Imperial College London
Department of Computing

Formulations, Algorithms and Software for Robust Optimisation

Johannes Wiebe

Advisor: Prof. Ruth Misener

Submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy at
Imperial College London, August 2021

I hereby declare that the content of this thesis is my own original work. Information derived from the work of others is acknowledged in the text and as a list of references in the bibliography.

Johannes Wiebe

Abstract

Equipment degradation is a common phenomenon in engineering applications. Unexpected equipment failures can lead to process downtime and significant cost. Information regarding the health of equipment is often uncertain and based on limited, potentially noisy or corrupted data. Optimising the production and maintenance of technical processes often requires taking these challenging aspects into consideration. To this end, this thesis applies data-driven robust optimization to applications with degrading equipment and develops new methods and tools in the process.

As a first step, we consider a variant of the state-task-network with uncertain equipment degradation. We use data-based stochastic process models common in condition-based maintenance to construct uncertainty sets. This allows us to incorporate information from the stochastic models into a process level mixed-integer optimization problem while retaining computational tractability. We apply this framework to several instances of the state-task-network and demonstrate that it can efficiently compromise between equipment availability and cost of maintenance.

Next, we extend our approach to an industrial drill scheduling case study. The uncertain degradation in this problem is represented by black-box constraints. We model these uncertain functions using (warped) Gaussian processes, use the models to generate uncertainty sets, and develop new reformulations for the resulting robust optimization problems. Unlike most robust optimization problems, this approach considers uncertain functions, not uncertain parameters. We analyze convexity conditions and propose a custom global optimization strategy for non-convex cases. The approach effectively mitigates uncertainty in the learned degradation curves.

Finally, we aim to make the methods developed and robust optimization as a whole more accessible to practitioners. To this end we introduce ROmodel, a Python package that extends the modeling capabilities of the popular modeling language Pyomo to robust optimization problems and automates the reformulation of robust optimization problems. ROmodel allows the definition of custom uncertainty sets using Pyomo constraints, but also contains a library containing commonly used uncertainty sets and our (warped) Gaussian process-based sets. ROmodel implements both robust reformulation and a cutting plane based solver. We demonstrate ROmodel's capabilities by applying it to a number of instances of several case studies.

Acknowledgements

This thesis would not have been possible without the support of a number of people. First and foremost I would like to thank my supervisor Ruth Misener for being an incredibly supportive mentor and always having words of advice for me.

Second, I would like to thank my supervisor at Schlumberger Cambridge Research, Inês Cecílio, for all her support throughout the years. I would also like to thank her and all other colleagues at Schlumberger, especially Jonathan Dunlop and Ali Özbek, for listening to my presentations and always giving me great feedback and helping me to ground my work in practially relevant problems.

Third, I would like to thank my colleagues Georgia Kouyalis, Simon Olofsson, Miten Mistry, Juan Campos Salazar, Dimitris Letsios, Francesco Ceccon, Alex Thebelt, Chryssa Kappatou, Calvin Tsay, Daniel Lengyel, and James Odgers. You have made the office such a friendly and enjoyable space. Thank you for our valuable discussions and for helping me procrastinate over coffee/tea/hot water and biscuits.

Fourth, I would like to thank everyone in HiPEDS for making my first year in the CDT space so enjoyable and insightful. Thanks to Dan Iorga, Georgios Rizos, and George Theodorakis for always being up for a drink at $\hbar$ (or elsewhere). Special thanks to Dan for always being up for a swim or the gym the next morning.

Finally, I would like to thank my friends and family for their support. Special thanks to Lara Waltersmann and Janine Möller for our regular phone calls, Sarah Kakadellis for lunch and slackline breaks, and Alexandra Stoikova for riverside walks and drinks. I'd like to thank Helena Arellano-Ballestero for her endless support and for keeping me sane during Covid lockdowns. Last but not least, I am grateful to my parents, Doris Röskenbleck and Frank Wiebe, for always supporting me and encouraging me to do what I am passionate about.

List of publications and presentations

0.0.1 Publications

- **Wiebe, J.**, Misener, R. (2021) ROmodel: Modeling robust optimization problems in Pyomo. *Pre-print, arXiv*: `https://arxiv.org/abs/2105.08598` .
- **Wiebe, J.**, Cecílio I., Dunlop, J., Misener, R. (2020) A robust approach to warped Gaussian process-constrainied optimization. *Pre-print, arXiv*: `http://arxiv.org/abs/2006.08222`.
- Letsios, D., Baltean-Lugojan, R., Ceccon, F., Mistry, M., **Wiebe, J.**, Misener, R. (2020) Approximation algorithms for process systems engineering. *Computers & Chemical Engineering*, 132:106599.
- **Wiebe, J.**, Cecílio I., Misener, R. (2019) Robust optimization for the pooling problem. *Industrial & Engineering Chemistry Research*, 58(28):12712-12722.
- **Wiebe, J.**, Cecílio I., Misener, R. (2018) Data-Driven Optimization of Processes with Degrading Equipment. *Industrial & Engineering Chemistry Research*, 57(50):17177-17191.

0.0.2 Presentations

- **Wiebe, J.**, A robust approach to warped Gaussian process-constrained optimization. *Virtual 2020 INFORMS Annual Meeting*, November 2020.
- **Wiebe, J.**, Robust optimization of pooling networks. *Foundations of Computer-Aided Process Design*, July 2019.
- **Wiebe, J.**, The robust pooling problem. *European Symposium on Computer Aided Process Engineering*, June 2019.
- **Wiebe, J.**, The robust pooling problem. *Young Professionals Conference on Process Engineering*, March 2019.
- **Wiebe, J.**, Robust planning and scheduling for processes with equipment degradation. *Annual AIChE Meeting*, November 2018.

0.0.3 Open-Source Software

- `https://github.com/cog-imperial/romodel`: A Python module which extends Pyomo for modeling and solving robust optimization problems.
- `https://github.com/cog-imperial/rogp`: A Python module for including Gaussian processes trained in GPy into Pyomo models.
- `https://github.com/cog-imperial/stn`: Pyomo implementation of a robust version of the state-task-network with combined scheduling, planning, and equipment degradation.
- `https://github.com/cog-imperial/drill-scheduling`: Pyomo implementation of a drill scheduling case study which uses the ROGP module to combine Gaussian Processes with robust optimization.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Robust optimization first emerged as a method for optimization under uncertainty when Soyster [6] introduced the concept in 1973 and gained traction after seminal works by Ben-Tal and Nemirovski [7] and Bertsimas and Sim [8]. Since then, the field has seen many contributions and subfields such as distributionally robust optimization, data-driven robust optimization, non-linear robust optimization, and more have emerged. Robust optimization has been applied to many applications, especially within process systems engineering, but has not yet been applied to uncertain equipment degradation.

1.1 Motivation and objectives

The purpose of this thesis is to explore the applicability of robust optimization to new, relevant engineering applications, in particular uncertain equipment degradation. To this end, this work aims to accomplish a number of objectives:

1. Apply robust optimization to novel, interesting applications with uncertain equipment degradation and explore the effect uncertainty has on optimal decision making in these problems.

2. Develop novel robust optimization methods for efficiently incorporating data-driven, stochastic models into (process-level) optimization problems. Stochastic models are often used to model equipment degradation in the condition-based maintenance literature.

3. Develop tools to make the transition from deterministic to robust optimization easier and to facilitate the development and comparison of new robust optimization reformulations and algorithms.

1.2 Contributions

In line with the objectives outlined above, the contributions of this work fall into three categories.

We explore the application of robust optimization to two engineering problems with uncertain equipment degradation. First, we consider a version of the classic state-task-network which combines scheduling with a planning horizon and maintenance scheduling. We consider uncertainty in equipment degradation and demonstrate how this uncertainty affects the optimal decision. We then consider a drill scheduling problem which combines maintenance scheduling with a non-linear, non-convex drillbit-rock interaction model. We again consider uncertainty in equipment degradation.

We develop methods for incorporating multiple types of stochastic problems into optimization problems using robust optimization. We use Lévy type stochastic processes which are commonly used as degradation models in condition-based maintenance to generate uncertainty sets for robust optimization. We also show how (warped) Gaussian processes can be incorporated into process level optimization problems using a chance-constraint reformulation and develop a Wolfe duality-based reformulation for optimization problems containing warped Gaussian processes.

Finally, we present two open-source projects. The first project, ROmodel, extends the Pyomo modeling language to facilitate intuitive modeling of robust optimization problems. It makes

it easy to transition from deterministic to robust optimization and to compare different types of uncertainty sets and solution methods. It is also extensible and can act as a platform for the development of new uncertainty set geometries, reformulations, and algorithms. The second project, ROGP, can be used to incorporate (warped) Gaussian processes into Pyomo models. The combination of both projects makes our methods for optimization with Gaussian Process-based constraints very easy to apply. (Warped) Gaussian processes trained in the Python library GPy can directly be used to specify custom uncertainty sets in ROmodel.

1.3 Thesis Outline

Chapter 2 introduces some important robust optimization concepts as well as the stochastic processes used throughout this thesis. Chapter 3 discusses the problem of uncertain equipment degradation using the example of the state-task-network and shows how robust optimization can be used to incorporate information from stochastic degradation models into process level optimization. Chapter 4 extends this by considering (warped) Gaussian processes as degradation models for a drill scheduling case study. To this end it develops new robust reformulations for optimization problems with black-box constraints modeled by (warped) Gaussian processes. Finally, Chapter 5 presents ROmodel, an open-source extension of the modeling language Pyomo which can model and solve robust optimization problems.

Chapter 2

Background theory

2.1 Introduction

This chapter provides a short summary of robust optimization. We also discuss data-driven approaches in robust optimization and review stochastic models commonly used in process systems engineering.

2.2 Robust optimization

Consider a generic robust optimization problem:

$$\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}) \tag{2.1a}$$

$$\text{s.t. } g(\boldsymbol{x},\boldsymbol{\xi}) \leq b \qquad \forall \boldsymbol{\xi} \in \mathcal{U}, \tag{2.1b}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the vector of decision variables, $\boldsymbol{\xi} \in \mathbb{R}^m$ the vector of uncertain parameters, $f(\cdot)$ the objective function, $g(\cdot,\cdot)$ an uncertain constraint, $\mathcal{U}$ a non-empty, compact set containing possible realizations of $\boldsymbol{\xi}$, and $\mathcal{X}$ is the certain feasible region (which does not depend on $\boldsymbol{\xi}$). We assume f and g are continuous. Note that the objective does not depend on $\boldsymbol{\xi}$ and that we are only considering a single constraint. This formulation is without loss of generality, because

an uncertain objective can always be transformed into an uncertain constraint using an epigraph formulation and because robust optimization usually considers each uncertain constraint separately (although there are settings where constraints can be modeled jointly [9, 10]). In robust optimization, it is often useful to rewrite Constraint (2.1b) as an inner maximization problem, leading to an equivalent bilevel formulation [11]:

$$\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}) \tag{2.2a}$$

$$\text{s.t.} \max_{\boldsymbol{\xi}\in\mathcal{U}} g(\boldsymbol{x},\boldsymbol{\xi}) \leq b. \tag{2.2b}$$

The complexity of Problem (2.1) and the applicable solution techniques largely depend on the convexity/concavity of $f(\boldsymbol{x})$, $g(\boldsymbol{x},\boldsymbol{\xi})$, $\mathcal{X}$, and $\mathcal{U}$ with respect to $\boldsymbol{x}$ and $\boldsymbol{\xi}$ [12]. Four categories of problems exist:

1. $g(\boldsymbol{x},\boldsymbol{\xi})$ and $f(\boldsymbol{x})$ are convex in $\boldsymbol{x}$ and concave in $\boldsymbol{\xi}$ and $\mathcal{U}$ and $\mathcal{X}$ are convex.

2. $g(\boldsymbol{x},\boldsymbol{\xi})$ is concave in $\boldsymbol{\xi}$ and $\mathcal{U}$ is convex, but $g(\boldsymbol{x},\boldsymbol{\xi})$ or $f(\boldsymbol{x})$ is non-convex in $\boldsymbol{x}$ or $\mathcal{X}$ is non-convex.

3. $g(\boldsymbol{x},\boldsymbol{\xi})$ and $f(\boldsymbol{x})$ are convex in $\boldsymbol{x}$ and $\mathcal{X}$ is convex, but $g(\boldsymbol{x},\boldsymbol{\xi})$ is non-concave in $\boldsymbol{\xi}$ or $\mathcal{U}$ is non-convex.

4. $g(\boldsymbol{x},\boldsymbol{\xi})$ is non-concave in $\boldsymbol{\xi}$ or $\mathcal{U}$ is non-convex and $g(\boldsymbol{x},\boldsymbol{\xi})$ or $f(\boldsymbol{x})$ is non-convex in $\boldsymbol{x}$ or $\mathcal{X}$ is non-convex.

Fig. 2.1 summarizes which methods are applicable to each category. Most work in the robust optimization literature falls into to category 1, also called convex robust optimization. category 1 includes linear programming problems with different types of convex uncertainty sets, e.g., box sets [6], ellipsoidal sets [7], and polyhedral sets [8]. It also includes non-linear convex problems, e.g., convex-quadratic problems [9, 13], semi-definite programs [9], or general convex constraints [14]. Both robust reformulation approaches and robust cutting planes have been applied extensively category 1 and solutions can often be obtained in polynomial time.

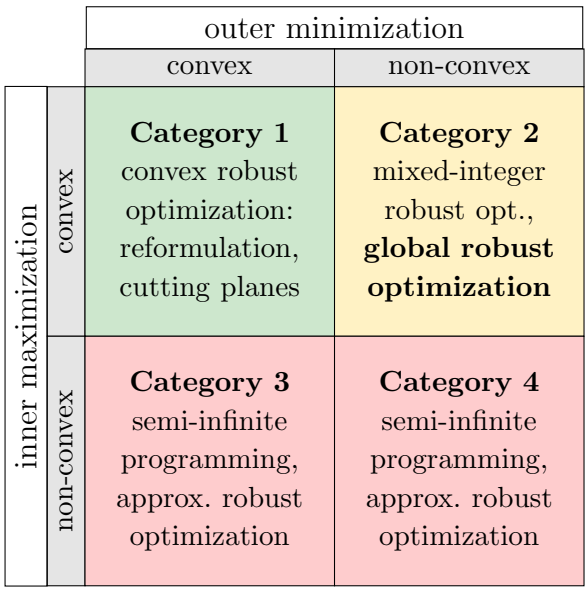


Figure 2.1: Complexity of Problem (2.1) for different convexity assumptions for the outer minimization and inner maximization in Problem (2.2). The inner maximization is non-convex when $g(\cdot)$ is non-concave or $\mathcal{U}$ is non-convex.

Categories 3 and 4 are generally much harder, because the inner maximization Problem (4.9b) is a non-convex optimization problem. Furthermore, even finding a feasible solution to Problem (2.1) requires solving this non-convex problem to global optimality. Problems in these categories can generally only be solved using semi-infinite programming techniques [15, 16, 17], which are limited to small scale problems, or approximate robust optimization schemes [18, 19, 20, 21].

Category 2 has received limited attention in the robust optimization and semi-infinite programming communities [22, 23, 24]. A particular subclass, (non-convex) MILP problems, has been studied extensively in the process systems engineering literature, e.g., several robust planning and scheduling applications [25, 26, 27, 28]. An example of a non-linear problem in category 2 to which robust optimization has been applied is the pooling problem [29]. The same methods (both reformulation and cutting plane approaches) which are used to solve problems in category 1 are generally also applicable to category 2, but solutions can rarely be found in polynomial time. Solving such problems effectively therefore requires a tighter integration between robust optimization and global optimization techniques.

2.2.1 Reformulation approaches

The robust reformulation approach was first proposed by Soyster [6] for LPs with a box uncertainty set and has since been developed for many classes of problems [7, 8, 9, 13, 14]. These approaches generally utilize strong duality by replacing the inner maximization in Problem (2.2) with its dual minimization problem:

$$\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}) \tag{2.3a}$$

$$\text{s.t.} \ \min_{\boldsymbol{\lambda}\in\Lambda} g^*(\boldsymbol{x},\boldsymbol{\lambda}) \leq b, \tag{2.3b}$$

where $g^*(\cdot,\cdot)$ is the objective of the dual problem, $\boldsymbol{\lambda}$ are the dual variables, and Λ is the feasible space. Problem (2.3) can clearly be written as a single level optimization problem which can often be solved using off-the-shelf solver software:

$$\min_{\boldsymbol{x}\in\mathcal{X},\boldsymbol{\lambda}\in\Lambda} f(\boldsymbol{x}) \tag{2.4a}$$

$$\text{s.t.} \ g^*(\boldsymbol{x},\boldsymbol{\lambda}) \leq b. \tag{2.4b}$$

2.2.2 Cutting plane approaches

Robust cutting plane approaches outer-approximate the feasible space by replacing the uncertainty set $\mathcal{U}$ with a finite number of uncertainty realizations $\hat{\mathcal{U}}_n = \{\boldsymbol{\xi}_1,\ldots,\boldsymbol{\xi}_n\}$. A robustly feasible solution is found by iteratively adding uncertainty realizations which violate the constraint. The key idea is to alternately solve a master problem:

$$\boldsymbol{x}_n = \arg\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}) \tag{2.5a}$$

$$\text{s.t} \ g(\boldsymbol{x},\boldsymbol{\xi}) \leq b, \quad \forall \boldsymbol{\xi} \in \hat{\mathcal{U}}_n, \tag{2.5b}$$

and a separation problem:

$$\boldsymbol{\xi}_{n+1} \in \arg\max_{\boldsymbol{\xi}\in\mathcal{U}} \; g(\boldsymbol{x}_n, \boldsymbol{\xi}), \tag{2.6}$$

which determines whether the current solution $\boldsymbol{x}_n$ is robustly feasible ($\max_{\boldsymbol{\xi}\in U} g(\boldsymbol{x}_n, \boldsymbol{\xi}) \leq 0$) and, if it is not, generates a new cut $\boldsymbol{\xi}_{n+1}$.

Robust cutting planes are similar to the concept of *backoff* in the control literature [30, 31, 32, 33]. The backoff approach generally also starts by solving the nominal problem and then subsequently "backing off" to suboptimal but more robust solutions which remain feasible in light of uncertainty. While the robust optimization approach assumes that the uncertain parameter is within a known uncertainty set, uncertainty in the backoff approach is usually due to process dynamics.

2.2.3 Data-driven robust optimization

A common challenge in robust optimization is defining a suitable uncertainty set for a given application. The approach in *data-driven robust optimization* is to construct the uncertainty set from available data. Early approaches in data-driven robust optimization focused on constructing uncertainty sets from data using confidence regions and statistical hypothesis testing [34]. Many recent contributions have utilized machine learning to construct data-driven uncertainty sets. Campbell et al. [35] construct unions of ellipsoidal uncertainty sets from Dirichlet process mixture models. Ning and You [36] also use Dirichlet process mixture models but construct polyhedral sets instead. The Dirichlet approach has the advantage that it can capture multimodal uncertainties. It may therefore be particularly applicable to, e.g., chemical engineering data sets where multimodal uncertainty is common due to distinct operating modes and other factors. Other contributions use unsupervised learning to construct data-driven uncertainty sets. E.g. Shang et all [37] use kernel-based support vector clustering to derive data-driven uncertainty sets while Goerigk and Kurtz (2020) [38] construct sets from the output of an unsupervised deep neural network. Deep neural networks have also been used in the context of

distributionally robust and chance constraint optimization. Zhao et al [39] utilize generative adversarial networks (GANs) to construct empirical distributions based on (noisy) data. They then create ambiguity sets for distributionally robust optimization based on these distributions.

2.3 Stochastic models

Many of the data-driven robust optimization approaches discussed above use stochastic processes to model the data. Stochastic models are often a better choice than deterministic models because they allow quantifying the uncertainty resulting from noisy or corrupted data. Stochastic processes can be interpreted as probability distributions over functions. As such, they are particularly useful for modeling uncertainty in functional dependencies based on noisy or corrupted data, i.e., black-box function uncertainty. Simple Lévy type stochastic processes, like the Wiener or Gamma process, are commonly used in data-based condition monitoring/equipment degradation [40]. Another commonly used class of stochastic processes are Gaussian processes. Gaussian processes are extensively used as surrogate models in chemical engineering, but have also been used for applications with black-box uncertainty due to noisy measurements of functional dependencies [41, 42]. Lévy type processes and (warped) Gaussian processes are discussed in greater detail in Sections 2.3.2 and 2.3.3. Other stochastic models which have been applied in the context of data-driven robust optimization include Dirichlet processes, Gaussian mixture models [35, 36, 43], and Bayesian networks [44, 45].

2.3.1 Lévy type processes

Lévy type processes are an important class of stochastic processes commonly used in degradation modeling [46]. They are defined as:

Definition 2.1. ***Lévy type process*** *[47]. A stochastic process* $S(t) = \{S_t : t \in T \subseteq \mathbb{R}\}$*, where* S_t *is a random variable, with*

1. *independent increments:* $S_{t_2} - S_{t_1}, \ldots, S_{t_n} - S_{t_{n-1}}$ *are independent for any* $0 < t_1 < t_2 < \ldots < t_n < \infty$,

2. *stationary increments:* $S_t - S_s$ *and* $S_{t-s} - S_0$ *have the same distribution for any* $s < t$,

3. *continuity in probability:* $\lim_{h \to 0} P(|S_{t+h} - S_t| > \epsilon) = 0$ *for any* $\epsilon > 0$, $t \geq 0$.

2.3.2 Gaussian processes

Gaussian Processes (GPs) are widely used for Bayesian optimization and non-parametric regression [48, 49, 50].

Definition 2.2 (Gaussian process)**.** *A continuous stochastic process* $\{G_{\boldsymbol{x}} : \boldsymbol{x} \in \mathbb{R}^n\}$ *where for every finite set of vectors* $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l$ *the random variables* $G_{\boldsymbol{x}_1}, \ldots, G_{\boldsymbol{x}_l}$ *follow a multivariate Gaussian distribution.*

A GP can be fully specified by a mean function $m(\cdot)$ and kernel function $k(\cdot, \cdot)$. Given a set of N data points $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N], \boldsymbol{y} = [y_1, \ldots, y_N]^\intercal$ and using a zero mean function, we can predict the mean $\boldsymbol{\mu}$ and covariance matrix Σ of the multivariate Gaussian distribution defined by a set of new test points $X_* = [\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_n^*]$:

$$\begin{aligned}
\boldsymbol{\mu}(X_*) &= K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} \boldsymbol{z} \\
\Sigma(X_*) &= K(X_*, X_*) \\
&\qquad - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} K(X, X_*),
\end{aligned}$$

where σ_n is the standard deviation of noise in the data, $K(X_*, X) = K(X, X_*)^\intercal$ is the $n \times N$ covariance matrix between test and training points, $K(X, X)$ the $N \times N$ covariance matrix between training points, $K(X_*, X_*)$ the $n \times n$ covariance matrix between test points, and I the identity matrix. We denote the ij-element of Σ as $\sigma_{ij}^2 = \sigma^2(\boldsymbol{x}_i^*, \boldsymbol{x}_j^*)$.

2.3.3 Warped Gaussian processes

The standard GP approach assumes that the data follows a multivariate Gaussian. While this assumption allows prediction using simple matrix multiplication, it may not be a reasonable assumption for non-Gaussian data [51]. A slightly more flexible model, which still retains many of the benefits of GPs, is the warped GP model. The key idea is to warp the observations $\boldsymbol{z}$ to a latent space $\boldsymbol{\xi}$ using a monotonic warping function $\boldsymbol{\xi} = h(\boldsymbol{z}, \boldsymbol{\Psi})$, where $\boldsymbol{\Psi}$ is a vector of parameters. A standard GP then models the data in the latent space. The Jacobian $\frac{\partial h(\boldsymbol{z})}{\partial y}$ is included in the likelihood function of the GP and the GP and warping parameters are learned simultaneously. A common warping function is the neural net style function:

$$\xi_i = h(z_i) = z_i + \sum_{j=1}^{n_w} a_j \tanh(b_j(z_i + c_j)), \tag{2.7}$$

where $a_j \geq 0, b_j \geq 0, \forall j$ to guarantee monotonicity [51, 52, 53] and n_w is the number of warping terms. Note that we use $\boldsymbol{h}(\cdot)$ to denote the vector version $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^n, \boldsymbol{h}(\boldsymbol{z}) = [h(z_1), \ldots, h(z_n)]^\intercal$, which warps each component individually.

Chapter 3

Uncertain equipment degradation

3.1 Introduction

Most technical processes contain equipment which degrades over time due to its usage. Degradation may lead to serious equipment failures, unless **preventive maintenance** actions are scheduled regularly to restore equipment conditions. While frequent preventive maintenance can keep equipment availability high, it also incurs significant cost. At the same time, unexpected equipment failures can lead to loss of production and high **corrective maintenance** costs. Finding the optimal balance between preventive and corrective maintenance is difficult, because degradation tends to be at least partially random and the health state of equipment can often only be estimated from data subject to uncertainty. To make things worse, scheduling maintenance activities is not independent from production planning and scheduling. A unit undergoing maintenance might, for example, be unavailable for production. Furthermore, the state of equipment health tends to depend not only on the selected maintenance strategy, but also on the process operating strategy. Operating a process with a high throughput might enable higher production volumes and more sales, but could also cause more equipment degradation and therefore a higher maintenance cost. These interactions between process conditions, maintenance strategy, and the equipment's uncertain state of health make finding optimal and compatible maintenance and operating strategies a very challenging data-driven optimization

problem under uncertainty.

One way of reducing the equipment maintenance cost is to determine maintenance schedules based on information regarding the equipment's state of health collected through condition monitoring [54]. This is the **condition-based maintenance** (CBM) paradigm [46, 54, 55, 56]. Due to the increased availability of cheap sensors and thereby large quantities of system health data, CBM is becoming more attractive [57]. While much attention has been paid to data collection and processing and prognostic modeling, the objective is usually to minimize the cost of maintaining a single unit [46]. This means that interaction between maintenance strategies for each piece of equipment and the operating strategy of the entire process has largely been neglected.

However, these interactions have been considered by multiple authors in the context of **integrated maintenance scheduling and process optimization**. Early approaches in this field which explicitly model degradation assume constant, known reliability or decay curves [58, 59, 60, 61, 62]. Dedopoulos and Shah [58, 59] combine short-term stochastic scheduling with long-term maintenance scheduling in a two-step procedure, while Vassiliadis and Pistikopoulos [61] determine optimal availability thresholds at which maintenance should be performed. Georgiadis et al. [63] optimize the cleaning and energy management of heat exchanger networks subject to fouling, which is assumed to follow a known profile. Liu et al. [64] consider scheduling of maintenance and biopharmaceutical batch production with a deterministic performance decay. Xenos et al. [65] optimize maintenance and production scheduling of a compressor network. The power consumed by the compressors is assumed to increase linearly with operating time (since maintenance was performed) due to fouling. Zulkafli and Kopanos [66, 67] develop an optimization framework for simultaneous operational planning and maintenance scheduling of production and utility systems. They consider extra energy costs caused by performance degradation. The degradation is assumed to depend on operating time and the production rate. Aguirre and Papageorgiou [68] consider integrated planning, scheduling and maintenance under schedule-dependent, deterministic performance decay. Rajagopalan et al. [69] analyze turnaround rescheduling and apply stochastic programming to manage unplanned outages. Biondi et al. [70] extend the state-task-network (STN), originally proposed by Kondili et al.

[1], to account for degrading equipment and different operating modes. They assume that each unit, after maintenance, has a given maximum residual lifetime and that each task performed on a unit in a certain operating mode reduces this residual lifetime by a given amount. Noticeably, none of these authors make use of the wealth of knowledge regarding degradation modeling and inference from data available from the CBM literature. Furthermore, degradation is assumed to be deterministic which may not be the case in practice.

Recent works have started to incorporate degradation models from CBM into process level Mixed-Integer Linear Programming (MILP) problems [71, 72, 73, 74, 75]. Yildirim et al. [71, 72] formulate an optimization model for generator maintenance and production scheduling. The cost of maintenance is calculated beforehand using a data-driven degradation model:

$$c_t = \frac{c^{prev}\left(1 - p_t^f\right) + c^{corr} p_t^f}{\int_0^t p_\tau^f d\tau},$$

where c_t is the predicted cost of performing maintenance at time t given the cost of preventive (c^{prev}) and corrective (c^{corr}) maintenance. The failure probability $p_t^f = P(\text{unit fails before } t)$ is calculated from the degradation model. The authors later applied the same approach to maintenance and operation of wind farms and extended it to include opportunistic maintenance [73]. While this approach starts to incorporate information from more sophisticated degradation models into process level optimization problems, degradation is still considered to be deterministic in the MILP optimization. Başçiftci et al. [74] extend this to consider sudden failures by using stochastic programming and generating scenarios from the underlying degradation model. To the best of our knowledge Başçiftci et al.'s [74] approach is the only work combining stochastic optimization with information from degradation models.

Unfortunately, the aforementioned approach cannot capture effects of the selected operating strategy on degradation. Since maintenance cost is calculated based on the degradation model before the optimization problem is solved, the degradation is assumed to be independent of the operating strategy. In practice this will often not be the case.

This chapter argues for a tighter integration between the sophisticated degradation models used

in CBM and process level maintenance scheduling and process optimization. To this end we make multiple contributions:

- We show how Lévy type models, a class of stochastic processes commonly used in degradation modeling, can be incorporated into an integrated maintenance and process MILP model. Lévy type models include the Wiener and Gamma processes – two very popular models in CBM. By making the Lévy models parameters depend on a set of operating modes, equipment degradation too depends on the operating strategy.

- We show how uncertainty and randomness in the equipment's degradation characteristics can be incorporated using adjustable robust optimization. We use results from the CBM literature to efficiently determine the robustness of the obtained solution.

- We prove that, in certain cases, feasible solutions to the adjustable robust optimization problem can be found by solving a deterministic approximation with worst case values for the uncertain parameters.

- Realizing that process planning and scheduling can be computationally expensive yet highly repetitive, we develop a computationally efficient, data-driven way of a priori estimating equipment failure probabilities. To this end, we generate data using a short-term scheduling model repeatedly. Using this data, we propose two methods based on Logistic Regression capable of cheaply generating a large number of long-term schedules which can be used to estimate failure probabilities.

- We propose Bayesian optimization for efficiently optimizing the uncertainty set. The uncertainty set size depends on a small number of parameters, but solving the robust MILP integrated maintenance and process optimization problem can be computationally expensive. Bayesian optimization is ideal for this kind of low dimensional problem with expensive function evaluations.

As a challenging case study, we apply the proposed method to an extension of the state-task-network (STN) [1, 70]. This model combines both planning and scheduling of production and

maintenance with operating mode dependent equipment degradation. We test our method on a number of STN instances [1, 2, 3, 4, 76].

3.2 Combining degradation modeling and robust optimization

Following Vassiliadis and Pistikopoulos [61], we assume an integrated production and maintenance scheduling problem of the form

$$\min_{\boldsymbol{x},\boldsymbol{m}} \quad \text{cost}(\boldsymbol{x},\boldsymbol{m}) \tag{3.1}$$

$$\text{s.t.} \quad \text{process model}(\boldsymbol{x},\boldsymbol{m}) \tag{1a}$$

$$\text{maintenance model}(\boldsymbol{x},\boldsymbol{m}), \tag{1b}$$

where $\boldsymbol{x}$ are the process variables (continuous and discrete) and $\boldsymbol{m}$ are the maintenance related variables. The process model includes, e.g., material balances, energy balances, unit constraints, and the maintenance model includes, e.g., maintenance crew constraints or constraints regarding different types of maintenance. Note that cost minimization could easily be replaced by profit maximization.

A health model added to Problem 3.1 accounts for equipment degradation:

$$\min_{\boldsymbol{x},\boldsymbol{m},\boldsymbol{h}} \quad \text{cost}(\boldsymbol{x},\boldsymbol{m},\boldsymbol{h}) \tag{3.2}$$

$$\text{s.t.} \quad \text{process model}(\boldsymbol{x},\boldsymbol{m},\boldsymbol{h}) \tag{2a}$$

$$\text{maintenance model}(\boldsymbol{x},\boldsymbol{m},\boldsymbol{h}) \tag{2b}$$

$$\text{health model}(\boldsymbol{x},\boldsymbol{m},\boldsymbol{h}), \tag{2c}$$

where $\boldsymbol{h}$ are health related variables and the health model includes all equipment health or degradation related constraints. Our first contribution is developing a generic health model based on the assumption that the equipments' state of health can be described by Lévy type

processes, a class of stochastic processes commonly used for modeling degradation in CBM.

3.2.1 Degradation modeling

The premise in degradation modeling is that a degradation signal $s^{meas}(t)$ describes the state of degradation of a unit over time. Signal $s^{meas}(t)$ can either be measured directly or obtained indirectly from measurements. Two common assumptions adopted in this chapter are that:

(SMAX) The unit fails and requires corrective maintenance when $s^{meas}(t)$ crosses a threshold s^{max} [77],

(AGAN) $s^{meas}(t)$ is reset back to initial value s^0 after preventive maintenance. The unit is as-good-as-new (AGAN) [78].

The degradation signal $s^{meas}(t)$ is often modeled by Lévy type stochastic processes [46], which we have defined in Section 2.3.1. Lévy type processes include both the Wiener and Gamma processes, which are the most commonly used stochastic processes in the degradation modeling literature [40, 46, 79, 80]. Due to their independence and stationarity, Lévy type process increments can be described by

$$S_t - S_{t-\Delta t} = D(\Delta t), \qquad D(\Delta t) \sim \mathcal{D}(\boldsymbol{\theta}, \Delta t), \quad \forall t, \tag{3.3}$$

where $D(\Delta t)$ is a random variable that follows a given distribution $\mathcal{D}(\boldsymbol{\theta}, \Delta t)$ with parameters $\boldsymbol{\theta}$. A difficulty, however, arises when $\mathcal{D}$ is also dependent on some of the operational variables $\boldsymbol{x}$:

$$D(\Delta t) \sim \mathcal{D}(\boldsymbol{\theta}(\boldsymbol{x}), \Delta t). \tag{3.4}$$

This dependence has been addressed by assuming that the operational variables $\boldsymbol{x}$ are piecewise constant, i.e., the process can only operate in a number of discrete operating modes $k \in K$ [81, 82]. Under this assumption Eqns. 3.3 and 3.4 simplify to

$$S_t - S_{t-\Delta t} = \sum_{k \in \mathcal{K}} x_{k,t} \cdot D_k(\Delta t), \quad D_k(\Delta t) \sim \mathcal{D}(\boldsymbol{\theta}_k, \Delta t), \tag{3.5}$$

where $x_{k,t}$ is 1 if the process operates in mode k at time t and 0 otherwise. Note that this approach is very similar to regime-switching Lévy models used extensively in finance [83]. Biondi et al. [70] use a similar approach in their STN extension.

Much of the degradation modeling literature focuses on estimating $\boldsymbol{\theta}$ and using, e.g., Bayesian approaches to update it regularly based on new available data [84, 85]. A major advantage of Bayesian approaches is that $\boldsymbol{\theta}$ can be estimated based on a population of units first and then individually adjusted to a particular unit [86].

3.2.2 Constructing a health model

We summarize the assumptions on which health model 2c hereafter is based: For each process unit j, a degradation signal $s_j^{meas}(t)$ can be obtained from measurements which is modeled well by a Lévy process $S_j(t)$, i.e., increments follow Eqn. 3.5. The unit fails when $S_j(t)$ reaches a maximum threshold s_j^{max} (SMAX) ($T^{fail} = \inf\{t \in T | S_{j,t} > s_j^{max}\}$) and $S_j(t)$ resets to an initial value s_j^0 after maintenance (AGAN). Based on these assumptions and assuming a discrete time formulation, the following health model replaces Eqn. 2c:

$$
\begin{aligned}
& S_{j,t} \leq s_j^{max} && && \forall t, j \in J \\
& S_{j,t} = \begin{cases} S_{j,t-1} + \sum_{k \in \mathcal{K}} x_{j,k,t} \cdot D_{j,k}, & \text{if } m_{j,t} = 0 \\ s_j^0, & \text{otherwise} \end{cases} && && \forall t, j \in J,
\end{aligned}
\tag{3.6}
$$

where J is the set of process units and $m_{j,t}$ is 1 if a maintenance action starts on unit j at time t and 0 otherwise. To address the random nature of degradation, the random variables $D_{j,k}$ and $S_{j,t}$ can be approximated by an uncertain parameter $\tilde{d}_{j,k}$ and a deterministic variable $s_{j,t}$ respectively. Assuming that $\tilde{d}_{j,k}$ is bounded by a compact uncertainty set $\mathcal{U}$, Problem 3.2 can

be robustified by requiring that all constraints hold for any $\tilde{d}_{j,k} \in \mathcal{U}$:

$$
\begin{aligned}
& s_{j,t} \leq s_j^{max} && && \forall t, j \in J \\
& s_{j,t} = \begin{cases} s_{j,t-1} + \sum_{k \in \mathcal{K}} x_{j,k,t} \cdot \tilde{d}_{j,k}, & \text{if } m_{j,t} = 0 \\ s_j^0 & \text{otherwise} \end{cases} && && \forall \tilde{d}_{j,k} \in \mathcal{U}, t, j \in J.
\end{aligned} \tag{3.7}
$$

This model explicitly considers preventive maintenance. Corrective maintenance becomes necessary only when realizations of $\tilde{d}_{j,k}$ lie outside the uncertainty set $\mathcal{U}$ and constraint $s_{j,t} \leq s_j^{max}$ is violated.

Notice that it is generally not possible to choose $s_{j,t}$ such that the equality constraint in Problem 3.7 holds for all values of $\bar{d}_{j,k}$ in $\mathcal{U}$, except for the trivial solution $x_{j,k,t} = 0, \forall j, k, t$. This is because the degradation signal $s_{j,t}$ is an analytical variable, not a decision variable. Interpreting the degradation signal instead as a second stage variable $s_{j,t}\left(\tilde{d}_{j,k}\right)$ turns Problem 3.7 into an adjustable robust optimization problem and a linear decision rule can be used to expresses $s_{j,t}$ as a function of $\tilde{d}_{j,k}$ [76]:

$$
s_{j,t}\left(\tilde{d}_{j,k}\right) = [s_{j,t}]_0 + \sum_k [s_{j,t}]_k \tilde{d}_{j,k}, \tag{3.8}
$$

where $[s_{j,t}]_0$ and $[s_{j,t}]_k$ are coefficients which become variables in the adjustable robust problem. Technically, $\tilde{d}_{j,k}$ should also be indexed by t as every time period constitutes an independent realization of $D_{j,k}$. Time-indexed uncertain parameters have been previously explored [76], but they can lead to a large increase in variables, especially for discrete time formulations. We therefore make the simplifying assumption that uncertainty is only revealed once after all variables except $s_{j,t}$ have been selected.

The health model 3.7 can be reformulated to remove the conditonal equality constraint, result-

ing in the final formulation:

$$
\begin{aligned}
\min_{\boldsymbol{x},\boldsymbol{m}} \quad & \text{cost}(\boldsymbol{x} = [x_{j,k,t}, \ldots]^\top, \boldsymbol{m} = [m_{j,t}, \ldots]^\top, \boldsymbol{h} = [s_{j,t}\left(\bar{d}_{j,k}\right)]^\top) \\
\text{s.t} \quad & \text{process model}(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{h}) \\
& \text{maintenance model}(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{h}) \\
& m_{j,t} s_j^0 \leq s_{j,t}\left(\tilde{d}_{j,k}\right) \leq s_j^{max} + m_{j,t} \cdot (s_j^0 - s_{j,max}) && \forall t, j \in J, \tilde{d} \in \mathcal{U} \\
& s_{j,t}\left(\tilde{d}_{j,k}\right) \geq s_{j,t-\Delta t}\left(\tilde{d}_{j,k}\right) + \sum_k x_{j,k,t} \tilde{d}_{j,k} + m_{j,t} \cdot (s_j^0 - s_j^{max}) && \forall t, j \in J, \tilde{d} \in \mathcal{U} \\
& s_{j,t}\left(\tilde{d}_{j,k}\right) \leq s_{j,t-\Delta t}\left(\tilde{d}_{j,k}\right) + \sum_k x_{j,k,t} \tilde{d}_{j,k} && \forall t, j \in J, \tilde{d} \in \mathcal{U}.
\end{aligned} \tag{3.9}
$$

By replacing $s_{j,t}\left(\tilde{d}_{j,k}\right)$ with Eqn. 3.8 in each constraint and using standard robust optimization reformulation techniques, the health model can be transformed into a deterministic robust counterpart (see Appendix A).

Consider a deterministic version of Problem 3.9 in which cost, process model, and maintenance model are not functions of $s_{j,t}\left(\tilde{d}_{j,k}\right)$ and $\tilde{d}_{j,k}$ has been replaced by $d_{j,k}^{max} = \max_{\mathcal{U}} \tilde{d}_{j,k}$:

$$
\begin{aligned}
\min_{\boldsymbol{x},\boldsymbol{m}} \quad & \text{cost}(\boldsymbol{x}, \boldsymbol{m}) \\
\text{s.t} \quad & \text{process model}(\boldsymbol{x}, \boldsymbol{m}) \\
& \text{maintenance model}(\boldsymbol{x}, \boldsymbol{m}) \\
& m_{j,t} s_j^0 \leq s_{j,t}, && \forall t, j \in J \\
& s_{j,t} \leq s_j^{max} + m_{j,t}(s_j^0 - s_j^{max}), && \forall t, j \in J \\
& s_{j,t} \geq s_{j,t-1} + \sum_k x_{j,k,t} d_{j,k}^{max} + m_t j, t(s_j^0 - s_j^{max}), && \forall t, j \in J \\
& s_{j,t} \leq s_{j,t-1} + \sum_k x_{j,k,t} d_{j,k}^{max}, && \forall t, j \in J,
\end{aligned} \tag{3.10}
$$

where $s_{j,t}$ is not a second stage variable anymore since there are no more semi-infinite constraints. Under certain circumstances, feasible solutions to robust Problem 3.9 can be found by solving deterministic Problem 3.10:

Theorem 3.1. *Given that cost, process model, and maintenance model are not functions of*

$\tilde{d}_{j,k}$ and that $s_j^0 \leq s_j^{init} = s_{j,t=t_0} \leq s_j^{max}$ and $\tilde{d}_{j,k} \geq 0, \forall \tilde{d}_{j,k} \in \mathcal{U}$, then a feasible solution ($\boldsymbol{x} = [x_{k,t}, \ldots], \boldsymbol{m} = [m_t, \ldots], \boldsymbol{h} = [s_t]$) to Problem 3.10 forms a feasible solution ($\boldsymbol{x} = [x_{k,t}, \ldots], \boldsymbol{m} = [m_t, \ldots], \boldsymbol{h} = [[s_t]_0, [s_t]_k]$) to Problem 3.9 with

$$[s_t]_0 = \begin{cases} s^{init} & t < t_{m,0} \\ s^0 & t \geq t_{m,0} \end{cases} \tag{3.11a}$$

$$[s_t]_k = \sum_{t'=t_{m,t}}^{t} x_{k,t}, \tag{3.11b}$$

where $s^{init} = s(t = 0)$, $t_{m,0}$ is the first point in time at which maintenance is performed, and $t_{m,t}$ is the most recent point in time at which maintenance was performed.

Proof. See Appendix B □

Theorem 3.1 only guarantees solution feasibility, not optimality. How well Problem 3.10 approximates Problem 3.9 also depends on the selected uncertainty set. Problem 10 is equivalent to outer-approximating the uncertainty set $\mathcal{U}$ with a box. The worst case realization for Problem 3.9 if $\mathcal{U}$ is a box uncertainty set is always the corner of the box defined by $d_{j,k}^{\max}$, because more degradation is always worse in the problem. If the uncertainty set $\mathcal{U}$ is a box, the outer-approximation is exact and the solutions of Problem 3.10 and Problem 3.9 are equivalent. For other uncertainty sets, a solution to Problem 3.10 is a feasible but potentially conservative solution to Problem 3.9. How conservative this solution is depends on how well the box set approximates the uncertainty set $\mathcal{U}$.

3.2.3 The uncertainty set

A major decision in robust optimization is the uncertainty set choice. This chapter uses a simple box uncertainty set

$$\mathcal{U} = \{\tilde{d}_{j,k} | \bar{d}_{j,k}(1 - \epsilon_{j,k}) \leq \tilde{d}_{j,k} \leq \bar{d}_{j,k}(1 + \epsilon_{j,k})\},$$

where $\bar{d}_{j,k}$ is the nominal value of $\tilde{d}_{j,k}$ and $\epsilon_{j,k}$ is a parameter determining the uncertainty set size. For this choice of uncertainty set, the worst case realization is attained when $d_{j,k} = d_{j,k}^{\max}$ and the robust problem is equivalent to the deterministic Problem 3.10. Note that this choice assumes that the random increments $D_{j,k}$ are independent, as a box uncertainty set cannot capture correlation between uncertain parameters. This assumption could be relaxed with a more complicated uncertainty set, e.g., a polyhedral set. Since degradation modeling assumes that the distribution of $D_{j,k}$ is known, $\epsilon_{j,k}$ can be determined using the inverse cumulative distribution function F^{-1}:

$$\epsilon_{j,k} = 1 - F^{-1}(\alpha)/\bar{d}_{j,k},$$

where $\alpha = P(D_{j,k} \leq \bar{d}_{j,k}(1 - \epsilon_{j,k}))$. If the distribution of $D_{j,k}$ is unknown, data-driven non-parametric methods such as Kernel Density Estimation can be used to estimate it [87].

By using F^{-1}, the uncertainty set size depends on a single parameter $\alpha \in [0, 0.5]$. For $\alpha = 0$, the uncertainty set includes all possible realizations of $D_{j,k}$ and for $\alpha = 0.5$ the uncertainty set is a singleton and the robust optimization problem is equivalent to the deterministic problem using the nominal values $\bar{d}_{j,k}$. While box uncertainty sets are often more conservative than most of the many other available uncertainty set types [88, 89], the solution robustness/conservatism in this formulation can be varied by adjusting α.

3.2.4 Evaluating robustness

Assume $\boldsymbol{x}_j^k = [k_1, k_2, \ldots, k_T]$, where $k_t = k \iff x_{j,k,t} = 1$, is the sequence of operating modes given by a solution to Problem 3.9. Its robustness can be measured by the probability p_j that unit j does not fail in the time horizon T

$$p_j = P(S_{j,t} \leq S_{j,max}, \forall t < T | \boldsymbol{x}_j^k),$$

or equivalently its probability of failure

$$p_j^f = 1 - p_j = P(\exists t \text{ such that } S_{j,t} > S_{j,max} | \boldsymbol{x}_j^k).$$

Assuming the parameters $\boldsymbol{\theta}_{j,k}$ of the distributions $\mathcal{D}_{j,k}$ are estimated from data, p_j^f can be calculated through Monte-Carlo simulation by randomly generating N realizations $\boldsymbol{s}_j^n = [s_{j,0}^n, s_{j,\Delta t}^n, \ldots, s_{j,T}^n]$ of $S_j(t, \boldsymbol{x}_j^k)$ with $\mathcal{D}_{j,k}(\boldsymbol{\theta}_{j,k}, \Delta t)$ distributed increments and checking how many violate $s_{j,t}^n \leq s_j^{max}$:

$$p_j^f = \frac{\sum_{n=1}^{N} \mathbb{1}(\exists t < T \text{ such that } s_{j,t}^n > s_j^{max})}{N}, \tag{3.12}$$

where $\mathbb{1}$ is the indicator function. This is illustrated in Fig. 3.1 for $N = 3$.

In the special case where $D_{j,k}$ is normal distributed $\mathcal{D}_{j,k} \sim \mathcal{N}(\mu_{j,k}\Delta t, \sigma_{j,k}^2 \Delta t)$, i.e., the Wiener process model is used, p_j^f can be efficiently calculated using analytical results for the crossing probability of a Brownian motion on a piecewise linear boundary [90, 91]. Instead of sampling from $\mathcal{D}_{j,k}$ at regular time intervals Δt, this approach only randomly samples at operating mode transitions ($k_t \neq k_{t+1}$). It requires far less Monte-Carlo samples and is therefore faster than the general method outlined above. A detailed description of this approach is given in the Appendix D.

3.2.5 Estimating failure probabilities

Evaluating the probability of failure p_j^f, e.g., using Eqn. 3.12, requires the exact sequence of operating modes $\boldsymbol{x}_j^k$ and maintenance actions to be known over the evaluation horizon T. Since maintenance tends to be infrequent, T has to be sufficiently long to obtain meaningful failure probabilities. Solving Problem 3.9 over a long time horizon may be computationally challenging. Instead, it may be possible to use existing data of past schedules to estimate p_j^f. If no historical data is available, it can be generated by solving Problem 3.9 over a shorter horizon. This section outlines two methods by which an upper estimate of p_j^f can be obtained from data.

Frequency approach

Assuming time discretization, a conceptually easy way to obtain an upper bound $\bar{p}_j^f$ on p_j^f is to generate the set $\mathcal{X}$ of all possible permutations of operating mode sequences $\boldsymbol{x}_j^k =$

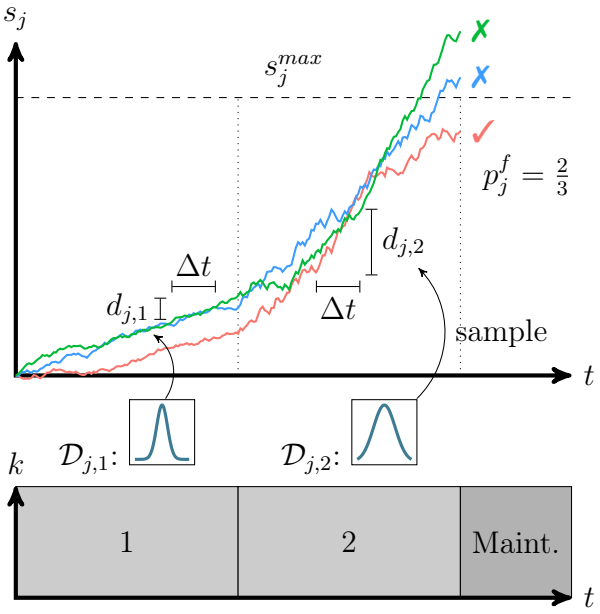


Figure 3.1: Example for calculating the failure probability p_j^f using Eqn. 3.12 and Monte-Carlo simulation. The operating mode schedule is $\boldsymbol{x}^k = [1, 2, \text{Maint.}]$.

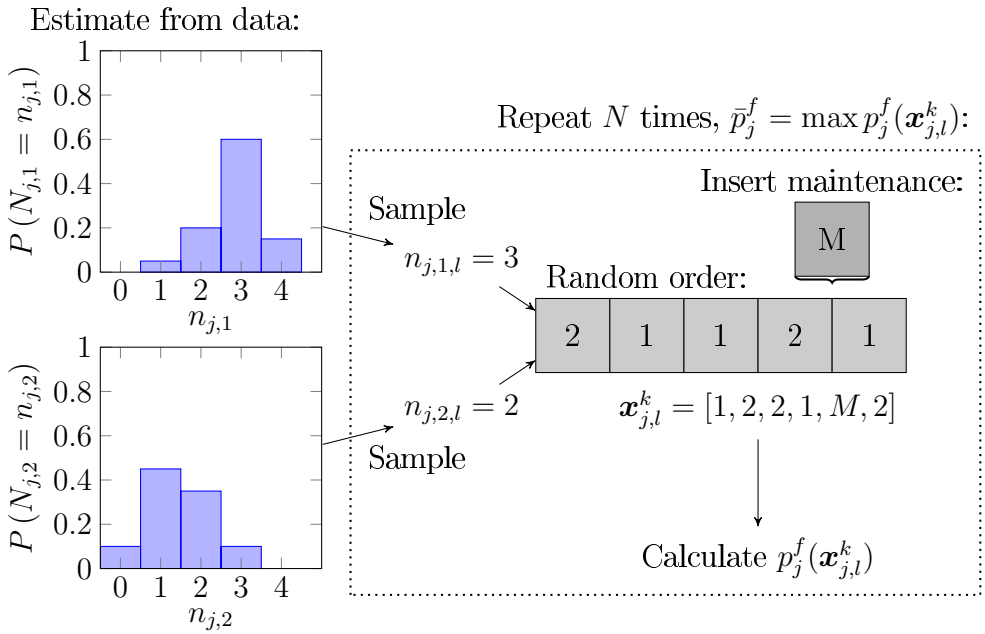


Figure 3.2: Frequency approach: Estimating p_j^f from historical data using Algorithm 1.

$[k_{j,1}, k_{j,2}, \ldots, k_{j,T}]$ and find the maximum probability of failure

$$\bar{p}_j^f = \max_{\boldsymbol{x} \in \mathcal{X}} p_j^f(\boldsymbol{x}_j^k).$$

For any realistic problem $\mathcal{X}$ will be very large, but there are two ways to reduce its size: First, the operating mode sequences can be generated without considering maintenance. Maintenance actions can then be inserted consecutively at the latest point in time $t_{m,l}$ which satisfies

$$\max_{\tilde{d}_{j,k} \in \mathcal{U}} \sum_{t'=t_{m,l-1}}^{t_{m,l}} \sum_k x_{j,k,t'} \tilde{d}_{j,k} < s_j^{max} - s_j^0, \tag{3.13}$$

where $t_{m,l-1}$ is the previous maintenance activity and $t_{m,0} = 0$. $\bar{p}_j^f$ remains an upper bound, because maintenance at a later point in time always causes a larger probability of failure. Secondly, it may be possible to estimate the frequency of occurrence $n_{j,k} = \sum_t x_{j,k,t}$ of each operating mode k from data. If these frequencies are modeled as random variables $N_{j,k}$, a smaller $\mathcal{X}$ can be obtained by only generating sequences which obey frequencies drawn from the distributions of $N_{j,k}$. This suggests the following algorithm for obtaining an estimate of $\bar{p}_j^f$ which is also visualized in Fig. 3.2:

Algorithm 1 Frequency approach [illustrated in Fig. 3.2]

procedure ESTIMATE $\bar{p}_j^f$
  $\eta_{n_{j,k}} = P(N_{j,k} = n_{j,k}) \leftarrow$ estimate from historical data $\forall n_k$
  $l \leftarrow 1$
  while $l \leq N$ **do**
    $n_{j,k,l} \leftarrow$ draw random sample from $P(N_{j,k} = n_{j,k})$ for each k
    $\boldsymbol{x}_{j,l}^k \leftarrow$ arrange $n_{j,l} = \sum_k n_{j,k,l}$ op. modes in random order $[k_1, k_2, \ldots, k_{n_{j,l}}]$
    $\boldsymbol{x}_{j,l}^k \leftarrow$ insert maintenance at last possible points in time ▷ Eqn. 3.13
    $p_{j,l}^f \leftarrow p_j^f(\boldsymbol{x}_{j,l}^k)$ ▷ Eqn. 3.12
  end while
  $\bar{p}_j^f \leftarrow \max_{l \leq N} p_{j,l}^f$
end procedure

If N is large enough and the estimated distribution of $N_{j,k}$ is accurate, $\bar{p}_j^f$ should be a good upper bound on p_j^f.

Markov chain approach

The second approach for estimating $\bar{p}_j^f$ is inspired by the use of Markov chains in regime-switching models in finance and to some extent also in the CBM literature for modeling different enviromental or operating regimes of a process [82, 83, 92, 93]. The key idea is to treat the occurrence of operating modes k over time as a Markov chain. Modeling the sequence of operating modes $\boldsymbol{x}_j^k$ on a unit by a memoryless Markov chain $X_j^k(t) = \{X_{j,t}^k : t \leq T\}$, the probability π_{k,k^*} of transitioning from one operating mode k to another k^* is given by

$$\pi_{k,k^*} = P(X_{j,t}^k = k^* | X_{j,t-1}^k = k).$$

The transition probabilities π_{k,k^*} can be estimated from data.

From this Markov chain random sequences of operating modes $\boldsymbol{x}_{j,l}^k$ can be generated. Maintenance can again be inserted at the latest possible point in time according to Eqn. 3.13. $\boldsymbol{x}_{j,l}^k$ may not be a feasible solution to Problem 3.9, but it can be used to estimate $\bar{p}_j^f$. The approach is summarized in Algorithm 2:

Algorithm 2 Markov chain approach

procedure ESTIMATE $\bar{p}_j^f$
  $\pi_{k,k^*} = P(X_{j,t}^k = k^* | X_{j,t-1}^k = k) \leftarrow$ estimate from historical data $\forall(k, k^*)$
  $l \leftarrow 1$
  while $l \leq N$ **do**
    $\boldsymbol{x}_{j,l}^k \leftarrow$ draw random operating mode sequence from Markov chain π_{k,k^*}
    $\boldsymbol{x}_{j,l}^k \leftarrow$ insert maintenance at last possible points in time ▷ Eqn. 3.13
    $p_{j,l}^f \leftarrow p_j^f(\boldsymbol{x}_{j,l}^k)$ ▷ Eqn. 3.12
  end while
  $\bar{p}_j^f \leftarrow \max_{l \leq N} p_{j,l}^f$
end procedure

Logistic regression

The optimal sequence of operating modes $\boldsymbol{x}_j^{k,*}$ depends not only on the structure of the process and the size of the uncertainty set $\mathcal{U}(\alpha)$, but also on parameters $\boldsymbol{\psi}(t)$ such as product demands

or environmental variables. The distributions of N_k and π_{k,k^*} are therefore not necessarily stationary:

$$\eta_{n_{j,k}}(\boldsymbol{\psi}(t)) = P(N_{j,k} = n_{j,k}|\boldsymbol{\psi}) \tag{3.14a}$$

$$\pi_{k,k^*}(\boldsymbol{\psi}(t)) = P(X^k_{j,t} = k^*|X^k_{j,t-1} = k, \boldsymbol{\psi}), \tag{3.14b}$$

where $\eta_{n_{j,k}}(\boldsymbol{\psi})$ is the probability that operating mode k occurs $n_{j,k}$ times in time period Δt.

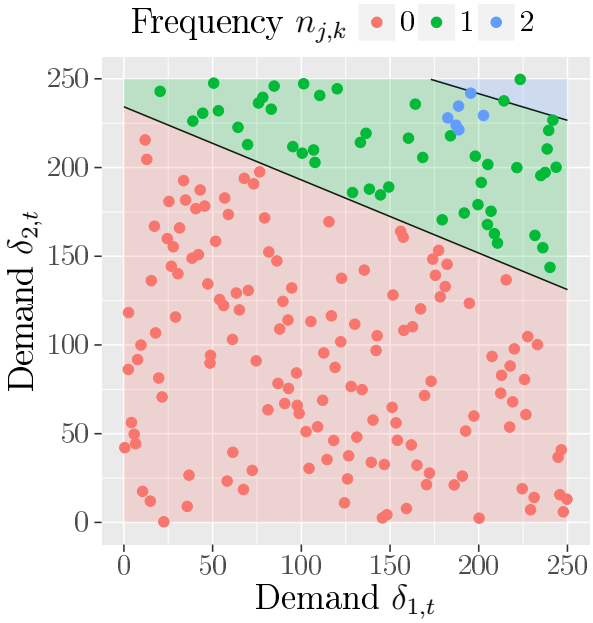


Figure 3.3: Frequency $n_{j,k}$ of operating mode k occurring on unit j for a scheduling problem with two product demands $\boldsymbol{\psi} = [\delta_{1,t}, \delta_{2,t}]^\top$. Points are training data generated by solving the scheduling model and shaded areas are predictions by logistic regression.

Covariate dependency of Markov chain transition probabilities has previously been modeled by using logistic regression [94, 95]. We model both $\eta_{n_{j,k}}$ and π_{k,k^*} using multinomial logistic regression in order to capture the influence of product demands:

$$\eta_{n_{j,k}}(\boldsymbol{\psi}(t)) = \frac{\exp(\boldsymbol{\beta}^\top_{n_{j,k}}\boldsymbol{\psi})}{\sum_{n'_k}\exp(\boldsymbol{\beta}^\top_{n'_{j,k}}\boldsymbol{\psi})} \tag{3.15a}$$

$$\pi_{k,k^*}(\boldsymbol{\psi}(t)) = \frac{\exp(\boldsymbol{\beta}^\top_{k,k^*}\boldsymbol{\psi})}{\sum_{k^+\in K}\exp(\boldsymbol{\beta}^\top_{k,k^+}\boldsymbol{\psi})}. \tag{3.15b}$$

We use Scikit-learn [96] for estimating parameters $\boldsymbol{\beta}$ based on data. Fig. 3.3 shows an example for a process with two product demands $\boldsymbol{\psi} = [\delta_{1,t}, \delta_{2,t}]^\top$. The shaded areas are the frequencies

$n_{j,k}$ predicted by logistic regression for a particular k and j (the $n_{j,k}$ with the largest $\eta_{n_{j,k}}(\boldsymbol{\psi})$) while the points are training data. We use logistic regression in this work because of its simplicity and interpretability, but it could be replaced by any classification method capable of probability estimation, e.g., artificial neural networks, support vector machines, k-nearest neighbours, decision trees, etc. [97, 98].

3.3 Optimizing the uncertainty set size

An important, non-trivial decision when using robust optimization is the size of the uncertainty set — or in this work the choice of parameter α. It governs a trade-off between the robustness of the solution and its cost. A common approach is to use a priori guarantees to determine an uncertainty set size that is guaranteed to have a probability of constraint violation below a predefined level. A priori guarantees are, however, not guaranteed to be tight and uncertainty sets based on them can be overly conservative. As demonstrated by Li and Li [99, 100], determining the optimal uncertainty set size can instead be seen as its own optimization problem. They minimize the uncertainty set size with the constraint that the solution remains feasible with a pre-defined probability. We propose a different formulation that does not require the decision maker to choose a probability of constraint satisfaction but is based purely on cost instead:

$$\min_{\alpha} c^*(\alpha) + \sum_j p_j^f(\alpha) \cdot c_j^f, \tag{3.16}$$

where $c^*(\alpha)$ is the minimal overall cost of the process as determined by solving Problem 3.9 for a given value of α, p_j^f is the corresponding probability of failure evaluated using Eqn. 3.12, and c_j^f is the cost incurred in case of an unplanned failure of unit j, i.e., the cost of corrective maintenance. Effectively, Problem 3.16 minimizes the trade-off between preventive and corrective maintenance. Note that this formulation assumes that each unit fails no more than once in the evaluated horizon T. This is reasonable under the assumption that the cost of failure c_j^f is high and therefore P_j^f tends to be low.

Problem 3.16 is a one-dimensional optimization problem, but determining $c^*(\alpha)$ and $p_j^f(\alpha)$ can

be computationally expensive because it requires solving a potentially large MILP problem and Monte-Carlo simulation. It can therefore be viewed as a black box optimization problem with expensive function evaluations. We propose Bayesian optimization, which is known to work well on expensive low dimensional objective functions, as an effective solution strategy [101]. Bayesian optimization has the further advantage that it can handle noise well. Both c^* and p_j^f can be noisy because it may not be possible to solve Problem 3.9 to optimality in a reasonable time frame. Further noise is introduced by the Monte-Carlo simulation used to evaluate p_j^f.

3.4 Case study

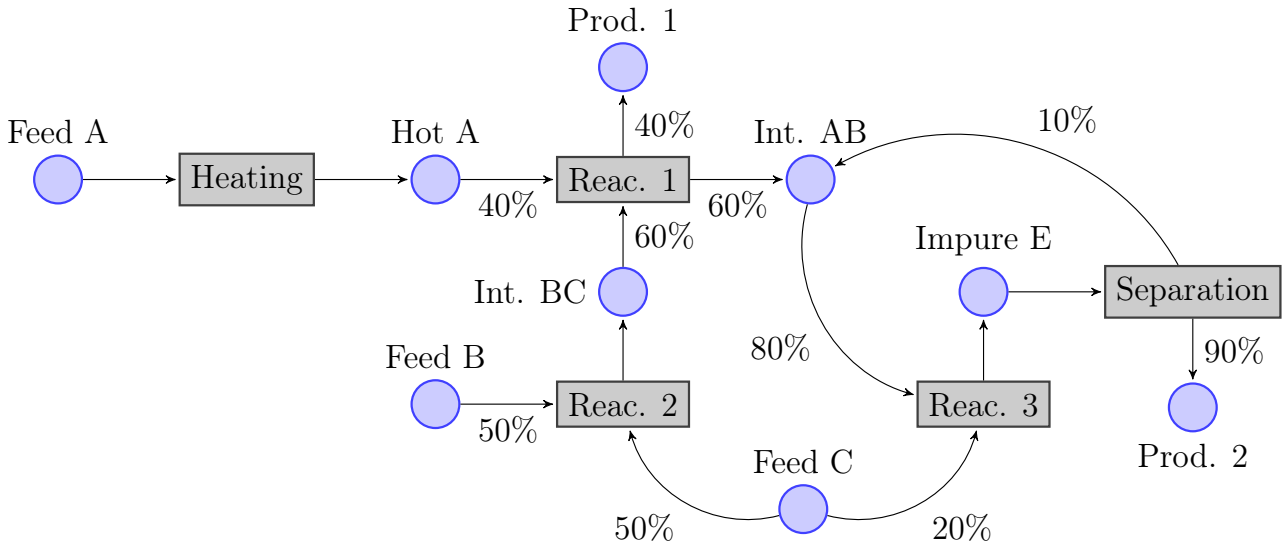


Figure 3.4: STN instance proposed by Kondili et al. [1]. Tasks are performed on four units: Heater, Reactor 1, Reactor 2, and Still.

The model by Biondi et al. [70], an extension of the state-task-network (STN) [1], forms the basis of our case study. The classic STN is a scheduling problem in which a set of tasks I has to be assigned to a set of units J. Biondi et al. [70] extend the STN by allowing each task i to be performed in a number of different operating modes $k \in K_i$. They add constraints reducing the residual lifetime $r_{j,t}$ of each unit j every time a task is performed, and restore $r_{j,t}$ by performing maintenance. Because the scheduling problem can only be solved for a short time horizon T_S but maintenance occurs infrequently, they add a planning horizon T_P to the problem. For the planning horizon, instead of an exact schedule, only the number of times $n_{i,j,k,t}$ a task i is

performed on unit j in operating mode k in each planning period t is calculated. Eqns. 3.17 to 3.20d give the modified version used as a case study in this work:

Objective function:

$$\begin{aligned}\text{cost} = \sum_{j\in J} c_j^{maint}\left(s_j^{fin}/s_j^{max} + \sum_{t\in T} m_{j,t}\right)\\ + c_s^{storage}\left(q_s^{fin} + \sum_{t\in T_p} q_{s,t}\right)\\ + U\left(\sum_{s\in S}\phi_s^d + \sum_{t\in T_S}\phi_{s,t}^q\right)\end{aligned} \tag{3.17}$$

Constraints scheduling horizon:

$$\sum_{k\in K_j}\sum_{i\in I_j}\sum_{t'=t-p_{i,j,k}+\Delta t_S}^{t} w_{i,j,k,t'} + \sum_{t'=t-\tau_j+\Delta t_s}^{t} m_{j,t'} \le 1 \qquad \forall J, t\in T_S \tag{3.18a}$$

$$v_{i,j}^{min} w_{i,j,k,t} \le b_{i,j,k,t} \le v_{i,j}^{max} w_{i,j,k,t} \qquad \forall J, i\in I_j, k\in K_j, t\in T_S \tag{3.18b}$$

$$\begin{aligned}q_{s,t} = q_{s,t-1} + \sum_{i\in\bar{I}_s}\bar{\rho}_{i,s}\sum_{j\in J_i}\sum_{k\in K_j} b_{i,j,k,t-p_{i,j,k}}\\ - \sum_{i\in I_s}\rho_{i,s}\sum_{j\in J_i}\sum_{k\in K_j} b_{i,j,k,t}\end{aligned} \qquad \forall s, t\in T_S \tag{3.18c}$$

$$0 \le q_{s,t} - \phi_{s,t}^q \le c_s \qquad \forall s, t\in T_S \tag{3.18d}$$

$$m_{j,t}s_j^0 \le s_{j,t} \le s_j^{max} + m_{j,t}\cdot(s_j^0 - s_j^{max}) \qquad \forall t, j\in J, D\in\mathcal{U} \tag{3.18e}$$

$$s_{j,t} \ge s_{j,t-\Delta t_S} + \sum_i\sum_k w_{i,j,k,t}\tilde{d}_{j,k} + m_{j,t}\cdot(s_j^0 - s_j^{max}) \qquad \forall t, j\in J, D\in\mathcal{U} \tag{3.18f}$$

$$s_{j,t} \le s_{j,t-\Delta t_S} + \sum_i\sum_k w_{i,j,k,t}\tilde{d}_{j,k} \qquad \forall t, j\in J, D\in\mathcal{U}, \tag{3.18g}$$

Constraints planning horizon:

$$\sum_{i\in I_j}\sum_{k\in K_j} p_{i,j,k}n_{i,j,k,t} + \tau_j m_{j,t} \le \Delta t_P \qquad \forall J, t\in T_P\backslash\{\bar{t}_P\} \tag{3.19a}$$

$$v_{i,j}^{min}\sum_{k\in K_j} n_{i,j,k,t} \le a_{i,j,t} \le v_{i,j}^{max}\sum_{k\in K_j} n_{i,j,k,t} \qquad \forall J, i\in I_j, k\in K_j, t\in T_P \tag{3.19b}$$

$$q_{s,t} = q_{s,t-1} + \sum_{i \in \bar{I}_s} \bar{\rho}_{i,s} \sum_{j \in J_i} a_{i,j,t} - \sum_{i \in I_s} \rho_{i,s} \sum_{j \in J_i} a_{i,j,t} - \delta_{s,t} \qquad \forall s, t \in T_P \backslash \{\bar{t}_P\} \tag{3.19c}$$

$$0 \leq q_{s,t} \leq c_s \qquad \forall s, t \in T_P \tag{3.19d}$$

$$n_{i,j,k,t} \leq U \cdot \omega_{j,k,t} \qquad \forall J, i \in I_j, k \in K_j, t \in T_P \tag{3.19e}$$

$$\sum_{k \in K_j} \omega_{j,k,t} = 1 \qquad \forall J, t \in T_P \tag{3.19f}$$

$$s_{j,t} \leq s_j^{max} \qquad \forall t, j \in J \tag{3.19g}$$

$$s_j^t \geq s_{j,t-\Delta t_P} + \sum_k n_{j,k,t} \tilde{d}_{j,k,t} + m_{j,t} \cdot (s_j^0 - s_j^{max}) \qquad \forall t, j \in J \tag{3.19h}$$

$$s_{j,t} \leq s_{j,t-\Delta t_P} + \sum_k n_{j,k,t} \tilde{d}_{j,k,t} \qquad \forall t, j \in J \tag{3.19i}$$

Constraints interface between scheduling and planning:

$$\begin{aligned} \sum_{k \in K_j} \sum_{i \in I_j} \sum_{t'=\bar{t}_S+2\Delta t_S-p_{i,j,k}}^{\bar{t}_S} & w_{i,j,k,t'} \left[p_{i,j,k} - (\bar{t}_S - t' + \Delta t_S) \right] \\ & + \sum_{t'=\bar{t}_S+2\Delta t_S-\tau_j}^{\bar{t}_S} m_{j,t'} \left[\tau_j - (\bar{t}_S - t' + \Delta t_S) \right] \\ & + \sum_{i \in I_j} \sum_{k \in K_j} p_{i,j,k} n_{i,j,k,\bar{t}_P} + \tau_j m_{j,\bar{t}_P} \leq \Delta t_P, \end{aligned} \qquad \forall j \in J \tag{3.20a}$$

$$\begin{aligned} q_s^{fin} = q_{s,\bar{t}_S} + \sum_{i \in \bar{I}_s} \bar{\rho}_{i,s} \sum_{j \in J_i} \sum_{k \in K_j} & b_{i,j,k,\bar{t}_S+1-p_{i,j,k}} \\ & - \delta_{s,\bar{t}_S} + \phi_s^d \end{aligned} \qquad \forall s \tag{3.20b}$$

$$0 \leq q_s^{fin} \leq c_s \qquad \forall s \tag{3.20c}$$

$$\begin{aligned} q_{s,\bar{t}_P} = q_s^{fin} + \sum_{i \in \bar{I}_S} \bar{\rho}_{i,s} \sum_{j \in J_i} \sum_{k \in K_j} \sum_{t'=\bar{t}_s+2-p_{i,j,k}}^{\bar{t}_S} & b_{i,j,k,t'} \\ & + \sum_{i \in \bar{I}_S} \bar{\rho}_{i,s} \sum_{j, J_i} a_{i,j,\bar{t}_P} \\ & - \sum_{i, I_s} \rho_{i,s} \sum_{j \in J_i} a_{i,j,\bar{t}_P} - \delta_{s,\bar{t}_P} \end{aligned} \qquad \forall s \tag{3.20d}$$

The decision variables are $m_{j,t}, q_{s,t}, w_{i,j,k,t}, n_{i,j,k,t}, b_{i,j,k,t}, a_{i,j,t}, s_{j,t}, \phi_{s,t}^q, \phi_s^d$ and $\omega_{j,k,t}$. The product demand $\delta_{s,t}$ has to be satisfied at the end of each planning period and at the end of any time

interval in the scheduling horizon. In practice, this model would be solved regularly in a rolling horizon fashion using recent demand estimates and degradation signal measurements $s_{j,0}$.

In comparison to Biondi et al. [70], the residual lifetime constraints have been replaced with the degradation signal based health model developed above (Eqn. 3.9). For the planning horizon, $s_{j,t}$ cannot be reset exactly to s_j^0, because the exact time at which maintenance is performed is unknown. Instead, it is merely enforced that $s_{j,t} \leq s_j^{max}$.

In addition, the objective function is slightly different. The term $c_j^{maint}(s_j^{fin}/s_j^{max})$ can be interpreted as a final cost of maintenance dependent on the final degradation signal s_j^{fin} (state of health) of unit j. Similar to Biondi et al.'s [70] penalty terms it avoids unnecessary degradation and ensures maintenance happens towards the end of a units residual lifetime.

Since the exact sequence of operating modes and maintenance actions is unknown in the planning horizon T_P, the probability of failure p_j^f can only be evaluated over the scheduling horizon T_S. In order to still evaluate p_j^f over a longer time period two possibilities exist: the schedule can be extended in length by solving the model repeatedly in a rolling horizon fashion or the Markov chain-based estimation approach in Algorithm 2 can be used. We compare both approaches to show that the proposed Markov chain estimate is indeed accurate. Note that, in order to facilitate a rolling horizon based solution approach, slack variables have been introduced in Eqns. 3.18d and 3.20b. This is necessary because the rolling horizon framework does not guarantee feasibility in subsequent time periods. Production targets from the planning model may, for example, not be achievable in the scheduling model. The slack variables ϕ_s^d and $\phi_{s,t}^q$ are penalized in the objective function.

We assume that the frequencies of operating mode occurrence $n_{j,k}$ and the transition probabilities $\pi_{k,k*}$ depend on the product demands in each planning period ($\boldsymbol{\phi}_t = [d_{s_1,t}, \ldots, d_{s_n,t}]$). We sample a range of demands using Latin Hypercube Sampling and solve just the scheduling horizon to generate data for estimating $n_{j,k}(\boldsymbol{\psi}_t)$ and $\pi_{k,k*}(\boldsymbol{\psi}_t)$.

Notice that the model above fulfills the assumptions in Theorem 3.1 and solutions can be obtained by solving the deterministic approximation 3.10.

3.5 Results

The framework outlined above was evaluated on five instances of the STN (see Table 3.1 and Appendix C). The model was implemented in Pyomo [102, 103] and solved using CPLEX

Instance	Toy	P1 [1]	P2 [2]	P4 [3]	P6 [4]
Units	2	4	5	3	6
Tasks	3	5	3	4	8
Op. modes	2	3	3	2	2
Products	2	2	1	2	4
Discrete vars	518	2492	1930	1869	1993
Continuous vars	1033	3630	2371	2777	4084
Constraints	1860	7332	5705	5699	7994
Avg. MIP gap [%]	0.0	3.0	5.8	10.9	1.02

Table 3.1: Evaluated STN instances (Details: see Appendix C)

12.7.1.0. All source code is publicly available under the MIT Licence [104]. Unless mentioned otherwise, we considered an evaluation horizon of 12 planning periods. The failure probability p_j^f for each unit j was evaluated for a range of values of the uncertainty set parameter α using both the frequency and Markov chain estimates as well as rolling horizon. The termination criteria for each CPLEX run were a maximum time limit between $1-5$ minutes (depending on the size of the instance) and a MIP gap of 2% (except for the toy instance which was solved to optimality). For each instance a low, average, and high demand scenario was considered with the high scenario being close to maximum process capacity. For Instance P1 [1] both the robust and deterministic Problems 3.9 and 3.10 were solved a number of times to evaluate the quality of the deterministic approximation. For all other instances only the deterministic approach was used. All calculations were carried on an i7-6700 CPU with 8×3.4GHz and 16GB RAM.

Fig. 3.5 shows three maintenance schedules for the original STN instance (P1) by Kondili et al. [1] with Biondi et al.'s [70] demand scenario (average scenario) with different values for α. The number of maintenance actions increases with increasing uncertainty set size (decreasing α). Essentially, hedging against more uncertainty and ensuring solution robustness for a larger set of possible realizations requires earlier maintenance. In the average demand scenario, maintenance actions increase by 22% when hedging against some of the uncertainty ($\alpha = 0.26$) and by 56% when hedging against almost all uncertainty ($\alpha = 0.02$). However, this trend also depends on

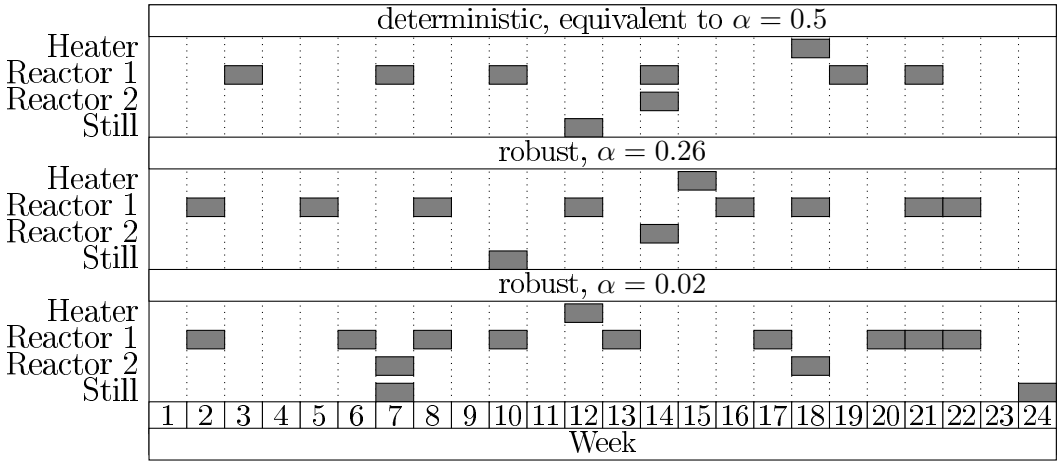


Figure 3.5: Comparison of maintenance schedules between deterministic solution ($\alpha = 0.5$) and two robust solutions with different values of α (Instance P1 [1], average demand scenario by Biondi et al. [70]). The number of required maintenance actions increases with increasing uncertainty set size (decreasing α).

product demand: for the low demand scenario, only an increase of 25% is necessary for $\alpha = 0.02$, while an increase of 64% is necessary in the high demand scenario. A higher demand increases unit utilization and therefore also the absolute number of maintenance actions required in a given time period.

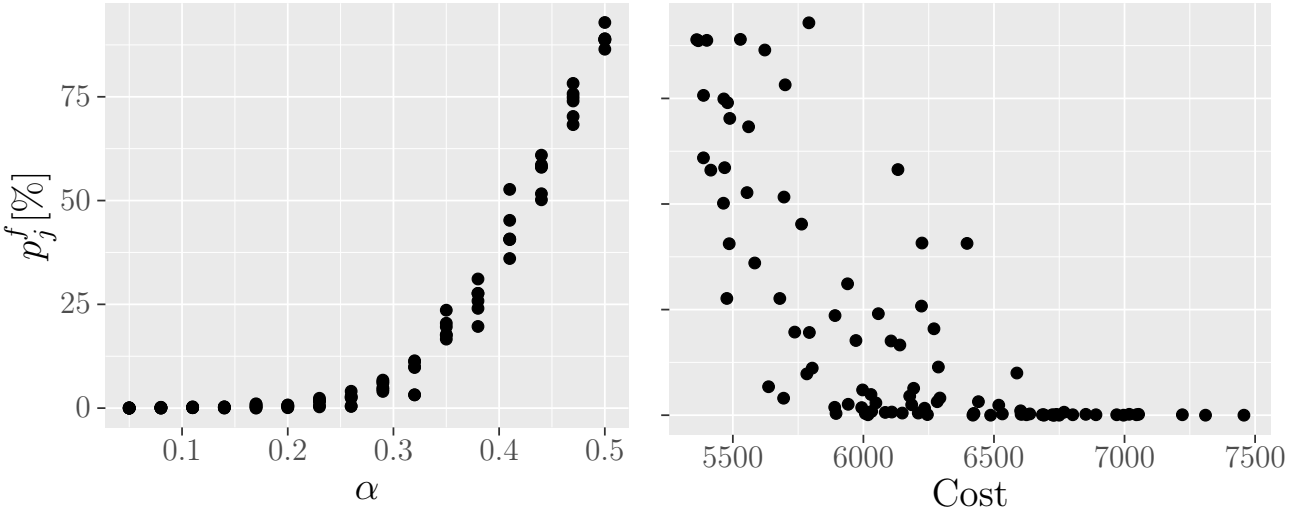


Figure 3.6: Probability of Reactor 1 failing p_j^f vs uncertainty set parameter α and cost (Instance P1 [1], average demand scenario). Each point is a solution to Problem 3.10 obtained by CPLEX. The probability of failure p_j^f was estimated using the Algorithm 2 Markov chain approach.

Fig. 3.6 shows the failure probability p_j^f for Reactor 1 as a function of both total cost (cost of storage and cost of maintenance) and the uncertainty set parameter α. As expected, p_j^f

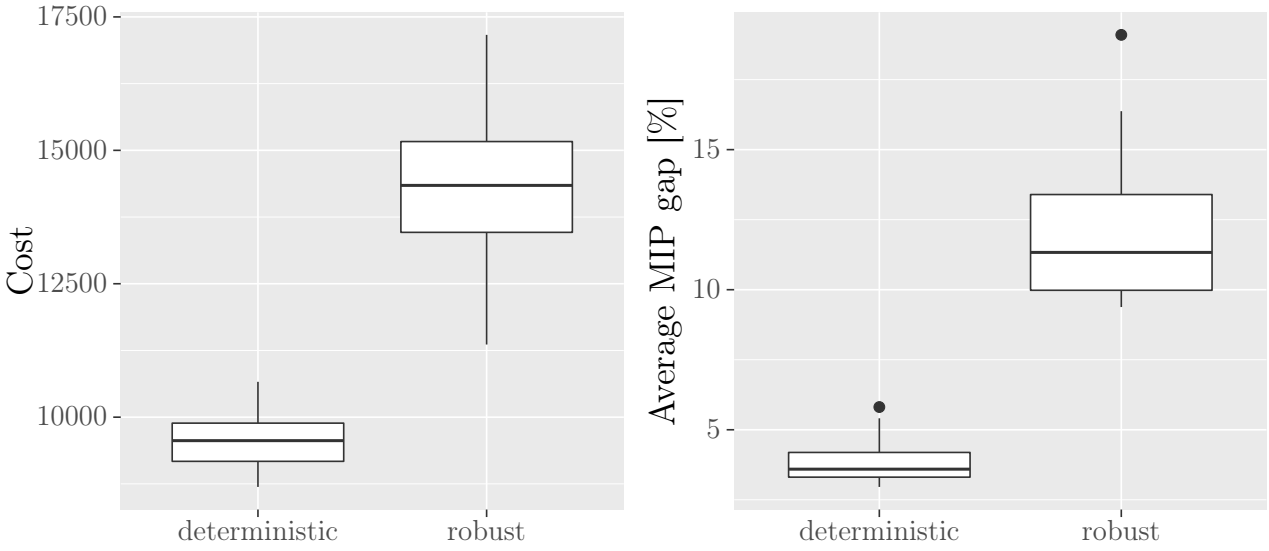


Figure 3.7: Robust vs. deterministic approach (Instance P1 [1]). Each model was solved 25 times with $\alpha = 0.41$ and a $120s$ time limit per rolling horizon iteration using out-of-the-box CPLEX. The robust model has 22514 variables and 14664 constraints while the deterministic model has 6122 variables and 7332 constraints.

increases for smaller uncertainty sets (large α's) and a low p_j^f comes at a significant cost – the price of robustness. Notice that, while calculating p_j^f using Monte-Carlo simulation only introduces modest noise, the calculated cost is very noisy due to the non-optimality of the solutions.

The results in Fig. 3.6 were obtained by solving the deterministic Problem 3.10. While Theorem 3.1 guarantees that these solutions are also feasible in the robust Problem 3.9, it does not prove that they are also optimal. Fig. 3.7 shows both total cost and average optimality gap for a number of rolling horizon solutions to the deterministic and robust version of Instance P1 [1]. The uncertainty set size was $\alpha = 0.41$ and a maximum time limit of $120s$ was used for each CPLEX run. Within this time limit, out-of-the-box CPLEX achieves an average optimality gap of 12.1% on the robust problem compared with 3.8% for the deterministic approximation with maximum values for $\tilde{d}_{j,k}$. Similarly, the deterministic approximation achieves significantly lower objective values. While many approaches could improve solution quality of the robust problem, e.g., solver parameter tuning or leveraging Satisfiability Modulo Theory [105], Constraint Programming [106], or Approximation Algorithms [107], it is likely that the deterministic approximation will remain favorable as it has significantly fewer variables and constraints (6122

vs. 22514 and 7332 vs. 14664 respectively). Assuming a box uncertainty set, we view it as a reasonable approximation for instances which cannot be solved to optimality in a reasonable amount of time. In the case of a more complex uncertainty set, replacing $\tilde{d}_{j,k}$ with its maximum value may lead to conservative solutions. General uncertainty sets require solving the robust problem.

Note that the large range of solution values in Fig. 3.7 is not only due to the differing MIP gaps, but also the rolling horizon approach which does not guarantee optimality.

Logistic regressions for the Frequency and Monte-Carlo approaches were trained based on 200 scheduling horizon only solutions. For the Frequency approach, the training points and their predicted values for Reaction 1 in mode Normal of the toy instance are shown in Fig. 3.8. Logistic regression predicts $n_{i,j,k}$ reasonably well but the rigid, linear classifier cannot capture some of the details. This is, however, not a major problem as the entire predicted probability distribution $\eta_{n_{i,j,k}}$ of operating mode occurrence frequencies $n_{i,j,k}$ is used in estimating $\bar{p}_j^f$. Near the predicted boundaries, $\eta_{n_{i,j,k}}$ of adjacent frequencies will be non-zero and they will be sampled in a significant number of operating mode sequences in Algorithm 1.

Fig. 3.9 shows the probability of failure p_j^f for each unit in the toy instance as a function of the uncertainty set parameter α for three different demand scenarios (average, high, and low). Since the rolling horizon framework does not guarantee optimality, solving the problem repeatedly for the same value of α can lead to different solutions and failure probabilities. The problem was therefore solved 10 times for each value of α. It can be seen that the probability of failure generally increases with demand. The figure furthermore shows the two bounds obtained using the Frequency and Markov chain based approaches, i.e. Algorithm 1 versus 2. For the reactor both approaches provide good upper bounds. For the heater, the frequency based approach performs very well, while the Markov chain approach underestimates p_j^f for the high demand scenario and overestimates it for the average and low demand scenarios. Finally notice that the a priori bound [108] given by the dotted lines greatly overestimates p_j^f. Fig. 3.10 shows similar trends for Instance P1 [1] (Kondili). Both approaches provide reasonable bounds for all units and scenarios except the average demand scenario on the Heater, for which the frequency

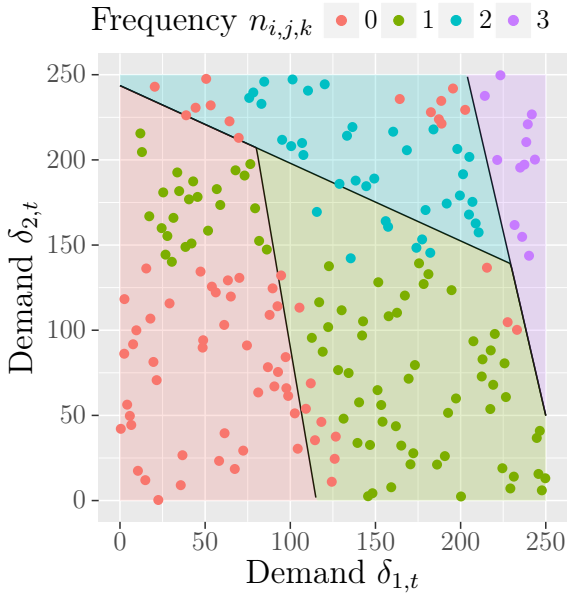


Figure 3.8: Toy instance: Frequency $n_{i,j,k}$ of Reaction 1 occurring in normal mode on unit Reactor within one scheduling horizon. Points are training data generated by solving the scheduling model repeatedly and shaded areas are predictions by logistic regression.

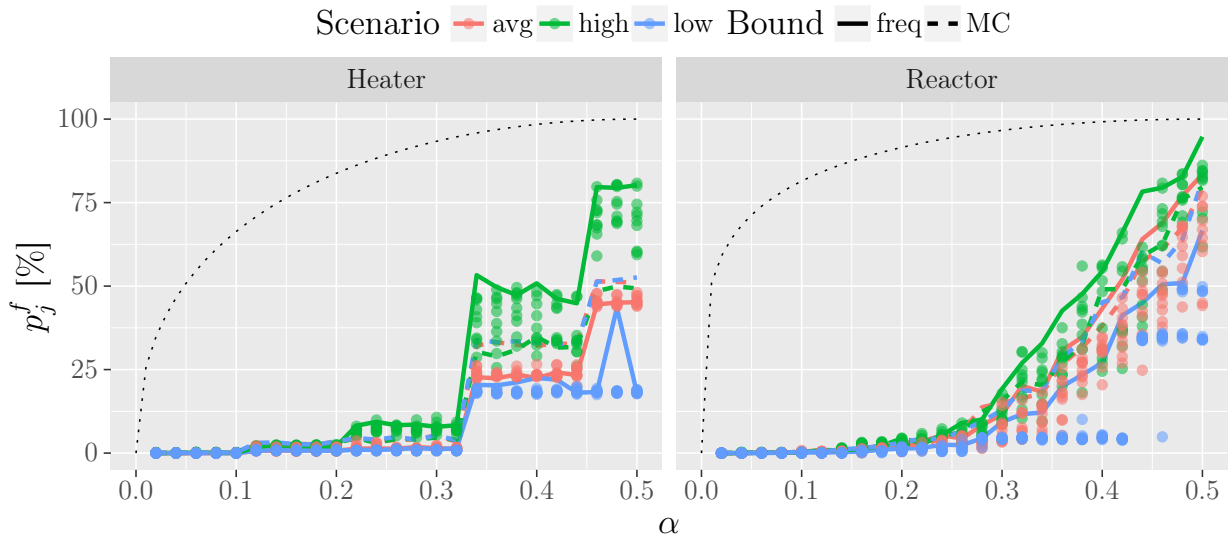


Figure 3.9: Toy instance: Frequency vs. Markov chain approach. Points are rolling horizon solutions. Colored lines are bounds from the Frequency and Markov chain approach. The dotted black lines show a priori bound B4 by Li et al. [108].

approach underestimates p_j^f. Notice that p_j^f is nearly zero for both the Heater and Reactor 2 at low demand irrespective of α. This is because for this scenario no maintenance occurs on either unit and $s_{j,t}$ does not get close to s_j^{max}.

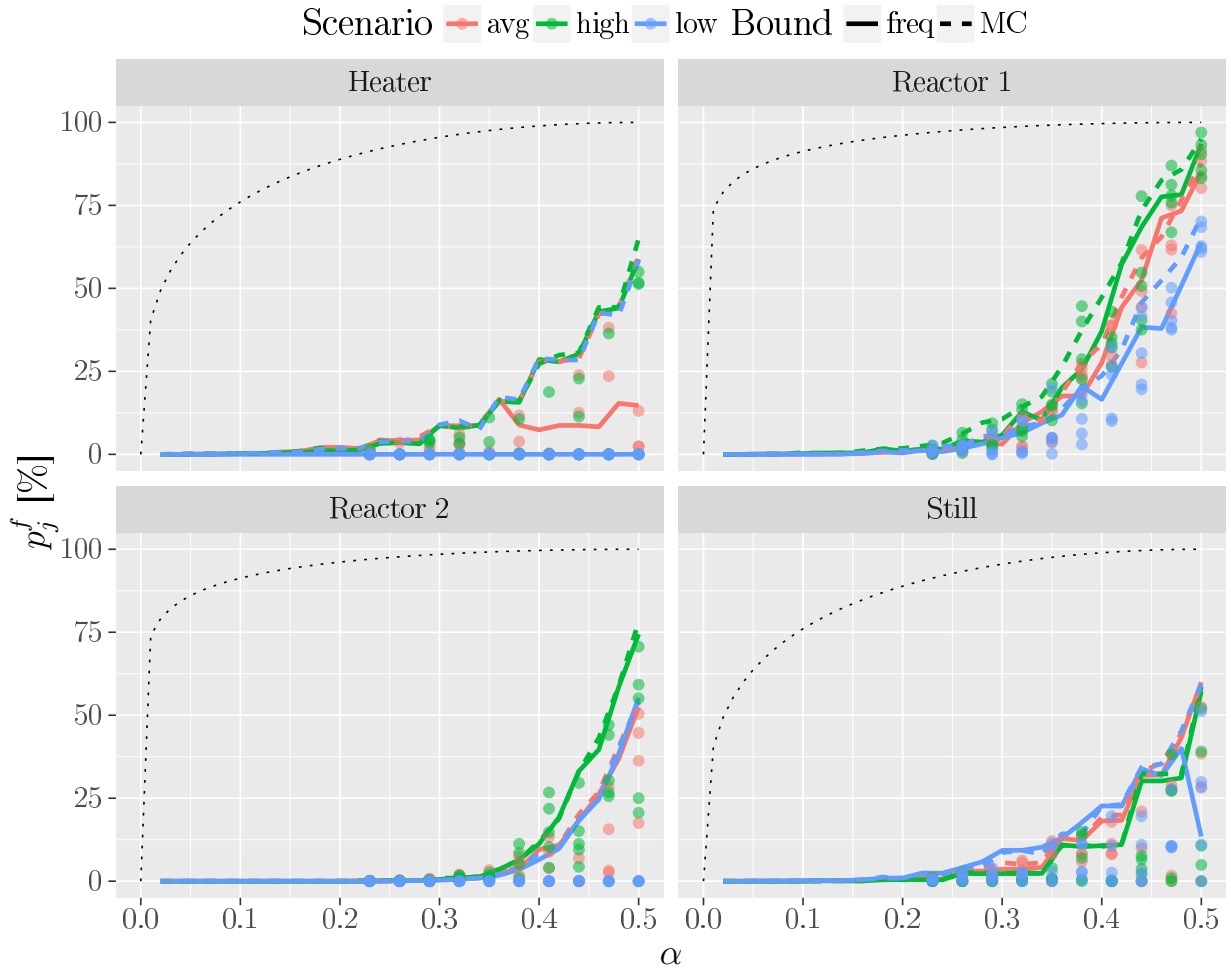


Figure 3.10: Instance P1 [1]: Frequency vs. Markov chain approach. Points are rolling horizon solutions. Colored lines are bounds from the Frequency and Markov chain approach. The dotted black lines show a priori bound B4 by Li et al. [108].

The performance of the Frequency and Markov chain based probability estimates was assessed using three metrics:

$$\text{rms}_{all}^2 = \frac{1}{N \cdot |A|} \sum_{n \in \{1..N\}, \alpha \in A} \left(\left[p_j^f \right]_{n,\alpha} - \bar{p}_j^f \right)^2, \tag{3.21a}$$

$$p_{out} = \frac{1}{N \cdot |A|} \sum_{n \in \{1..N\}, \alpha \in A} \mathbb{1} \left(\left[p_j^f \right]_{n,\alpha} > \bar{p}_j^f \right), \text{ and} \tag{3.21b}$$

$$\text{rms}_{out}^2 = \frac{1}{p_{out} \cdot N \cdot |A|} \sum_{n \in \{1..N\}, \alpha \in A} \mathbb{1} \left(\left[p_j^f \right]_{n,\alpha} > \bar{p}_j^f \right) \left(\left[p_j^f \right]_{n,\alpha} - \bar{p}_j^f \right)^2, \tag{3.21c}$$

where rms_{all} is the root-mean-squared deviation between the estimate and all rolling horizon solutions, p_{out} is the percentage of rolling horizon solutions with a larger p_j^f than the estimated bound, and rms_{out} is the root-mean-squared deviation of all underestimated points. While rms_{all} evaluates the estimate $\bar{p}_j^f$'s quality as a predictor of p_j^f, p_{out} and rms_{out} assess its quality as an upper bound.

instance	bound	rms_{all}	p_{out}	rms_{out}
toy	freq	8.00	17.54	0.90
toy	mc	10.41	9.62	2.86
P1 [1]	freq	12.61	18.08	5.80
P1	mc	17.25	10.13	1.81
P2 [2]	freq	7.40	48.19	2.24
P2	mc	13.68	40.56	1.10
P4 [3]	freq	10.09	13.77	4.10
P4	mc	11.40	11.91	2.67
P6 [4]	freq	16.35	29.40	4.48
P6	mc	20.16	21.27	3.23
all	freq	10.89	25.40	3.50
all	mc	14.58	18.70	2.34

Table 3.2: Average performance metrics for probability estimates - all instances

Table 3.2 shows values for all three instances averaged over demand scenarios and units for all tested STN instances. It can be seen that the frequency approach is generally a better estimator for p_j^f than the Markov chain approach (smaller values of rms_{all}) but also has a larger rate of misclassification p_{out}. While rms_{all} can be large due to noise in the rolling horizon solutions and p_{out} values of up to 48% show that $\bar{p}_j^f$ is not a perfect upper bound, rms_{out} is generally small with average values of 3.50 and 2.34% for the Frequency and Markov chain approach respectively. This means that, when $\bar{p}_j^f$ underestimates p_j^f, it does not do so by much. Considering the noise introduced by non-optimal solutions and the rolling horizon framework, the error introduced by estimating p_j^f through $\bar{p}_j^f$ is small. Because it is a slightly better upper bound, the Markov chain approach was used for all subsequent experiments unless mentioned otherwise.

Both probability estimation approaches are dependent on the number N of operating mode sequences generated. Fig. 3.11 shows that increasing N from 100 to 1000 for Reactor 1 in Instance P1 [1] only has a small effect on p_j^f, especially for the Markov chain based approach. $N = 100$ is therefore deemed sufficient.

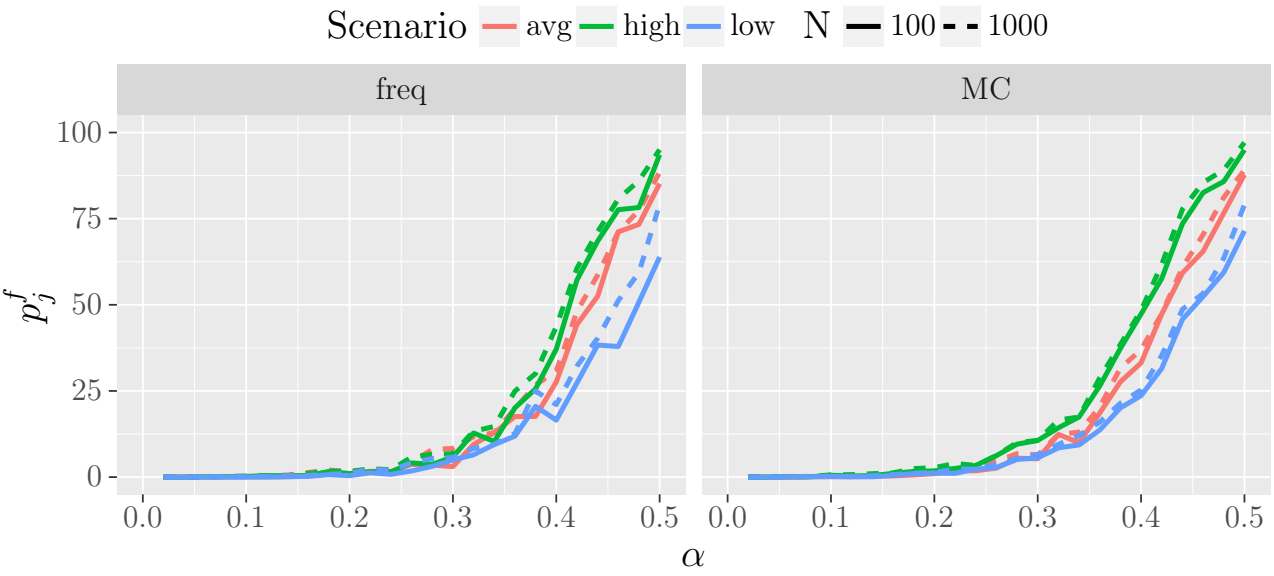


Figure 3.11: Effect of number of Monte-Carlo samples N on failure probability p_j^f of Reactor 1 (Instance P1 [1]).

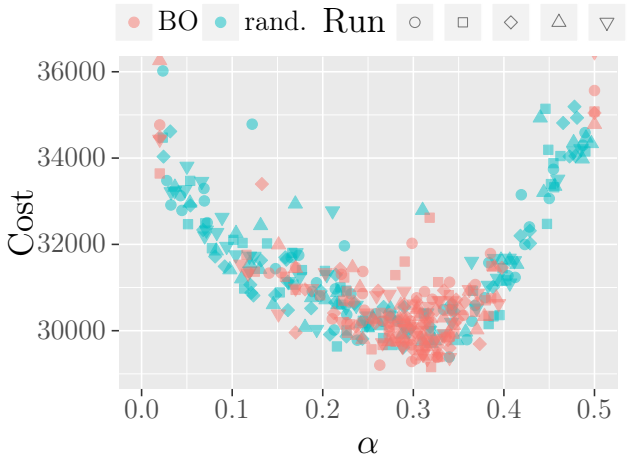


Figure 3.12: Bayesian optimization vs. random search. Five runs of BO and random search were conducted. BO efficiently explores values near the optimal α.

Figs. 3.12 and 3.13 compare Bayesian optimization (BO) with random search for Instance P1 [1]. Overall cost (Eqn. 3.16) was evaluated over a horizon of 24 weeks and both Bayesian optimization and random search were repeated five times. For the Bayesian optimization, four points were sampled evenly from the interval $\alpha \in [0.02, 0.5]$ initially. Fig. 3.12 shows the obtained objective values as a function of α. There is clearly a trade-off between high cost of preventive maintenance in conservative solutions (small α) and high cost of corrective maintenance for less conservative but also less robust solutions (large α). Bayesian optimization very efficiently samples from the area around the optimal $\alpha \approx 0.3$ while random search naturally samples from the entire interval. Fig 3.13 shows the lowest objective value previously obtained as a function of the number of samples (averaged over five runs). Bayesian optimization consistently finds lower cost solutions than random search and achieves a good compromise between preventive and corrective maintenance within about 20 iterations.

3.6 Conclusion

This work integrates equipment degradation effects in process level optimization problems. The demonstrated methodology is summarized in Fig. 3.14. We combine commonly used methods from the degradation modeling literature, which allow unit health characteristics to be estimated and updated from data, with robust optimization. This is highly relevant since, realistically, almost all equipment in chemical and manufacturing processes will be subject to performance degradation and failures.

Solving realistic integrated maintenance and production scheduling and planning problems is a hard task by itself, because models tend to be large and computationally expensive. Combining such models with robust optimization increases the need for solving problems efficiently. Furthermore, the scheduling task is highly repetitive and should become easier as historical data is collected. To reduce computational expense, we show conditions where robust optimization problems can be solved by solving a deterministic approximation and develop data-based methods for estimating failure probabilities.

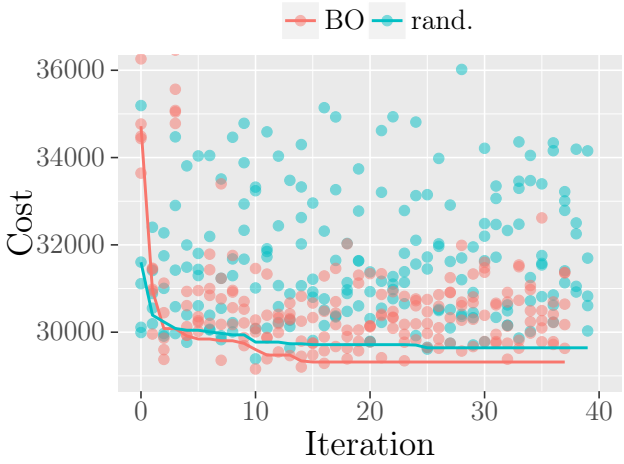


Figure 3.13: Bayesian optimization vs. random search. Points are individual values obtained at a given iteration. Lines represent the best previously achieved solution value averaged over five runs. BO consistently finds better solutions.

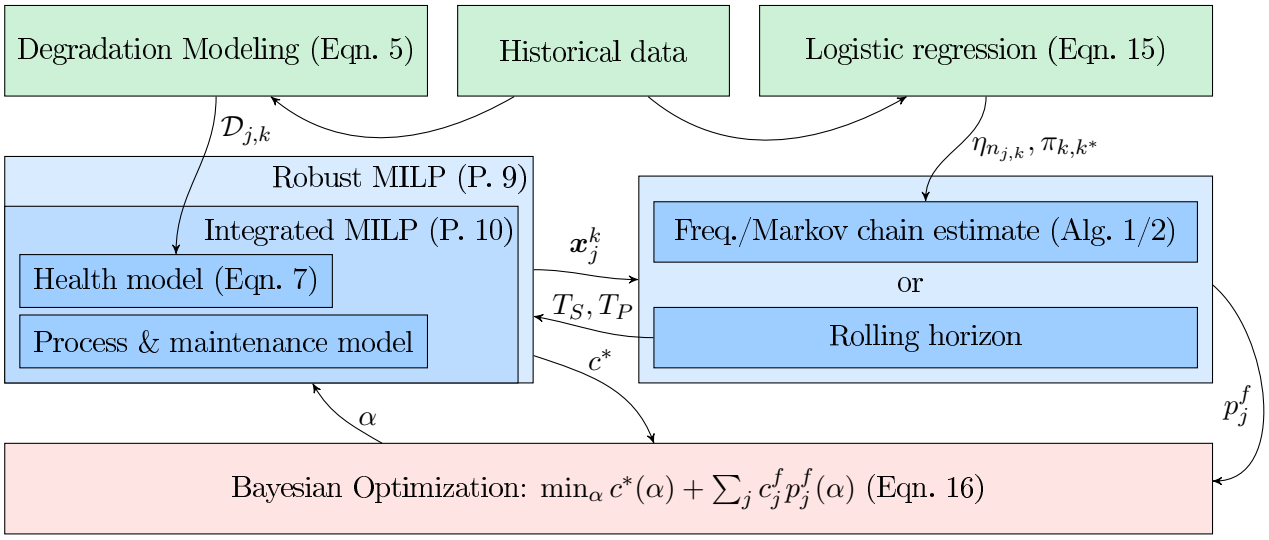


Figure 3.14: Flowchart of the proposed method.

In the context of computationally expensive models, we show that Bayesian optimization can be used effectively to optimize the uncertainty set in robust optimization and balance the trade-off between preventive and corrective maintenance.

Chapter 4

A robust approach to black-box constrained optimization

4.1 Introduction

In mathematical programming, optimization under uncertainty often focuses on parametric uncertainty [109, 110, 111, 112, 6]. But many application areas rely on uncertain, expensive to evaluate black-box functions, e.g., automatic chemical design, production planning, scheduling with equipment degradation, adaptive vehicle routing, automatic control and robotics, and biological systems [113, 114, 115, 116, 117, 118, 119].

Bayesian optimization optimizes such functions by (i) fitting a Gaussian process to a small number of collected data points and (ii) subsequently choosing new sampling points using an acquisition function [49, 120, 121]. The Bayesian optimization literature also considers problems with black-box constraints, e.g., by multiplying the acquisition function with the probability of constraint satisfaction [122, 123, 124]. The global optimization community often handles black-box constraints by (i) generating a small data set from the black box function, (ii) fitting a surrogate model to this data, and (iii) replacing the black box constraint by the surrogate model [125, 126, 127, 128, 129, 130, 131]. This approach, however, rarely considers uncertainty in the black box function.

One way of including uncertain black-box functions into the optimization problem is to consider the surrogate model's parameters to be uncertain and use classical parametric uncertainty methods. Hüllen et al. [132] recently demonstrated this approach for polynomial surrogates using robust optimization. This chapter proposes a more direct approach utilizing probabilistic surrogate models to model the uncertain curves. We study optimization problems with constraints which aggregate black-box functions:

$$\sum_i \tilde{a}_i x_i \leq b \tag{4.1a}$$

$$\tilde{a}_i = g(\boldsymbol{z}_i), \tag{4.1b}$$

where x_i is a decision variable and $\tilde{a}_i$ depends on a vector of decision variables $\boldsymbol{z}_i \in \mathbb{R}^k$ through a black-box function $g(\cdot)$. Constraint (4.1) occurs in many highly relevant applications. In production planning, one may limit the total allowed equipment degradation $\sum_i r(p_i)\Delta t_i \leq b$, where $r(p_i)$ is the black-box degradation rate depending on production p_i in time period i and Δt_i is equipment operation time in period i [119]. A second example is vehicle routing, where the total travelling time $\sum_i \Delta t(t_i, s_i, d_i)\gamma_i$ is the sum of travelling times $\Delta t(t_i, s_i, d_i)$ for individual legs i, dependent on starting time t_i, source s_i, and destination d_i, and γ_i is a binary variable indicating whether leg i is part of the route. A third example is project scheduling under uncertainty in which duration uncertainty may be aggregated over multiple activities [133]. Lastly, the drill scheduling case study described in detail later in this chapter is an industrially relevant example.

When black-box constraints are risk or safety-critical, hedging solutions against uncertainty is essential. Evaluating black-box functions may require expensive computer simulations or physical experiments, so available data is generally limited and may be subject to model errors and measurement noise. We therefore consider the function $g(\cdot)$ to be uncertain and aim to find solutions for which Constraint (4.1) holds with confidence $1 - \alpha$:

$$P\left(\sum_i g(\boldsymbol{z}_i)x_i \leq b\right) \geq 1 - \alpha. \tag{4.2}$$

To capture the uncertainty in $g(\cdot)$, we model it by stochastic surrogate models. A common stochastic surrogate is the Gaussian process (GP) model. Depending on the underlying data generating distribution, however, a GP may be an inadequate model. Warped Gaussian processes, which map observations to a latent space using a warping function, are an alternative, more flexible model [51]. This chapter considers both standard and warped GPs.

We note that other contributions have connected Bayesian optimization with robust optimization [134, 135, 136, 137]. These works have focused on implementation errors. An adversary can perturb the input $\boldsymbol{x}$ to a black-box function f by $\boldsymbol{\delta} \in \mathcal{U}$. Robust solutions optimize performance under the worst-case perturbation realization: $\min_{\boldsymbol{x}\in\mathbb{R}^n} \max_{\boldsymbol{\delta}\in\mathcal{U}} f(\boldsymbol{x}+\boldsymbol{\delta})$. Our work does not consider implementation errors. We instead focus on the setting where the uncertainty is in the output of the black-box function. Most of these works also focus on uncertain black-box objective functions with some making extensions to uncertain black-box constraints [134, 135]. Our work focuses on aggregated black-box constraints which are relevant in safety-critical applications. To the best of our knowledge, no prior work has connected either warped GPs or aggregated black-box constraints with robust optimization.

Contributions. For the standard GP model, we show how chance constraint Eq. (4.2) can be exactly reformulated as a deterministic constraint using existing approaches. For the warped case, we develop a robust optimization approach which conservatively approximates the chance constraint. By constructing decision-dependent uncertainty sets from confidence ellipsoids based on the warped GP models, we obtain probabilistic constraint violation bounds. We utilize Wolfe duality to reformulate the resulting robust optimization problem and obtain explicit deterministic robust counterparts. This reformulation expresses uncertain constraints, modeled by GPs, as deterministic constraints with a guaranteed probability of constraint satisfaction, i.e., deterministic approximations to a chance constraint. We analyze convexity conditions of the warping function under which the Wolfe duality based reformulation is applicable. For non-convex cases, we develop a global optimization strategy which utilizes problem structure. To reduce solution conservatism, we furthermore propose an iterative a posteriori procedure of selecting the uncertainty set size which complements the obtained a priori guarantee.

We show how the proposed approach hedges against uncertainty in learned curves for two case studies: i) a production planning-inspired case study with an uncertain price-supply curve and ii) an industrially relevant drill-scheduling case study with uncertain motor degradation characteristics. For the drill-scheduling case study we develop a custom strategy for dealing with discrete decisions.

Notation. See Appendix H for a table of notation.

4.2 Method

We have discussed standard and warped Gaussian processes in Sections 2.3.2 and 2.3.3. Section 4.2.1 discusses an exact chance-constraint reformulation for the standard Gaussian process-based approach while Sections 4.2.2 and 4.2.3 outline our proposed robust approximation approach.

4.2.1 Standard GPs: chance constrained optimization

When $g(\cdot)$ is modeled well by a standard GP $G_{\boldsymbol{x}}$, chance constraint Eq. (4.2) can be exactly replaced by a deterministic equivalent [138]. Since $\{G_{\boldsymbol{z}_1}, \dots G_{\boldsymbol{z}_n}\} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ is normal distributed, the linear combination:

$$\beta = \sum_{i=1}^{n} G_{\boldsymbol{z}_i} x_i$$

is also normal distributed with distribution:

$$\beta \sim \mathcal{N}\left(\sum_{i=1}^{n} \mu_i x_i, \sum_{i=1}^{n}\sum_{j=1}^{n} x_i \sigma_{i,j}^2 x_j\right).$$

Note that we have suppressed the dependence of $\boldsymbol{\mu}$ and Σ on $\boldsymbol{z}_i$ for notational simplicity. For a given confidence level α, we can therefore replace chance constraint Eq. (4.2) by:

$$\sum_{i \in S} \mu_i x_i + F(1-\alpha) \cdot \sqrt{\sum_{i,j \in S} x_i \sigma_{i,j}^2 x_j} \leq b, \tag{4.3}$$

where $F(\cdot)$ is the cumulative distribution function of the standard normal distribution. If the GP models $g(\cdot)$ well, Eq. (4.3) is an exact deterministic reformulation of chance constraint Eq. (4.2).

4.2.2 Warped GPs: robust approximation

If $g(\cdot)$ is insufficiently modeled by a standard GP, a warped GP may be a more suitable model [51]. In this case, a direct reformulation of the chance constraint as outlined above for the standard GP case is not known. Such chance constraints are generally addressed by (i) sample approximation [139, 140, 141] or (ii) safe outer-approximation [142, 143, 144, 145, 146, 147]. Instead, we develop a robust approximation. Recall that a warped GP uses a strictly increasing warping function $h : \mathbb{R} \to \mathbb{R}$ to warp observations y_i into a latent space: $\xi_i = h(y_i)$. A standard GP is then fit to the data in the latent space. Consider an optimization problem containing a nominal version of Constraint (4.1), where the black box function $g(\boldsymbol{y}_i)$ is replaced by the mean prediction $\xi_i = \mu(\boldsymbol{y}_i)$ of the GP, warped back into the observation space using the inverse of the warping function $y_i = h^{-1}(\xi_i)$:

$$\min_{(\boldsymbol{x},\boldsymbol{z}) \in \mathcal{X}} \quad f(\boldsymbol{x}, \boldsymbol{z}) \tag{4.4a}$$

$$\text{s.t.} \quad \sum_i h^{-1}(\mu(\boldsymbol{z}_i)) x_i \leq b, \qquad i \in [n] \tag{4.4b}$$

where $\boldsymbol{z}$ is the vector containing all elements of $\boldsymbol{z}_i, \forall i$ and the objective function $f : \mathcal{X} \to \mathbb{R}$ is assumed to be known explicitly, i.e., it is not a black-box function. Clearly, a solution to Problem (4.4) is not guaranteed to be feasible in practice if the prediction $\mu(\boldsymbol{z}_i)$ is uncertain. Using the full multivariate distribution generated by the sampling points $\{\boldsymbol{z}_i\}$, we can construct

an α–confidence ellipsoid in the latent space:

$$\mathcal{E}^{\alpha}(\boldsymbol{z}) = \left\{ \boldsymbol{\xi} : (\boldsymbol{\xi} - \boldsymbol{\mu}(\boldsymbol{z}))^{\intercal} \Sigma^{-1}(\boldsymbol{z}) (\boldsymbol{\xi} - \boldsymbol{\mu}(\boldsymbol{z})) \leq F_n^{1-\alpha} \right\}. \tag{4.5}$$

Here, $F_n^{1-\alpha}$ is the cumulative distribution function of the χ^2 distribution with n degrees of freedom. Note that when the GP kernel is positive definite, the covariance matrix Σ is also positive definite and the inverse Σ^{-1} exists. Assuming that the warped GP models the black-box function well, $\mathcal{E}^{\alpha}(\boldsymbol{z})$ contains the true value $h(g(\boldsymbol{z}_i))$ with probability at least $1 - \alpha$. We therefore construct the following robust optimization problem:

$$\min_{(\boldsymbol{x},\boldsymbol{z}) \in \mathcal{X}} f(\boldsymbol{x}, \boldsymbol{z}) \tag{4.6a}$$

$$\text{s.t. } \boldsymbol{y}^{\intercal}\boldsymbol{x} \leq b \qquad \forall \boldsymbol{y} : h(\boldsymbol{y}) \in \mathcal{E}^{\alpha}(\boldsymbol{z}) \tag{4.6b}$$

Any solution to Problem (4.6) is feasible with probability at least $1 - \alpha$ given that the warped GP models the underlying data generating distribution well. Alternatively, we can take the warping into the uncertainty set:

$$\min_{(\boldsymbol{x},\boldsymbol{z}) \in \mathcal{X}} \quad f(\boldsymbol{x}, \boldsymbol{z}) \tag{4.7a}$$

$$\text{s.t.} \quad \boldsymbol{y}^{\intercal}\boldsymbol{x} \leq b \qquad \forall \boldsymbol{y} \in \mathcal{U}(\boldsymbol{z}) \tag{4.7b}$$

where $\mathcal{U}$:

$$\mathcal{U}(\boldsymbol{z}) = \left\{ \boldsymbol{y} \in \mathbb{R}^l : (h(\boldsymbol{y}) - \boldsymbol{\mu}(\boldsymbol{z}))^{\intercal} \Sigma^{-1}(\boldsymbol{z}) (h(\boldsymbol{y}) - \boldsymbol{\mu}(\boldsymbol{z})) \leq F_n^{1-\alpha} \right\}. \tag{4.8}$$

Note that Problem (4.7) can also be interpreted as approximating a robust problem with an uncertainty set over functions $\tilde{g} \in \mathcal{U}^g$ (see Theorem F.1, Appendix F). Fig. 4.1 shows an example of the ellipsoidal and warped sets $\mathcal{E}^{\alpha}$ and $\mathcal{U}$ for $n = 2$. The warped set $\mathcal{U}$ (Eqn. 4.8) may or may not be convex, depending on the warping function $h(\cdot)$.

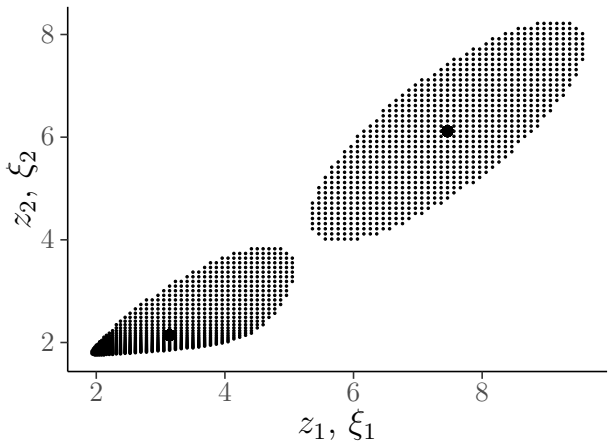


Figure 4.1: Example of uncertainty sets $\mathcal{E}^\alpha$ in latent and $\mathcal{U}$ in observation space.

Reformulation for convex warped sets $\mathcal{U}$

We first assume that the warped set $\mathcal{U}$ retains convexity and utilize Wolfe duality to reformulate the semi-infinite Problem (4.7) into a deterministic problem with a finite number of constraints. Consider the min-max equivalent of Problem (4.7):

$$\min_{(\boldsymbol{x},\boldsymbol{z})\in\mathcal{X}} \quad f(\boldsymbol{x},\boldsymbol{z}) \tag{4.9a}$$

$$\text{s.t.} \quad \max_{\boldsymbol{y}\in\mathcal{U}(\boldsymbol{z})} \boldsymbol{y}^\intercal \boldsymbol{x} \leq b \tag{4.9b}$$

When $\mathcal{U}$ is convex, the inner maximization in Eq. (4.9b) is convex:

$$\max_{\boldsymbol{y}} \boldsymbol{y}^\intercal \boldsymbol{x} \tag{4.10a}$$

$$\text{s.t. } (\boldsymbol{h}(\boldsymbol{y}) - \boldsymbol{\mu})^\intercal \Sigma^{-1} (\boldsymbol{h}(\boldsymbol{y}) - \boldsymbol{\mu}) \leq F_n^{1-\alpha}, \tag{4.10b}$$

Note that we suppress the dependence of $\boldsymbol{\mu}$ and Σ on $\boldsymbol{z}$ for notational simplicity from here onward. Problem (4.10) generally does not have a simple closed form solution. Instead, we can use Wolfe duality to transform Problem (4.10) into an equivalent minimization problem, leading to a deterministic reformulation of Problem (4.7):

$$\min_{(\boldsymbol{x},\boldsymbol{z})\in X,\boldsymbol{y},u} \quad f(\boldsymbol{x}) \tag{4.11a}$$

$$\text{s.t} \quad \boldsymbol{y}^\intercal \boldsymbol{x} + u \cdot \left((\boldsymbol{h}(\boldsymbol{y}) - \boldsymbol{\mu})^\intercal \Sigma^{-1} (\boldsymbol{h}(\boldsymbol{y}) - \boldsymbol{\mu}) - F_n^{1-\alpha} \right) \leq b \tag{4.11b}$$

$$\boldsymbol{x} + 2u \cdot \nabla \boldsymbol{h}(\boldsymbol{y}) \Sigma^{-1} (\boldsymbol{h}(\boldsymbol{y}) - \boldsymbol{\mu}) = \boldsymbol{0} \tag{4.11c}$$

$$u \geq 0, \tag{4.11d}$$

where u is a dual variable, $\nabla \boldsymbol{h}(\boldsymbol{y}) = \text{diag}(h'(y_i))$, and Constraint (4.11c) is the Karush-Kuhn-Tucker (KKT) stationarity condition. Note that when $F_n^{1-\alpha} > 0$, i.e. for any non-zero $1-\alpha$, the uncertainty set $\mathcal{U}$ contains a feasible interior point and Slater's condition holds. Furthermore, since $\mathcal{U}$ is convex and the objective function is concave and continously differentiable, strong duality holds. The dual feasibility constraint $u \geq 0$ ensures that any point which satisfies the stationarity condition is a maximum, not a minimum. Also note that, unless $\boldsymbol{x} = \boldsymbol{0}$, the stationarity condition means that $u \neq 0$ and, due to complementary slackness, $w(\boldsymbol{y}, \boldsymbol{z}) = 0$, i.e.:

$$(\boldsymbol{h}(\boldsymbol{y}) - \boldsymbol{\mu})^\intercal \Sigma^{-1} (\boldsymbol{h}(\boldsymbol{y}) - \boldsymbol{\mu}) = F_n^{1-\alpha}. \tag{4.12}$$

Furthermore, we can reformulate Eq. (4.11c) to:

$$\boldsymbol{h}(\boldsymbol{y}) - \boldsymbol{\mu} = -\frac{1}{2u} \Sigma (\nabla \boldsymbol{h}(\boldsymbol{y}))^{-1} \boldsymbol{x}$$

Substituting this in Eq. (4.12) yields:

$$\frac{1}{4u^2} \boldsymbol{x}^\intercal \left(\nabla \boldsymbol{h}(\boldsymbol{y}) \right)^{-1} \Sigma \left(\nabla \boldsymbol{h}(\boldsymbol{y}) \right)^{-1} \boldsymbol{x} = F_n^{1-\alpha}.$$

This leads to a slightly different formulation which has the advantage that it does not depend on the inverse of the covariance matrix Σ^{-1}:

$$\min_{(\boldsymbol{x}, \boldsymbol{z}) \in \mathcal{X}, \boldsymbol{y}, u} \quad f(\boldsymbol{x}, \boldsymbol{z}) \tag{4.13a}$$

$$\text{s.t} \quad \boldsymbol{y}^\intercal \boldsymbol{x} \leq b \tag{4.13b}$$

$$\Sigma \left(\nabla \boldsymbol{h}(\boldsymbol{y}) \right)^{-1} \boldsymbol{x} + 2u \cdot (\boldsymbol{h}(\boldsymbol{y}) - \boldsymbol{\mu}) = \boldsymbol{0} \tag{4.13c}$$

$$4u^2 F_n^{1-\alpha} = \boldsymbol{x}^\intercal \left(\nabla \boldsymbol{h}(\boldsymbol{y}) \right)^{-1} \Sigma \left(\nabla \boldsymbol{h}(\boldsymbol{y}) \right)^{-1} \boldsymbol{x} \tag{4.13d}$$

$$u \geq 0, \tag{4.13e}$$

where the inverse $(\nabla \boldsymbol{h}(\boldsymbol{y}))^{-1}$ exists when $h'(y) \neq 0$ because $\nabla \boldsymbol{h}(\boldsymbol{y})$ is a diagonal matrix. Note that $h(y_i) > 0, \ \forall y_i \in \mathbb{R}$ for the neural net style warping function in Eq. (2.7). Formulation (4.13) has the advantages that it does not depend on the inverse $\Sigma^{-1}(\boldsymbol{z})$, which generally cannot be formulated explicitly, and that it can be solved using off-the-shelf solver software. While Problem (4.13) is non-convex due to bilinearities between $\boldsymbol{y}$ and $\boldsymbol{x}$, the potentially non-convex objective function f, and the dependence of μ and Σ on $\boldsymbol{z}$, even a non-optimal solution will be robustly feasible.

Convexity conditions

Section 4.2.2 relies on the convexity of the inner maximization problem. If $\mathcal{U}$ is non-convex, Problem (4.13) is not necessarily equivalent to Problem (4.7) as there may be more than one KKT point. Since $\mathcal{U}$ is the confidence set of a multivariate distribution, however, may often be convex, especially when the distribution is unimodal. The following section analyzes conditions where the Wolfe duality approach is justified.

First consider the inner maximization Problem (4.10) transformed into the latent space by substituting $\boldsymbol{y} = \boldsymbol{h}^{-1}(\boldsymbol{\xi})$:

$$\max_{\boldsymbol{\xi}} \quad \boldsymbol{x}^\intercal \boldsymbol{h}^{-1}(\boldsymbol{\xi}) \tag{4.14a}$$

$$\text{s.t.} \quad (\boldsymbol{\xi} - \boldsymbol{\mu})^\intercal \Sigma^{-1} (\boldsymbol{\xi} - \boldsymbol{\mu}) \leq F_n^{1-\alpha}, \tag{4.14b}$$

which depends on the generally not explicitly known inverse warping function h^{-1}. We further state the well known result on the derivative of inverse functions [148]:

Lemma 4.1. *If $f : \mathbb{R} \to \mathbb{R}$ is continuous, bijective, and differentiable and $f'(f^{-1}(x)) \neq 0$, then $[f^{-1}]'(x) = \frac{1}{f'(f^{-1}(x))}$.*

Using this, we can show the following proposition.

Theorem 4.1. *Let the warping function $h(\cdot)$ be strictly concave (convex) and let $x_i \geq 0$ (≤ 0), $\forall i$, then the inner maximization Problem (4.10) has a unique KKT point.*

Proof. Note that Problem (4.14) is convex when $\boldsymbol{h}^{-1}$ is concave (convex) and $x_i \geq 0$ ($x_i \leq 0$), $\forall i$. The KKT conditions for Problems (4.10) and (4.14) are:

$$\boldsymbol{x} + 2u\nabla \boldsymbol{h}(\boldsymbol{y})\Sigma^{-1}(\boldsymbol{h}(\boldsymbol{y}) - \boldsymbol{\mu}) = \boldsymbol{0} \tag{4.15a}$$

$$(\boldsymbol{h}(\boldsymbol{y}) - \boldsymbol{\mu})^\intercal \Sigma^{-1}(\boldsymbol{h}(\boldsymbol{y}) - \boldsymbol{\mu}) = F_n^{1-\alpha} \tag{4.15b}$$

and:

$$\nabla \boldsymbol{h}^{-1}(\boldsymbol{\xi})\boldsymbol{x} + 2u\Sigma^{-1}(\boldsymbol{\xi} - \boldsymbol{\mu}) = \boldsymbol{0} \tag{4.16a}$$

$$(\boldsymbol{\xi} - \boldsymbol{\mu})^\intercal \Sigma^{-1}(\boldsymbol{\xi} - \boldsymbol{\mu}) = F_n^{1-\alpha}, \tag{4.16b}$$

where:

$$[\nabla \boldsymbol{h}^{-1}(\boldsymbol{\xi})]_{i,j} = \begin{cases} h^{-1}(\xi_i), & i = j \\ 0, & i \neq j \end{cases}. \tag{4.17}$$

By Lemma 4.1:

$$[\nabla \boldsymbol{h}^{-1}(\boldsymbol{\xi})]_{i,j} = \begin{cases} h^{-1}(\xi_i), & i = j \\ 0, & i \neq j \end{cases} = \begin{cases} \frac{1}{h'(h^{-1}(\xi_i))}, & i = j \\ 0, & i \neq j \end{cases} = [\nabla \boldsymbol{h}(\boldsymbol{h}^{-1}(\boldsymbol{\xi}))]_{i,j}^{-1}. \tag{4.18}$$

So Problem 4.16 is equivalent to:

$$\nabla[\boldsymbol{h}(\boldsymbol{h}^{-1}(\boldsymbol{\xi}))]^{-1}\boldsymbol{x} + 2u\Sigma^{-1}(\boldsymbol{\xi} - \boldsymbol{\mu}) = \boldsymbol{0} \tag{4.19a}$$

$$(\boldsymbol{\xi} - \boldsymbol{\mu})^\intercal \Sigma^{-1}(\boldsymbol{z} - \boldsymbol{\mu}) = F_n^{1-\alpha}. \tag{4.19b}$$

Let $\boldsymbol{y}^*$ be a KKT point for Problem (4.10), then $\boldsymbol{z}^* = h(\boldsymbol{y}^*)$ is clearly a solution to Problem (4.19), and therefore a KKT point for Problem (4.14). Since Problem (4.14) is convex, $\boldsymbol{y}^*$

is unique. □ □

Strategy for non-convex warped sets $\mathcal{U}$

When $\mathcal{U}$ is non-convex, we need to globally optimize the inner maximization problem efficiently. To this end we develop a custom divide and conquer strategy which makes use of the problem's special structure. We first note the following properties of the inner maximization problem.

Lemma 4.2. *Let $\mathcal{Y}$ be the set of solutions of Problem 4.10, then $\exists \boldsymbol{y}^* \in \mathcal{Y}$ which is on the boundary of $\mathcal{U}$, i.e., $\boldsymbol{y}^* \in \partial\mathcal{U}$.*

Proof. See Appendix G. □

Lemma 4.3. *The bounding box of an ellipsoid $(\boldsymbol{x}-\boldsymbol{\mu})^\intercal\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \leq r^2$ is given by the extreme points $\bar{x}_i = \mu_i + r\sigma_{ii}$ and $\underline{x}_i = \mu_i - r\sigma_{ii}$.*

Proof. See Appendix G. □

Lemma 4.4. *Consider a version of Problem (4.14) in which the ellipsoidal feasible region is replaced by its bounding box:*

$$\max_{\boldsymbol{\xi}} \ \boldsymbol{x}^\intercal \boldsymbol{h}^{-1}(\boldsymbol{\xi}) \tag{4.20a}$$

$$\text{s.t. } \mu_i - r\sigma_{ii} \leq \xi_i \leq \mu_i + r\sigma_{ii} \qquad \forall i. \tag{4.20b}$$

If $x_i \geq 0, \forall i$, the corner $\boldsymbol{\xi}^*$ of the bounding box with $\xi_i^* = \mu_i + r\sigma_{ii}, \forall i$, is an optimal solution to this problem.

Proof. Let $\boldsymbol{\xi}^*$ be an optimal solution to Problem (4.20) that $\boldsymbol{\xi}^*$ lies on the boundary of the feasible space (Lemma G.2). Assume $\exists i$, s.t., $\xi_i^* < \mu + r\sigma_{ii}$. Because h^{-1} is strictly monotonically increasing and $x_i \geq 0$, $x_i h^{-1}(\xi_i) > x_i h^{-1}(\mu + r\sigma_{ii})$. Therefore we can construct a new solution $\hat{\boldsymbol{\xi}}$:

$$\hat{\xi_j} = \begin{cases} \xi_j^* & j \neq i \\ \mu_j + r\sigma_{jj} & j = i, \end{cases} \tag{4.21}$$

for which $\boldsymbol{x}^\intercal \hat{\boldsymbol{\xi}} \geq \boldsymbol{x}^\intercal \boldsymbol{\xi}^*$, which is a contradiction. □

Theorem 4.2. *Let $\bar{\boldsymbol{\xi}}$ $(\boldsymbol{\xi})$ be $\bar{\xi}_i = \mu_i + r\sigma_{ii}$ $(\xi_i = \mu_i - r\sigma_{ii})$ and $\boldsymbol{\xi}^*$ an optimal solution to Problem (4.14). Then (i) $\boldsymbol{x}^\intercal \boldsymbol{h}^{-1}(\boldsymbol{\xi}) \leq \boldsymbol{x}^\intercal \boldsymbol{h}^{-1}(\boldsymbol{\xi}^*) \leq \boldsymbol{x}^\intercal \boldsymbol{h}^{-1}(\bar{\boldsymbol{\xi}})$ and (ii) $\frac{\boldsymbol{x}^\intercal \boldsymbol{h}^{-1}(\bar{\boldsymbol{\xi}})}{\boldsymbol{x}^\intercal \boldsymbol{h}^{-1}(\boldsymbol{\xi}^*)} \leq \frac{\boldsymbol{x}^\intercal \boldsymbol{h}^{-1}(\boldsymbol{\mu}+r\boldsymbol{\sigma})}{\boldsymbol{x}^\intercal \boldsymbol{h}^{-1}(\mu+r\boldsymbol{\sigma}/\sqrt{\boldsymbol{\sigma}^\intercal \Sigma^{-1}\boldsymbol{\sigma}})}$ where $\boldsymbol{\sigma} = [\sigma_{11}, \ldots, \sigma_{nn}]^\intercal$.*

Proof. Part (i) follows immediately from Lemma 4.4 because Problem (4.20) is a relaxation of Problem (4.14). To prove (ii), first consider a scaled version of the bounding box with corner $\hat{\boldsymbol{\xi}} = \boldsymbol{\mu} + \lambda r \boldsymbol{\sigma}$. Assume that scaling factor $\lambda \in [0, 1]$ is chosen so that the corner of the scaled box $\hat{\boldsymbol{\xi}}$ lies on the ellipsoid, i.e.:

$$\left(\hat{\boldsymbol{\xi}} - \boldsymbol{\mu}\right)^\intercal \Sigma^{-1} \left(\hat{\boldsymbol{\xi}} - \boldsymbol{\mu}\right) = r^2.$$

Substituting $\hat{\boldsymbol{\xi}} = \boldsymbol{\mu} + \lambda r \boldsymbol{\sigma}$ and rearranging leads to

$$\lambda = \frac{1}{\sqrt{\boldsymbol{\sigma}\Sigma^{-1}\boldsymbol{\sigma}}}$$

Furthermore, since $\hat{\boldsymbol{\xi}}$ is contained in the ellipsoid, $\frac{\boldsymbol{x}^\intercal \boldsymbol{h}^{-1}(\bar{\boldsymbol{\xi}})}{\boldsymbol{x}^\intercal \boldsymbol{h}^{-1}(\boldsymbol{\xi}^*)} \leq \frac{\boldsymbol{x}^\intercal \boldsymbol{h}^{-1}(\bar{\boldsymbol{\xi}})}{\boldsymbol{x}^\intercal \boldsymbol{h}^{-1}(\hat{\boldsymbol{\xi}})}$. Substituting $\bar{\boldsymbol{\xi}} = \boldsymbol{\mu} + r\boldsymbol{\sigma}$ and $\hat{\boldsymbol{\xi}} = \boldsymbol{\mu} + \frac{r\boldsymbol{\sigma}}{\sqrt{\boldsymbol{\sigma}\Sigma^{-1}\boldsymbol{\sigma}}}$ concludes the proof. □

Using Lemma G.2 and Theorem 4.2 we develop the spatial branching strategy shown in Algorithm 3. Algorithm 3 works similarly to other elimination-based algorithms [149, 150]. It starts by outer-approximating the ellipsoid by its bounding box and evaluating the objective $\boldsymbol{x}^\intercal \boldsymbol{h}^{-1}(\boldsymbol{\xi})$ at the two corner points $(\boldsymbol{\xi}, \bar{\boldsymbol{\xi}})$, obtaining an upper and lower bound (Theorem 4.2). The algorithm then branches on the dimension of largest width. Boxes can be pruned if they are fully inside or outside the ellipsoid (Lemma G.2). Theorem 4.2 also provides an a posteriori bound for the tightness of the outer approximation. In the special case when the ellipsoid is an n-D ball with zero mean, the bound can be simplified to an a priori bound: $\frac{\boldsymbol{x}^\intercal \boldsymbol{h}^{-1}(\bar{\boldsymbol{\xi}})}{\boldsymbol{x}^\intercal \boldsymbol{h}^{-1}(\boldsymbol{\xi}^*)} \leq \frac{h^{-1}(r)}{h^{-1}(\frac{r}{\sqrt{n}})}$.

Algorithm 3 Globally optimize inner maximization problem

```
lower bound, upper bound ← x^⊤ h^{-1}(ξ), x^⊤ h^{-1}(ξ̄)
nodes ← [(ξ, ξ̄)]
while (upper bound - lower bound)/upper bound ≤ ε do
    (ξ, ξ̄) ← choose element in nodes with largest x^⊤ h^{-1}(ξ̄)
    upper bound ← x^⊤ h^{-1}(ξ̄)
    children ← split (ξ, ξ̄) along single axis
    for (ξ, ξ̄) in children do
        if (ξ, ξ̄) contains boundary point of ellipsoid and lower bound ≤ x^⊤ h^{-1}(ξ̄) then
            add (ξ, ξ̄) to nodes
            lower bound ← min{x^⊤ h^{-1}(ξ), lower bound}
        end if
    end for
end while
```

4.2.3 Iterative a posteriori approximation

The a priori probabilistic bound implied by $\mathcal{E}^\alpha$ may be overly conservative. Alg. 4 is an alternative, less conservative strategy that iteratively determines the uncertainty set size. Starting

Algorithm 4 Posteriori approximation

```
α ← ε_0
while ‖ε − ε_0‖ ≥ δ do
    x ← solution of Problem (4.13) with α
    ε ← percentage of 1000 samples ξ_i drawn from N(μ(z), Σ(z)) with x^⊤ h^{-1}(ξ_i) ≤ b
    if ε − ε_0 ≥ 0 then
        α_U ← α, ε_U ← ε
    else
        α_L ← α, ε_L ← ε
    end if
    α ← (α_L + α_U)/2                                   ▷ Bisection search
end while
```

with the confidence level α equal to the target feasibility ϵ_0, Alg. 2 iteratively solves the robust optimization problem, estimates the feasibility of the solution, and consequently adjusts the confidence level α using bisection search. To estimate the feasibility of a solution, we generate 10000 random samples $\boldsymbol{\xi}_i$ from the warped GP distribution $\mathcal{N}(\boldsymbol{\mu}(\boldsymbol{z}), \Sigma(\boldsymbol{z}))$, evaluate the constraint $\boldsymbol{x}^\intercal \boldsymbol{h}^{-1}(\boldsymbol{\xi}_i)$ for each sample i, and calculate the percentage of samples for which $\boldsymbol{x}^\intercal \boldsymbol{h}^{-1}(\boldsymbol{\xi}_i) \leq b$. The search terminates when a solution has been found that is sufficiently close (tolerance δ) to the target feasibility ϵ_0.

4.3 Case studies

4.3.1 Production planning

Our first case study is inspired by production planning. Assume that a company wants to decide how much product x_t to produce in a number of subsequent time periods $\boldsymbol{x} = [x_1, \ldots, x_t, \ldots, x_T]$. There is a known cost of production c_t which may vary from period to period. The company seeks to maximize its profit ψ, which depends on the total production cost $\sum_t c_t x_t$ and revenue $\sum_t \tilde{p}_t x_t$. Here $\tilde{p}_t$ is the price at which the product can be sold in period t. The company has to sell all its product in the same time period, e.g., because the product is perishable. The sale price depends on the amount the company produces in that period $\tilde{p}_t = \tilde{p}(x_t)$, e.g., because the company has a very large market share.

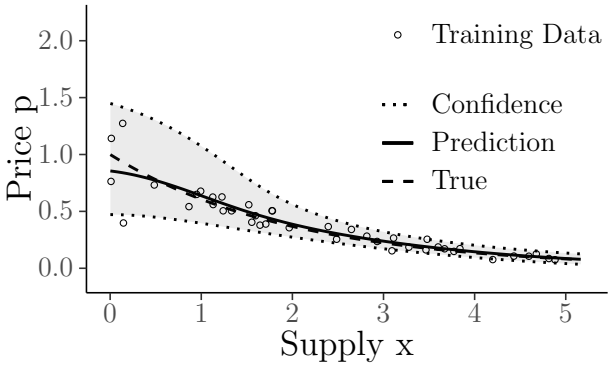


Figure 4.2: GP trained using 50 observations from the price-supply curve $p(x_t) = \exp(-x_t) + \epsilon$ with non-uniform Gaussian noise $\epsilon \sim \mathcal{N}(0, 4 \cdot 0.3 \cdot \exp(-x/2))$. The confidence region is two standard deviations wide.

The company uses GP regression to predict $\tilde{p}(x_t)$ based on limited historical data. Additional features, e.g., season and general state of the economy, could be part of this regression but are irrelevant for our purpose as they are not decision variables. The prediction has to be considered uncertain and the company wants a production plan guaranteeing a certain profit with some confidence. This decision problem can be formulated as a chance constrained optimization problem:

$$\max_{\boldsymbol{x} \in \mathbb{R}^T, \psi} \quad \psi \tag{4.22a}$$

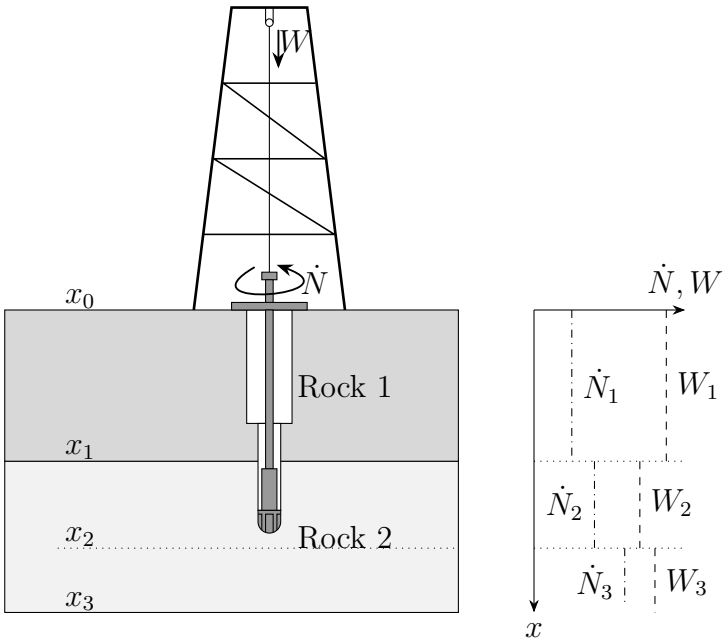


Figure 4.3: Illustration of drill scheduling problem with two rock types. The rock type changes at x_1, maintenance is scheduled at x_2, and the target depth is x_3. The right side shows an example schedule of the decision variables $\dot{N}$ and W

$$\text{s.t} \quad P\left(\sum_{t=1}^{T} \left(\tilde{p}(x_t) - c_t\right) x_t \geq \psi\right) \geq 1 - \alpha \tag{4.22b}$$

Choosing $p(x_t) = \exp(-x_t)$ as ground truth for the price-supply curve, we generate noisy data $\tilde{p}(x_t) = p(x_t) + \epsilon$ and fit a GP surrogate as shown in Fig. 4.2. We consider uniform Gaussian noise ($\epsilon \sim \mathcal{N}(0, \sigma_{\text{noise}})$) and non-uniform Gaussian noise ($\epsilon \sim \mathcal{N}(0, 4 \cdot \sigma_{\text{noise}} \cdot \exp(-x/2))$), where σ_{noise} is a parameter determining the amount of noise. We use a squared exponential kernel for this case study, but the proposed method does not generally rely on a specific choice for $k(\cdot, \cdot)$.

4.3.2 Drill scheduling

The objective in drilling oil wells is generally minimizing total well completion time. The aim of the drill scheduling problem, illustrated in Fig. 4.3, is to find a schedule of the two decision variables, rotational speed $\dot{N} \in \mathbb{R}$ and weight on bit $W \in \mathbb{R}$, as a function of depth $x \in \mathbb{R}$. Current practice often consists of minimizing the total drilling time, which depends on $\dot{N}$ and W through a non-linear bit-rock interaction model [151] and the motor's power-curves (see

Appendix I). Total well completion time, however, also depends on maintenance time. Current practice may increase maintenance time because drilling quickly can have a detrimental effect on motor degradation. Furthermore, the motors degradation characteristics are subject to uncertainty and are often obtained through a mixture of experiments and expensive numerical simulations [5]. Other works have considered uncertain equipment degradation in scheduling applications [119, 152], but not with predicted degradation rates.

To find the optimal trade off between drilling and maintenance time, we propose a drill scheduling model which explicitly considers uncertainty in the motor degradation characteristics. First consider a model which discretizes the drill trajectory into n_d equidistant intervals:

$$\min_{\boldsymbol{W},\dot{\boldsymbol{N}},\boldsymbol{y},\boldsymbol{z},\boldsymbol{V},\boldsymbol{\Delta p},\boldsymbol{R}} \sum_{i=1}^{n_d} \left(\frac{\Delta x_i}{V_i} + z_i \Delta t_i^{\text{maint}} \right) \tag{4.23a}$$

$$\text{s.t } V_i = f(\dot{N}_i^{\text{top}}, W_i, \Delta p_i) \qquad \forall i \in [n_d] \tag{4.23b}$$

$$0 \leq R_i = \sum_{j=1}^{i} \left(\frac{\Delta x_j}{V_j} \cdot r(\Delta p_j) - y_j \right) \leq 1 \qquad \forall i \in [n_d] \tag{4.23c}$$

$$z_i \geq y_i, z_i \in \{0, 1\} \qquad \forall i \in [n_d], \tag{4.23d}$$

The rate of penetration V_i in each segment depends on the drill parameters ($\dot{N}_i$ and W_i) through the non-linear model in Appendix I. The rate of degradation $r(\cdot)$ is a black-box function of the differential pressure across the motor Δp. We model $r(\cdot)$ with a warped GP based on 10 data points from a curve obtained by Ba et al. [5] through a combination of experiments and numerical simulation. The maintenance indicator R_i keeps track of the total cumulative degradation of the motor. We assume the motor fails when R_i reaches 1. Binary variable z_i indicates whether maintenance is scheduled in segment i. If maintenance is scheduled, the continuous variable y_i resets the total degradation indicator R_i to zero. Note that the bit-rock interaction model depends on rock parameters which can change from segment to segment.

A major disadvantage of Model (4.23) is that it requires a large number of segments in order to get a good resolution on the optimal maintenance depth. To avoid this we propose, in analogy with continuous time formulations [153, 154], an alternative continuous depth scheduling

formulation:

$$\min_{\boldsymbol{W},\dot{\boldsymbol{N}},\boldsymbol{x},\boldsymbol{\Delta p},\boldsymbol{V},\boldsymbol{R}} \sum_{i\in N}\left(\frac{x_i - x_{i-1}}{V_i}\right) + \sum_{m\in M}\Delta t^{\text{maint}}(x_m) \tag{4.24a}$$

$$\text{s.t } V_i = f(\dot{N}_i, W_i, \Delta p_i) \qquad \forall i \in N = [n_d] \tag{4.24b}$$

$$R_m = \sum_{j=m^-}^{m}\left(\frac{x_j - x_{j-1}}{V_j}\cdot r(\Delta p_j)\right) \leq 1 \qquad \forall m \in M \cup \{n_d\} \tag{4.24c}$$

Model (4.24) only considers geological segments (segments with constant rock parameters) and maintenance induced segments. The vector $\boldsymbol{x}$ is ordered and contains the fixed rock formation depths as well as the variable maintenance depths. M is a set containing the indices of the variable maintenance depths. Fig. 4.3 shows an example instance where x_0, x_1, x_3 are fixed depths and x_2 is the variable depth of a maintenance event, i.e., $\boldsymbol{x} = [x_0, x_1, x_2, x_3]$ and $M = \{2\}$. The indices $i \in M$ of the variable maintenance depths are determined a priori, i.e., we decide both the number of maintenance events as well as the geological segment in which they occur a priori. m^- is either the index of the previous maintenance event or 1 if m is the first element in M.

While Problem (4.24) cannot decide the optimal number of maintenance events, it is easier to solve than Problem (4.23) because it does not contain integer variables and generally has a much smaller number of segments, i.e., fewer variables and constraints. The following discusses strategies for deciding the optimal number and segment assignment of maintenance events.

Integer strategy

In drill scheduling, the number of maintenance events m is generally small ($m \leq 4$). The number of geological segments n_d can be large in practice but will not be known a priori. We therefore consider groupings of segments into a small number ($n_d \leq 10$) of longer segments with average rock parameters which are known a priori. Given n_d and m, the combinatorial complexity of enumerating the maintenance-segment assignment problem is $N = \binom{n_d+m-1}{m}$. However, the optimal number of maintenance events m is a decision variable. Therefore, finding a globally optimal maintenance-segment assignment also requires enumerating different values of m.

Algorithm 5 Deriving upper bounds for m and x_m

1: $\boldsymbol{x}, \boldsymbol{R}, \hat{\boldsymbol{V}} \leftarrow$ solution of Problem (4.24) without Constraint (4.24c), $M = \emptyset$
2: $m \leftarrow \lfloor R_n \rfloor$ ▷ Upper bound on number of maint.
3: $\hat{\boldsymbol{x}}^m \leftarrow \mathbf{0} \in \mathbb{R}^m$ ▷ Vector of maintenance depths
4: **for** $i \in \{m, \dots, 1\}$ **do**
5: $\quad \hat{x}_i^m \leftarrow$ smallest depth s.t. Cons. (4.24c) is satisfied
6: **end for**

Alg. 5 derives upper bounds for the number of maintenance events m as well as their location. It starts by solving Problem (4.24) without any maintenance events and ignoring the upper bound on the degradation indicator $R_{n_d} \not\leq 1$. The floor of the maintenance indicator at the target depth x_{n_d}, $\lfloor R_{n_d} \rfloor$ is an upper bound for the necessary number of maintenance events m. Alg. 5 then starts at the target depth x_{n_d} and inserts $\lfloor R_{n_d} \rfloor$ maintenance events at the earliest possible points that satisfy the maintenance constraint. The locations $\hat{x}_m$ are upper bounds for the maintenance locations:

Lemma 4.5. *Let $\hat{\boldsymbol{x}}^m$ be the maintenance depths determined by Alg. 5. Let $\boldsymbol{x}^{*,m}$ contain a set of globally optimal maintenance depths. Let $\boldsymbol{x}^{m,*}$ further be padded with zeroes at the beginning such that $\hat{\boldsymbol{x}}^m$ and $\boldsymbol{x}^{*,m}$ have the same length. Then $x_i^{*,m} \leq \hat{x}_i^m$.*

Proof. Assume $x_i^{*,m} \leq \hat{x}_i^m$ but $x_{i+1}^{*,m} > \hat{x}_{i+1}^m$. Construct a new solution $(\boldsymbol{x}'^{,m}, \boldsymbol{V}')$ by moving $x_{i+1}^{*,m}$ to $\hat{x}_{i+1}^m$ and drilling at maximum speed $\hat{V}_{i+1}$ between $\hat{x}_{i+1}^m$ and $x_{i+1}^{*,m}$:

$$x_k'^{,m} = \begin{cases} x_k^{*,m} & k \neq i+1 \\ \hat{x}_k^m & k = i+1 \end{cases}, \begin{cases} V_k^* & k \neq i+1 \\ \hat{V}_k & k = i+1 \end{cases}.$$

$(\boldsymbol{x}'^m, \boldsymbol{V}')$ has drilling and maintenance cost lower than $(\boldsymbol{x}^{*,m}, \boldsymbol{V}^*)$, which is a contradiction. Therefore $x_i^{*,m} \leq \hat{x}_i^m \implies x_{i+1}^{*,m} \leq \hat{x}_{i+1}^m$. Furthermore, note that $x_m^{*,m} \leq \hat{x}_m^m$ (last maintenance event) has to be true by the same logic as above. The Lemma follows by induction. □

Lemma 4.5 reduces the number of maintenance-segment assignments to enumerate.

Note. *Let $\hat{\boldsymbol{x}}$ be the upper bounds on maintenance locations from Alg. 5. Let n_i be the segment containing $\hat{x}_i$. The complexity of enumerating the maintenance-segment assignment problem*

using the upper bounds from Alg. 5 is:

$$N = \sum_{i_1=1}^{n_1} \sum_{i_2=i_1}^{n_2} \dots \sum_{i_m=i_{m-1}}^{n_m} 1 = \sum_{i_1=1}^{n_1} \sum_{i_2=i_1}^{n_2} \dots \sum_{i_{m-1}=i_{m-2}}^{n_{m-1}} n_m - i_{m-1} + 1.$$

Heuristics

Alg. 5 is equivalent to minimizing the drilling cost without considering degradation — a strategy often used in practice. It provides feasible but likely suboptimal solutions to Problem (4.24), i.e., it can be used as a heuristic. We call this the *no–degradation heuristic* and propose a second, improved heuristic: the *boundary heuristic*, outlined in Alg. 6. Alg. 6 starts with the solution of the no–degradation heuristic (Alg. 5). It improves the solution by iteratively solving Problem (4.24) and reassigning maintenance events occurring at geological boundaries to the adjacent segment. It terminates after finding a solution with all maintenance events occurring in the interior of their segment. Note that moving a maintenance event occurring at a geological

Algorithm 6 Boundary heuristic

1: $\hat{M} \leftarrow$ no–degradation heuristic (Alg. 5)
2: $\boldsymbol{x} \leftarrow$ solve Problem (4.24) with $M = \hat{M}$
3: **while** $\exists m \in M$, s.t. x_m at geological boundary **do**
4: $\quad \hat{M} \leftarrow$ reassign m to neighboring segment, drop maintenance event if at x_0.
5: $\quad \boldsymbol{x} \leftarrow$ solve Problem (4.24) with $M = \hat{M}$
6: **end while**

boundary to the adjacent segment cannot lead to a worse solution, i.e. Alg. 6 is an anytime algorithm.

While it does not guarantee global optimality of the maintenance-segment assignment, the boundary heuristic may be useful for very large instances when enumeration is prohibitive.

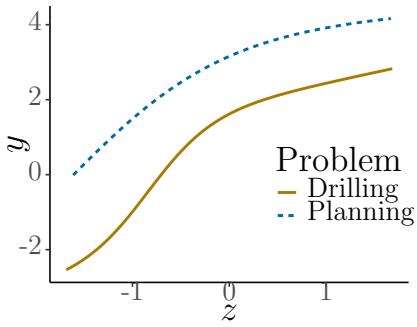


Figure 4.4: Warping functions for the drilling and production planning case studies. Input values are normalized to zero mean and $\sigma = 1$.

4.4 Results

The deterministic reformulations of both case studies were implemented in Pyomo (Version 5.6.8) [155, 156], an algebraic modeling language for expressing optimization problems. As part of this work, we developed a Python (Version 3.6.8) module which takes a GP model trained using the Python library GPy (Version 1.9.6) [157] and predicts $\mu(\boldsymbol{x})$ and $\Sigma(\boldsymbol{x})$ as Pyomo expressions. The module is available open source on GitHub [158]. This allows the easy incorporation of GP models into Pyomo optimization models. We use the interior-point convex optimization solver Ipopt [159] with a multistart strategy to solve the problem. Each instance was solved 30 times with a random starting point. The multistart procedure ends prematurely if it finds the same optimal solution (with a relative tolerance of 10^{-4}) 5 times.

Fig. 4.4 shows the warping functions for both case studies. Since the production planning warping function is concave and the production amounts x_t are strictly positive, Theorem 4.1 applies and the warped set $\mathcal{U}$ is convex. Theorem 4.1 cannot be applied to the drill scheduling case, because its warping function is neither convex nor concave. However, because the warping function is only slightly non-convex, the warped set $\mathcal{U}$ may still be convex for many instances. To avoid solving the bilevel problem directly we therefore use the following strategy: (i) solve the robust reformulation (Eq. 4.13), (ii) check feasibility of the obtained solution using Algorithm 3 (to a tolerance of 10^{-2}), and (iii) only solve the bilevel problem (Eq. 4.7) directly if the obtained solution is infeasible. For the instances considered in this work, the obtained solution always turns out to be feasible.

Period	1	2	3	4	5	6
Cost	0.1	0.05	0.01	0.02	0.1	0.15
Period	7	8	9	10	11	12
Cost	0.04	0.03	0.1	0.11	0.25	0.1

Table 4.1: Production costs c_t for each time period t.

4.4.1 Production planning

For the production planning case study, we consider 4 model instances with $T = 1, 2, 3$ and 6 time periods. Table 4.1 shows the cost of production $\boldsymbol{c}$. We solve each instance for 30 different

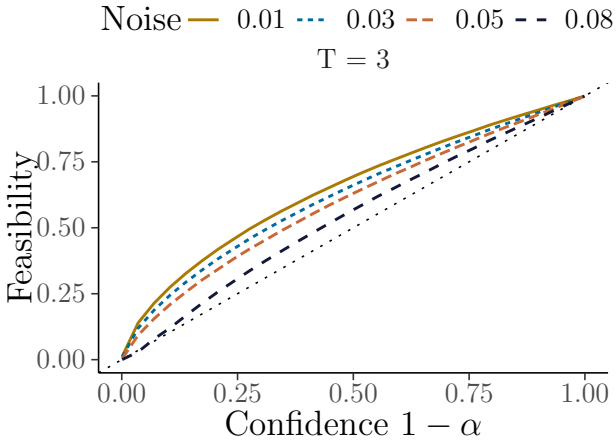


Figure 4.5: Fraction of feasible solutions as a function of confidence $1 - \alpha$ for the planning problem with three time periods. $1 - \alpha = 0$ corresponds to the nominal case and $1 - \alpha = 1$ to 0% chance of constraint violation. The noise in the data is uniform Gaussian with $\sigma_{\text{noise}} = 0.01, 0.03, 0.05$ and 0.08 and a standard GP model was used. The smaller the noise, the closer the actual feasibility is to the expected confidence (dotted line).

confidence values $1-\alpha$. The GP was trained based on 20 randomly generated data points using both uniform and non-uniform Gaussian noise with $\sigma_{\text{noise}} = 0.01, 0.03, 0.05$, and 0.08.

Standard GP: Fig. 4.5 shows results for the chance constrained approach using a standard GP model. We plot the fraction of feasible scenarios out of 1 million random samples from the true underlying distribution. Fig. 4.5 shows results for four different noise scenarios. By varying the confidence $1 - \alpha$, we adjust the robustness of the obtained solution. Clearly, the resulting feasibility does not exactly match the expected feasibility (shown as a dotted line) determined by the confidence level $1-\alpha$. This is due to a mismatch between the true underlying

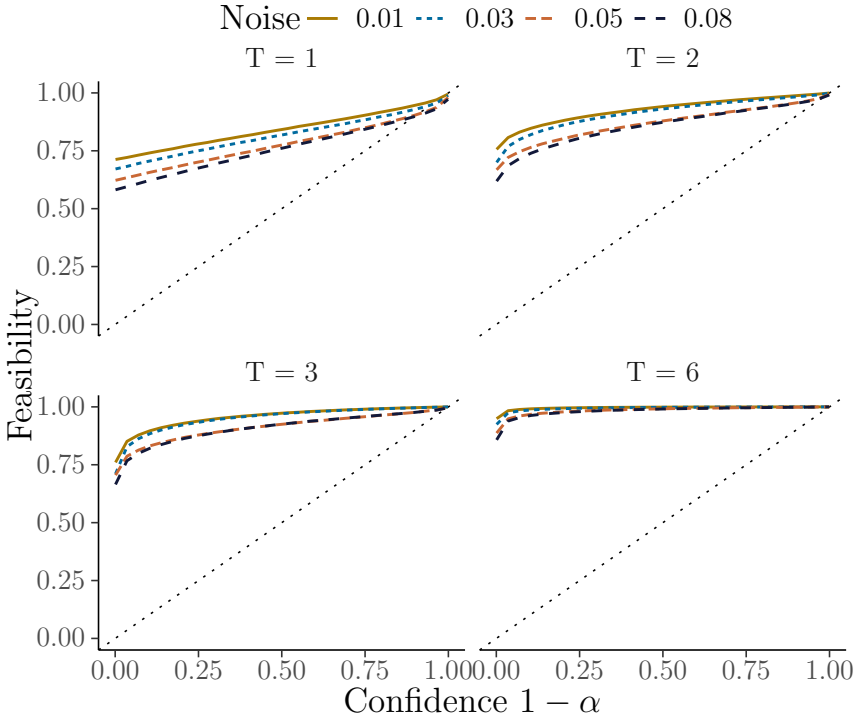


Figure 4.6: Fraction of feasible solutions as a function of confidence $1 - \alpha$ for non-uniform Gaussian noise with $\sigma_{\text{noise}} = 0.01, 0.03, 0.05$, and 0.08 for the production planning case study. Results are shown for four different numbers of time periods $T = 1, 2, 3$ and 6. The dotted line shows the a priori bound. With increasing numbers of time periods, the robust approximation becomes increasingly conservative.

distribution and the normal distribution estimated by the GP. As the amount of noise increases, the GP estimate deteriorates and the mismatch between feasibility and confidence increases.

Warped GP: Fig. 4.6 shows solution feasibility as a function of confidence $1 - \alpha$ for non-uniform noise using a warped GP model and the proposed robust approach. We show results for four different numbers of time periods. In the nominal case ($1 - \alpha = 0$), the feasibility is always close to 50% because a solution which is valid for the mean price-supply curve will also be valid for many scenarios with higher prices. In the robust case, as expected, feasibility increases as the size of the uncertainty set, i.e. $1 - \alpha$, increases. Notice that the robust approach is almost always a conservative approximation to the chance constraint, as the achieved feasibility is generally larger than the confidence $1 - \alpha$. Small violations of the a priori bound (dotted line) can still occur due to a mismatch between the GP model and the true underlying data generating distribution. The solution conservatism also varies with the number of time periods

considered. The a priori bound is less tight for larger values of T.

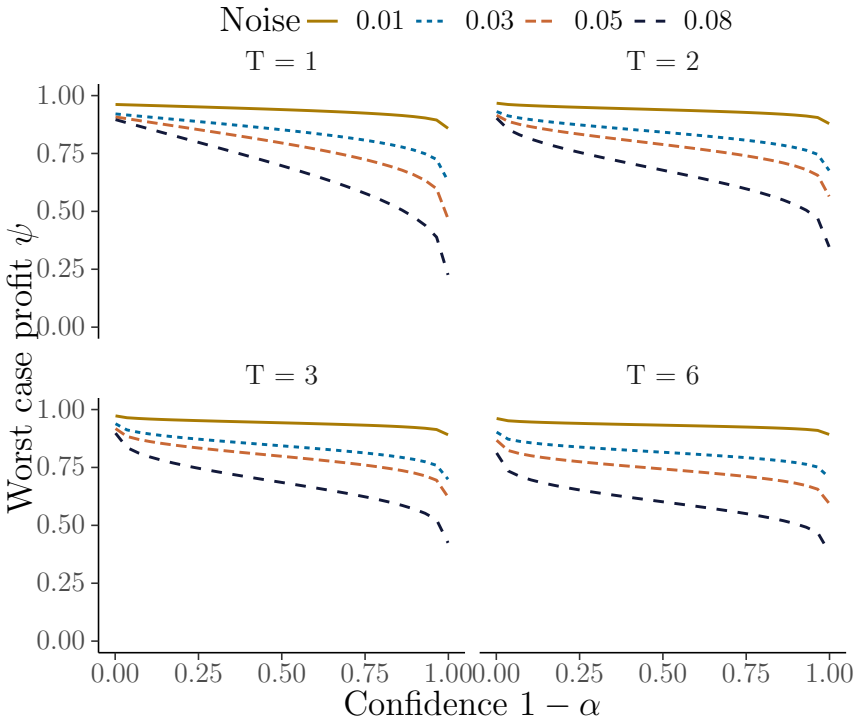


Figure 4.7: Profit, normalized with respect to nominal profit with $\sigma_{\text{noise}} = 0.01$ (objective of Problem (4.4)), as a function of confidence $1-\alpha$ for four different noise scenarios and time periods $T = 1, 2, 3$ and 6. As expected, the objective value decreases with increasing confidence $1-\alpha$, because more extreme worst case scenarios are considered.

Fig. 4.7 shows the worst case profit, normalized with respect to the nominal profit for $\sigma_{\text{noise}} = 0.01$, achieved as a function of the confidence level $1-\alpha$. As expected, increasing the confidence $1-\alpha$ leads to a lower worst case profit, because a larger confidence hedges against more uncertain price outcomes. Note that results are shown for values of $1-\alpha$ between 0.001 and 0.999. At $1-\alpha = 1$, the profit is always zero, because the uncertainty set includes negative prices and the optimal solution is to not produce anything. For a fixed confidence level, noisier data will generally lead to a smaller objective value as there is more uncertainty to hedge against.

Iterative procedure: Fig. 4.8 shows solution feasibility for the iterative a posteriori procedure (Alg. 4). We use confidence values between 0.55 and 0.999, since smaller confidences can often not be achieved using the iterative approach (the smallest achievable confidence is the feasibility of the nominal solution, i.e., $\sim 50\%$). The a posteriori approach is clearly less conservative than the a priori approach, however, this comes at the cost of additional compu-

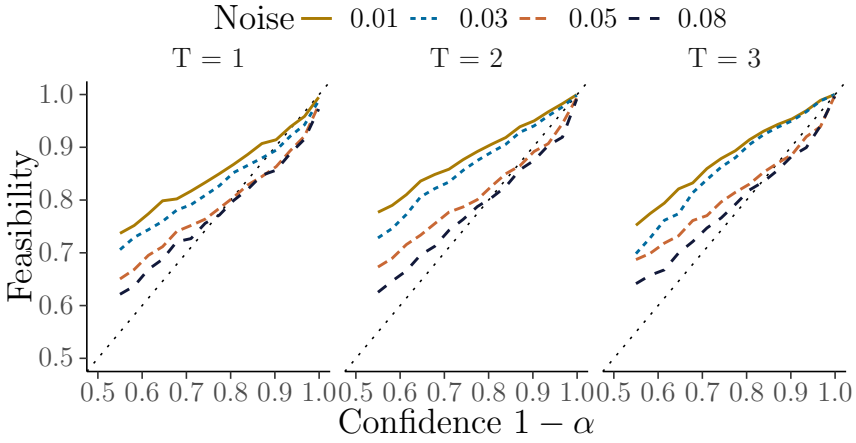


Figure 4.8: Fraction of feasible solutions vs confidence $1 - \alpha$ for the iterative a posteriori procedure (Alg. 4). If the noise is small, feasibility generally tracks the expected confidence (dotted line) well. For larger noise, deviations can occur due to mismatch between the warped GP model and the true data generating distribution.

tational expense and also potential bound violations when the warped GP does not model the underlying distribution perfectly. The a posteriori approach could therefore be a viable less conservative alternative in relatively low noise scenarios or when more training data is available.

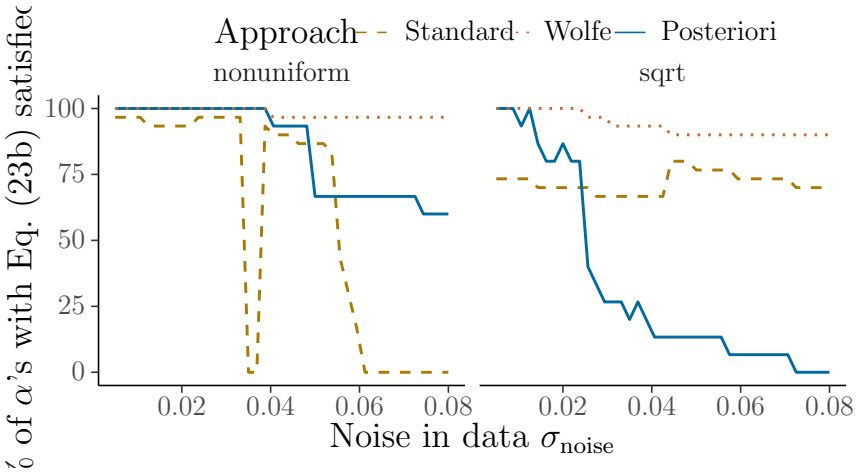


Figure 4.9: Percentage of α's for which the chance constraint is violated for non-uniform Gaussian data and Gaussian data warped through a square root function. Results are shown for the standard GP chance constraint approach (Standard), the warped GP robust approximation (Wolfe), and the a posteriori approach (Posteriori). For an ideal approximation of the chance-constraint Eq. 4.22b, the percentage of α's would always be 100%. All results are for the production planning case study with $T = 3$.

Comparison: Fig. 4.9 and 4.10 compare the performance of the chance constraint reformulation using the standard GP, the Wolfe duality based robust approximation of the warped

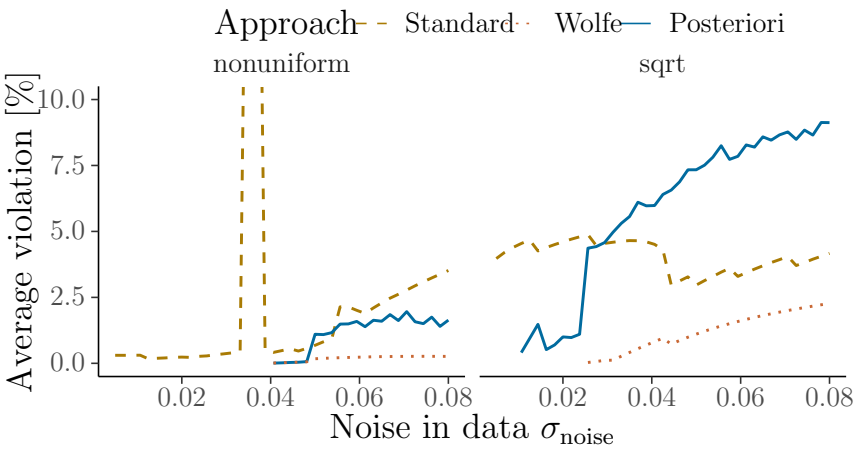


Figure 4.10: Violation of the chance constraint (when it is violated), averaged over different values of α, for non-uniform Gaussian data and Gaussian data warped through a square root function. Results are shown for the standard GP chance constraint approach (Standard), the warped GP robust approximation (Wolfe), and the a posteriori approach (Posteriori). All results are for the production planning case study with $T = 3$. Some lines do not start at $\sigma_{\text{noise}} = 0$ because for small noise values all instances are feasible and the average violation doesn't exist.

GP, and the a posteriori approach. In addition to the non-uniform Gaussian distribution, we also generate training data by sampling from a Gaussian distribution and warping the samples through a square root function. Fig. 4.9 and 4.10 show results for $T = 3$ only, but the results for other uncertainty set dimensions follow the same patterns.

Fig. 4.9 shows the percentage of 30 solutions with different values of α for which the chance constraint is feasible. To determine if the chance constraint is feasible for a given solution we draw 10000 samples from the true data-generating distribution, calculate the percentage of samples for which the constraint is satisfied, and compare it to the intended feasibility of the chance constraint $1 - \alpha$. I.e., Fig. 4.9 shows the percentage of instances for which the probability of constraint violation is actually smaller than α. Ideally, this percentage should always be 100%. The standard GP chance constraint reformulation comes fairly close to 100% for the non-uniform Gaussian data with small noise, but does considerably worse for the square root warped data and for larger noise in general. The robust approximation leads to the highest percentage of feasible solutions for both data generating distributions. Solutions also remain largely feasible even as the noise in the data becomes very large. In safety-critical applications with non-Gaussian noise, the proposed approach is therefore clearly beneficial. The proposed

a posteriori strategy achieves high rates of feasibility when the noise in the data is small, but deteriorates quickly as the noise increases. It may therefore be beneficial in applications with relatively low noise and where the robust approximation leads to overly conservative solutions.

Fig 4.10 shows a similar consideration. Here, the average violation of the chance constraint in percentage points, i.e., the average absolute difference between the actual feasibility of the solution and the intended feasibility $1 - \alpha$ is shown. Only instances for which the chance constraint is indeed violated are included in the average. The graph shows that, even when the chance constraint is violated, the robust approximation leads to smaller violations than the standard GP chance constraints approach.

4.4.2 Drill scheduling

For the drill scheduling case study, we consider two different geologies with 2 and 6 geological segments. We consider a range of target depths and confidence values. Fig. 4.11 shows the drilling, maintenance, and total cost for a target depth of 2200m as a function of the confidence parameter $1 - \alpha$. In the deterministic case ($1 - \alpha = 0.5$), the optimal strategy is to not do maintenance at all and drill as fast as possible. As we increase $1 - \alpha$ to obtain more robust solutions, we eventually reach a point where the average rate of penetration is slightly lower in order to reduce degradation and guarantee that the well can be completed without a motor failure. For the 2-segment geology the increased cost of drilling outweighs the zero maintenance cost at around $1 - \alpha = 0.92$. After this point the optimal strategy is to do maintenance once.

Results are shown for both the no–degradation and boundary heuristics as well as total enumeration. For this instance, the boundary heuristic leads to the same solution as the globally optimal enumeration strategy. The no–degradation heuristic, on the other hand, leads to sub-optimal solutions when the optimal maintenance number is lower than the upper bound $\lfloor R_n \rfloor$.

Fig. 4.12 shows the same cost components as Fig. 4.11 as a function of the target depth x_{n_d}. Results are shown for three different values of $1 - \alpha$ (0.5, 0.75, and 0.99). A larger confidence

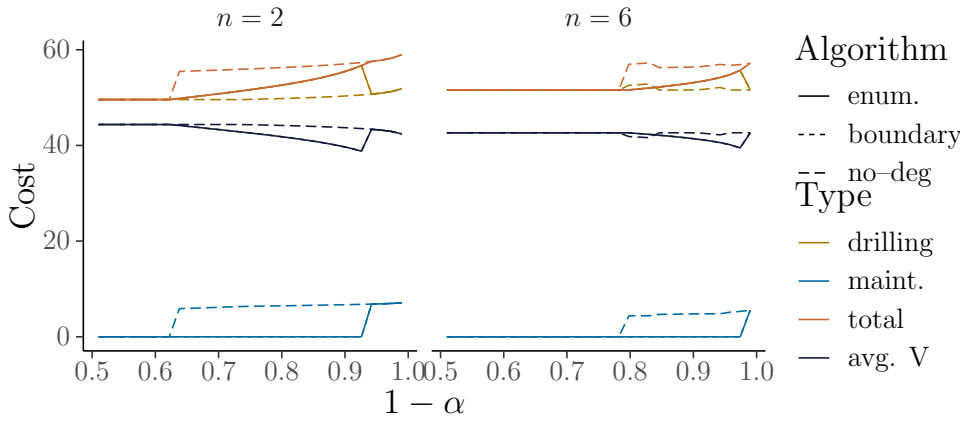


Figure 4.11: Cost of drilling to a depth of 2200 meters through a geologies with 2 and 6 segments for different values of confidence parameter α. Results are shown for three different integer strategies. The boundary heuristic gives the same results as total enumeration, while the no–degradation heuristic gives suboptimal solutions.

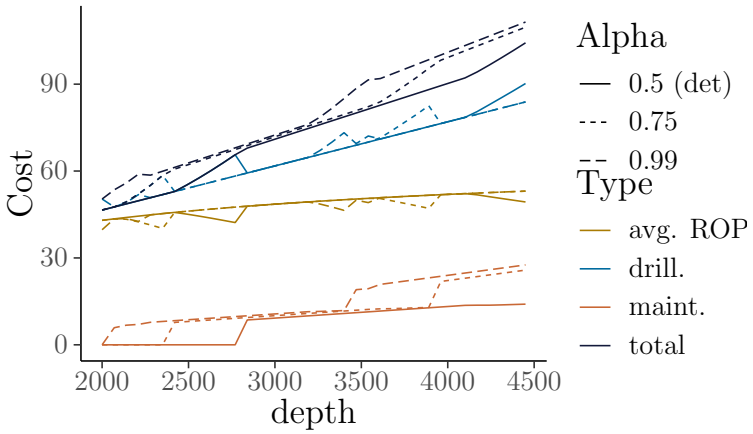


Figure 4.12: Cost of drilling as a function of target depth for three different for a geology with two rock types for and for three different values of confidence parameter α. All results are obtained with the globally optimal enumeration strategy.

always leads to a higher cost, as would be expected, but the difference between the deterministic solution and a 99%–confidence robust solution can be large or small, depending on the target depth, e.g., for a target depth of $x_{n_d} = 3000$m hedging against uncertainty does not lead to significant cost increases.

Finally, Fig. 4.13 shows the total solution time for the three integer strategies for the instance with 6 geological segments as a function of confidence parameter $1 - \alpha$. While the no–degradation heuristic often leads to suboptimal solutions, as seen above, it is computationally very cheap. The boundary heuristic comprises a good compromise: it frequently finds

the global optimum while being much cheaper computationally. Especially for instances with many geological segments and maintenance events, where the combinatorial complexity of the enumeration strategy becomes prohibitive, it may therefore be a good alternative.

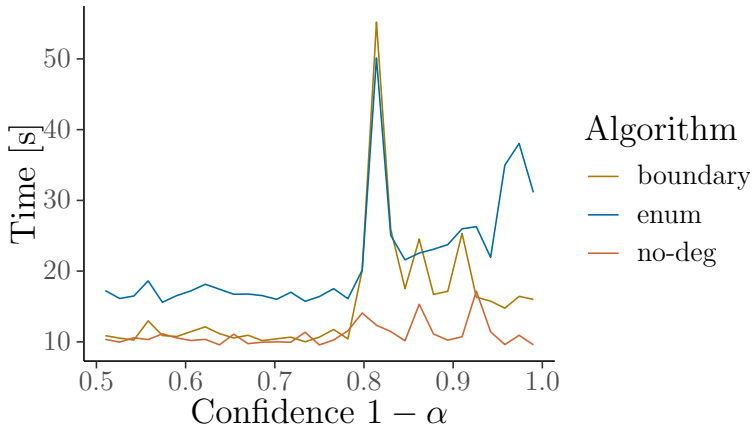


Figure 4.13: Total time to solve an instance with 6 rock types as a function of the confidence parameter α. While the enumeration strategy is the only approach which is guaranteed to find the globally optimal solution, the boundary heuristic often finds the same solution in significantly less time.

4.5 Conclusion

Our approach reformulates uncertain black-box constraints, modeled by warped Gaussian processes, into deterministic constraints guaranteed to hold with a given confidence. We achieve this deterministic reformulation of chance constraints by constructing confidence ellipsoids and utilizing Wolfe duality. We show that this approach allows the solution conservatism to be controlled by a sensible confidence probability choice. This could be especially useful in safety-critical settings where constraint violations should be avoided.

Chapter 5

Modeling and solving robust problems in Pyomo

5.1 Introduction

Robust optimization is a common way of managing optimization under uncertainty in process systems engineering: applications range from production scheduling to flexible chemical process design [25, 44, 87, 160, 161, 162]. New developments in robust optimization include distributionally and adjustable robust optimization to reduce solution conservatism [162], data-driven robust optimization to model uncertainty sets based on available data [34], and approximate robust optimization to make non-linear problems more tractable [19]. This large number of alternative approaches and the required domain knowledge can discourage practitioners from making the transition from deterministic optimization to optimization under uncertainty. Furthermore, the lack of a platform that allows the easy implementation and application of new algorithms means that it is difficult for researchers to compare different approaches [163].

This chapter introduces ROmodel, a Python package extending the popular, Python-based algebraic modeling language Pyomo [102, 164] to facilitate modeling of robust optimization problems and implementation of robust optimization algorithms. ROmodel aims to (i) make robust optimization more accessible to practitioners and facilitate moving from deterministic

to robust optimization, (ii) demonstrate how robust optimization problems can be modeled in close analogy to their mathematical formulation, and (iii) provide an open-source platform for researchers to implement and compare new robust reformulations and uncertainty sets. To this end, ROmodel introduces intuitive modeling which closely resembles the optimization problem's underlying mathematical formulation and will feel familiar to Pyomo users. ROmodel uses the richness of Pyomo's solver interfaces and methods for model transformations and Python's data processing capabilities. It supports both automatic reformulation and cutting plane algorithms. Uncertainty sets can be chosen from a library of common geometries, or custom defined using Pyomo constraints. ROmodel also support adjustable robust optimization through linear decision rules. Because ROmodel and Pyomo are open-source, ROmodel can be extended to incorporate additional uncertainty set geometries and reformulations. As an example, we implemented uncertainty sets based on (warped) Gaussian processes for black-box constrained problems.

There are other software packages for *solving* robust optimization problems, e.g. ROME [165], RSOME [166], ROC++ [167], and PyROS [168]. All four approaches are designed as robust optimization *solvers* and, except for PyROS which is based on Pyomo, do not rely on a general purpose algebraic modeling language. In contrast, ROmodel focuses on *modeling* robust optimization problems. By building on Pyomo, ROmodel simplifies the transition from deterministic to robust optimization, since both the deterministic and robust model can be implemented in the same environment. ROmodel also allows access to a much larger number of deterministic solvers than these other solvers. In contrast to AIMMS, which has some capabilities for modeling robust optimization problems [169], ROModel is open source and allows possibilities for extension. JumPeR [170], which extends Julia's modeling language JumP to robust optimization problems, is the most similar to ROModel. In contrast to JumPeR, ROModel is more tightly tied to its respective algebraic modeling language, e.g. developing new convex uncertainty sets in ROmodel only requires adding Pyomo constraints while JumPeR would require adding Julia functionality. ROmodel also automatically recognizes uncertainty set geometries and applies the corresponding transformations without the user having to specify which geometry they are using. Table 5.1 summarizes the key features of ROmodel as compared

to other tools.

	ROmodel	JumPeR	PyROS	ROC++	RSOME
Modeling language	Pyomo	JumP	Pyomo	C++	MATLAB
Extends modeling language	yes	yes	no	no	no
Solvers	reformulation, cutting plane, nominal	reformulation, cutting plane	cutting plane	reformulation	reformulation
Adjustable RO	yes	yes	yes	yes	yes
Data-driven sets	GP-based sets	no	no	no	ambiguity sets
Extensible	uncertainty sets, reformulations	uncertainty sets, reformulations	uncertainty sets	no	no

Table 5.1: Comparison of ROmodel's key features with other tools.

A further advantage of ROmodel is that it is based on Python, which is popular in data analytics and machine learning. ROmodel therefore allows data-based techniques to be integrated seamlessly with robust optimization methods. As an example, we implement (warped) Gaussian process-based uncertainty sets in ROmodel [171]. These sets seamlessly integrate Gaussian processes trained in the Python library GPy [157] into robust optimization problems. ROmodel is open source and available on Github [172].

The rest of this chapter is structured as follows. Section 5.2 introduces ROmodel's new modeling objects and shows how they can be used to model robust optimization problems. Section 5.3 introduces ROmodel's three solvers: a reformulation based solver, a cutting plane solver, and a nominal solver for obtaining nominal solutions of robust problems. Section 5.4 discusses how ROmodel can be extended and demonstrates our implementation of Gaussian process-based uncertainty sets for black-box constrained problems. Section 5.5 introduces six case studies and presents results.

5.2 Modeling

Consider the following generic robust optimization problem:

$$\min_{\boldsymbol{x}\in\mathcal{X},\boldsymbol{z}(\boldsymbol{\xi})} \max_{\boldsymbol{\xi}\in\mathcal{U}(\boldsymbol{x})} \quad f(\boldsymbol{x},\boldsymbol{z}(\boldsymbol{\xi}),\boldsymbol{\xi}) \tag{5.1a}$$

$$\text{s.t} \quad g(\boldsymbol{x},\boldsymbol{z}(\boldsymbol{\xi}),\boldsymbol{\xi}) \leq 0 \qquad \forall \boldsymbol{\xi} \in \mathcal{U}(\boldsymbol{x}) \tag{5.1b}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ are "here and now" variables, determined before the uncertainty is revealed, while $\boldsymbol{z}(\boldsymbol{\xi})$ are adjustable variables, determined after the uncertainty is revealed. The uncertain parameter vector $\boldsymbol{\xi}$ is bounded by the uncertainty set $\mathcal{U}(\boldsymbol{x})$, which may depend on $\boldsymbol{x}$ and which contains the nominal values $\bar{\boldsymbol{\xi}}$. Optimization Problem 5.1, a generic robust optimization problem with uncertainty in the objective and constraints, is the basis for ROmodel. Note that we are limiting ourselves here to one uncertain constraint for simplicity of notation only, ROmodel can handle multiple robust constraints. Also note that there can be an arbitrary number of deterministic constraints which define the set $\mathcal{X}$.

ROmodel introduces three new modeling objects to represent robust optimization problems like Problem 5.1 within Pyomo:

1. `UncParam`: A class similar to Pyomo's `Param` and `Var` class used to model uncertain parameters $\boldsymbol{\xi}$. One can supply a `nominal` argument, which defines the vector of nominal values $\bar{\boldsymbol{\xi}}$ used to replace the uncertain parameters when solving the deterministic (nominal) problem.

2. `UncSet`: Each `UncParam` object has an `UncSet` object associated with it. The `UncSet` class, based on Pyomo's `Block` class, models the uncertainty sets $\mathcal{U}$. Uncertainty sets $\mathcal{U}$ can be defined by (i) constructing generic sets by adding Pyomo constraints to the `UncSet` object, or (ii) through a library of common uncertainty set geometries, using their matrix representation as an input.

3. `AdjustableVar`: A class similar to Pyomo's `Var` class used to model adjustable variables

which can be determined after some of the uncertainty has been resolved, i.e. $\boldsymbol{z}(\boldsymbol{\xi})$. ROmodel currently only implements linear decision rules for adjustable variables.

These three new modeling objects are sufficient for modeling quite generic robust optimization problems. They are set up to allow modeling problems in an intuitive way which is closely related to their mathematical formulation (Problem 5.1). We discuss each modeling object and how it is used to construct robust optimization problems in the subsequent sections.

5.2.1 Uncertain parameters

Indexed uncertain parameters are constructed in analogy to Pyomo's `Var` type for variables:

```
m.c = UncParam(range(3), nominal=[0.1, 0.2, 0.3], uncset=m.U)
```

The `nominal` argument specifies a list of nominal values $\bar{\boldsymbol{\xi}}$ for the uncertain paramters $\boldsymbol{\xi}$. The `uncset` argument specifies the uncertainty set to use for these parameters. The two approaches for constructing the uncertainty set `m.U` are discussed in the next two sections.

5.2.2 Generic uncertainty sets

Generic uncertainty sets are constructed with the `UncSet` class. This class inherits from Pyomo's `Block` class. Users can construct generic uncertainty sets by adding Pyomo constraints to an `UncSet` object in the same way as they would add constraints to a `Block` object in Pyomo. The following example shows how a polyhedral set can be modeled using the `UncSet` class:

```
from romodel import UncSet, UncParam
# Define uncertainty set
m.U = UncSet()
# Define uncertain parameter
m.w = UncParam(range(2), uncset=m.U, nominal=[0.5, 0.5])
# Add constraints to uncertainty set
m.U.cons1 = Constraint(expr=m.w[0] + m.w[1] <= 1)
m.U.cons2 = Constraint(expr=m.w[0] - m.w[1] <= 1)
```

```
...
```

The `uncset` and `nominal` arguments define the uncertainty set and a vector of nominal values for the uncertain parameter `m.w`. ROmodel's strategy of modeling uncertainty sets using Pyomo's `Constraint` modeling object is analogous to typical robust optimization formulations. ROmodel can therefore be used to define any uncertainty set which can be expressed in Pyomo. However, not every set that can be modeled with Pyomo constraints can necessarily also be solved. Section 5.3 discusses which types of uncertainty sets can be solved using which solver. Users can also define multiple uncertainty sets and replace one by another. The uncertainty set for any uncertain parameter `m.w` can be set using the `uncset` property:

```
# Define second uncertainty set
m.U2 = ro.UncSet()
# Swap uncertainty sets and solve
m.w.uncset = m.U2
solver = pe.SolverFactory('romodel.cuts')
solver.solve(m)
```

5.2.3 Library uncertainty sets

For commonly used, standard uncertainty sets, the generic approach (Section 5.2.2) is unnecessarily complicated. Therefore, ROmodel implements custom classes which can define an uncertainty set using its matrix representation. ROmodel implements polyhedral and ellipsoidal sets. The user can define polyhedral sets of the form $\mathcal{U} = \{w \mid Pw \leq b\}$ by passing the matrix P and the right hand side b to the `PolyhedralSet` class:

```
from romodel.uncset import PolyhedralSet
# Define polyhedral set
m.U = PolyhedralSet(mat=[[ 1,  1],
                         [ 1, -1],
                         [-1,  1],
                         [-1, -1]],
                    rhs=[1, 1, 1, 1])
```

ROmodel creates ellipsoidal sets of the form $(w-\mu)\Sigma^{-1}(w-\mu) \leq 1$ using the `EllipsoidalSet` class, the covariance matrix Σ and the mean vector μ:

```
from romodel.uncset import EllipsoidalSet
# Define ellipsoidal set
m.U = EllipsoidalSet(cov=[[1, 0, 0],
                          [0, 1, 0],
                          [0, 0, 1]],
                     mean=[0.5, 0.3, 0.1])
```

Many other sets commonly used in robust optimization, e.g. box sets or norm sets can either be modeled using the polyhedral library set, or using the generic approach for constructing general convex sets. ROmodel's uncertainty set library can also be extended. In Section 5.4 we discuss how additional library sets can be added to ROmodel using data-driven uncertainty sets based on (warped) Gaussian processes as an example.

5.2.4 Constructing uncertain constraints

After defining uncertain parameters and an uncertainty set, the user can construct uncertain constraints implicitly by using the uncertain parameters in a Pyomo constraint. Consider the following deterministic Pyomo constraint:

```
# deterministic
m.x = Var(range(3))
c = [0.1, 0.2, 0.3]
m.cons = Constraint(expr=sum(c[i]*m.x[i] for i in m.x) <= 0)
```

If the coefficients c are uncertain, we can model the robust constraint $c^T x \leq 0,\ \forall c \in \mathcal{U}$ as:

```
# robust
m.x = Var(range(3))
m.c = UncParam(range(3), nominal=[0.1, 0.2, 0.3], uncset=m.U)
m.cons = Constraint(expr=sum(m.c[i]*m.x[i] for i in m.x) <= 0)
```

The uncertain parameter `m.c` can be used in Pyomo constraints in the same way as Pyomo `Var` or

`Param` objects. ROmodel's solvers automatically recognize constraints and objectives containing containing uncertain parameters.

5.2.5 Adjustable variables

ROmodel also has capabilities for modeling adjustable variables. Adjustable variables $\boldsymbol{z}(\boldsymbol{\xi})$ are variables which can be determined after some (or all) of the uncertainty has been revealed. Defining an adjustable variable is analogous to defining a regular variable in Pyomo, with an additional `uncparam` argument specifying a list of uncertain parameters which the adjustable variable depends on:

```
# Define uncertain parameters
m.w = UncParam(range(3), nominal=[1, 2, 3])
# Define adjustable variable which depends on uncertain parameter
m.y = AdjustableVar(range(3), uncparams=[m.w], bounds=(0, 1))
```

The uncertain parameters can also be set individually for each element of the adjustable variables index using the `set_uncparams` function:

```
# Set uncertain parameters for individual indices
m.y[0].set_uncparams([m.w[0]])
m.y[1].set_uncparams([m.w[0], m.w[1]])
```

ROmodel only implements linear decision rules for solving adjustable robust optimization problems. If a model contains adjustable variables in a constraint or objective, ROmodel automatically replaces it by a linear decision rule based on the specified uncertain parameters.

5.3 Solvers

ROmodel has three solvers: A robust reformulation based solver, a cutting plane based solver, and a nominal solver.

5.3.1 Reformulation

The reformulation-based solver, illustrated in Fig. 5.1, implements standard duality based techniques for reformulating robust optimization problems into deterministic counterparts [8]. First,

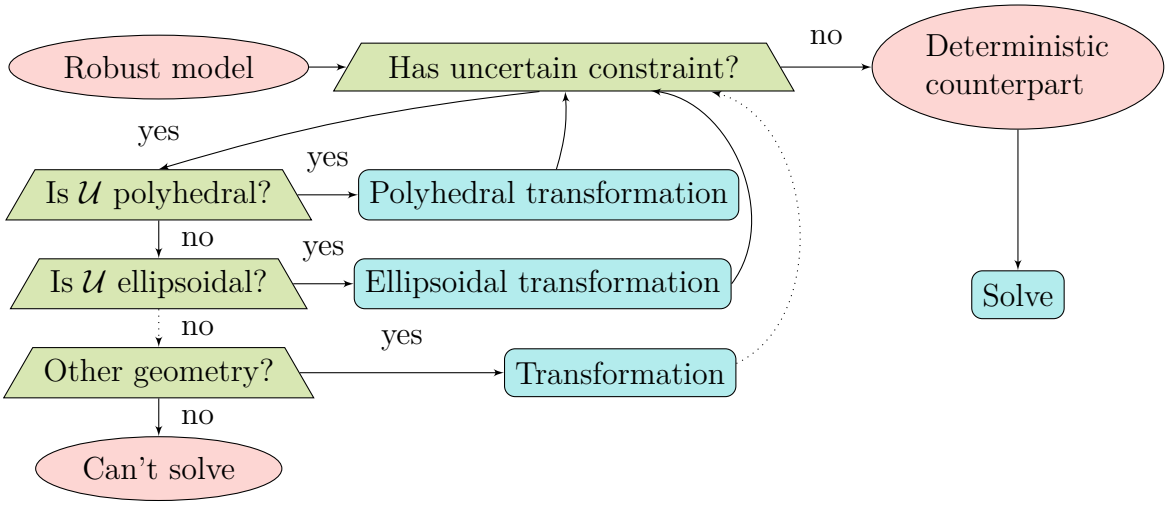


Figure 5.1: Schematic of reformulation solver. For each uncertain constraint, the reformulation solver goes through all known transformations and identifies the geometry of the constraint/uncertainty set. It then applies the corresponding model transformation. If all uncertain constraints can be reformulated, the resulting deterministic counterpart is solved using one of Pyomo's solver interfaces. If one or more constraints cannot be reformulated, the problem cannot be solved.

it detects every constraint containing uncertain parameters. Second, it checks the structure of each uncertain constraint and the corresponding uncertainty set to determine if a known reformulation is applicable. To do this, it goes through the available reformulations in a pre-defined order to determine whether they are applicable or not. If multiple reformulations are applicable, the first one on the list is applied. Finally, it applies a model transformation, generating the deterministic counterpart of each robust constraint. The deterministic counterpart is then solved using an appropriate solver available in Pyomo. The structure of the optimization problem and the uncertainty set geometry determine which solvers are applicable. If no applicable transformation can be identified for one or more constraints, the problem cannot be solved and ROmodel will raise an error.

ROmodel implements standard reformulations for ellipsoidal and polyhedral uncertainty sets and linear uncertain constraints [7, 8]. It also implements reformulations for black-box con-

strained problems [171]. These are discussed in more detail in Section 5.4, which also discusses how ROmodel can be extended to include further reformulations.

5.3.2 Cutting planes

The cutting plane solver, outlined in Fig. 5.2, implements an iterative strategy for solving robust optimization problems [173]. It replaces each uncertain constraint and objective by a `CutGenerator` object which initially just contains the nominal constraint. The solver then iteratively solves the master problem and generates cuts to cut off solutions which are not robustly feasible. A solution x^* is considered to be robustly feasible when for each uncertain

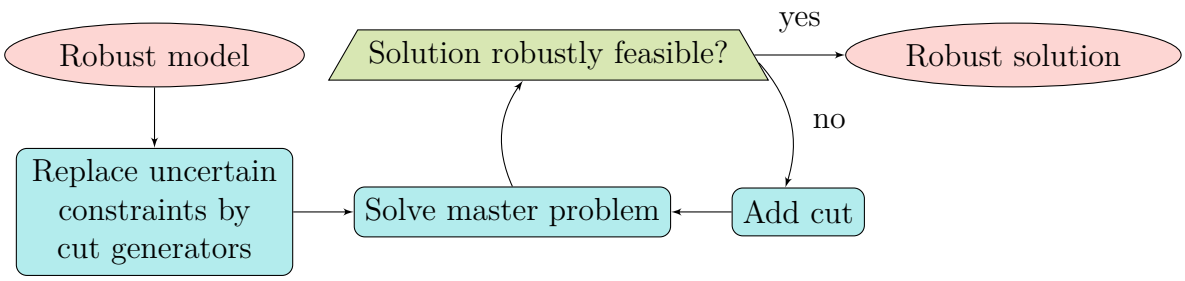


Figure 5.2: Schematic of cutting plane solver. The cutting plane solver iteratively solves the master problem and adds cutting planes until a robustly feasible solution is found.

constraint $g(\boldsymbol{x}^*, \boldsymbol{z}(\boldsymbol{\xi}), \boldsymbol{\xi}) \leq 0$, the objective value of the separation problem is smaller than some tolerance ϵ:

$$\max_{\boldsymbol{\xi} \in \mathcal{U}} g(\boldsymbol{x}^*, \boldsymbol{z}(\boldsymbol{\xi}), \boldsymbol{\xi}) \leq \epsilon \tag{5.2}$$

ROmodel's cutting plane solver can generally be applied to any convex uncertainty set. The Pyomo solvers for solving the master and separation problems can be set individually using:

```
solver = SolverFactory('romodel.cuts')
# Master solver
solver.options['solver'] = 'gurobi'
# Sub solver
solver.options['subsolver'] = 'ipopt'
```

The solvers need to be appropriate for the corresponding problem, i.e., the above choice would be sensible if the master problem is a mixed-integer linear problem and the uncertain constraints and uncertainty set are continuous and convex. If the solvers are not appropriate, Pyomo will raise an error.

5.3.3 Nominal

ROmodel also includes a nominal solver. This solver replaces all occurrences of the uncertain parameters by their nominal values and solves the resulting deterministic problem:

```
# Obtaining the nominal solution
solver = SolverFactory('romodel.nominal')
solver.solve(m)
```

The nominal solver allows users to combine their implementations of the nominal and robust problem. An implementation of the robust model can be used to obtain the solution of the nominal problem.

5.4 Extending ROmodel for black-box constrained problems

ROmodel can be extended to incorporate additional reformulations and uncertainty set geometries. This section outlines how ROmodel can be extended using the (warped) Gaussian process-based uncertainty sets for black-box constrained problems, discussed in detail in Chapter 4, as an example. This example showcases the ease with which ROmodel can integrate Python's machine learning and data analytics capabilities with Pyomo's mathematical optimization modeling.

Recall that the Gaussian process-based sets are applicable to constraints containing a black-box

function $g(\cdot)$ of the following form:

$$P\left(\sum_i g(\boldsymbol{y}_i)x_i \leq b\right) \geq 1-\alpha \tag{5.3}$$

In order to make these approaches available in ROmodel, we need to (i) implement two library uncertainty sets, `GPSet` and `WarpedGPSet`, which collect the relevant data, and (ii) implement two corresponding model transformations which perform the reformulations for standard and warped Gaussian process-based sets. The implementation is based on the Python module ROGP [158], which is includes Gaussian process models trained in the Python library GPY [157] in Pyomo models.

5.4.1 Implementing new library sets

Implementing a new library set mainly requires a new Python class collecting the necessary data. For the standard and warped Gaussian process set, this data consist of three arguments:

```
from romodel.uncset import GPSet, WarpedGPSet
# Define variables
m.y = Var([0, 1])
# Define uncertainty set based on standard GP
m.uncset_standard = GPSet(gp_standard, m.z, 0.95)
# Define uncertainty set based on warped GP
m.uncset_warped = WarpedGPSet(gp_warped, m.z, 0.95)
```

The first argument `gp_standard/warped` is a (warped) Gaussian process object trained in GPy. The second is an indexed Pyomo variable on which the GP depends, i.e. $\boldsymbol{z}$ in Eq. 5.3. The third parameter specifies the confidence level $1-\alpha$ with which the true parameter is contained in the uncertainty set. I.e., in this case the confidence that the true parameter vector is an element of the uncertainty set is at least 95%.

The new sets can be used in the same way as other library sets, e.g.:

```
# Define variables
m.x = Var([0, 1])
```

```
    # Define uncertain parameters
    m.w = UncParam([0, 1], uncset=m.uncset_warped)
    # Define constraint
    m.cons = Constraint(expr=m.x[0]*m.w[0] + m.x[1]*m.w[1] <= 1)
```

Constraints which use this type of uncertainty set need to be linear in the uncertain parameter. Note that ROmodel requires the indices of `m.y` and `m.w` to be identical in the formulation above. ROmodel also allows specifying the relationship between `m.w` and `m.y` through a dictionary. This is for example useful, if the black-box function depends on more than one variable:

```
    # Define variables
    m.y = Var([0, 1], ['a', 'b'])
    # Define uncertainty set
    y_dict = {0: [m.y[0, 'a'], m.y[0, 'b']],
              1: [m.y[1, 'a'], m.y[1, 'b']]},
    m.uncset_warped = WarpedGPSet(gp, y_dict, 0.95)
```

The dictionary indicates that the uncertain parameter `m.w[0]` depends on the variables `m.y[0, 'a']` and `m.y[0, 'b']` through the black-box function $g(\cdot)$, modeled in GPy by the Gaussian process `gp`.

Note that ROmodel's cutting plane solver is not applicable to the Gaussian process-based sets because the sets are decision dependent. Attempting to solve a problem with one of theses sets therefore results in an error. When implementing new library sets which *can* be solved using cutting planes, an additional Python function `generate_cons_from_lib`, which generates Pyomo constraints for the uncertainty set based on the data collected by the library set, is required. For an example, see `romodel/uncset/ellipsoidal.py` on the ROmodel Github [172].

5.4.2 Implementing new reformulations

For ROmodel to be able to solve models containing the two new Gaussian process-based sets, we need to implement the corresponding reformulations. Adding new reformulations to ROmodel generally requires two Python functions: (i) a function `_check_applicability` which de-

tects whether a constraint and uncertainty set have the required structure, and (ii) a function `_reformulate` which generates the robust counterpart. The former function is only required if the reformulation is supposed to work with generically constructed uncertainty sets as described in Section 5.2.2. For library sets like the Gaussian process-based sets, only the latter function is required. This function takes data describing the constraints and uncertainty set as an input and returns a Pyomo block containing the deterministic counterpart. For a full example see the implemented reformulations in `romodel/reformulate/` on the ROmodel Github [172].

5.5 Results

We use ROmodel to model and solve six case studies:

1. A portfolio optimization problem with uncertain returns [8],
2. A knapsack problem with uncertain item weights,
3. A pooling problem instance [174] with uncertain product demands,
4. A capacitated facility location problem as an example for adjustable robust optimization, where the decision which facilities to build has to be made under demand uncertainty, while the decision from which facility to supply individual customers can be made once the uncertainty is resolved,
5. A production planning problem where the price at which products can be sold depends on the amount produced through an uncertain black-box function modelled by a (warped) Gaussian process [171],
6. And a drill scheduling problem in which the equipment used to drill a well degrades at a rate which depends on other drill parameters through a black-box function [171].

For the reformulation approach, we use Gurobi 9.0.3 to solve the reformulation for case studies 1–4 and Ipopt 3.12.12 for case studies 5 and 6. For the cutting plane approach, we use Gurobi

Problem	Type	# Variables	# Constraints	# Unc. params
Portfolio	LP	6	2	5
Knapsack	MILP	4	1	4
Pooling	NLP	13	45	4
Facility	adj. MILP	4	9	5
Planning	NLP	4	1	3
Drilling	NLP	63	39	12

Table 5.2: Type of problem, number of variables, constraints, and uncertain parameters for each case study.

9.0.3 as the main and subsolver for all problems. All experiments were run on an i7-6700 CPU with 8x3.40GHz and 16GB RAM. All examples except for the drill scheduling example are included with ROmodel and can be used as follows:

```
import romodel.example as ex
portfolio = ex.Portfolio()
knapsack = ex.Knapsack()
pooling = ex.Pooling()
facility = ex.Facility()
planning = ex.ProductionPlanning(alpha=0.95, warped=True)

```

The implementation of the drill scheduling example is separately available on Github [175]. Table 5.2 shows a summary of the type of problem and number of variables, constraints, and uncertain parameters for each case study.

We solve the portfolio, knapsack, pooling, and facility location problems with both the reformulation and cutting plane solver for ellipsoidal and polyhedral uncertainty sets and using both the library approach to generating uncertainty sets as well as the generic, Pyomo constraint-based approach. We solve the production planning and drill scheduling problems using the reformulation solver with uncertainty sets based on both standard and warped Gaussian processes. We solve 30 instances with different uncertainty set sizes for each case study.

Table 5.3 shows the median time in milliseconds taken to solve each problem for a given uncertainty set geometry and solver. The reformulation solver generally outperforms the cutting plane solver with median times of 74ms and 330ms respectively. An exception is the the non-

Problem	Unc. Set	Reformulation		Cuts		Overall	
Knapsack	Polyhedral	54	(3.7)	272	(1.4)	85	(1.9)
	Ellipsoidal	50	(2.2)	183	(1.4)	91	(1.8)
Pooling	Polyhedral	74	(7.7)	329	(2.9)	173	(7.4)
	Ellipsoidal	638	(11.6)	331	(2.9)	349	(6.7)
Portfolio	Polyhedral	50	(3.0)	276	(0.8)	126	(2.3)
	Ellipsoidal	49	(2.7)	1659	(0.8)	129	(1.2)
Facility	Polyhedral	261	(106.0)	13353	(37.1)	5588	(74.5)
	Ellipsoidal	–	(–)	31275	(71.4)	31275	(71.4)
Planning	Standard	2776	(871.7)		NA	2776	(871.7)
	Warped	8536	(997.9)		NA	8536	(997.9)
Drilling	Standard	13646	(272.8)		NA	13646	(272.8)
	Warped	75325	(577.8)		NA	75325	(577.8)
Overall		74	(7.5)	330	(2.8)	271	(3.2)

Table 5.3: Median time in milliseconds taken to solve the six example problems with different uncertainty set geometries. Times are shown for the reformulation and cutting plane solvers and both combined (Overall). The median time taken to *reformulate* the model is shown in parantheses. The cutting plane solver is not applicable (NA) for Gaussian process-based sets and the reformulation solver cannot solve the facility problem with ellipsoidal sets to optimality (–).The *Overall* columns show the median time over all instances irrespective of the solver used. The *Overall* row shows the median time over all instances of the different case studies.

linear, non-convex pooling problem with an ellipsoidal set. For this instance, the cutting plane solver achieves significantly better results, which is in line with previous work on robust pooling problems [29]. Similarly, for the facility location problem with an ellipsoidal set, the reformulation approach does not solve the problem to optimality within a 10 minute time frame, while the cutting plane solver does. For the production planning and drills scheduling examples only the reformulation solver can be applied. The Wolfe duality-based reformulation for warped Gaussian processes generally takes longer to solve than the chance-constraint reformulation for standard Gaussian processes. Note that most of this time is the time taken by the subsolvers. The transformations which ROmodel performs are generally very quick: the median transformation time across all instances is 3.2 milliseconds, while the maximum transformation time is 1.7 seconds. The median transformation time for each problem and geometry/solver is shown in parantheses in Table 5.3.

Fig. 5.3 shows the objective value (normalized with the objective of the nominal solver) as a function of uncertainty set size for each of the six examples. Note that the facility location and drill scheduling problems are minimization problems, while the rest are maximization

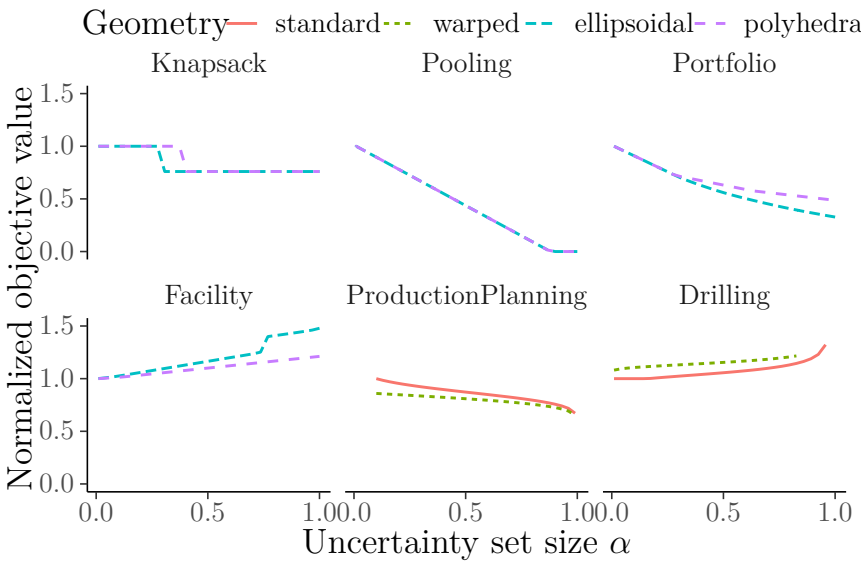


Figure 5.3: The figure shows the normalized objective value as a function of the uncertainty set size for knapsack, portfolio and pooling case studies and an ellipsoidal and polyhedral uncertainty set.

problems. By construction, the ellipsoidal sets tested always fully contain the polyhedral sets for a given α. Correspondingly, they are always more conservative, i.e., for a given α the objective value achieved using the ellipsoidal set is larger for minimization and smaller for maximization problems than the value achieved using the polyhedral set. For the Gaussian process-based sets, the standard approach is always less conservative than the warped approach. However, the limited ability of the standard Gaussian process to model non-Gaussian noise may mean that the actual probability of constraint violation is larger than the intended confidence would suggest. For a more detailed comparison of these two approaches see [171].

5.6 Conclusion

ROmodel formulates robust versions of common optimization problems. The modeling environment it provides makes (adjustable) robust optimization methods more readily available to practitioners and makes trying different solution approaches and uncertainty sets very easy. ROmodel is open source and available free of charge and could play a vital role as a platform for prototyping novel robust optimization algorithms and comparing them to existing approaches.

Chapter 6

Conclusion

This thesis has applied robust optimization to novel applications with uncertain equipment degradation. In the process, we have developed new methods for incorporating stochastic models, which are often used to model equipment degradation, into process level optimization and scheduling problems. These developments include a new reformulation for optimization problems with uncertain black box constraints modeled by warped Gaussian processes. We have analysed and discussed in detail the relationship between maintenance, robustness to unexpected failures, and expected process costs. We have also developed ROmodel, an open source Python library for modeling robust optimization problems, in order to make transitioning from deterministic to robust optimization easier. ROmodel also implements the reformulations for Gaussian process-based sets we have developed to make them more accessible to practitioners. ROmodel allows close coupling of training data-based models and using them to generate uncertainty sets for robust optimization.

There are many ways in which this work could be extended. Processes often continuously collect data about equipment health. It would therefore make sense to re-generate data-driven uncertainty sets as new data comes in. Bayesian approaches for updating degradation models already exist and could be incorporated into this methodology. This may also be coupled with the cutting plane approach for robust optimization by adaptively removing and adding cuts as the uncertainty set changes, instead of solving the model from scratch every time.

There are also many other facets of equipment degradation and maintenance scheduling that could and should be explored in the context of optimization under uncertainty. These range from different types of assumptions about degradation characteristics, such as equipment which cannot be restored to its original state but can be maintained to increase it's lifetime, to more complex maintenance strategies such as opportunistic maintenance.

The (warped) Gaussian process-based uncertainty sets could be extended and applied to different types of problems. Adapting recent methods for global optimization of Gaussian processes may allow for solving the reformulation proposed in this thesis to global optimality. Using the Gaussian process-based set in mixed integer problems may also require further work. In general, uncertain black-box constraints should be a topic of further investigation. There are many other stochastic models, e.g. Bayesian belief networks or Bayesian neural networks, which may be used to model them. Incorporating such models into optimization problems requires the development of new methods and robust reformulations.

Finally, ROmodel can be extended to include many more uncertainty set geometries and reformulations. This includes other simple geometries, e.g. box sets, intersections of common geometries, or novel approaches for data-driven uncertainty sets. Especially the latter could make full use of Python's data-processing capabilities to offer very practical uncertainty sets. There are also many other techniques in and adjacent to robust optimization which would be valuable features in ROmodel. Examples include distributionally robust optimization, which may be implemented in ROmodel by introducing an ambiguity set component, k-adaptability or non-affine decision rules, or chance constraint approaches. In the long-term ROmodel could even be extended to stochastic programming approaches through the introduction of scenario sets. ROmodel is designed for *modeling* robust optimization problems and could be extended to offer interfaces to other robust optimization solvers, e.g. PyROS or ROC++. This would make it much easier for practicioners and researchers to compare different solution approaches. ROmodel's cutting plane solver could also be integrated more tightly with some of the underlying Pyomo solvers, e.g. by using callbacks to generate robust cuts when a feasible candidate solution is identified.

Bibliography

[1] E. Kondili, C.C. Pantelides, and R.W.H. Sargent. A general algorithm for short-term scheduling of batch operations - I. MILP formulation. *Computers and Chemical Engineering*, 17(2):211–227, 1993.

[2] Iftekhar A. Karimi and Conor M. McDonald. Planning and Scheduling of Parallel Semi-continuous Processes. 2. Short-Term Scheduling. *Industrial & Engineering Chemistry Research*, 36(7):2701–2714, 1997.

[3] Christos T. Maravelias and Ignacio E. Grossmann. New general continuous-time state - Task network formulation for short-term scheduling of multipurpose batch plants. *Industrial & Engineering Chemistry Research*, 42(13):3056–3074, 2003.

[4] M. G. Ierapetritou and C. A. Floudas. Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes. *Industrial & Engineering Chemistry Research*, 37(11):4341–4359, 1998.

[5] Samba BA, Maxim Pushkarev, Anton Kolyshkin, Lijun Song, and Ling Ling Yin. Positive Displacement Motor Modeling: Skyrocketing the Way We Design, Select, and Operate Mud Motors. In *Abu Dhabi International Petroleum Exhibition & Conference*. Society of Petroleum Engineers, 2016.

[6] A L Soyster. Technical Note — Convex Programming with Set-Inclusive Constraints and Applications to Inexact Linear Programming. *Oper. Res.*, 21(5):1154–1157, 1973.

[7] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–13, 1999.

[8] Dimitris Bertsimas and Melvyn Sim. The price of robustness. *Oper. Res.*, 52:35–53, 2004.

[9] A Ben-Tal and A Nemirovski. Robust Convex Optimization. *Mathematics of Operations Research*, 23(4):769–805, 1998.

[10] Yuan Yuan, Zukui Li, and Biao Huang. Robust optimization approximation for joint chance constrained optimization problem. *Journal of Global Optimization*, 67(4):805–827, 2017.

[11] Elijah Polak. *Optimization: algorithms and consistent approximations*. Springer Science & Business Media, 2012.

[12] Sven Leyffer, Matt Menickelly, Todd Munson, Charlie Vanaret, and Stefan M Wild. Nonlinear Robust Optimization. 2018. preprint ANL/MCS-P9040-0218, Argonne National Laboratory, Mathematics and Computer Science Division.

[13] Ahmadreza Marandi, Aharon Ben-Tal, Dick den Hertog, and Bertrand Melenberg. Extending the Scope of Robust Quadratic Optimization. Technical report, Optimization Online, 2017.

[14] Aharon Ben-Tal, Dick den Hertog, and Jean Philippe Vial. Deriving robust counterparts of nonlinear uncertain inequalities. *Mathematical Programming*, 149(1-2):265–299, 2014.

[15] Angelos Tsoukalas, Berç Rustem, and Efstratios N Pistikopoulos. A global optimization algorithm for generalized semi-infinite , continuous minimax with coupled constraints and bi-level problems. *Journal of Global Optimization*, 44:235–250, 2009.

[16] Alexander Mitsos. Global optimization of semi-infinite programs via restriction of the right-hand side. *Optimization*, 1934, 2011.

[17] Oliver Stein. How to solve a semi-infinite optimization problem. *European Journal of Operational Research*, 223(2):312–320, 2012.

[18] Moritz Diehl, Hans Georg Bock, and Ekaterina Kostina. An approximation technique for robust nonlinear optimization. *Mathematical Programming*, 107(1-2):213–230, 2006.

[19] Boris Houska and Moritz Diehl. Nonlinear robust optimization via sequential convex bilevel programming. *Math. Program.*, 142:539–577, 2013.

[20] Yuan Yuan, Zukui Li, and Biao Huang. Nonlinear robust optimization for process design. *AIChE Journal*, 64(2), 2017.

[21] Nam Ho-Nguyen and Fatma Kılınç-Karzan. Exploiting problem structure in optimization under uncertainty via online convex optimization. *Mathematical Programming*, 2018.

[22] Oliver Stein and Georg Still. Solving Semi-Infinite Optimization Problems with Interior Point Techniques. *SIAM Journal on Control and Optimization*, 42(3):769–788, 2003.

[23] M Diehl, B Houska, O Stein, and P Steuermann. A lifting method for generalized semi-infinite programs based on lower level Wolfe duality. *Computational Optimization and Applications*, 54(1):189–210, 2012.

[24] Stuart M Harwood and Paul I Barton. Lower level duality and the global solution of generalized semi-infinite programs. *Optimization*, 65(6):1129–1149, 2016.

[25] Zukui Li and Marianthi G. Ierapetritou. Robust Optimization for Process Scheduling Under Uncertainty. *Ind. Eng. Chem. Res.*, 47(12):4148–4157, 2008.

[26] Zukui Li, Ran Ding, and Christodoulos Floudas. A Comparative Theoretical and Computational Study on Robust Counterpart Optimization: I. Robust Linear Optimization and Robust Mixed Integer Linear Optimization. *Industrial & Engineering Chemistry Research*, 50(18):10567–10603, 2011.

[27] Lisia S. Dias and Marianthi G. Ierapetritou. Integration of scheduling and control under uncertainties: Review and challenges. *Chemical Engineering Research & Design*, 116: 98–113, 2016.

[28] Pedro M. Castro, Ignacio E. Grossmann, and Qi Zhang. Expanding scope and computational challenges in process scheduling. *Computers & Chemical Engineering*, 114:14–42, 2018.

[29] Johannes Wiebe, Inês Cecílio, and Ruth Misener. Robust optimization for the pooling problem. *Ind. Eng. Chem. Res.*, 2019.

[30] Lawrence T Narraway and John D Perkins. Selection of Process Control Structure Based on Linear Dynamic Economics. *Industrial & Engineering Chemistry Research*, 32(11): 2681–2692, 1993.

[31] Parka A Bahri, Jose A Bandoni, and Jose A Romagnoli. Effect of Disturbances in Optimizing Control : Steady-State Open-Loop Backoff Problem. *AIChE Journal*, 42(4): 983–994, 2006.

[32] Robert W. Koller, Luis A. Ricardez-Sandoval, and Lorenz T. Biegler. Stochastic back-off algorithm for simultaneous design, control, and scheduling of multiproduct systems under uncertainty. *AIChE Journal*, 64(7):2379–2389, 2018.

[33] Mina Rafiei and Luis A. Ricardez-Sandoval. Stochastic Back-Off Approach for Integration of Design and Control under Uncertainty. *Industrial & Engineering Chemistry Research*, 57(12):4351–4365, 2018.

[34] Dimitris Bertsimas, Vishal Gupta, and Nathan Kallus. Data-driven robust optimization. *Math. Program.*, 167:235–292, 2018.

[35] Trevor Campbell and Jonathan P. How. Bayesian nonparametric set construction for robust optimization. *Proceedings of the American Control Conference*, 2015-July:4216–4221, 2015.

[36] Chao Ning and Fengqi You. Data-driven stochastic robust optimization: General computational framework and algorithm leveraging machine learning for optimization under uncertainty in the big data era. *Computers and Chemical Engineering*, 111:115–133, 2018.

[37] Chao Shang, Xiaolin Huang, and Fengqi You. Data-driven robust optimization based on kernel learning. *Computers and Chemical Engineering*, 106:464–479, 2017.

[38] Marc Goerigk and Jannis Kurtz. Data-driven robust optimization using unsupervised deep learning, 2020.

[39] Shipu Zhao and Fengqi You. Distributionally robust chance constrained programming with generative adversarial networks (gans). *AIChE Journal*, 66, 2020.

[40] Khanh T.P. Nguyen, Mitra Fouladirad, and Antoine Grall. Model selection for degradation modeling and prognosis with health monitoring data. *Reliability Engineering & System Safety*, 169(August 2017):105–116, 2018.

[41] Cong Liu, Simon X. Yang, Xiaofang Li, Lijuan Xu, and Lie Deng. Noise level penalizing robust gaussian process regression for nir spectroscopy quantitative analysis. *Chemometrics and Intelligent Laboratory Systems*, 201, 2020.

[42] Eric Bradford, Lars Imsland, Dongda Zhang, and Ehecatl Antonio del Rio Chanona. Stochastic data-driven model predictive control using gaussian processes. *Computers and Chemical Engineering*, 139, 2020.

[43] Tao Chen and Jie Zhang. On-line multivariate statistical monitoring of batch processes using gaussian mixture model. *Computers and Chemical Engineering*, 34:500–507, 2010.

[44] Qi Zhang, Ignacio E. Grossmann, Clara F. Heuberger, Arul Sundaramoorthy, and Jose M. Pinto. Air separation with cryogenic energy storage: Optimal scheduling considering electric energy and reserve markets. *AIChE Journal*, 61(5):1547–1558, 2015.

[45] Prerna Jain, Antik Chakraborty, Efstratios N. Pistikopoulos, and M. Sam Mannan. Resilience-based process upset event prediction analysis for uncertainty management using bayesian deep learning: Application to a polyvinyl chloride process system. *Industrial and Engineering Chemistry Research*, 57:14822–14836, 2018.

[46] Suzan Alaswad and Yisha Xiang. A review on condition-based maintenance optimization models for stochastically deteriorating system. *Reliability Engineering & System Safety*, 157:54–63, 2017.

[47] David Applebaum. Lévy processes-from probability to finance and quantum groups. *Notices of the American Mathematical Society*, 51(11):1336–1347, 2004.

[48] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning.* The MIT Press, Cambridge, Massachusetts, 2006.

[49] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando De Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[50] Christopher K. I. Williams and Carl Edward Rasmussen. Gaussian processes for regression & classification. In *NIPS*, pages 514–520, 2008.

[51] Edward Snelson, Carl E. Rasmussen, and Zoubin Ghahramani. Warped Gaussian processes. In *NIPS*, 2003.

[52] Peng Kou, Feng Gao, and Xiaohong Guan. Sparse online warped Gaussian process for wind power probabilistic forecasting. *Appl. Energy*, 108:410–428, 2013.

[53] Anna Mateo-Sanchis, Jordi Muñoz-Marí, Adrián Pérez-Suay, and Gustau Camps-Valls. Warped Gaussian Processes in Remote Sensing Parameter Estimation and Causal Inference. *IEEE Geoscience and Remote Sensing Letters*, 15(11):1647–1651, 2018.

[54] Andrew K.S. Jardine, Daming Lin, and Dragan Banjevic. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, 20(7):1483–1510, 2006.

[55] Diana Barraza-Barraza, Jorge Limón-Robles, and Mario G Beruvides. Opportunities and challenges in Condition-Based Maintenance research. *IIE Annual Conference and Expo 2014*, pages 3035–3043, 2014.

[56] Alexandros Bousdekis, Babis Magoutas, Dimitris Apostolou, and Gregoris Mentzas. Review, analysis and synthesis of prognostic-based decision support methods for condition based maintenance. *Journal of Intelligent Manufacturing*, pages 1–14, 2015.

[57] William Q. Meeker and Yili Hong. Reliability Meets Big Data: Opportunities and Challenges. *Quality Engineering*, 26(1):102–116, 2014.

[58] Ilias T. Dedopoulos and Nilay Shah. Optimal Short-Term Scheduling of Maintenance and Production for Multipurpose Plants. *Industrial & Engineering Chemistry Research*, 34 (1):192–201, 1995.

[59] I.T. Dedopoulos and N. Shah. Preventive maintenance policy optimization for multipurpose plant equipment. *Computers & Chemical Engineering*, 19:693–698, 1995.

[60] Constantinos Georgiou Vassiliadis. *Integration of Maintenance Optimization in Process Design and Operation under Uncertainty*. PhD thesis, Imperial College of Science, Technology and Medicine, 1999.

[61] C.G. Vassiliadis and E.N. Pistikopoulos. Maintenance scheduling and process optimization under uncertainty. *Computers and Chemical Engineering*, 25(2-3):217–236, 2001.

[62] J. Casas-Liza, J.M. Pinto, and L.G. Papageorgiou. Mixed Integer Optimization for Cyclic Scheduling of Multiproduct Plants Under Exponential Performance Decay. *Chemical Engineering Research and Design*, 83(10):1208–1217, 2005.

[63] Michael C. Georgiadis, Lazaros G. Papageorgiou, and Sandro Macchietto. Optimal Cleaning Policies in Heat Exchanger Networks under Rapid Fouling. *Industrial & Engineering Chemistry Research*, 39(2):441–454, 2000.

[64] Songsong Liu, Ahmed Yahia, and Lazaros G. Papageorgiou. Optimal Production and Maintenance Planning of Biopharmaceutical Manufacturing under Performance Decay. *Industrial & Engineering Chemistry Research*, 53(44):17075–17091, 2014.

[65] Dionysios P. Xenos, Georgios M. Kopanos, Matteo Cicciotti, and Nina F. Thornhill. Operational optimization of networks of compressors considering condition-based maintenance. *Computers and Chemical Engineering*, 84:117–131, 2016.

[66] Nur I. Zulkafli and Georgios M. Kopanos. Planning of production and utility systems under unit performance degradation and alternative resource-constrained cleaning policies. *Applied Energy*, 183:577–602, 2016.

[67] Nur I. Zulkafli and Georgios M. Kopanos. Integrated condition-based planning of production and utility systems under uncertainty. *Journal of Cleaner Production*, 167:776–805, 2017.

[68] Adrian M. Aguirre and Lazaros G. Papageorgiou. Medium-term optimization-based approach for the integration of production planning, scheduling and maintenance. *Computers and Chemical Engineering*, 0:1–21, 2018.

[69] Sreekanth Rajagopalan, Nikolaos V. Sahinidis, Satyajith Amaran, Anshul Agarwal, Scott J. Bury, Bikram Sharda, and John M. Wassick. Risk analysis of turnaround reschedule planning in integrated chemical sites. *Computers & Chemical Engineering*, 107:381–394, 2017.

[70] Matteo Biondi, Guido Sand, and Iiro Harjunkoski. Optimization of multipurpose process plant operations: A multi-time-scale maintenance and production scheduling approach. *Computers and Chemical Engineering*, 99:325–339, 2017.

[71] Murat Yildirim, Xu Andy Sun, and Nagi Z Gebraeel. Sensor-Driven Condition-Based Generator Maintenance Scheduling - Part I: Maintenance Problem. *IEEE Transactions on Power Systems*, 31(6):4253–4262, 2016.

[72] Murat Yildirim, Xu Andy Sun, and Nagi Z Gebraeel. Sensor-Driven Condition-Based Generator Maintenance Scheduling - Part II: Incorporating Operations. *IEEE Transactions on Power Systems*, 31(6):4263–4271, 2016.

[73] Murat Yildirim, Nagi Z. Gebraeel, and Xu Andy Sun. Integrated Predictive Analytics and Optimization for Opportunistic Maintenance and Operations in Wind Farms. *IEEE Transactions on Power Systems*, 32(6):4319–4328, 2017.

[74] Beste Başçiftci, Shabbir Ahmed, Nagi Z. Gebraeel, and Murat Yildirim. Stochastic Optimization of Maintenance and Operations Schedules under Unexpected Failures. *IEEE Transactions on Power Systems*, 8950(c):1–1, 2018.

[75] Adriaen Verheyleweghen and Johannes Jäschke. Framework for Combined Diagnostics, Prognostics and Optimal Operation of a Subsea Gas Compression System. *IFAC-PapersOnLine*, 50(1):15916–15921, 2017.

[76] Nikolaos H Lappas and Chrysanthos E Gounaris. Multi-stage adjustable robust optimization for process scheduling under uncertainty. *AIChE Journal*, 62(5):1646–1667, 2016.

[77] Peng Wang and David Coit. Reliability and Degradation Modeling with Random or Uncertain Failure Threshold. In *2007 Proceedings - Annual Reliability and Maintainability Sympsoium*, pages 392–397. IEEE, 2007.

[78] Laurent Doyen and Olivier Gaudoin. Classes of imperfect repair models based on reduction of failure intensity or virtual age. *Reliability Engineering & System Safety*, 84(1): 45–56, 2004.

[79] Zhi-Sheng Ye and Min Xie. Stochastic modelling and analysis of degradation for highly reliable products. *Applied Stochastic Models in Business and Industry*, 31(1):16–32, 2015.

[80] Xiao-Sheng Si, Wenbin Wang, Chang-Hua Hu, and Dong-Hua Zhou. Remaining useful life estimation - A review on the statistical data driven approaches. *European Journal of Operational Research*, 213(1):1–14, 2011.

[81] Haitao Liao and Zhigang Tian. A framework for predicting the remaining useful life of a single unit under time-varying operating conditions. *IIE Transactions*, 45(9):964–980, 2013.

[82] Qi Li, Zhanbao Gao, Diyin Tang, and Baoan Li. Remaining useful life estimation for deteriorating systems with time-varying operational conditions and condition-specific failure zones. *Chinese Journal of Aeronautics*, 29(3):662–674, 2016.

[83] Julien Chevallier and Stéphane Goutte. On the estimation of regime-switching Lévy models. *Studies in Nonlinear Dynamics and Econometrics*, 21(1):3–29, 2017.

[84] Nagi Z. Gebraeel, Mark A. Lawley, Rong Li, and Jennifer K. Ryan. Residual-life distributions from component degradation signals: A Bayesian approach. *IIE Transactions*, 37(6):543–557, 2005.

[85] Linkan Bian and Nagi Gebraeel. Computing and updating the first-passage time distribution for randomly evolving degradation signals. *IIE Transactions*, 44(11):974–987, 2012.

[86] Nagi Gebraeel and Jing Pan. Prognostic degradation models for computing and updating residual life distributions in a time-varying environment. *IEEE Transactions on Reliability*, 57(4):539–550, 2008.

[87] Chao Ning and Fengqi You. A data-driven multistage adaptive robust optimization framework for planning and scheduling under uncertainty. *AIChE Journal*, 63(10):4343–4369, 2017.

[88] Yannis A. Guzman, Logan R. Matthews, and Christodoulos A. Floudas. New a priori and a posteriori probabilistic bounds for robust counterpart optimization: I. Unknown probability distributions. *Computers & Chemical Engineering*, 84:568–598, 2016.

[89] Zukui Li, Ran Ding, and Christodoulos Floudas. A Comparative Theoretical and Computational Study on Robust Counterpart Optimization: I. Robust Linear Optimization and Robust Mixed Integer Linear Optimization. *Industrial & Engineering Chemistry Research*, 50(18):10567–10603, 2011.

[90] Klaus; Pötzelberger and Liqun Wang. Boundary Crossing Probability for Brownian Motion and General Boundaries. *Journal of Applied Probability*, 34(1):54–65, 1997.

[91] Linkan Bian and Nagi Gebraeel. A stochastic methodology for prognostics under time-varying environmental future profiles. In *Proceedings of the 2011 Conference on Intelligent Data Understanding, CIDU 2011*, 2011.

[92] Lothar Breuer. Occupation times for Markov-modulated Brownian motion. *Journal of Applied Probability*, 49(2):549–565, 2012.

[93] Souleyman Ozekici. Optimal maintenance policies in random environments. *European Journal of Operational Research*, 82(2):283–294, 1995.

[94] Lewis Paton, Matthias C M Troffaes, Nigel Boatman, Mohamud Hussein, and Andy Hart. Multinomial Logistic Regression on Markov Chains for Crop Rotation Modelling. *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 476–485, 2014.

[95] Narayan Chanra Sinha, M. Ataharul Islam, and Kazi Saleh Ahamed. Logistic Regression Models for Higher Order Transition Probabilities of Markov Chain for Analyzing the Occurrences of Daily Rainfall Data. *Journal of Modern Applied Statistical Methods*, 10 (1):337–348, 2011.

[96] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2012.

[97] Cristian Bravo, Gaston L'Huillier, Jose Luis Lobato, and Richard Weber. Probability Estimation for Multiclass Problems Combining SVMs and Neural Networks. *Neural Network World*, 20(4):475–489, 2010.

[98] Stephan Dreiseitl and Lucila Ohno-Machado. Logistic regression and artificial neural network classification models: A methodology review. *Journal of Biomedical Informatics*, 35(5-6):352–359, 2002.

[99] Zhuangzhi Li and Zukui Li. Chance constrained planning and scheduling under uncertainty using robust optimization approximation. *IFAC-PapersOnLine*, 28(8):1156–1161, 2015.

[100] Zhuangzhi Li and Zukui Li. Optimal robust optimization approximation for chance constrained optimization problem. *Computers and Chemical Engineering*, 74:89–99, 2015.

[101] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13:455–492, 1998.

[102] William E. Hart, Carl D. Laird, Jean-Paul Watson, David L. Woodruff, Gabriel A. Hackebeil, Bethany L. Nicholson, and John D. Siirola. *Pyomo — Optimization Modeling in Python*, volume 67. Springer International Publishing, 2017.

[103] William E. Hart, Jean Paul Watson, and David L. Woodruff. Pyomo: Modeling and solving mathematical programs in Python. *Mathematical Programming Computation*, 3 (3):219–260, 2011.

[104] Johannes Wiebe. STN with degradation. DOI: 10.5281/zenodo.1313718, 2018. URL `https://doi.org/10.5281/zenodo.1313718`.

[105] Miten Mistry, Andrea Callia D'Iddio, Michael Huth, and Ruth Misener. Satisfiability modulo theories for process systems engineering. *Computers and Chemical Engineering*, 113:98–114, 2018.

[106] Andre A. Ciré, Elvin Çoban, and John N. Hooker. Logic-based Benders decomposition for planning and scheduling: A computational analysis. *Knowledge Engineering Review*, 31(5):440–451, 2016.

[107] D. Letsios and R. Misener. Exact Lexicographic Scheduling and Approximate Rescheduling. *ArXiv e-prints*, arXiv:1805.03437, 2018.

[108] Zukui Li, Qiuhua Tang, and Christodoulos A. Floudas. A Comparative Theoretical and Computational Study on Robust Counterpart Optimization: II. Probabilistic Guarantees on Constraint Satisfaction. *Industrial & Engineering Chemistry Research*, 51(19): 6769–6788, 2012.

[109] Dimitris Bertsimas, David B Brown, and Constantine Caramanis. Theory and Applications of Robust Optimization. *SIAM Review*, 53(3):464–501, 2011.

[110] Dimitris Bertsimas and Melvyn Sim. The Price of Robustness. *Oper. Res.*, 52(1):35–53, 2004.

[111] John R. Birge and François Louveaux. *Introduction to Stochastic Programming*. 2011.

[112] Nikolaos V. Sahinidis. Optimization under uncertainty: State-of-the-art and opportunities. *Comput. Chem. Eng.*, 28(6-7):971–983, 2004.

[113] Atharv Bhosekar and Marianthi Ierapetritou. Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Comput. Chem. Eng.*, 108:250–267, 2018.

[114] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian Processes for Data-Efficient Learning in Robotics and Control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423, 2015.

[115] Nima Dolatnia, Alan Fern, and Xiaoli Fern. Bayesian Optimization with Resource Constraints and Production. *Twenty-Sixth International Conference on Automated Planning and Scheduling (ICAPS)*, pages 115–123, 2016.

[116] Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained Bayesian Optimization for Automatic Chemical Design. In *NIPS*, 2018.

[117] Siyuan Liu, Yisong Yue, and Ramayya Krishnan. Adaptive collective routing using Gaussian process dynamic congestion models. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13*, page 704. ACM Press, 2013.

[118] Doniyor Ulmasov, Caroline Baroukh, Benoit Chachuat, Marc Peter Deisenroth, and Ruth Misener. Bayesian optimization with dimension scheduling: Application to biological systems. In Zdravko Kravanja and Miloš Bogataj, editors, *26th European Symposium on Computer Aided Process Engineering*, volume 38, pages 1051 – 1056. Elsevier, 2016.

[119] Johannes Wiebe, Inês Cecílio, and Ruth Misener. Data-Driven Optimization of Processes with Degrading Equipment. *Ind. Eng. Chem. Res.*, 57(50):17177–17191, 2018.

[120] Jonas Mockus. On Bayesian methods for seeking the extremum. In *Proceedings of the IFIP Technical Conference*, pages 400–404, London, UK, UK, 1974. Springer-Verlag.

[121] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *NIPS*, pages 2951–2959, USA, 2012.

[122] Jacob R. Gardner, Matt J. Kusner, Zhixiang Xu, Kilian Q. Weinberger, and John P. Cunningham. Bayesian optimization with inequality constraints. In *ICML*, 2014.

[123] Michael A. Gelbart, Jasper Snoek, and Ryan P. Adams. Bayesian Optimization with Unknown Constraints. *arXiv e-prints*, art. arXiv:1403.5607, 2014.

[124] Victor Picheny, Robert B Gramacy, Stefan Wild, and Sebastien Le Digabel. Bayesian optimization under mixed constraints with a slack-variable augmented Lagrangian. In *NIPS*, 2016.

[125] Burcu Beykal, Fani Boukouvala, Christodoulos A. Floudas, Nadav Sorek, Hardikkumar Zalavadia, and Eduardo Gildin. Global optimization of grey-box computational systems using surrogate functions and application to highly constrained oil-field operations. *Comput. Chem. Eng.*, 114:99–110, 2018.

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

[127] Fani Boukouvala, Ruth Misener, and Christodoulos A. Floudas. Global optimization advances in Mixed-Integer Nonlinear Programming, MINLP, and Constrained Derivative-Free Optimization, CDFO. *Eur. J. Oper. Res.*, 252(3):701–727, 2016.

[128] Ignacio E Grossmann. Pyrolysis of Heavy Oil in the Presence of Supercritical Water: The Reaction Kinetics in Different Phases. *AICHE Journal*, 61(3):857–866, 2015.

[129] Miten Mistry, Dimitrios Letsios, Gerhard Krennrich, Robert M. Lee, and Ruth Misener. Mixed-Integer Convex Nonlinear Optimization with Gradient-Boosted Trees Embedded. 2018. URL `http://arxiv.org/abs/1803.00952`.

[130] Juliane Müller, Christine A. Shoemaker, and Robert Piché. SO-MI: A surrogate model

algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems. *Comput. Oper. Res.*, 40(5):1383–1400, 2013.

[131] Rommel G. Regis and Christine A. Shoemaker. Constrained global optimization of expensive black box functions using radial basis functions. *Journal of Global Optimization*, 31(1):153–171, 2005.

[132] Gordon Hüllen, Jianyuan Zhai, Sun Hye Kim, Anshuman Sinha, Matthew J. Realff, and Fani Boukouvala. Managing Uncertainty in Data-Driven Simulation-Based Optimization. *Comput. Chem. Eng.*, page 106519, 2019.

[133] Pradeep Varakantham, Na Fu, and Hoong Chuin Lau. A proactive sampling approach to project scheduling under uncertainty. In *AAAI*, 2016.

[134] Justin J Beland and Prasanth B Nair. Bayesian Optimization Under Uncertainty. In *NIPS*, 2017.

[135] Dimitris Bertsimas, Omid Nohadani, and Kwong Meng Teo. Nonconvex robust optimization for problems with constraints. *INFORMS J. Comput.*, 22(1):44–58, 2010.

[136] Dimitris Bertsimas, Omid Nohadani, and Kwong Meng Teo. Robust Optimization for Unconstrained Simulation-Based Problems. *Oper. Res.*, 58(1):161–178, 2010.

[137] Ilija Bogunovic, Jonathan Scarlett, Stefanie Jegelka, and Volkan Cevher. Adversarially Robust Optimization with Gaussian Processes. In *NIPS*, 2018.

[138] A. Charnes and W. W. Cooper. Deterministic Equivalents for Optimizing and Satisficing under Chance Constraints. *Oper. Res.*, 11(1):18–39, 1963.

[139] James Luedtke and Shabbir Ahmed. A sample approximation approach for optimization with probabilistic constraints. *SIAM J. Optim.*, 19(2):674–699, 2008.

[140] Arkadi Nemirovski and Alexander Shapiro. Scenario Approximations of Chance Constraints. In G. Calafiore and F. Dabbene, editors, *Probabilistic and Randomized Methods for Design under Uncertainty*, number 2, pages 3–47. Springer Verlag, 2006.

[141] B. K. Pagnoncelli, S. Ahmed, and A. Shapiro. Sample average approximation method for chance constrained programming: Theory and applications. *J. Optim. Theory Appl.*, 142(2):399–416, 2009.

[142] Shabbir Ahmed. Convex relaxations of chance constrained optimization problems. *Optim. Lett.*, 8(1):1–12, 2014.

[143] Zhuangzhi Li and Zukui Li. Optimal robust optimization approximation for chance constrained optimization problem. *Comput. Chem. Eng.*, 74:89–99, 2015.

[144] Arkadi Nemirovski. On safe tractable approximations of chance constraints. *Eur. J. Oper. Res.*, 219(3):707–718, 2012.

[145] Arkadi Nemirovski and Alexander Shapiro. Convex approximations of chance constrained programs. *SIAM J. Optim.*, 17(4):959–996, 2006.

[146] J. Pintér. Deterministic approximations of probability inequalities. *ZOR Zeitschrift für Operations Research Methods and Models of Operations Research*, 33(4):219–239, 1989.

[147] Weijun Xie and Shabbir Ahmed. Distributionally Robust Chance Constrained Optimal Power Flow with Renewables: A Conic Reformulation. *IEEE Transactions on Power Systems*, 33(2):1860–1867, 2018.

[148] P. R. Baxandall and H. Liebeck. *Vector calculus*. Dover Publications, Mineola, N.Y., Dover ed. edition, 2008.

[149] D R Jones, C D Perttunen, B E Stuckman, and L C W Dixon. Lipschitzian optimization without the lipschitz constant. *JOURNAL OF OPTIMIZATION THEORY AND APPLICATION*, 79, 1993.

[150] Rémi Munos. Optimistic optimization of a deterministic function without the knowledge of its smoothness. pages 783–791, 2011.

[151] Emmanuel Detournay, Thomas Richard, and Mike Shepherd. Drilling response of drag bits: Theory and experiment. *International Journal of Rock Mechanics and Mining Sciences*, 45(8):1347–1360, 2008.

[152] Beste Başçiftci, Shabbir Ahmed, Nagi Gebraeel, and Murat Yildirim. Integrated Generator Maintenance and Operations Scheduling under Uncertain Failure Times. *IEEE Transactions on Power Systems*, 2018.

[153] G Schilling and C C Pantelides. A Simple Continuous-Time Process Scheduling Formulation and a Novel Solution Algorithm. *Comput. Chem. Eng.*, 20(96):1221–1226, 1996.

[154] Christodoulos A. Floudas and Xiaoxia Lin. Continuous-time versus discrete-time approaches for scheduling of chemical processes: A review. *Comput. Chem. Eng.*, 28(11): 2109–2129, 2004.

[155] William E. Hart, Jean Paul Watson, and David L. Woodruff. Pyomo: Modeling and solving mathematical programs in Python. *Math. Progam. Comput.*, 3(3):219–260, 2011.

[156] William E. Hart, Carl D. Laird, Jean-Paul Watson, David L. Woodruff, Gabriel A. Hackebeil, Bethany L. Nicholson, and John D. Siirola. *Pyomo–optimization modeling in python*, volume 67. Springer Science & Business Media, second edition, 2017.

[157] GPy. GPy: A Gaussian process framework in . `http://github.com/SheffieldML/GPy`, since 2012.

[158] Johannes Wiebe. ROGP: Robust GPs in Pyomo. `https://github.com/cog-imperial/rogp`, 2020.

[159] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106 (1):25–57, 2006.

[160] Stacy L. Janak and Christodoulos A. Floudas. Advances in robust optimization approaches for scheduling under uncertainty. *Comput. Chem. Eng.*, 20(C):1051–1056, 2005.

[161] Chao Shang and Fengqi You. Distributionally robust optimization for planning and scheduling under uncertainty. *Comput. Chem. Eng.*, 110:53–68, 2018.

[162] Ignacio E. Grossmann, Robert M. Apap, Bruno A. Calfa, Pablo García-Herreros, and Qi Zhang. Recent advances in mathematical programming techniques for the optimization of process systems under uncertainty. *Comput. Chem. Eng.*, 91:3–14, 2016.

[163] Anita Goerigk Marc and Schöbel. Algorithm engineering in robust optimization. *Algorithm Engineering: Selected Results and Surveys*, pages 245–279, 2016.

[164] William E Hart, Jean-Paul Watson, and David L Woodruff. Pyomo: modeling and solving mathematical programs in Python. *Mathematical Programming Computation*, 3 (3):219–260, 2011.

[165] Joel Goh and Melvyn Sim. Robust optimization made easy with rome. *Oper. Res.*, 59: 973–985, 2011.

[166] Zhi Chen, Melvyn Sim, and Peng Xiong. Robust stochastic optimization made easy with rsome. *Management Science*, 66:3329–3339, 2020.

[167] Phebe Vayanos, Qing Jin, and George Elissaios. Roc++: Robust optimization in c++., 2020.

[168] N. M. Isenberg, J. D. Siirola, and C. E. Gounaris. Pyros: A pyomo robust optimization solver for robust process design. In *2020 Virtual AIChE Annual Meeting*, 2020.

[169] AIMMS. `http://aimms.com`. Accessed: 2021-03-25.

[170] Iain R. Dunning. *Advances in Robust and Adaptive Optimization: Algorithms, Software, and Insights*. PhD thesis, Sloan School of Management, MIT, 2016.

[171] Johannes Wiebe, Inês Cecílio, Jonathan Dunlop, and Ruth Misener. A robust approach to warped Gaussian process-constrained optimization, 2020.

[172] Johannes Wiebe and Ruth Misener. Romodel 0.1.0. `github.com/cog-imperial/romodel`, 2020.

[173] Almir Mutapcic and Stephen Boyd. Cutting-set methods for robust convex optimization with pessimizing oracles. *Optim. Method. Softw.*, 24:381–406, 2009.

[174] Nilanjan Adhya, Mohit Tawarmalani, and Nikolaos V. Sahinidis. A Lagrangian Approach to the Pooling Problem. *Ind. Eng. Chem. Res.*, 38(5):1956–1972, 1999.

[175] Johannes Wiebe. Drill scheduling. `github.com/cog-imperial/drill-scheduling`, 2020.

[176] Linkan Bian and Nagi Gebraeel. Stochastic methodology for prognostics under continuously varying environmental profiles. *Statistical Analysis and Data Mining*, 6(3):260–270, 2013.

[177] David Siegmund. Boundary Crossing Probabilities and Statistical Applications. *The Annals of Statistics*, 14(2):361–404, 1986.

Appendix A

Formulating the robust counterpart

The reformulation of the semi-infinite constraints

$$m_{j,t}s_j^0 \leq s_{j,t} \qquad \forall t, j \in J, \tilde{d}_{j,k} \in \mathcal{D} \qquad \text{(A.1a)}$$

$$s_{j,t} \leq s_j^{max} + m_{j,t} \cdot (s_j^0 - s_j^{max}) \qquad \forall t, j \in J, \tilde{d}_{j,k} \in \mathcal{U} \qquad \text{(A.1b)}$$

$$s_{j,t} \geq s_{j,t-\Delta t} + \sum_k x_{j,k,t}\tilde{d}_{j,k} + m_{j,t} \cdot (s_j^0 - s_j^{max}) \qquad \forall t, j \in J, \tilde{d}_{j,k} \in \mathcal{U} \qquad \text{(A.1c)}$$

$$s_{j,t} \leq s_{j,t-\Delta t} + \sum_k x_{j,k,t}\tilde{d}_{j,k} \qquad \forall t, j \in J, \tilde{d}_{j,k} \in \mathcal{U} \qquad \text{(A.1d)}$$

into a deterministic robust counterpart in this work is based on the approach by Lappas and Gounaris [76]. $s_{j,t}$ is replaced by the affine decision rule

$$s_{j,t} = [s_{j,t}]_0 + \sum_k [s_{j,t}]_k \tilde{d}_{j,k}$$

in all constraints, where $[s_{j,t}]_0$ and $[s_{j,t}]_k$ are coefficients which become new variables in the reformulated constraints. The first constraint (Eqn. A.1a) can be reformulate in the following

way:

$$
\begin{aligned}
& m_{j,t}s_j^0 \leq [s_{j,t}]_0 + \sum_k [s_{j,t}]_k \tilde{d}_{j,k} \\
& \Rightarrow -\sum_k [s_{j,t}]_k \tilde{d}_{j,k} \leq [s_{j,t}]_0 - m_{j,t}s_j^0 \\
& \Rightarrow \Theta^* \leq [s_{j,t}]_0 - m_{j,t}s_j^0,
\end{aligned}
$$

where

$$
\begin{aligned}
\Theta^* = \max_{\tilde{d}_{j,k}} \quad & -\sum_k [s_{j,t}]_k \tilde{d}_{j,k} \\
\text{s.t} \quad & -\tilde{d}_{j,k} \leq -\bar{d}_{j,k}(1-\epsilon) \qquad \forall k \\
& \tilde{d}_{j,k} \leq \bar{d}_{j,k}(1+\epsilon) \qquad \forall k.
\end{aligned}
$$

The dual of this is

$$
\begin{aligned}
\Theta^* = \min_{u^{1,j,t}, l^{1,j,t}} \quad & \sum_k \bar{d}_{j,k} \left[(1+\epsilon)u_k^{1,j,t} - (1-\epsilon)l_k^{1,j,t}\right] \\
\text{s.t} \quad & u_k^{1,j,t} - l_k^{1,j,t} \geq -[s_{j,t}]_k \qquad \forall k,
\end{aligned}
$$

with dual variables $u_k^{1,j,t}$ and $l_k^{1,j,t}$. Dropping the minimization leads to the final reformulation:

$$
\begin{aligned}
& \sum_k \bar{d}_{j,k} \left[(1+\epsilon)u_k^{1,j,t} - (1-\epsilon)l_k^{1,j,t}\right] \leq [s_{j,t}]_0 - m_{j,t}s_j^0 \quad \forall t, j \in J \\
& u_k^{1,j,t} - l_k^{1,j,t} \geq -[s_{j,t}]_k \qquad \forall j, t, k
\end{aligned}
$$

Similar analysis for the second inequality (Eqn. A.1b) leads to reformulation

$$
\begin{aligned}
& \sum_k \bar{d}_{j,k} \left[(1+\epsilon)u_k^{2,j,t} - (1-\epsilon)l_k^{2,j,t}\right] \\
& \leq s_j^{max} - [s_{j,t}]_0 + m_{j,t} \cdot (s_j^0 - s_j^{max}) \quad \forall t, j \in J \\
& u_k^{2,j,t} - l_k^{2,j,t} \geq [s_{j,t}]_k \qquad \forall j, t, k,
\end{aligned}
$$

the third inequality (Eqn. A.1c) yields

$$\begin{aligned}
&\sum_k \bar{d}_{j,k} \left[(1+\epsilon)u_k^{3,j,t} - (1-\epsilon)l_k^{3,j,t}\right] \\
&\leq [s_{j,t}]_0 - [s_{j,t-\Delta t}]_0 + m_{j,t} s_j^{max} && \forall t, j \in J \\
&u_k^{3,j,t} - l_k^{3,j,t} \geq [s_{j,t-\Delta t}]_k - [s_{j,t}]_k + x_{j,k,t} && \forall j, t, k,
\end{aligned}$$

and the fourth (Eqn. A.1d)

$$\begin{aligned}
&\sum_k \bar{d}_{j,k} \left[(1+\epsilon)u_k^{4,j,t} - (1-\epsilon)l_k^{4,j,t}\right] \\
&\leq -[s_{j,t}]_0 + [s_{j,t-\Delta t}]_0 && \forall t, j \in J \\
&u_k^{4,j,t} - l_k^{4,j,t} \geq -[s_{j,t-\Delta t}]_k + [s_{j,t}]_k - x_{j,k,t} && \forall j, t, k.
\end{aligned}$$

Appendix B

Equivalence to deterministic optimization with maximal parameters

Note that for convenience and readability the index j has been dropped in all equations in this appendix.

Consider the case where the cost, process model, and maintenance model in Problem 9 are not functions of the uncertain parameters $\tilde{d}_k$:

$$\min_{\boldsymbol{x},\boldsymbol{m}} \quad \text{cost}(\boldsymbol{x},\boldsymbol{m}) \tag{B.1a}$$

$$\text{s.t} \quad \text{process model}(\boldsymbol{x},\boldsymbol{m}) \tag{B.1b}$$

$$\text{maintenance model}(\boldsymbol{x},\boldsymbol{m}) \tag{B.1c}$$

$$m_t s_0 \leq [s_t]_0 + \sum [s_t]_k \tilde{d}_k, \qquad \forall t, \tilde{d}_k \in \mathcal{U} \tag{B.1d}$$

$$[s_t]_0 + \sum [s_t]_k \tilde{d}_k \leq s^{max} + m_t(s^0 - s^{max}), \qquad \forall t, \tilde{d}_k \in \mathcal{U} \tag{B.1e}$$

$$[s_t]_0 + \sum [s_t]_k \tilde{d}_k \geq [s_{t-1}]_0 + \sum [s_{t-1}]_k \tilde{d}_k + \sum_k x_{k,t} \tilde{d}_k + m_t(s^0 - s^{max}), \qquad \forall t, \tilde{d}_k \in \mathcal{U} \tag{B.1f}$$

$$[s_t]_0 + \sum [s_t]_k \tilde{d}_k [s_{t-1}]_0 + \sum [s_{t-1}]_k \tilde{d}_k + \sum_k x_{k,t-1} \tilde{d}_k, \qquad \forall t, \tilde{d}_k \in \mathcal{U} \tag{B.1g}$$

Furthermore consider a deterministic version of Problem B.1

$$\min_{\boldsymbol{x},\boldsymbol{m}} \quad \text{cost}(\boldsymbol{x},\boldsymbol{m}) \tag{B.2a}$$

$$\text{s.t} \quad \text{process model}(\boldsymbol{x},\boldsymbol{m}) \tag{B.2b}$$

$$\text{maintenance model}(\boldsymbol{x},\boldsymbol{m}) \tag{B.2c}$$

$$m_t s^0 \leq s_t, \qquad \forall t \tag{B.2d}$$

$$s_t \leq s^{max} + m_t(s^0 - s^{max}), \qquad \forall t \tag{B.2e}$$

$$s_t \geq s_{t-1} + \sum_k x_{k,t} d_k^{max} + m_t(s^0 - s^{max}), \qquad \forall t \tag{B.2f}$$

$$s_t \leq s_{t-1} + \sum_k x_{k,t} d_k^{max}, \qquad \forall t \tag{B.2g}$$

in which $\tilde{d}_k$ has been replaced by

$$d_k^{max} = \max_{\tilde{d}_k \in \mathcal{U}} \tilde{d}_k.$$

Theorem B.1. *Given that cost, process model, and maintenance model are not functions of* $\tilde{d}_{j,k}$ *and that* $s_j^0 \leq s_j^{init} = s_{j,t=t_0} \leq s_j^{max}$ *and* $\tilde{d}_{j,k} \geq 0, \forall \tilde{d}_{j,k} \in \mathcal{U}$*, then a feasible solution* ($\boldsymbol{x} = [x_{k,t}, \ldots], \boldsymbol{m} = [m_t, \ldots], \boldsymbol{h} = [s_t]$) *to Problem B.2 forms a feasible solution* ($\boldsymbol{x} = [x_{k,t}, \ldots], \boldsymbol{m} = [m_t, \ldots], \boldsymbol{h} = [[s_t]_0, [s_t]_k]$) *to Problem B.1 with*

$$[s_t]_0 = \begin{cases} s^{init} & t < t_{m,0} \\ s^0 & t \geq t_{m,0} \end{cases} \tag{B.3a}$$

$$[s_t]_k = \sum_{t'=t_{m,t}}^{t} x_{k,t}, \tag{B.3b}$$

where $s^{init} = s(t=0)$*,* $t_{m,0}$ *is the first point in time at which maintenance is performed, and* $t_{m,t}$ *is the most recent point in time at which maintenance was performed.*

Proof. First we show that the Inequality B.1d

$$m_t s^0 \leq [s_t]_0 + \sum [s_t]_k \tilde{d}_k \tag{B.1d}$$

holds for any $\tilde{d}_k \geq 0$ given ($\boldsymbol{x} = [x_{k,t}, \ldots], \boldsymbol{m} = [m_t, \ldots], \boldsymbol{h} = [[s_t]_0, [s_t]_k]$): From the assumption $s^0 \leq s^{init} \leq s^{max}$ and Eqn. B.3a it follows that $s^0 \leq [s_t]_0 \leq s^{max}$. Furthermore, it directly follows from Eqn. B.3b that $[s_t]_k \geq 0$. Eqn. B.1d is therefore guaranteed to hold for any $\tilde{d}_k \in \mathcal{U}, \tilde{d}_k \geq 0$.

Next, we show that Inequalities B.1f and B.1g,

$$[s_t]_0 + \sum [s_t]_k \tilde{d}_k \geq [s_{t-1}]_0 + \sum [s_{t-1}]_k \tilde{d}_k + \sum_k x_{k,t} \tilde{d}_k + m_t(s^0 - s^{max}) \tag{B.1f}$$

and

$$[s_t]_0 + \sum [s_t]_k \tilde{d}_k \leq [s_{t-1}]_0 + \sum [s_{t-1}]_k \tilde{d}_k + \sum_k x_{k,t} \tilde{d}_k \tag{B.1g}$$

respectively, hold for any $\tilde{d}_k \in \mathcal{U}$ as long as Inequality B.1e is satisfied. We first assume that $m_t = 0$. In this case $[s_t]_k = [s_{t-1}]_k + x_{k,t}$ (from Eqn. B.3b), $[s_t]_0 = [s_{t-1}]_0$ (from Eqn. B.3a) and Eqns. B.1f and B.1g simplify to

$$[s_t]_0 \geq [s_{t-1}]_0$$

and

$$[s_t]_0 \leq [s_{t-1}]_0,$$

which is true for any $\tilde{d}_k$. Next we assume that $m_t = 1$. In this case $[s_t]_0 = s^0$ (from Eqn. B.3a), $[s_t]_k = 0$ (from Eqn. B.3b), and $x_{k,t} = 0$ (assuming that a unit can not be operated while maintenance is performed). Substituting this into Eqns. B.1f and B.1g and rearranging yields

$$[s_{t-1}]_0 + \sum [s_{t-1}]_k \tilde{d}_k \leq s^{max} \tag{B.4}$$

and

$$s^0 \leq [s_{t-1}]_0 + \sum [s_{t-1}]_k \tilde{d}_k \tag{B.5}$$

respectively. Eqn. B.4 is guaranteed to be satisfied as long as Inequality B.1e holds for $t = t-1$ and Eqn. B.5 holds for any $\tilde{d}_k \in \mathcal{U}, \tilde{d}_k \geq 0$ since $[s_{t-1}]_0 \geq S_0$ and $[s_{t-1}]_k \geq 0$.

Finally we show that the Inequality B.1e

$$[s_t]_0 + \sum [s_t]_k \tilde{d}_k \leq s^{max} + m_t(s^0 - s^{max}) \tag{B.1e}$$

holds for any $\tilde{d}_k \in \mathcal{U}$. To this end we notice that

$$\arg\max_{\tilde{d}_k \in \mathcal{U}} [s_t]_0 + \sum_k [s_t]_k \tilde{d}_k = d_k^{max}$$

since $[s_t]_k \geq 0$. Therefore, if Eqn. B.1e holds for $\tilde{d}_k = d_k^{max}$, it holds for any $\tilde{d}_k \in \mathcal{U}$. Since $(\boldsymbol{x} = [x_{k,t}, \ldots], \boldsymbol{m} = [m_t, \ldots], \boldsymbol{h} = [s_t])$ is a solution to Problem B.2,

$$s_t \leq s^{max} + m_t(s^0 - s^{max}). \tag{B.6}$$

Lastly, noticing that the definition of $[s_t]_0$ and $[s_t]_k$ (Eqns. B.3a and B.3b) ensure that s_t can always be decomposed as

$$s_t = [s_t]_0 + \sum_k [s_t]_k d_k^{max}$$

if s_t satisfies Problem B.2, it follows that

$$s_t = [s_t]_0 + \sum_k [s_t]_k d_k^{max} \leq s^{max} + m_t(s^0 - s^h max)$$

and Eqn. B.1e holds for all $\tilde{d}_k \in \mathcal{U}$. □

Appendix C

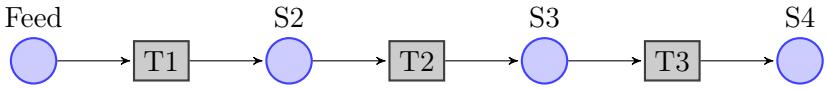
Instances of the STN

States	Feed A	Feed B	Feed C	Hot A	Int. BC	Int. AB	Impure E	Prod. 1	Prod. 2
Capacity [kg]	∞	∞	∞	100	200	150	100	∞	∞
Initial [kg]	∞	∞	∞	0	0	0	0	0	0
Storage cost	0	0	0	1	1	1	1	5	5

Units	Heater	Reactor 1	Reactor 2	Still
v_j^{min} [kg]	40	32	20	80
v_j^{max} [kg]	100	80	50	200
s_j^{max}	80	150	160	100
s_j^{init}	30	50	120	40
τ_j [hr]	15	21	24	15
c_j^{maint}	300	900	2000	1200
c_j^{f}	2000	3000	3000	1500

Task	Mode	Unit							
		Heater		Reactor 1		Reactor 2		Still	
		$p_{i,j,k}$	$\bar{d}_{i,j,k}/\sigma_{i,j,k}$	p	$\bar{d}/\sigma$	p	$\bar{d}/\sigma$	p	$\bar{d}/\sigma$
Heating	Slow	9	1/0.27						
	Normal	6	2/0.54						
	Fast	3	3/0.81						
Reaction 1	Slow			27	4/1.08	30	4/1.08		
	Normal			15	5/1.35	18	5/1.35		
	Fast			8	8/2.43	12	10/2.7		
Reaction 2	Slow			36	1/0.27	33	2/0.54		
	Normal			21	3/0.81	18	4/1.08		
	Fast			15	5/1.35	12	4/1.08		
Reaction 3	Slow			30	3/0.81	24	2/0.54		
	Normal			18	7/1.89	21	5/1.35		
	Fast			6	8/2.16	12	9/2.43		
Separation	Slow							15	2/0.54
	Normal							9	5/1.35
	Fast							6	6/1.62

Period	Scenario					
	Low		Average		High	
	Product 1	Product 2	Product 1	Product 2	Product 1	Product 2
1	76	136	150	200	190	294
2	116	162	88	150	231	323
3	101	115	125	197	198	307
4	91	141	67	296	217	335
5	60	147	166	191	181	293
6	60	103	203	193	244	328
7	54	148	90	214	243	326
8	110	113	224	294	182	296
9	92	105	174	247	246	296
10	99	175	126	313	189	348
11	51	177	66	226	199	319
12	117	164	119	121	222	346

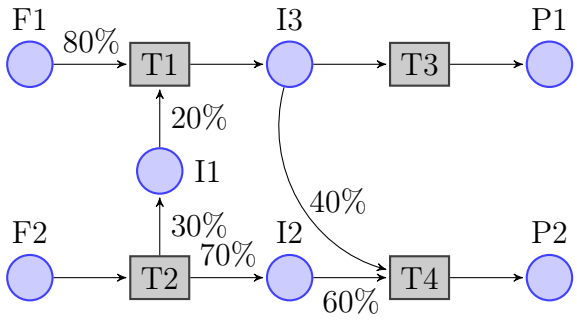


States	S1	S2	S3	S4
Capacity [kg]	∞	∞	∞	∞
Initial [kg]	∞	0	0	0
Storage cost	0	1	1	1

Units	U1	U2	U3	U4	U5
v_j^{min} [kg]	0	0	0	0	0
v_j^{max} [kg]	100	150	200	150	150
s_j^{max}	120	100	150	90	80
s_j^{init}	10	70	45	60	30
t_j [hr]	21	15	18	9	13
c_j^{maint}	600	600	500	400	400

Task	Mode	Unit									
		U1		U2		U3		U4		U5	
		$p_{i,j,k}$	$\bar{d}_{i,j,k}/\sigma_{i,j,k}$	p	$\bar{d}/\sigma$	p	$\bar{d}/\sigma$	p	$\bar{d}/\sigma$	p	$\bar{d}/\sigma$
T1	Slow	33	3/0.66	30	3/0.66						
T1	Normal	25	5/1.35	21	6/1.62						
T1	Fast	15	7/2.17	12	9/2.79						
T2	Slow					24	4/0.88				
T2	Normal					18	6/1.62				
T2	Fast					12	10/3.1				
T3	Slow							21	2/0.44	18	2/0.44
T3	Normal							15	4/1.08	12	4/1.08
T3	Fast							9	7/2.17	6	6/1.86

Period	Scenario			Period	Scenario		
	Low	Average	High		Low	Average	High
1	725	1167	2002	13	446	947	1847
2	587	1110	2141	14	201	1426	2178
3	397	1087	1668	15	305	1090	2159
4	558	906	1977	16	447	1040	2015
5	411	1188	1692	17	378	917	1662
6	678	1191	1805	18	566	1190	1782
7	252	1436	2007	19	409	953	1646
8	415	1020	2174	20	797	1109	2135
9	539	1110	1713	21	605	1298	2113
10	414	1266	2162	22	413	1275	1760
11	214	1042	2155	23	550	1364	1958
12	612	1169	2018	24	362	1158	1779

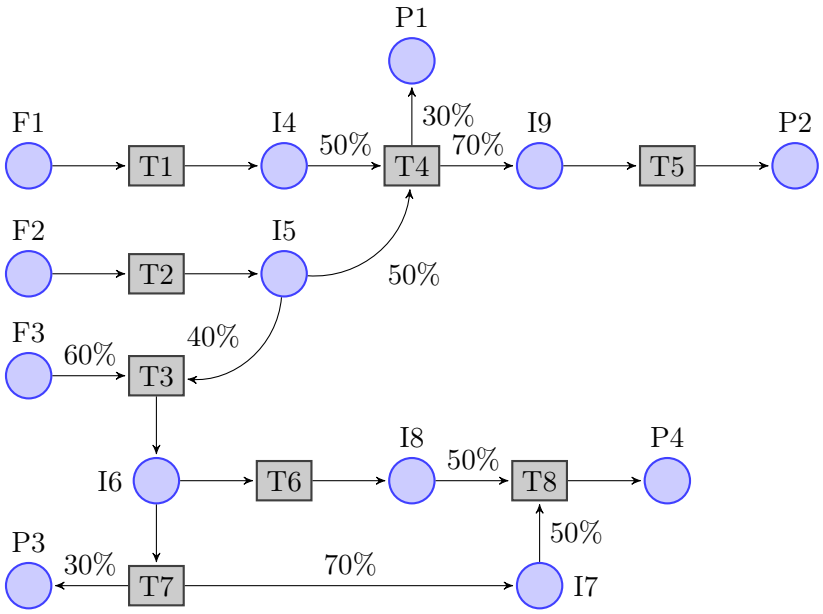


States	F1	F2	I1	I2	I3	P1	P2
Capacity [kg]	∞	∞	200	100	500	1000	1000
Initial [kg]	∞	∞	0	0	0	0	0
Storage cost	0	0	1	1	1	5	5

Units	R1	R2	R3
v_j^{min} [kg]	40	25	40
v_j^{max} [kg]	80	50	80
s_j^{max}	70	120	70
s_j^{init}	10	10	10
τ_j [hr]	21	21	21
c_j^{maint}	600	600	600

Task	Mode	Unit					
		R1		R2		R3	
		$p_{i,j,k}$	$\bar{d}_{i,j,k}/\sigma_{i,j,k}$	p	$\bar{d}/\sigma$	p	$\bar{d}/\sigma$
T1	Slow	24	3/0.66	24	3/0.66		
T1	Normal	15	5/1.35	15	5/1.35		
T1	Fast	9	8/2.16	9	8/2.16		
T2	Slow	36	3/0.66	36	3/0.66		
T2	Normal	24	5/1.35	24	5/1.35		
T2	Fast	15	8/2.16	15	8/2.16		
T3	Slow					12	3/0.66
T3	Normal					9	5/1.35
T3	Fast					6	8/2.16
T4	Slow					24	3/0.66
T4	Normal					15	5/1.35
T4	Fast					9	8/2.16

Period	Scenario						Period	Scenario					
	Low		Average		High			Low		Average		High	
	P1	P2	P1	P2	P1	P2		P1	P2	P1	P2	P1	P2
1	190	102	271	231	311	305	13	197	192	213	280	314	376
2	172	156	230	282	381	347	14	120	100	257	270	370	392
3	102	103	274	226	381	321	15	180	115	286	223	335	370
4	130	172	289	281	310	310	16	178	186	201	281	386	358
5	130	104	270	212	317	371	17	135	138	288	289	398	309
6	174	192	205	205	339	328	18	140	115	284	253	304	396
7	167	194	260	248	379	348	19	128	104	200	273	334	351
8	185	175	259	282	317	392	20	155	179	275	210	326	300
9	179	180	211	233	300	387	21	158	120	298	253	315	397
10	131	120	261	292	346	326	22	160	186	252	284	301	396
11	104	196	271	212	364	364	23	125	105	223	220	396	378
12	104	188	202	289	309	346	24	187	134	205	285	316	393



States	F1	F2	F3	I4	I5	I6	I7	I8	I9	P1	P2	P3	P4
Capacity [kg]	∞	∞	∞	1000	1000	1500	2000	1000	3000	∞	∞	∞	∞
Initial [kg]	∞	∞	∞	0	0	0	0	0	0	0	0	0	0
Storage cost	0	0	0	1	1	1	1	1	1	5	5	5	5

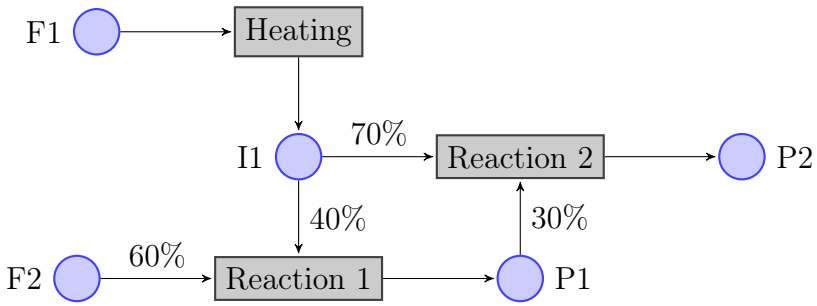
Units	R1	R2	R3	R4	R5	R6
v_j^{min} [kg]	0	0	0	0	0	0
v_j^{max} [kg]	1000	2500	3500	1500	1000	4000
s_j^{max}	100	100	100	100	100	100
s_j^{init}	77	80	90	17	40	33
τ_j [hr]	21	21	21	21	21	21
c_j^{maint}	1000	1700	2000	1200	1000	2100

Task	Mode	Unit											
		R1		R2		R3		R4		R5		R6	
		$p_{i,j,k}$	$\bar{d}_{i,j,k}/\sigma_{i,j,k}$	p	$\bar{d}/\sigma$	p	$\bar{d}/\sigma$	p	$\bar{d}/\sigma$	p	$\bar{d}/\sigma$	p	$\bar{d}/\sigma$
T1	Slow	18	3/0.66										
T1	Normal	12	5/1.35										
T2	Slow							24	3/0.66				
T2	Normal							15	5/1.35				
T3	Slow			33	5/0.66								
T3	Normal			21	8/1.35								
T4	Slow					42	5/0.66						
T4	Normal					21	11/1.35						
T5	Slow											45	7/0.66
T5	Normal											30	10/1.35
T6	Slow									18	3/0.66		
T6	Normal									12	5/1.35		
T7	Slow			33	6/0.66								
T7	Normal			21	9/1.35								
T8	Slow											45	6/0.66
T8	Normal											30	10/1.35

Table C.4: Instance P6 [4]

Period	Scenario											
	Low				Average				High			
	P1	P2	P3	P4	P1	P2	P3	P4	P1	P2	P3	P4
1	715	501	801	933	1277	1356	1739	1349	1605	1844	1898	1631
2	593	878	888	739	1533	1361	1384	1374	1687	1882	1650	1732
3	743	995	817	563	1727	1400	1323	1351	1510	1805	1893	1699
4	620	963	636	698	1702	1701	1260	1605	1557	1717	1693	1908
5	991	612	535	686	1521	1424	1600	1315	1929	1769	1657	1918
6	919	819	914	626	1451	1412	1667	1406	1539	1818	1676	1521
7	648	799	920	549	1447	1471	1732	1620	1999	1663	1632	1603
8	609	728	969	925	1746	1444	1694	1512	1673	1987	1924	1742
9	741	575	604	814	1691	1456	1384	1305	1826	1559	1982	1882
10	968	682	853	532	1436	1297	1564	1356	1987	1737	1598	1910
11	624	789	816	728	1357	1730	1441	1638	1696	1660	1761	1778
12	840	929	700	733	1384	1283	1461	1423	1779	1722	1558	1655
13	516	790	705	743	1387	1594	1696	1533	1596	1528	1857	1745
14	556	643	974	890	1722	1493	1528	1533	1782	1829	1994	1512
15	940	715	797	638	1470	1377	1635	1303	1949	1738	1933	1782
16	894	896	693	853	1563	1467	1526	1565	1837	1593	1938	1852
17	792	994	509	647	1561	1593	1614	1531	1694	1869	1879	1593
18	725	718	856	789	1739	1681	1373	1255	1902	1663	1814	1953
19	820	886	971	531	1576	1706	1635	1628	1875	1988	1648	1512
20	950	788	580	859	1627	1358	1469	1694	1642	1519	1999	1595
21	641	773	681	877	1257	1433	1581	1420	1890	1942	1854	1826
22	793	963	950	634	1250	1641	1644	1404	1795	1628	1658	1961
23	504	741	531	671	1472	1617	1311	1519	1967	1768	1877	1739
24	830	529	819	594	1390	1645	1632	1614	1844	1945	1620	1563

Table C.5: Instance P6 [4] continued



States	F1	F2	I1	P1	P2
Capacity [kg]	∞	∞	∞	∞	∞
Initial [kg]	∞	∞	0	0	0
Storage cost	0	0	15	9	5

Units	Heater	Reactor
v_j^{min} [kg]	40	30
v_j^{max} [kg]	100	140
s_j^{max}	80	120
s_j^{init}	43	30
τ_j [hr]	2	3
c_j^{maint}	300	300

Task	Mode	Unit			
		Heater		Reactor	
		$p_{i,j,k}$	$\bar{d}_{i,j,k}/\sigma_{i,j,k}$	p	$\bar{d}/\sigma$
Heating	Slow	9	5.5/1.49		
Heating	Normal	6	11/2.97		
Reaction 1	Slow			6	7/1.89
Reaction 1	Normal			4	9/2.43
Reaction 2	Slow			10	5/1.35
Reaction 2	Normal			6	13/3.51

Period	Scenario						Period	Scenario					
	Low		Average		High			Low		Average		High	
	P1	P2	P1	P2	P1	P2		P1	P2	P1	P2	P1	P2
1	52	61	121	113	184	201	13	49	45	141	128	180	225
2	60	32	137	149	228	207	14	29	44	115	120	210	192
3	42	59	149	128	204	218	15	69	35	148	106	220	192
4	51	40	146	127	192	200	16	53	51	125	148	224	225
5	24	42	105	103	206	225	17	38	49	102	136	229	218
6	34	52	111	100	194	212	18	47	77	112	125	204	201
7	32	55	105	143	223	216	19	54	53	141	134	197	226
8	30	62	141	140	182	218	20	26	77	118	131	223	215
9	65	30	144	128	185	197	21	61	68	116	128	199	183
10	32	69	147	122	218	180	22	31	71	132	101	212	196
11	59	37	119	119	182	197	23	53	42	107	121	181	222
12	24	44	133	139	218	195	24	39	47	113	143	220	211

Table C.6: Toy instance

Except for the toy instance, all instances are taken from the benchmark collection by Lappas and Gounaris [76]. All parameters were converted to a discrete time formulation and degradation parameters were added. For Instance P1 [1] parameters from Biondi et al. [70] were used. The time steps and horizons used in each instance are given in Table C.7.

Instance	Toy	P1 [1]	P2 [2]	P4 [3]	P6 [4]
T_S	30	168	168	168	168
Δt_S	1	3	3	3	3
T_P	720	4032	4032	4032	4032
Δt_P	30	168	168	168	168

Table C.7: Time horizons of STN instances

The degradation of all units is assumed to follow a Wiener process. The distribution of increments is

$$S_{j,t+p_{i,j,k}} - S_{j,t} = D_{i,j,k}, \quad D_{i,j,k} \sim \mathcal{N}\left(\bar{d}_{i,j,k}, \sigma^2_{i,j,k}\right),$$

where $\bar{d}_{i,j,k}$ is the nominal amount of degradation when task i is performed on unit j in mode k. When no task is being processed the degradation signal is assumed to vary with zero mean and a small variance:

$$S_{j,t+\Delta t} - S_{j,t} = D_{0,\Delta t}, \quad D_{0,\Delta t} \sim \mathcal{N}\left(0, 0.05^2 \Delta t\right). \tag{C.1}$$

A detailed list of all parameter values used in each case study can be found in Tables C.1 through C.6.

Appendix D

Crossing probabilities of a Brownian motion for a piecewise linear boundary

When the Wiener process is used as a degradation model, the failure probability p_j^f can be calculated efficiently based on analytical result [90, 91, 176]. The probability of a Wiener process with piecewise constant parameters $\boldsymbol{\theta_{j,k}} = [\mu_{j,k}, \sigma_{j,k}]$ crossing a fixed threshold s_j^{max} is equivalent to the probability of a standard Brownian motion $W(t)$ (a Wiener process with $\mathcal{N}(0,1)$ distributed increments) crossing a piecewise linear boundary. The probability of $W(t)$ crossing a linear boundary $at + b$ is known to be inverse gaussian distributed

$$P(W(t) \geq at + b, t \leq T) = 1 - \Phi(\frac{aT + b}{\sqrt{T}}) + \exp^{-2ab} \Phi(\frac{aT - b}{\sqrt{T}}), \tag{D.1}$$

where $\Phi(\cdot)$ is the standard normal distribution function [177].

Based on this, the probability of failure $p_j^{u,v}$ between two consecutive maintenance times $t_{m,u}$ and $t_{m,v}$ can be calculated as

$$\begin{aligned} p_j^{u,v} &= 1 - \mathbb{E}h(\boldsymbol{y}) \\ &= 1 - \prod_{l=1}^{n} \mathbb{1}(y_l > 0) \left(1 - \exp\left[-\frac{2y_{l-1}y_l}{t_l - t_{l-1}}\right]\right), \end{aligned}$$

where $t_l, l \in \{1, \ldots, n\}$ are the n points in time between $t_{m,u}$ and $t_{m,v}$ at which the operating mode k changes [90].

Here $\boldsymbol{y}$ is a vector representing the values of $W(t)$ at each t_l. It is defined as

$$\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{M}\boldsymbol{D^{1/2}}\boldsymbol{u},$$

where $\boldsymbol{M}$ is a lower triangular matrix of ones,

$$\boldsymbol{D^{1/2}} = \mathrm{diag}(\sqrt{t_1 - t_{m,u}}, \sqrt{t_2 - t_1}, \ldots, \sqrt{t_{m,v} - t_n}),$$

$\boldsymbol{u}$ is a random vector with $u_l \sim \mathcal{N}(0, \sigma^2_{j,k}(t_l))$, and $\boldsymbol{c}$ is the piecwise linear boundary

$$\boldsymbol{c} = (s_j^{max} - s_j^{init}) \cdot [1, \ldots, 1]^\top - \boldsymbol{M}\mathrm{diag}\left(0, \mu_{j,k}\left(t_1\right), \ldots, \mu_{j,k}\left(t_n\right)\right)\boldsymbol{\Delta t}$$

with

$$\boldsymbol{\Delta t} = \left[0, t_1 - t_0, \ldots, t_n - t_{n-1}\right]^\top.$$

The overall probability of failure over the evaluation horizon T can then be calculated as

$$p_j^f = 1 - \prod_{u=1}^{n+1}\left(1 - p_j^{u-1,l}\right), \tag{D.2}$$

where $t_{m,0} = 0$, $t_{m,n+1} = T$, and $t_{m,u}, u \in \{1, \ldots, n\}$ are the n points in time at which maintenance is carried out on unit j in the evaluation horizon.

Appendix E

Nomenclature

$\boldsymbol{h}$	health variables
$\boldsymbol{m}$	maintenance variables
$\boldsymbol{x}$	process variables

Indices

i	task
j	unit
k	operating mode
s	state
t	time

Sets

I	set of tasks
I_j	set of tasks i available on unit j
I_s	set of tasks i consuming state s
$\bar{I}_s$	set of tasks i producing state s
J	set of process units
J_i	set of units j on which task i can be performed
K_i	set of operating modes allowed for task i
K_j	set of operating modes k available on unit j

T	evaluation horizon
T_P	planning horizon
T_S	scheduling horizon
$\mathcal{U}$	uncertainty set
$\mathcal{X}$	set of operating mode sequences $\boldsymbol{x}_j^k$

Discrete Variables

$m_{j,t}$	1 if maintenance is performed on unit j at time t
$n_{i,j,k,t}$	number of times task i is performed on unit j in mode k in time period t
$w_{i,j,k,t}$	1 if task i starts on unit j in mode k at time t, 0 otherwise
$x_{j,k,t}$	1 if unit j is operated in mode k at time t
$\boldsymbol{x}_j^k$	sequence of operating modes $[k_1, k_2, \ldots, k_T]$
$\omega_{j,k,t}$	1 if unit j operates in mode k in period t, 0 otherwise

Continuous Variables

$a_{i,j,k,t}$	amount of material processed by task i in unit j in mode k in time period t
$b_{i,j,k,t}$	amount of material committed to task i on unit j in mode k at time t
c^*	minimal cost determined by solving Problem 3.9
c_j^f	cost of unit j failing
D	random variable modeling increment of $S(t)$
$N_{j,k}$	random variable modeling $n_{j,k}$
$n_{j,k}$	number of times mode k occurs on unit j in a given Δt
p_j^f	probability of failure
$\bar{p}_j^f$	estimated upper bound on p_j^f
q_s^{fin}	quantity of state s stored at end of planning horizon
$q_{s,t}$	quantity of state s stored at time t
$S(t)$	stochastic process modeling $s^{meas}(t)$
$s_{j,t}^n$	realization of $S_j(t)$ at time t
s_j^{fin}	value of degradation signal at end of planning horizon
$s^{meas}(t)$	measured degradation signal

S_t	random variable modeling $s^{meas}(t)$ at time t
$X_j^k(t)$	memoryless Markov chain modeling $\boldsymbol{x}_j^k$
$X_{j,t}^k$	state of Markov chain at time t
ϕ_s^d	slack variable for unfulfilled demand of state s
$\phi_{s,t}^q$	slack variable for storage capacity violation of state s at time t
$\boldsymbol{\psi}$	process/environmental parameters

Parameters

c_j^{maint}	cost of maintenance for unit j
$c_s^{storage}$	per unit cost of storage for state s
c_s	storage capacity of state s
$\mathcal{D}$	distribution of D
$\bar{d}_{j,k}$	nominal value of $\tilde{d}_{j,k}$
$\tilde{d}_{j,k}$	uncertain parameter modeling increment of $S(t)$
$d_{j,k}^{max}$	maximum of $\tilde{d}_{j,k}$ in $\mathcal{U}$
N	number of samples in Monte-Carlo simulation
$p_{i,j,k}$	processing time of task i on unit j in mode k
$r_{j,t}$	residual lifetime of unit j at time t
$[s_{j,t}]_0, [s_{j,t}]_k$	parameters for affine decision rule
s^0	reset value degradation signal
s^{init}	initial value of degradation signal
s^{max}	failure threshold degradation signal
$\bar{t}_P$	first time period in planning horizon
$\bar{t}_S$	last time period in scheduling horizon
Δt_P	length of planning period
Δt_S	length of scheduling period
U	large number
$v_{i,j}^{max}$	maximum batch size for task i on unit j
$v_{i,j}^{min}$	minimum batch size for task i on unit j
α	size parameter of $\mathcal{U}$

$\delta_{s,t}$	demand for s at time t
$\epsilon_{j,k}$	size parameter of $\mathcal{U}$
$\eta_{j,k}$	probability of k occuring $n_{j,k}$ times
$\boldsymbol{\theta}$	parameter vector of $\mathcal{D}$
$\mu_{j,k}$	mean of $\mathcal{D}_{j,k}$
$\pi_{k,k*}$	transition probability Markov chain
$\bar{\rho}_{i,s}$	fraction of state s of material produced by task i
$\rho_{i,s}$	fraction of state s of material consumed by task i
$\sigma_{j,k}$	standard deviation of $\mathcal{D}_{j,k}$
τ_j	duration of maintenance on unit j

Appendix F

Connection to uncertain functions

Consider the following robust optimization problem:

$$\min_{(\boldsymbol{x},\boldsymbol{z})\in\mathcal{X}} \quad f(\boldsymbol{x},\boldsymbol{z}) \tag{F.1a}$$

$$\text{s.t} \quad \sum_{i=1}^{n} \tilde{g}(\boldsymbol{z}_i)x_i \leq b \qquad \forall \tilde{g} \in \mathcal{U}^g \tag{F.1b}$$

$$\boldsymbol{z}_i \in \mathbb{R}^k, k \leq n. \tag{F.1c}$$

Instead of uncertain parameters, Problem (F.1) considers an uncertainty set $\mathcal{U}^g$ over uncertain functions $\tilde{g}(\cdot)$. We are interested in defining $\mathcal{U}^g$ in a way that it contains "likely" realizations of the GP.

Recall that for any finite set of points $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l$, $l \in \mathbb{N}$, $G_{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_l} = [G(\boldsymbol{z}_1), \ldots, G(\boldsymbol{z}_l)]^\intercal$ is a multivariate Gaussian with mean $\boldsymbol{\mu}(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l)$ and covariance matrix $\Sigma(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l)$. For any such $G_{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_l}$, we can construct a confidence ellipsoid $\mathcal{E}^\alpha(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l)$ containing the true values $[g(\boldsymbol{z}_1), \ldots, g(\boldsymbol{z}_l)]^\intercal$ with probability $1 - \alpha$:

$$\mathcal{E}^\alpha(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l)_l = \left\{ \begin{array}{l} \boldsymbol{y} \in \mathbb{R}^l \\ \text{s.t. } (h(\boldsymbol{y}) - \boldsymbol{\mu})^\intercal \Sigma^{-1} (h(\boldsymbol{y}) - \boldsymbol{\mu}) \\ \qquad \leq F_l^{1-\alpha} \end{array} \right\},$$

where $F_l^{1-\alpha} = F_l(1-\alpha)$ is the cumulative distribution function of the χ^2 distribution with l degrees of freedom. We then construct a set $\mathcal{U}^g$ over functions $\tilde{g}(\cdot)$ for which $[\tilde{g}(\boldsymbol{z}_1), \dots, \tilde{g}(\boldsymbol{z}_l)]$ lies in the corresponding α-confidence ellipsoid $\mathcal{E}^\alpha(\boldsymbol{z}_1, \dots, \boldsymbol{z}_l)_l$ for any $l \in \mathbb{N}$ and $\boldsymbol{z}_1, \dots, \boldsymbol{z}_l$ with $\boldsymbol{z}_i \in \mathbb{R}^k$:

$$\mathcal{U}^{\mathcal{E}} = \left\{ \begin{array}{l} \tilde{g} : \mathbb{R}^k \to \mathbb{R} \text{ s.t.} \\ [\tilde{g}(\boldsymbol{z}_1), \dots, \tilde{g}(\boldsymbol{z}_l)]^\intercal \in \mathcal{E}^\alpha(\boldsymbol{z}_1, \dots, \boldsymbol{z}_l), \\ \forall \{\boldsymbol{z}_1, \dots, \boldsymbol{z}_l\}, \boldsymbol{z}_i \in \mathbb{R}^k, l \in \mathbb{N} \end{array} \right\}$$

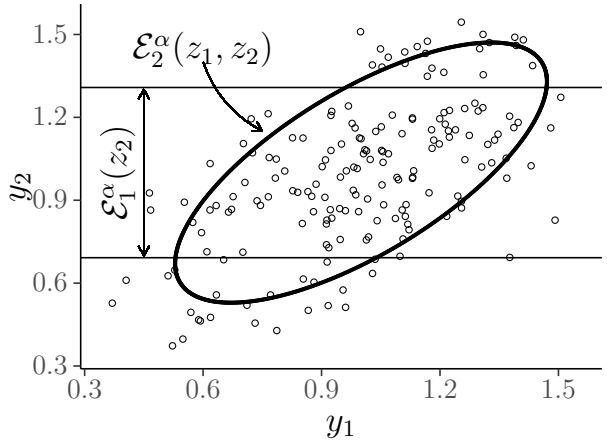


Figure F.1: Any l_1-dimensional α-confidence ellipsoid $\mathcal{E}_{l_1}^\alpha$ is a strict subset of the projection of higher order α-confidence ellipsoids $\mathcal{E}_{l_2}^\alpha,\ l_2 > l_1$ onto the l_1-dimensional space.

Replacing $\mathcal{U}^g$ with $\mathcal{U}^{\mathcal{E}}$ transforms Problem (F.1) into a robust optimization problem with an uncertainty set over functions defined by an infinite number of confidence ellipsoids which can have arbitrarily many dimensions. This set is not semialgebraic and it is not clear how it could practically be used in optimization. In practice, however, we are only interested in evaluating the GP at a finite number of points. Here, the number of evaluation points is the number of times $|S|$ that the GP occurs in the optimization problem. Consider the following robust optimization problem:

$$\min_{(\boldsymbol{x},\boldsymbol{z}) \in \mathcal{X}} \quad f(\boldsymbol{x}, \boldsymbol{z}) \tag{F.2a}$$

$$\text{s.t} \quad \boldsymbol{y}^\intercal \boldsymbol{x} \leq b \qquad \forall \boldsymbol{y} \in \mathcal{E}^\alpha(\boldsymbol{z}) \tag{F.2b}$$

$$\boldsymbol{z}_i \in \mathbb{R}^k, k \leq n. \tag{F.2c}$$

Theorem F.1. *A vector $\boldsymbol{x}^*$ which is a feasible solution to Problem (F.2) is also a feasible solution to Problem (F.1).*

Proof. Assume $\boldsymbol{x}^*$ is a solution to Problem (F.2) but not to Problem (F.1). Then $\exists \hat{g} \in \mathcal{U}^g$ s.t. $\sum_{i \in S} \hat{g}(\boldsymbol{z}_i^*) x_i^* > 0$. The definition of $\mathcal{U}^g$ implies that $[\hat{g}(\boldsymbol{z}_i^*) : i \in S]^\intercal \in \mathcal{E}^\alpha(\boldsymbol{z}_i^* : i \in S)$. Choosing $\hat{\boldsymbol{y}} = [\hat{g}(\boldsymbol{z}_i^*) : i \in S]^\intercal$, it follows that $\sum_{i \in S} \hat{y}_i x_i^* = \sum_{i \in S} \hat{g}(\boldsymbol{z}_i^*) x_i^* > 0$, meaning that $\{\boldsymbol{x}^*, \hat{\boldsymbol{y}}\}$ is not feasible in Problem (F.2). But $\hat{\boldsymbol{y}} \in \mathcal{E}^\alpha(\boldsymbol{z}_i^* : i \in [n])$, which is a contradiction. □

Fig. (F.1) shows that the converse of Theorem (F.1) is not necessarily true. Because all confidence ellipsoids are symmetric and centered at the mean of the distribution, any lower dimensional ellipsoid $\mathcal{E}_l^\alpha = \mathcal{E}^\alpha(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l), l < n$ is a strict subset of the projection of $\mathcal{E}_n^\alpha = \mathcal{E}^\alpha(\boldsymbol{z})$ onto the l-dimensional space (otherwise it would have to contain a larger probability mass). Problem (F.2) therefore conservatively approximates Problem (F.1). Furthermore, the α-confidence ellipsoid $\mathcal{E}^\alpha(\boldsymbol{z})$ implies that a solution to Problem (F.2) is a feasible solution to the black-box constrained problem with a probability of at least $1 - \alpha$ (see Theorem 1).

Appendix G

Globally optimizing non-convex inner maximization problems

Lemma G.2. *Let $\mathcal{Y}$ be the set of solution of Problem 4.10, then $\exists \boldsymbol{y}^* \in \mathcal{Y}$ which is on the boundary of $\mathcal{U}$, i.e., $\boldsymbol{y}^* \in \partial\mathcal{U}$.*

Proof. When $\boldsymbol{x} = 0$ the objective function is constant and the Lemma is trivially true. For $\boldsymbol{x} \neq 0$ we prove the Lemma by contradiction. For the sake of contradiction assume $\boldsymbol{y}^* \in int(\mathcal{U})$, then $\exists\, \epsilon > 0$ s.t. $\boldsymbol{y}_0 \in \mathcal{U}$, $\forall \boldsymbol{y}_0 \in \{\boldsymbol{y}_0 \mid \|\boldsymbol{y}^* - \boldsymbol{y}_0\| < \epsilon\}$. Let:

$$\hat{\boldsymbol{y}} = \boldsymbol{y}^* + \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}\frac{\epsilon}{2},$$

then:

$$\|\boldsymbol{y}^* - \hat{\boldsymbol{y}}\| = \|\boldsymbol{y}^* - \boldsymbol{y}^* - \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}\frac{\epsilon}{2}\| = \frac{\epsilon}{2} < \epsilon,$$

and therefore $\hat{\boldsymbol{y}} \in \mathcal{U}$, but:

$$\boldsymbol{x}^\intercal\hat{\boldsymbol{y}} = \boldsymbol{x}^\intercal\left(\boldsymbol{y}^* - \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}\frac{\epsilon}{2}\right) = \boldsymbol{x}^\intercal\boldsymbol{y}^* + \frac{\boldsymbol{x}^\intercal\boldsymbol{x}}{\|\boldsymbol{x}\|}\frac{\epsilon}{2} > \boldsymbol{x}^\intercal\boldsymbol{y}^*,$$

which is a contradiction. □

Lemma G.3. *The bounding box of an ellipsoid* $(\boldsymbol{x}-\boldsymbol{\mu})^\intercal\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \leq r^2$ *is given by the extreme points* $\bar{x}_i = \mu_i + r\sigma_{ii}$ *and* $\underline{x}_i = \mu_i - r\sigma_{ii}$.

Proof. Consider the optimization problem:

$$\max_{\boldsymbol{x}} x_i \tag{G.1a}$$

$$\text{s.t. } (\boldsymbol{x}-\boldsymbol{\mu})^\intercal\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) = r^2 \tag{G.1b}$$

It's stationarity condition is:

$$\boldsymbol{\delta} = 2\lambda\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}), \tag{G.2}$$

Pre-multiplying by $(\boldsymbol{x}-\boldsymbol{\mu})^\intercal$ and substituting primal feasibility leads to the expression:

$$\lambda = \frac{x_i-\mu_i}{2r^2}. \tag{G.3}$$

Substituting this back into the stationarity condition and rearranging gives:

$$\boldsymbol{x}-\boldsymbol{\mu} = \frac{r^2}{x_i-\mu_i}\Sigma\delta, \tag{G.4}$$

which, substituted into the primal constraint leads to the desired results:

$$\bar{x}_i = \mu_i + r\sigma_{ii},\ \underline{x}_i = \mu_i - r\sigma_{ii}. \tag{G.5}$$

□

Appendix H

Table of notation

	General
$\tilde{a}_i$	uncertain parameter
$F_n^{1-\alpha}$	CDF of the χ^2
u	dual variable
$\boldsymbol{x}, \boldsymbol{z}$	decision variable vectors
$\boldsymbol{z}_i$	subset of decision variables $\boldsymbol{z}$
$\boldsymbol{y}$	observation vector in original space
$f(\cdot)$	black-box objective function
$g(\cdot)$	black-box constraint
$h(\cdot)$	warping function
$K(\cdot, \cdot)$	kernel function of GP
$w(\cdot)$	constraint defining $\mathcal{U}$
$\mathcal{E}^\alpha$	α-confidence ellipsoid
$\mathcal{U}$	(warped) uncertainty set
$\mathcal{X}$	deterministic feasible set
α	probability of constraint violation
$\boldsymbol{\delta}$	disturbances vector
ϵ	estimated feasibility
ϵ_0	target feasibility

$\boldsymbol{\xi}$	observation vector in latent space
$\boldsymbol{\psi} = \{a_j, b_j, c_j\}$	parameters of warping function
$\boldsymbol{\mu}$	mean of GP at $\boldsymbol{z}_i$
σ_{ij}^2	ij-element of covariance matrix
Σ	covariance matrix of GP at $\boldsymbol{z}_i$
	Production planning
c_t	production cost in period t
$\tilde{p}_t$	uncertain price in period t
x_t	production amount in period t
	Drill scheduling
W	weight on bit
$\dot{N}$	rotational speed
V	rate of penetration
Δp	differential pressure
R	degradation indicator
M	set of maintenance depths

Appendix I

Drill scheduling model

In order to connect the penetration rate V and degradation rate r to the drilling parameters, weight-on-but W and rotational speed $\dot{N}$, we require two models:

- A *bit-rock interaction model* [151] connecting W and $\dot{N}$ with V and differential pressure across the mud motor Δp
- A *mud motor degradation model* [5] connecting the degradation rate r with the differential pressure Δp.

I.1 Detournay's bit-rock interaction model

To model the connection between W, $\dot{N}$, V, and Δp, we combine the bit-rock interaction model by Detournay et al. [151] with the PDM's powercurve. There are three relevant rotational speeds in the drilling process: The drill-string speed $\dot{N}_{top}$, the PDM speed (relative to the drill string) $\dot{N}_{PDM}$, and the drill-bit speed $\dot{N}_{bit}$:

$$\dot{N}_{bit} = \dot{N}_{top} + \dot{N}_{PDM} \tag{I.1}$$

Based on Detournay et al. [151], the following drilling response model can be formulated

relating N_{bit} with the weight-on-bit W and the rate of penetration V:

$$V = d \cdot \dot{N}_{bit} \qquad \text{[151, Eq. 4]} \tag{I.2a}$$

$$w = \frac{W}{a(1-\rho)} \qquad \text{[151, Eq. 4]} \tag{I.2b}$$

$$d = \begin{cases} \frac{w}{S^*} \\ \frac{w^*}{S^*} + \frac{w - w^*}{\xi\epsilon} \end{cases} \qquad \text{[151, Eqs. 24,37]} \tag{I.2c}$$

(I.2d)

where d is the depth of cut per revolution, w is a scaled weight-on-bit, and a, ρ, S^*, w^*, $\xi\epsilon$, N^{max}, and W^{max} are rock and equipment parameters.

The relationship between torque T and weight-on-bit W is given by:

$$t = \frac{2T}{a^2(1-\rho^2)} \qquad \text{[151, Eqn. 4]}$$

$$t = \begin{cases} \mu\gamma' w \\ \frac{1}{\xi}\left(w - (1-\beta)w^*\right) \end{cases} \qquad \text{[151, Eqns. 29,38]} \tag{I.3}$$

For the bit parameters $a = 100.4$ and $\rho = 0.0$ was used. Rock parameters are available for Lower Jurassic shale and Sandstone in the open literature [151]:

Parameter	Lower Jurassic shale	Sandstone
S^* [MPa]	278	315
w^* [N/mm]	199	59
$\xi\epsilon$ [MPa]	125	50
$\mu\gamma'$ [-]	0.48	0.93
$(1-\beta)w_{f*}$ [N/mm]	157	33
ξ [-]	0.98	0.65

Using the PDM's power curve (Fig. I.1), the bit rotational speed $\dot{N}_{\text{bit}}$ can be determined as a function of $\dot{N}$, T, and Δp. Fig. I.1 shows the relationship between T, $\dot{N}_{PDM}$, the differential

pressure over the PDM Δp, and the flow rate through the PDM $\dot{Q}$. Since torque T is specified through W (Eqn. I.3), Δp can be determined from the power curve (Fig. I.1). If additionally the flow $\dot{Q}(t)$ is given, $\dot{N}_{PDM}$ is also fully specified.

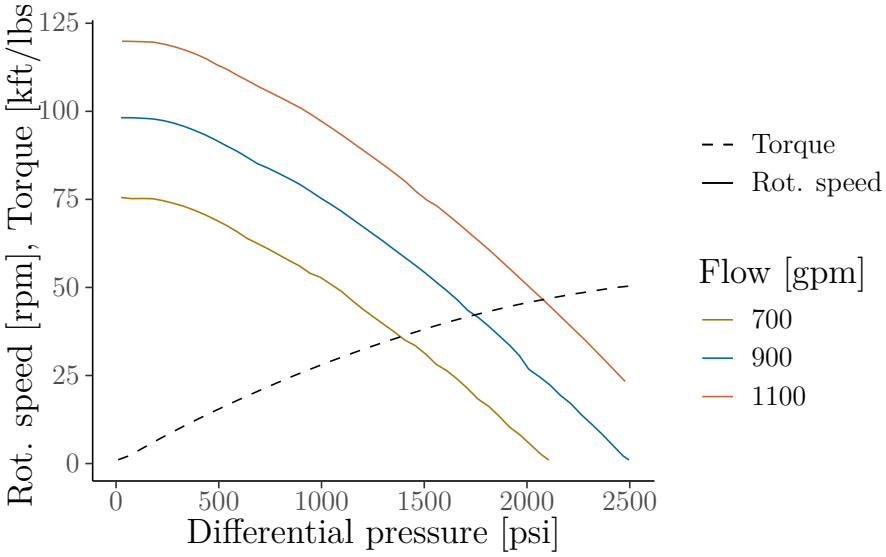


Figure I.1: Example of a PDM power curve [5].

Putting this together, we obtain the following model relating V to W and $\dot{N}$:

$$V = d\left(\dot{N}_{top} + \dot{N}_{PDM}\right) \tag{I.4a}$$

$$w = \frac{W}{a(1-\rho)} \tag{I.4b}$$

$$t = \frac{2T}{a^2(1-\rho^2)} \tag{I.4c}$$

$$t = \begin{cases} \mu\gamma' w \\ \frac{1}{\xi}\left(w - (1-\beta)w^*\right) \end{cases} \tag{I.4d}$$

$$d = \begin{cases} \frac{w}{S^*} \\ \frac{w^*}{S^*} + \frac{w-w^*}{\xi\epsilon} \end{cases} \tag{I.4e}$$

$$\dot{N}_{PDM} = f\left(T, \dot{Q}\right) \qquad \text{(from Fig. I.1)} \tag{I.4f}$$

$$\dot{N}_{top} \leq \dot{N}^{max} \tag{I.4g}$$

$$W \leq W^{max} \tag{I.4h}$$

$$\text{(safety constraints)}, \tag{I.4i}$$

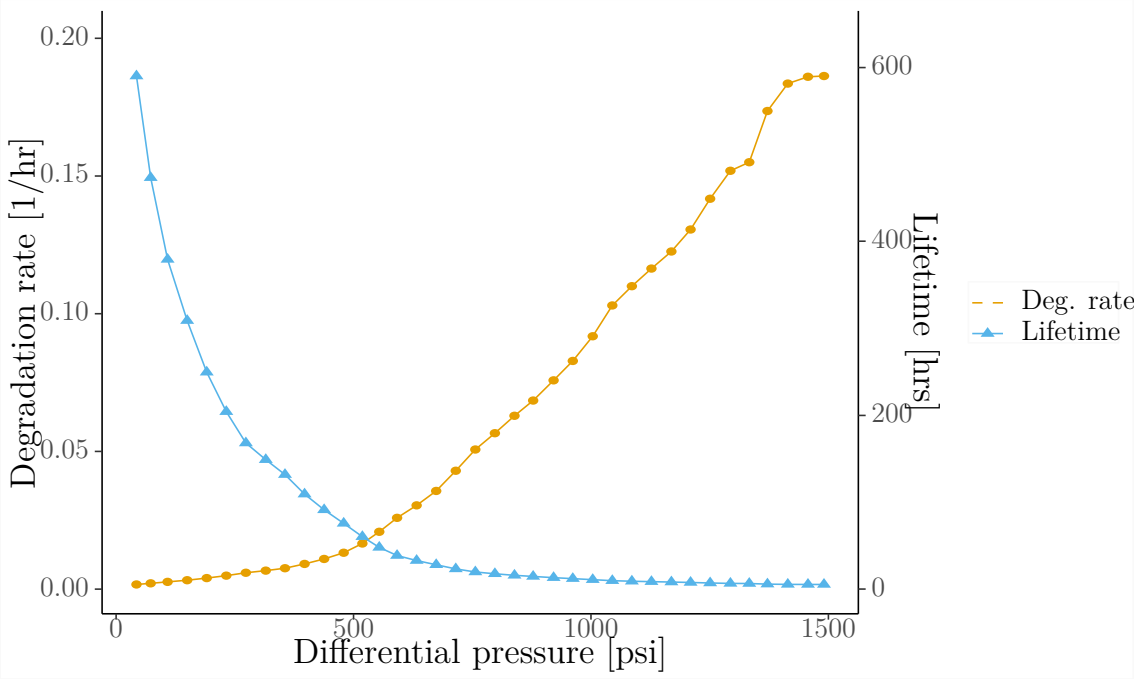


Figure I.2: Maximum lifetime of a PDM as a function of differential Δp (for a given PDM geometry and elastomer, mud, flow, and temperature) [5].

Assuming that the flow rate $\dot{Q}(t)$ is treated as a parameter, the only decision variables are $W(t)$, and $\dot{N}_{top}(t)$. For the purpose of this work we model the above power curves using quadratic equations. Notice that the variables w, t, d, and $\dot{N}_{\text{PDM}}$ could easily be eliminated, resulting in a more compact albeit less intuitive/physically meaningful formulation.

I.2 Mud motor degradation model

For the mud motor degradation characteristics we use data obtained by Ba et al. [5], determined through a combination of simulation and experiments, shown in Fig. I.2.