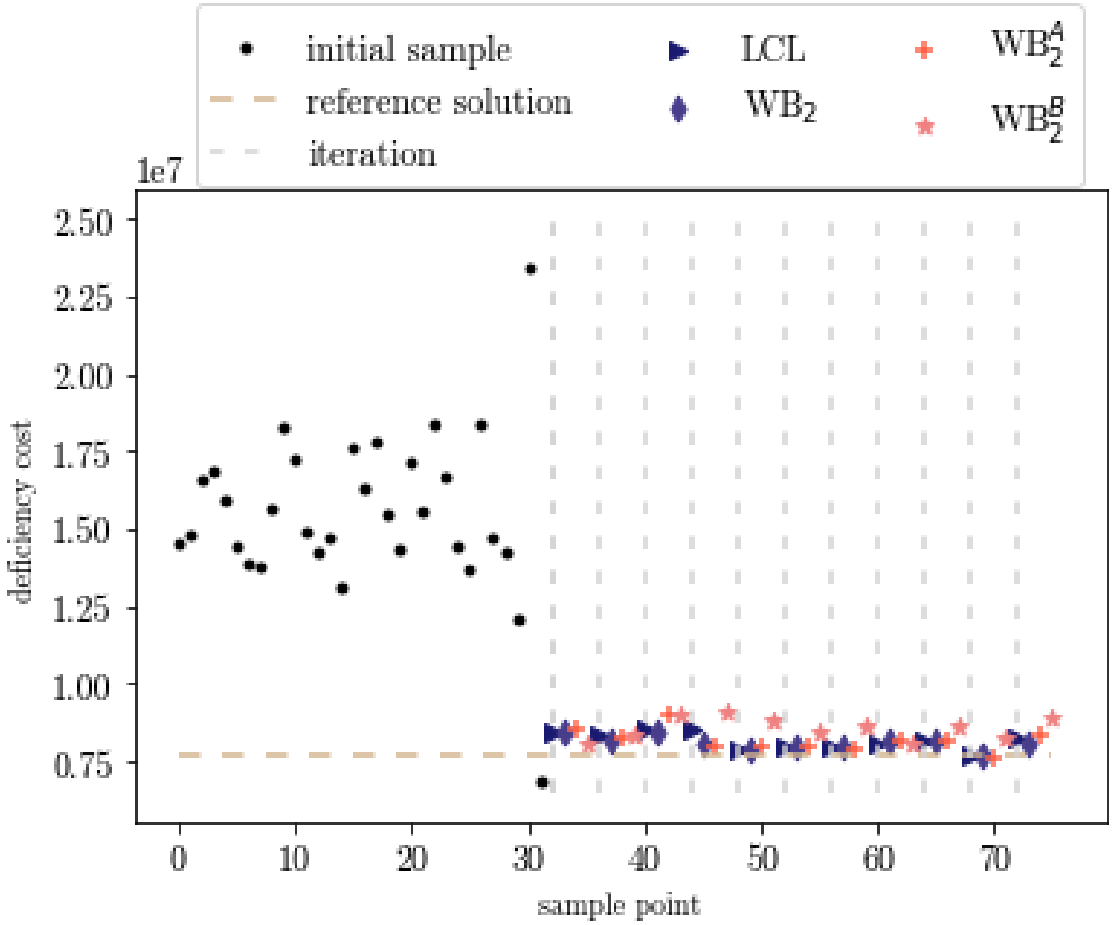


Figure 44: Deficiency costs for each of the sample points evaluated using the KPLS algorithm on problem setting S1 and real airline data, when only exploiting ISCs are considered.

The Kriging prediction errors calculated using (45) are presented in Figure 46. The errors are very low; the largest error is slightly above 3 %. No evident trend of neither increasing nor decreasing errors can be observed.

Lastly, in Figure 47, the corresponding computation times are visualized. The total computation time from start to termination is approximately 10 hours, where 1 hour and 8 minutes of those are spent on computations outside Jeppesen's optimizer. The best solution is found after roughly 7.5 hours, where the algorithm computations take 48 minutes.

The simulation presented above is associated with the lowest cost found. Resulting variables are presented in Table 8 together with those of two other simulations, to show that also in this case the algorithm is relatively robust.

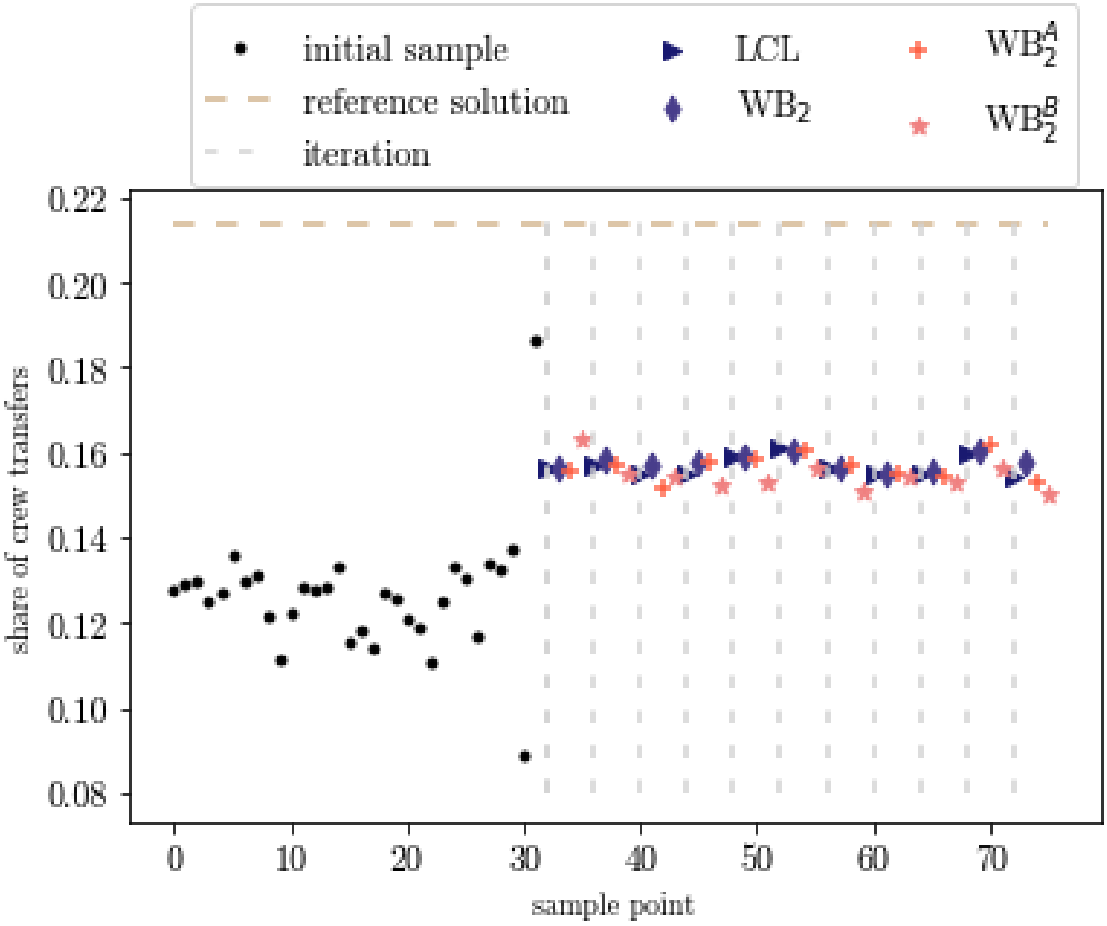


Figure 45: Share of crew transfers for each of the sample points evaluated using the KPLS algorithm on problem setting S1 and real airline data, when only exploiting ISCs are considered.

Table 8: Three evaluations of the KPLS algorithm using problem setting S1 and considering only the exploiting ISC.

Lowest cost	ISC	Iteration	Time reaching best solution [h:min]	Time including black-box evaluation	Total # iterations	Total time [h:min]	Total time including black-box evaluation
20,717,286	WB_2^B	8	0.48	≈ 7:30	11	1:08	≈ 10:00
20,816,133	WB_2	11	1:11	≈ 10:00	14	1:56	≈ 13:30
20,987,046	WB_2^B	3	0:07	≈ 2:30	6	0:21	≈ 5:30

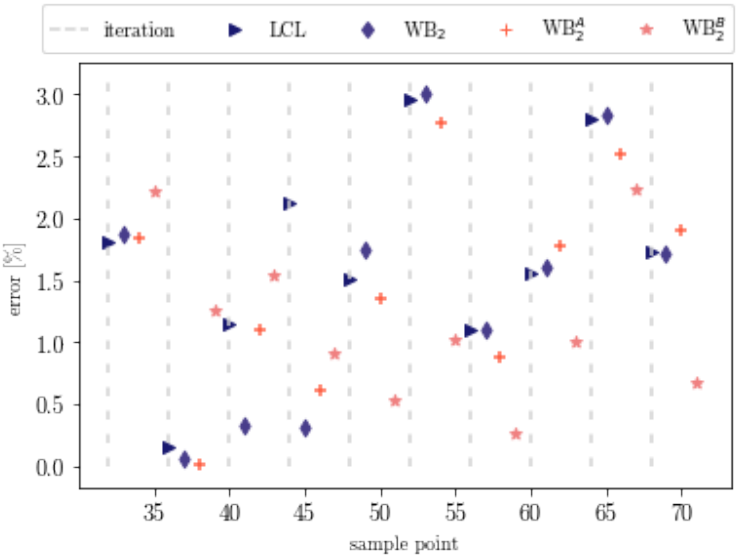


Figure 46: Kriging predictor errors given by (45) when running the KPLS algorithm with parameter setting S1 on real airline data and only considering the exploiting ISC for finding new evaluation points.

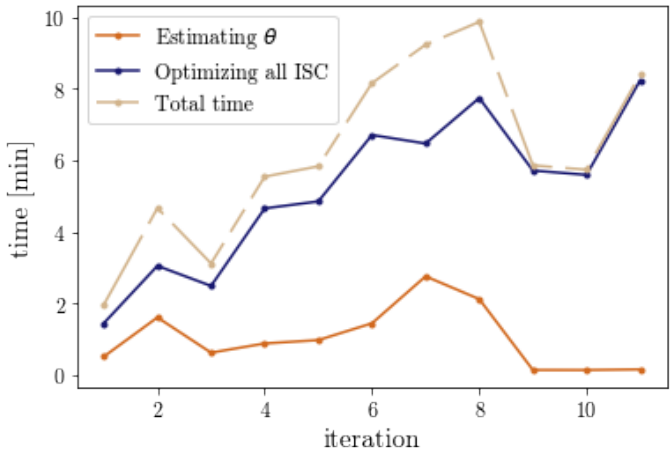


Figure 47: Computation times for each iteration of the KPLS algorithm with parameter setting S1 on real airline data, when only the exploiting ISC are considered for finding new evaluation points.

8 Discussion and future research

The goal of this thesis work was to build an algorithm for modifying vacancy announcement and the amount of staffing deviation allowed (so-called backfilling) within the Manpower Planning framework used at Jeppesen. The algorithm was implemented in Python, in order to be easily integrated into the already existing software. The algorithm, which is referred to as KPLS in the literature, was implemented to solve general planning problems. So was also the extension KPLS+K, which includes an additional estimation of the surrogate model parameters. Both versions of the algorithm were evaluated using a benchmark function with a known minimum value, in order to assess their performance quality. Thereafter, they were evaluated using real airline data for two different problem

settings. In this section, the results presented in the previous section will be discussed, as well as the algorithm in general. Possible improvements will be presented, as well as future research.

8.1 Method selection

Constructing the surrogate model

In constructing the surrogate model, Kriging was chosen as the basis method. The reason to choose Kriging was its ability to measure the uncertainty in the predictor itself. Other, similar methods have been considered, such as surrogate models based on radial basis functions. The (in literature stated) improved performance of the Kriging method compared to the RBF method was preferred in this case. However, the RBF method is simpler, requiring only the solution of a system of linear equations, compared to maximizing a likelihood function as required in the Kriging method. The trade-off between performance and simplicity have not been investigated further in this project.

Except surrogate modelling techniques as Kriging and RBF, there are other possible approaches on how to face the problem. In these days, machine learning algorithms have become very popular, e.g., artificial neural networks (ANN) and support vector machines (SVM). However, they generally require a huge amount of data in order to provide good results. That is the main reason why such algorithms are discarded, since it would take a vast amount of time to generate the data needed to train the algorithm. Surrogate models like Kriging, which do not require the evaluation of particularly many sample points, was the better option for this very problem.

Due to the problem being high-dimensional, dimensionality reduction techniques were invoked, to decrease computation times. The KPLS model (and its extension KPLS+K) was chosen due to its ability of maintaining good accuracy while saving CPU time, as stated in [13]. Within this project, no explicit comparison of the accuracy and the CPU time between KPLS and full-dimensional Kriging was performed. However, since the estimation of the parameter η (a local optimization, consisting of only one iteration of the L-BFGS-B algorithm) usually took several minutes, we expect a full-dimensional estimation of the parameter θ to be an unreasonably heavy computation.

In [13], the number of principal components is set using a leave-one-out cross-validation procedure. First, h different KPLS models are constructed, where $h = 1, 2, ...$ denotes the number of principal components used. Then, the number h is chosen as the number of components which minimizes the leave-out cross-validation error. Since it takes a noticeable time to estimate θ, especially for larger values of h, we chose to fix $h = 3$ within this project. The choice $h = 3$ is a good trade-off between surrogate model accuracy and computation time and the same approach is used in, e.g., [18].

The optimization algorithm was constructed mainly using the ideas presented in [13], [15], and [18]. In these papers, the parameter $p \equiv 2$ (in the correlation function (8)) as a simplification. Since the problem considered in this thesis is high dimensional, the estimation of an additional parameter vector would make the computation times increase considerably. Recalling the interpretation of the parameter vector p, it determines the function's smoothness in each direction $j = 1, ..., d$. Values p_j close to 2 serve to model smooth functions, while values of p_j close to 0 are more suitable for modeling less smooth functions. The benchmark problems yielded solutions with good quality, which can be explained by the fact that the benchmark function (44) is very smooth. When it comes to the black-box function, we can not establish its smoothness with certainty. However, the observation that some pair of points which differed only by a few units gave rise to essentially different costs indicates that the black-box function is less smooth. As a future revision of the algorithm, the value of the parameter p should therefore be decreased.

Design of experiments

As seen in Figure 18, the distances between points do not decrease as the sample size does. The reason for this is that the search space is huge. For the S1 problem instance, the total number of points within the search space is $30^{37} \approx 5 \cdot 10^{54}$, which means that it is impossible to fill the space with as few as $\approx 10^2$ points. This motivates a larger initial sample size. On the other hand, an increased sample size

increases computation times both for estimating parameters and for finding new points to evaluate, and hence this trade-off must be taken into consideration. In the evaluations above, the start samples have a size of 32, where 30 of the points are constructed using Latin Hypercube design and the remaining two samples constitutes the minimum and maximum values of the variable domains. No further investigation of the algorithm performance using different start samples have been performed in this project, but can easily be done. However, the variable domains must be chosen in accordance, such that the LHD sample can be constructed properly. For example, if a LHD sample of size 50 is constructed for problem setting S1, the range of the integral variable domain Ω must be a multiple of 50 (or the other way around), i.e. $\Omega = [-50k + 1, 0], k \geq 1$, or $\Omega = [-50/k + 1, 0]$ such that $-50/k \in \mathbb{N}$, which changes the solution set. The choice $\Omega = [-29, 0]$ is a suitable variable range for the airline data considered, and that motivates the use of a LHD sample consisting of 30 points. However, using the logic just presented a LHD sample of size $30 \cdot 2 = 60$ would do as well, but the decreased computation time using a 30 point LHD sample was preferred in this case.

Infill sampling criteria

Another trade-off which have been considered is how many sample points to evaluate in each iteration. We want to utilize the fact that evaluations of the black-box function can be done in parallel, by adding several points to the sample in each iteration. By doing so, the number of iterations needed for the algorithm to find good solutions is decreased. However, as mentioned in Section 5.1.3, it is not beneficial to add a huge number of points in each iteration. From the start, the surrogate model is approximated from a small number of sample points, which means that it is not very accurate. Therefore, the points found by the different infill sampling criteria are optimized based on a deficient surrogate model and will unlikely coincide with the true minimum. As we have seen, the computation times generally increase with an increased sample size, and thus a reasonable number of sample points should be added in each iteration, so that computation times do not become too heavy already from the start.

In the cases evaluated in Section 7, eight different ISC were considered, which focused on both exploration and exploitation. We observed that the ISC focusing on exploration yielded points associated with high costs, which motivated us to consider a new setting where only exploiting ISCs were used to find new points to evaluate. The result of running the KPLS algorithm on this new, modified S1 setting was presented in Table 8. The KPLS algorithm found good solutions, one even better than the best solution found using the full-ISC setting. From these observations, we conclude that the exploiting ISC are the most important criteria for finding good solutions and that the sample points found by the exploring ISC are more or less redundant. This might as well mean that the surrogate model used is unnecessarily complex, and that a simpler method than Kriging could have been sufficient. One advantage with the Kriging method is its ability to predict the uncertainty in the model itself, but since the ISC focusing on maximizing the model uncertainty were more or less redundant, the uncertainty prediction was not particularly useful. Therefore, a general suggestion is to consider a surrogate model based on a simpler method than Kriging.

Numerical optimization

The L-BFGS-B algorithm was used rather than the COBYLA algorithm, due to lack in performance of the latter (e.g., COBYLA produced worse solutions compared to the L-BFGS-B algorithm and did not maintain the box constraints). However, there are still a few problems related to the use of the L-BFGS-B method within the `minimize` method in the Python package `scipy.optimize`. It happens that the L-BFGS-B function does not manage to find an improved solution, compared to the initial vector sent as input argument. In the case of estimating the parameter vector θ, this means that the surrogate model is not updated between two iterations, since the start vector $\theta^{(0)}$ is set as the parameter vector θ obtained in the previous iteration. This behavior can be observed in many of the plots visualizing computation times, e.g., in Figures 27, 37, and 47. The time of estimating θ

is dropping as no improved surrogate model can be found by the L-BFGS-B method. Ideally, the surrogate model should be improved in each iteration of the algorithm, and if it is not it may result in incorrect points being evaluated. In the KPLS+K case, the problem becomes less significant, since that algorithm includes an additional parameter estimation which can retrieve the improper initial parameter estimation. However, the fact that the algorithm lacks in performance is still inconvenient.

Another problem which had to be dealt with concerning the L-BFGS-B function was the fact that some updates produced within the L-BFGS-B procedure become infeasible. For example, for different choices of the parameter vector θ the correlation matrix (8) can become singular and thus non-invertible. Such cases have to be taken care of separately, e.g., by penalizing (giving high costs to) parameters θ related to correlation matrices with determinants very close to zero. Such problems appeared often for parameters θ and η including very small entries. In fact, both θ and η are bounded to be strictly above zero. In the implementation, these lower bounds have to be chosen explicitly. In order to avoid the singularity problems, sufficiently large bounds are chosen, namely $\theta \geq 10^{-4}$ and $\eta \geq 10^{-6}$.

The reason why singularity problems appear is probably because of sample points being almost equal, due to some of the infill sampling criteria being similar. Even if we relax the requirement that at least n_L points must be added in each iteration (which results in similar sample points, due to the fact that the noise applied only changes a few of the variables by one unit), the singularity problem will still be present. This is because the convergence to a minimum means that several points are sampled in the same area. The problem becomes even more evident since the variables are integer valued and the variable ranges are relatively small.

Apart from all these inconveniences, L-BFGS-B was the best choice among the methods available in the `scipy` package in Python. For an improved algorithm, another numerical optimization tool should be used, e.g., the partitioning algorithm DIRECT (shorthand for DIviding RECTangles). The absence of an easy-to-use implementation of DIRECT led to the rejection of the method within this project, and it can be considered as a future revision. Implementing a numerical optimization procedure from scratch is also a possible approach.

8.2 Algorithm evaluation

Starting off with the evaluation of the algorithms using the benchmark function (44), we were not able to find the true minimum before termination, but points relatively close to the optimum were discovered. Beginning with points further away than 59 length units (Euclidean distance in $d = 37$ dimensions), the KPLS+K algorithm was able to find a point 5.83 length units away from the true minimum, with a corresponding cost of 20,030,600 (0.15% greater than the minimum). The fact that the algorithm performed well on the benchmark problem motivated us to stay with the implementation and apply it on real airline data. However, as mentioned in the previous section, the benchmark function is much smoother than the black-box function representing the cost produced by Jeppesen's optimizer. This becomes evident when evaluating the algorithms on real airline data, observing, e.g., that small changes in a few of the variables can give rise to great cost increments.

It is not surprising that the algorithm performed better using a convex objective function. When the surrogate model was approximated and when new points were to be evaluated, the numerical optimization algorithm L-BFGS-B algorithm was employed, which is based on the approximation of the Hessian matrix. The approximation of the Hessian matrix will probably be more precise considering the benchmark function (44), since it is in fact constant (the true objective is a second order polynomial). When it comes to the Hessian of the black-box function, there is no explicit formulation to make use of, and little is known about its characteristics, except that there are many irregularities. The irregularities, such that small changes in a few number of variables may result in large cost changes, probably makes it more difficult for the L-BFGS-B algorithm to produce accurate surrogate models.

In contrast to the evaluation of the benchmark problem, the KPLS+K algorithm did not perform better than the KPLS algorithm when real airline data was considered. For the S1 problem setting, the

overall results were similar, but the KPLS algorithm managed to find a slightly lower cost compared to the KPLS+K algorithm. Since the KPLS computation times are lower than the KPLS+K computation times, we can conclude that the KPLS algorithm is preferred rather than the KPLS+K algorithm in this case. This means that the parameter estimation of the KPLS algorithm is sufficient, or at least that the improved parameter estimation in the KPLS+K algorithm does not improve the fit of the surrogate model to the black-box model. Nevertheless, both algorithms find many solutions with lower costs compared to the manually constructed reference solution. Even though the computation times are heavy, they probably improve upon the time it takes to construct a good solution manually, i.e., modify all vacancies and decide how much staffing deviation to allow in each of the groups. Moreover, in addition to the manually constructed reference solution, a useful idea would be to construct one more reference solution. Instead of being constructed manually, this other reference solution should rather be the result of some basic heuristic approach. As a future work, it would be interesting to see if the algorithms presented in this work are able to find better solutions compared to the results obtained via a basic heuristic approach, which has not been implemented here.

Considering the S2 problem setting, neither of the algorithms perform better than when the S1 problem setting was used. However, with the S2 setting, the KPLS+K algorithm still managed to find solutions associated with lower costs compared to the reference solution, but the KPLS algorithm did not. The reason why the algorithms perform better on the S1 setting may be due to the vacancy announcements being relatively good, while there is much room for improvements when it comes to the staffing deviation, which is the only thing being optimized using problem setting S1.

Another behavior which can be observed from many of the evaluations in Section 7 is that the evolution of sample points does not converge to a minimum, the costs are rather increased after the lowest cost have been found. One reason for this behavior may be that an interpolating surrogate model is used. As mentioned in Section 2.1, interpolation can lead to over-fitting, meaning that the model used for interpolating the sample points becomes overly complex. Over-fitted models can, e.g., result in artificial bumps, which makes the Kriging prediction of the minimum erroneous. As a future work, the over-fitting problem can be studied in more detail.

9 Conclusions

In order to modify the announced job vacancies and to allow staffing deviations, two optimization algorithms were presented, referred to as KPLS and KPLS+K. These algorithms were based on surrogate modeling, meaning that the true objective is approximated by a function, given a small set of sample points where the true costs are known. Both of the algorithms performed well on a benchmark problem, where the black-box function constituted a known, smooth function. The lowest cost found by the KPLS algorithm was 0.88 % larger than the actual function minimum, while the KPLS+K algorithm found a point with a corresponding cost which was only 0.15 % larger than the true minimum. The algorithms were then evaluated on real airline data provided by Jeppesen, which had a less smooth behavior. Two different settings were considered, referred to as S1 and S2. In the S1 setting, only groups where no job vacancy was announced were considered for optimization, while for the S2 setting, the announced vacancies were optimized as well. For the S1 setting, both the KPLS and the KPLS+K algorithms found solutions with lower costs compared to a manually constructed reference solution. The performance of the algorithms on the S2 setting was worse, but the KPLS+K algorithm still managed to find a few solutions associated with lower costs compared the reference solution, while the KPLS algorithm did not.

During simulations, it was observed that several of the infill sampling criteria (ISC) – those focusing on exploration – generally gave rise to high costs. Therefore, an additional evaluation of the S1 setting was performed, where only four exploiting ISCs were employed. The result even improved upon the result associated with the full-ISC setting. From the observations, we conclude that, for this problem, the exploring ISC are not important in order to find good solutions. This means that the Kriging method is probably unnecessarily complex for this problem, since its ability to predict the

model uncertainty is barely taken advantage of.

Several revisions of the model have been proposed, such as decreasing the parameter p (in the kernel function), testing algorithm robustness by using different start samples of different sizes, using a different approach for numerical estimation of the parameters θ and η, and optimizing the different ISCs. However, due to the indications of the Kriging method being unnecessarily complex, the first thing to consider before improving this particular algorithm should be implementing a simpler surrogate model, e.g., based on RBFs. Moreover, there are many interesting problems and extensions to be investigated within the Manpower Planning framework. For example, a similar problem to this can be found at the beginning of the Manpower Planning procedure presented in Figure 2, where also uncertainty is present.

References

[1] L. A. Karlberg, "Doktorstäta Carmen Systems säljs för 1 miljard," *NyTeknik*, 2006-03-03. url: https://www.nyteknik.se/digitalisering/doktorstata-carmen-systems-saljs-for-1-miljard-6438840, 2020-04-20.

[2] P. Jiang, Q. Zhou, and X. Shao, "Surrogate-model-based design and optimization," in *Surrogate Model-Based Engineering Design and Optimization*, pp. 2–34, Springer, 2020.

[3] D. G. Krige, "A statistical approach to some basic mine valuation problems on the Witwatersrand," *Journal of the Southern African Institute of Mining and Metallurgy*, vol. 52, no. 6, pp. 119–139, 1951.

[4] J. P. Kleijnen, "Design and analysis of simulation experiments," in *International Workshop on Simulation*, pp. 3–22, Springer, 2015.

[5] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of Global Optimization*, vol. 21, no. 4, pp. 345–383, 2001.

[6] H. Wendland, *Scattered Data Approximation*, vol. 17. Cambridge, UK: Cambridge Monographs on Applied and Computational Mathematics, 2005.

[7] S. Jakobsson, M. Patriksson, J. Rudholm, and A. Wojciechowski, "A method for simulation based optimization using radial basis functions," *Optimization and Engineering*, vol. 11, no. 4, pp. 501–532, 2010.

[8] A. M. Yaglom, "Correlation theory of stationary and related random functions.," *Volume I: Basic Results.*, vol. 526 of Springer Series in Statistics, 1987.

[9] M. Schonlau, "Computer Experiments and Global Optimization," University of Waterloo, 1997.

[10] F. Bachoc, "Cross validation and maximum likelihood estimations of hyper-parameters of Gaussian processes with model misspecification," *Computational Statistics & Data Analysis*, vol. 66, pp. 55–69, 2013.

[11] M. J. Sasena, *Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations*. PhD thesis, University of Michigan Ann Arbor, MI, 2002.

[12] J. Sacks, S. B. Schiller, and W. J. Welch, "Designs for computer experiments," *Technometrics*, vol. 31, no. 1, pp. 41–47, 1989.

[13] M. A. Bouhlel, N. Bartoli, A. Otsmane, and J. Morlier, "Improving kriging surrogates of high-dimensional design models by partial least squares dimension reduction," *Structural and Multidisciplinary Optimization*, vol. 53, no. 5, pp. 935–952, 2016.

[14] R. A. Pérez and G. González-Farias, "Partial least squares regression on symmetric positive-definite matrices," *Revista Colombiana de Estadística*, vol. 36, no. 1, pp. 177–192, 2013.

[15] M. A. Bouhlel, N. Bartoli, A. Otsmane, and J. Morlier, "An improved approach for estimating the hyperparameters of the kriging model for high-dimensional problems through the partial least squares method," *Mathematical Problems in Engineering*, vol. 2016, 2016.

[16] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA, 1950.

[17] R. Manne, "Analysis of two partial-least-squares algorithms for multivariate calibration," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1–3, pp. 187–197, 1987.

[18] M. A. Bouhlel, N. Bartoli, R. G. Regis, A. Otsmane, and J. Morlier, "Efficient global optimization for high-dimensional constrained problems by using the kriging models combined with the partial least squares method," *Engineering Optimization*, vol. 50, no. 12, pp. 2038–2053, 2018.

[19] C. Li, S. Gupta, S. Rana, V. Nguyen, S. Venkatesh, and A. Shilton, "High dimensional Bayesian optimization using dropout," *arXiv preprint arXiv:1802.05400*, 2018.

[20] E. R. van Dam, B. Husslage, D. den Hertog, and H. Melissen, "Maximin latin hypercube designs in two dimensions," *Operations Research*, vol. 55, no. 1, pp. 158–169, 2007.

[21] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[22] K. D. Tocher, "The art of simulation," English Universities Press, London, 1967.

[23] D. Raj, "Sampling theory," McGraw-Hill, New York, 1968.

[24] B. Tang, "Selecting Latin Hypercubes using correlation criteria," *Statistica Sinica*, vol. 8, pp. 965–977, 1998.

[25] M. D. Morris and T. J. Mitchell, "Exploratory designs for computational experiments," *Journal of Statistical Planning and Inference*, vol. 43, no. 3, pp. 381–402, 1995.

[26] I. O. Bohachevsky, M. E. Johnson, and M. L. Stein, "Generalized simulated annealing for function optimization," *Technometrics*, vol. 28, no. 3, pp. 209–217, 1986.

[27] W. W. Li and C. Jeff Wu, "Columnwise-pairwise algorithms with applications to the construction of supersaturated designs," *Technometrics*, vol. 39, no. 2, pp. 171–179, 1997.

[28] H. J. Kushner, "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise," *Journal of Basic Engineering*, vol. 86, no. 1, pp. 97–106, 1964.

[29] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of Bayesian methods for seeking the extremum," *Towards Global Optimization*, vol. 2, no. 117-129, p. 2, 1978.

[30] M. J. Powell, "A direct search optimization method that models the objective and constraint functions by linear interpolation," in *Advances in Optimization and Numerical Analysis*, pp. 51–67, Springer, 1994.

[31] J. Nocedal, "Updating quasi-Newton matrices with limited storage," *Mathematics of Computation*, vol. 35, no. 151, pp. 773–782, 1980.

[32] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, no. 1-3, pp. 503–528, 1989.