Maintenance Optimization for Railway Infrastructure Networks

Zhou Su

Maintenance Optimization for Railway Infrastructure Networks

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology,
by the authority of the Rector Magnificus Prof.dr. ir. T.H.J.J. van der Hagen,
chair of the Doctorate Board
to be defended publicly on
Thursday 13 September 2018 at 10:00 o'clock
by

Zhou SU,
Master of Science in Automation & Robotics, Technische Universität Dortmund, Germany
born in Chongqing, China.

This dissertation has been approved by the promotors.

Research sponsored by the NWO/ProRail project "Multi-party risk management and key performance indicator design at the whole system level (PYRAMIDS)", project 438-12-300, which is partly financed by the Netherlands Organisation for Scientific Research (NWO).

TRAIL Thesis Series T2018/6, the Netherlands TRAIL Research School
P.O. Box 5017
2600 GA Delft, The Netherlands
E: info@rstrail.nl

Published and distributed by: Zhou Su
E-mail: suzhouhust@gmail.com

ISBN 978-90-5584-238-4

Keywords: maintenance optimization, railway infrastructures, travelling salesman problem, model predictive control, reverse Stackelberg games

Acknowledgements

The last four years as a PhD candidate has been a wonder journey in my life, and I know very well that without the amazing people accompany me along the way, the destination would not be reached so easily.
First and foremost, I would like to send my greatest thanks to my daily supervisor and promoter Bart De Schutter, who offered me the opportunity to start such a wonderful journey four years ago. He is the best possible supervisor one can imagine: responsible and meticulous, while giving me enough freedom to conduct my own research at the same time. I would like to thank my copromoter Simone Baldi as well, for his effort and dedication in improving the quality of my research.
I would like to thank all the amazing people that have helped me with my research. First I would like to thank my project members Alfredo Núñez and Ali Jamshidi. As my closest collaborators, they have always answered my research questions so patiently and provided solid contribution in almost all my publications. I would like to thank my coauthor Mernout Burger, who first introduced to me the fascinating traveling salesman problem. I would like to thank Noortje Groot, my predecessor in reverse Stackelberg games. Finally, special thanks to my ProRail research coach Robert Degenhart, who has helped me so much in the practical aspects of my dissertation, including providing the cover photo.
Furthermore, I would like to thank my friends and colleagues who made the last four years so enjoyable. I would like to thank Sophie for all the lovely weekend drinks and movie nights we spent together. I would like to thank Anqi for accompanying me the 2018 Rotterdam Marathon and CDC 2016. I would like to thank Laura, who made possible the unforgettable 2016 road trip in Nevada and South California. I would like to thank Christian and Hubert for the amazing 2017 summer holiday in Südtirol. I would like to thank Sjoerd for the Dutch translation of my thesis summary. I would like to thank Reinier and Farid for all the weekly discussions on DISC homework. I would like to thank Le as my first guide in DCSC. Finally, I would like to thank Anahita and Na for being such friendly and understanding office mates.
Last but not the least, greatest thanks to my parents for all their support.

Zhou Su
Delft, August 2018

Contents

Chapter 1

Introduction

1.1 Scope and Motivation

Railway transportation has been an important means of public transportation since their birth in the sixteenth century [45]. Compared with other means of transportation, railway transportation is advantageous because of its high capacity, high reliability, high safety, high degree of automation, and low environmental impact [45].
Maintenance is crucial to guarantee the reliability, availability, and safety of a railway infrastructure network. This thesis focuses on track maintenance, which takes more than 40% of the 1 billion EURO annual maintenance budget in the Dutch railway network [175]. One important track maintenance intervention is grinding, which is applied to treat squats (see Figure 1.1), a type of rolling contact fatigue, that first appear on the rail surface and can lead to rail breakage if not treated properly. Another important track maintenance intervention is tamping, which is intended for ballast degradation (see Figure 1.2) and which repairs track irregularities by correcting track geometry parameters [69, 95].
How to plan costly track maintenance interventions without sacrificing the safety and reliability of the whole network has become the primary concern of railway infrastructure managers. In practice, railway infrastructure managers face several challenges in track maintenance planning:

- High setup costs: track maintenance interventions usually engage heavy track main-

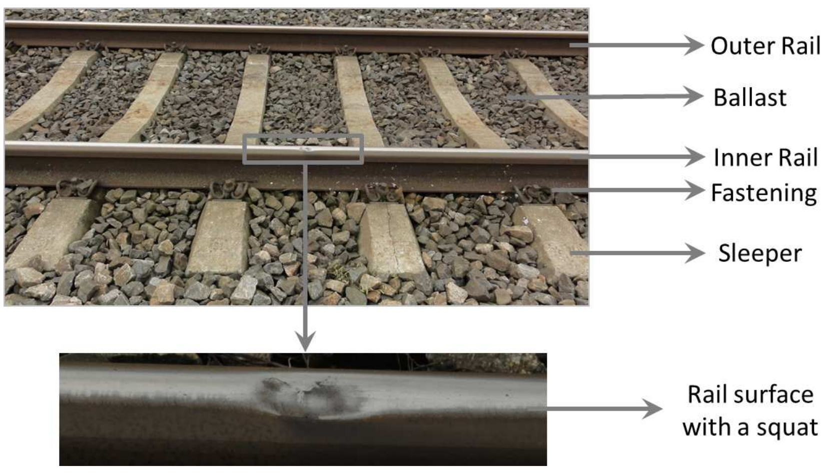


Figure 1.1: A severe squat on the rail surface.

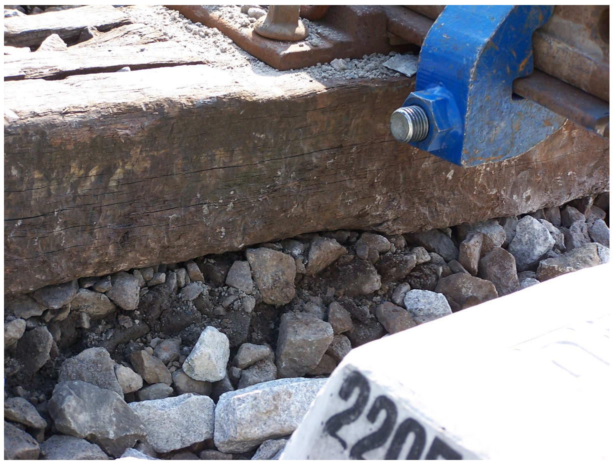


Figure 1.2: Severely worn ballast stones under an old sleeper. Compared with the new ballast stones nearby with sharp edges, the worn ballast stones have become rounded in shape and covered by dust. This problem is called the "foul" of the ballast, meaning that the ballast stones have been crushed, provide less drainage, move less, and have less elasticity, leading to a reduction in their friction dissipation.

tenance machines, which are expensive to use. For example, it takes more than 10 000 EURO to rent a grinding machine for 10 hours in the Netherlands.

- Limited resources, in terms of track possession time and machinery: for a busy network like the Dutch railway network, the time slot for track maintenance is usually less than 6 hours. The number of heavy track maintenance machines is also small (e.g. there is only one grinding machine for rails in the Netherlands).

- Negative effects on the railway infrastructure brought by track maintenance: for instance, the metal dust left after a grinding operation can cause short circuits in insulated joints.

The high setup costs, limited resources, and negative effects of maintenance interventions motivate railway infrastructure managers and maintenance contractors to search for more cost-efficient maintenance strategies, other than the traditional time-based maintenance strategies, which perform maintenance at an optimal interval of time/usage. Efficient track maintenance is especially important under limited resources. For example, due to the short maintenance time slots in the Dutch railway network, approximately 10% of the cyclic rail grinding operations cannot cover the planned range, and the uncovered part of the rail has to wait for the next maintenance cycle (at least six months) to be ground, resulting in reduced riding comfort of passengers. If the uncovered part contains many severe squats, an unplanned grinding must be performed before the next maintenance cycle in the worst case. In this thesis, the inefficiency of time-based maintenance strategies is also the motivation to adopt a condition-based maintenance strategy [1, 79], where maintenance interventions are suggested based on the information obtained from real-time condition monitoring.

1.2 Problem Statement

The main topic of this thesis is a model-based methodology for railway track maintenance optimization, with the following contributions:

- A multi-level approach that covers both long-term condition-based maintenance planning and short-term maintenance crew scheduling for railway track maintenance.
- A chance-constrained formulation to achieve a robust but non-conservative maintenance plan, in the presence of model uncertainties.
- Tractable solution methods to address the computational complexities of the optimization problems; distributed optimization schemes to improve the scalability of the proposed approach.

The second contribution is a novel 2-indexed Mixed Integer Linear Programming formulation for the Fixed-destination Multi-Depot Multiple Traveling Salesmen Problem, which serves as the basis for the railway track maintenance crew scheduling problem. The last contribution is a systematic numerical solution method to obtain the optimal nonlinear leader function for reverse Stackelberg games with incomplete information, and general, non-concave utility functions.

1.3 Structure of the Thesis

The outline of this thesis is given in Figure 1.3. The background and preliminaries of all the topics discussed in this thesis are summarized in Chapter 2. Chapter 3 and 4 deal with the main research topic, i.e. maintenance optimization of railway infrastructure networks. Both adopt a multi-level scheme that considers the long-term condition-based maintenance planning and short-term maintenance crew scheduling. The difference is that the approach developed in Chapter 3 is most suitable for small-scale railway networks, as it solves the optimal maintenance intervention planning problem in a centralized fashion, while Chapter 4 proposes a distributed optimization solution approach for the sake of scalability, in the case of large-scale railway networks. Chapter 5 presents an exact 2-indexed formulation for the Fixed-Destination Multi-depot Multiple Traveling Salesman Problem, which forms the basis of the maintenance crew scheduling problem in the multi-level maintenance optimization approach in Chapter 4. A numerical solution approach to obtain the optimal nonlinear leader function for reverse Stackelberg games with incomplete information is presented in Chapter 6. Finally, the conclusions and future work on the whole thesis are provided in Chapter 7.

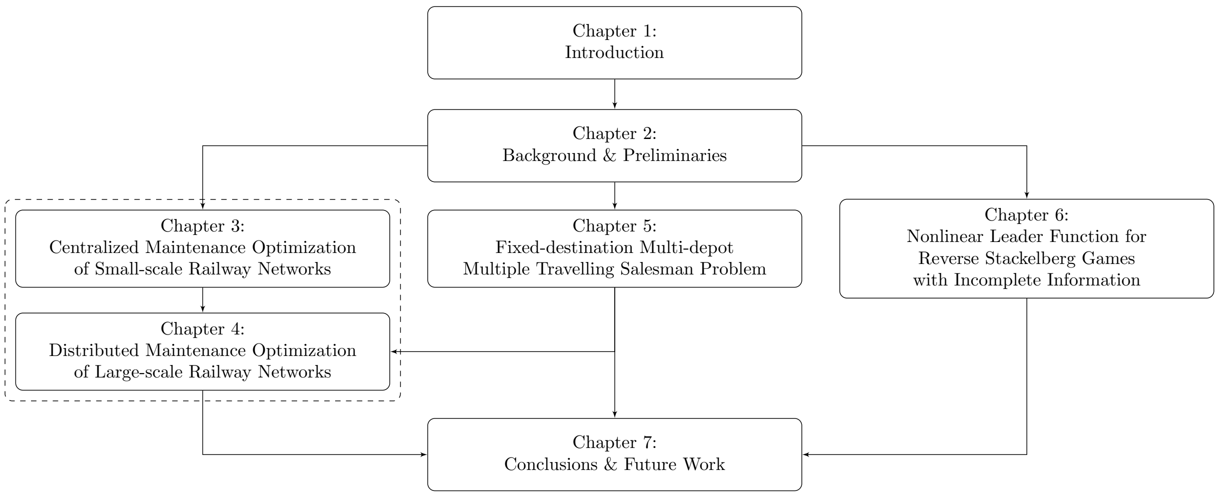

Chapter 1:
Introduction
Chapter 2:
Background & Preliminaries
Chapter 3:
Centralized Maintenance Optimization
of Small-scale Railway Networks
Chapter 4:
Distributed Maintenance Optimization
of Large-scale Railway Networks
Chapter 5:
Fixed-destination Multi-depot
Multiple Travelling Salesman Problem
Chapter 6:
Nonlinear Leader Function for
Reverse Stackelberg Games
with Incomplete Information
Chapter 7:
Conclusions & Future Work

Figure 1.3: Outline of the thesis

Chapter 2

Background and Preliminaries

2.1 Track Defects and Maintenance Interventions

Maintenance is crucial for the proper functioning and lifetime extension of a railway network, which is composed of various infrastructures with different functions. In particular, the Dutch railway network, one of the most intensively used railway networks in Europe, consists of tracks (6830 km), tunnels (5100), overhead wiring (4500 km), switches (7508), signaling systems, stations (388), and safety control systems. The degradation of an infrastructure can severely impair the performance of the whole railway network. In this thesis we focus on the optimal treatment of *track defects*. One example is a squat, a typical type of Rolling Contact Fatigue (RCF), that accelerates rail degradation, which can potentially lead to derailment if not treated properly [140]. Grinding is effective for the treatment of early-stage squats, while rail replacement would be the only solution for severe-stage squats. Another example is ballast degradation, which can affect track geometries, causing unreliable support for sleepers and potential rail buckling and derailment [69, 95]. In this case, tamping is applied to correct the track geometry.
In this section, we briefly explain generic defects for railway infrastructures and the corresponding maintenance interventions. Note that full renewal is also considered as a maintenance intervention. Figure 2.1 shows the deterioration process of a generic track defect [45], e.g. a ballast defect, for one component of a railway asset, e.g. a section of ballast. The condition of the component is represented by one single measurable factor. The natural degradation[1] is shown by the green dashed line, which will eventually hit the operational limit, potentially triggering a great hazard like derailment, if no adequate intervention is performed. An intervention is performed to improve the condition, when the predicted condition is close to the maintenance limit, which is usually much smaller than the operational limit to allow for sufficient safety margin. An intervention brings an abrupt improvement of the condition, quantified by the vertical drop of the degradation level after the intervention. Full renewal can always restore a component to an “as good as new" condition. In comparison, corrective maintenance, like tamping and grinding, is different in the sense that the level of improvement that can be achieved by a corrective maintenance becomes less the more it is applied. This is also shown in the monotonically increasing dashed red line connecting the conditions immediately after each maintenance intervention. Moreover, the deterioration also becomes faster after each maintenance intervention, demonstrated by the more

[1]The natural degradation is only assumed to be exponential in this example for demonstration purposes. In general, it can be described by any monotonically increasing function.

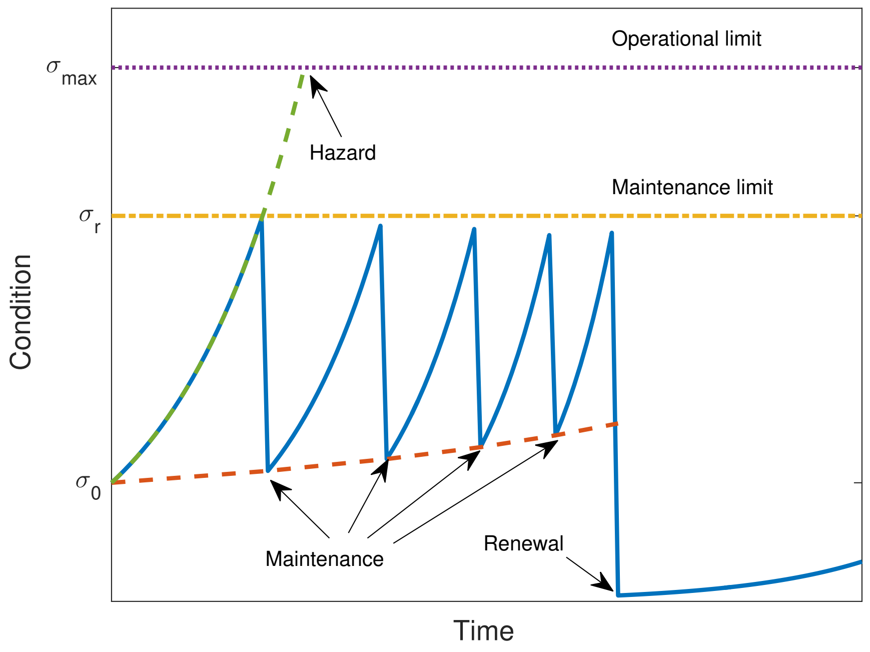


Figure 2.1: Deterioration process of a generic defect with an exponential deterioration. A higher value indicates a worse condition. The initial condition is denoted by σ_0, while σ_r and $\sigma_{\max}$ represent the maintenance and operational limit, respectively.

steep natural degradation between two consecutive maintenance interventions. Finally, renewal becomes the only option when a maintenance intervention becomes so inefficient that it can no longer improve the condition.

2.2 Condition-Based Maintenance Planning of Railway Infrastructure Networks

Maintenance can be either reactive or proactive. A shift from reactive maintenance to proactive solutions can be identified in several European countries in recent years [2, 174]. Proactive maintenance policies are especially popular for multi-component systems, i.e. systems consist of multiple dependent or independent components. This is because under a proactive maintenance strategy, techniques like grouping and balancing can be utilized to reduce maintenance costs in a multi-component system without compromising system performance [121]. Condition-based maintenance [13, 84], a proactive maintenance strategy where decision making is based on the observed “condition" of an asset, has received growing attention in various industrial fields [49, 79]. Unlike time-based maintenance strategies(e.g. the current cyclic track maintenance strategy in the Netherlands), condition-based maintenance is efficient as it can avoid unnecessary maintenance, reducing the maintenance costs. The resources saved from unnecessary maintenance (e.g. available maintenance time) can then be allocated to perform necessary maintenance for severely deteriorated parts, improving the safety of the whole asset [55]. The cost-saving potential of

condition-based maintenance on multi-component systems is demonstrated in [173] by a comparison with failure based policy and time-based policy. Condition-based maintenance is considered as the most promising maintenance strategy, as most system failures are preceded by one or more indicative signals [1]. A recent review on condition-based maintenance policies for general multi-component systems is provided in [83].
In this thesis we consider condition-based maintenance optimization based on a mathematical model describing the deterioration process of the condition of the asset. The model can be either deterministic, e.g. [47, 166], or stochastic, e.g. [102, 160]. A linear model is used in [166] for the natural degradation of track quality, and a Mixed Integer Linear Programming (MILP) problem is formulated in [166] to optimize tamping for a railway line over a finite planning horizon. An exponential model for track geometry deterioration is developed in [47] to minimize total track possession time caused by tamping over a finite planning horizon, while keeping the track geometry quality within safe limits. Optimal condition-based tamping is formulated as an MILP problem in [66], including the setup costs of tamping operations. Note that the deterioration models used in [47, 66, 166] are all deterministic models considering only nominal deterioration behavior. The resulting maintenance strategies might not be robust enough in the presence of various randomness like model uncertainties, measurement errors, and missing data. In this case stochastic models, which describe the deterioration dynamics either by a stochastic process, or by a random-variable model [53], are preferred because of the robustness of the resulting maintenance strategy. A binary Mixed Integer Nonlinear Programming (MILNP) problem is developed in [160] for optimal condition-based maintenance planning based on a stochastic deterioration model characterized by the Dagum probabilistic distributions. Other notable examples of condition-based track maintenance optimization approaches based on stochastic deterioration models include [102], where a bi-variate Gamma process describing the evolution of both the longitudinal and transverse levels is developed for the optimal planning of tamping operations for a French high-speed line, and [128], where a grey-box model is proposed to describe the ageing process of track geometry. A fuzzy Takagi-Sugeno internal model is used in [77] to capture the most important dynamics of squats evolution over time, and the effects of grinding and rail replacement are also modelled considering different representative scenarios.

2.2.1 Deterioration Model

In this section, we explain the deterioration model used in both Chapter 3 and 4. This discrete-time state space model describes the deterioration process of a generic defect of a railway infrastructure, e.g. ballast defect or squat growth in a certain length of track in the railway network. The infrastructure is composed of n components, e.g. n segments of track, with independent condition deterioration dynamics. Let the vector

$$x_j(k) = \begin{bmatrix} x_j^{\text{con}}(k) \\ x_j^{\text{aux}}(k) \end{bmatrix} \in \mathscr{X}_j$$

denote the state of the j-th component of the infrastructure at time step k. The state of the j-th component includes its condition x_j^{con}, as well as other auxiliary variables collected in the vector x_j^{aux}. These auxiliary variables, e.g. the condition after the last maintenance intervention, are necessary to model the inefficiency of corrective maintenance. Let $\mathscr{U} = \{a_0, \ldots, a_N\}$

denote the set of all possible actions[2] (including no maintenance) that can be applied to a component, where N is the number of possible interventions. The first action a_0 represents no maintenance, and the last action a_N represents full renewal. Let $u_j(k) \in \mathcal{U}$ denote the maintenance action applied to the j-th component at time step k. Furthermore, we define $u(k) = [u_1(k)^{\mathrm{T}} \dots u_n(k)^{\mathrm{T}}]^{\mathrm{T}} \in \mathcal{U}^n$ as the maintenance action performed on the multi-component infrastructure at time step k.
The deterioration process is also affected by various uncertainties like measurement errors and model inaccuracies. We denote $\theta_j(k) \in \Theta_j$ as the realization of the uncertainties related to the deterioration of component j. Similarly, we define $\theta(k) = [\theta_1^{\mathrm{T}}(k) \dots \theta_n^{\mathrm{T}}(k)]^{\mathrm{T}} \in \Theta$ as the realization of the uncertainties for the whole asset. The following generic model is proposed to describe the stochastic deterioration process of the j-th component of the asset:

$$\begin{aligned} x_j(k+1) &= f_j(x_j(k), u_j(k), \theta_j(k)) \\ &= \begin{cases} f_j^0(x_j(k), \theta_j(k)) & \text{if } u_j(k) = a_0 \text{ (no maintenance)} \\ f_j^q(x_j(k), \theta_j(k)) & \text{if } u_j(k) = a_q \quad \text{with } q \in \{1, \dots, N-1\} \\ f_j^N(\theta_j(k)) & \text{if } u_j(k) = a_N \text{ (full renewal)} \end{cases} \\ &\forall j \in \{1, \dots, n\}. \end{aligned} \tag{2.1}$$

The natural degradation f_j^0 only depends on the current condition and uncertainties. Full renewal f_j^N restores the component to an "as good as new" condition, regardless of the current condition or the history of maintenance interventions. However, renewal is also stochastic, and the exact condition after a renewal is uncertain. The effect of other interventions on a component depends both on the current condition and the history of maintenance.
The deterioration dynamics of the whole asset can then be written as:

$$x(k+1) = f(x(k), u(k), \theta(k)) \tag{2.2}$$

where $f = [f_1^{\mathrm{T}} \dots f_n^{\mathrm{T}}]^{\mathrm{T}}$ is a vector-valued function.
In practice, constraints must be considered for each individual component. We call these constraints local constraints, as they are only dependent on the condition, action, and uncertainties of the individual component. One crucial local constraint is that the condition of each component should not exceed the maintenance limit. In addition to local constraints for individual components, we also consider global constraints on the whole asset. Such global constraints usually arise from limited resources available for maintenance and renewal of the asset, e.g. budget and working time limits. To summarize, we define the constraints that need to be considered at time step k for the whole system as:

$$g(x(k), u(k), \theta(k)) \leq 0 \tag{2.3}$$

In summary, the deterioration process of the infrastructure can then be described by the whole-system dynamics (2.2) subject to the constraint (2.3).

[2]For the sake of simplicity we consider the same set of available actions for each component of the infrastructure.

2.2.2 Model Predictive Control for a Class of Hybrid Systems

We use Model Predictive Control (MPC) [26, 132] as the basic scheme for the long-term optimal planning of maintenance interventions over a finite planning horizon. Based on a dynamic model for the process and the current state, an MPC controller determines the optimal sequence of control actions that optimizes an objective function subject to constraints for a given prediction horizon. According to the receding horizon principle, only the first entry of the sequence of control actions is applied to the system, and the controller moves to the next time step, solving a new optimization problem using an updated state, which is obtained either from measurements, estimation, or simulation.

MPC has been widely applied to several real-world problems including supply chain management [108, 144], risk management of irrigation canals [169], and drinking water network management [62]. Some of these problems involve a system with both continuous and discrete dynamics, which we call hybrid systems. The generic deterioration of a railway infrastructure is one example of a hybrid system because the choice of maintenance activities can only take discrete values.

Such hybrid systems are usually modeled using the Mixed Logical Dynamical (MLD) framework [12], and a sequence of discrete control actions is determined by solving an Mixed Integer Programming (MIP) problem at each time step. An alternative is the Time-Instant Optimization (TIO) scheme, where a sequence of continuous time instants indicating the occurrence of each discrete control action is optimized, resulting in a continuous, albeit non-smooth, optimization problem at each time step. For some real-world applications, like water level control for irrigation canals [139, 161], the number of admissible control actions, e.g. opening and closing of barriers, is relatively small even for a long prediction horizon. In this case, only a small number of time instants are needed to indicate when each control action occurs. This usually results in an optimization problem that is easier to solve than the large MIP problem of MLD-MPC. A comparison between the MLD and the TIO framework is given in Figure 2.2, where $u(k)$ denotes the action performed at time step k, while t_{maint} and t_{renewal} are vectors containing all the time instants of the maintenance and renewal actions[3]. In this example, at most two maintenance actions and one renewal action can be performed within the 12-month prediction horizon. The MLD-MPC controller needs to optimize a discrete sequence of length 12 specifying which action (maintenance, renewal, or doing nothing) to be applied at each time step. However, the TIO-MPC controller only needs to optimize a continuous sequence of length 3, containing the time instants at which the two maintenance actions and one renewal action are performed, respectively. In this example, the TIO framework is a better choice than the MLD framework as it only needs a small number of continuous decision variables.

2.2.3 Distributed Optimization

Because an NP-hard problem (an MILP or non-smooth optimization problem) must be solved at each time step, hybrid MPC is in general very computationally demanding for large-scale systems. For the sake of scalability, a distributed optimization scheme is then usually adopted. However, there is a lack of distributed implementations of hybrid MPC in literature [99]. A distributed MPC method based on primal decomposition is developed in [97] for a class

[3]We use the notation $(v)_i$ to indicate the i-th element of the vector v.

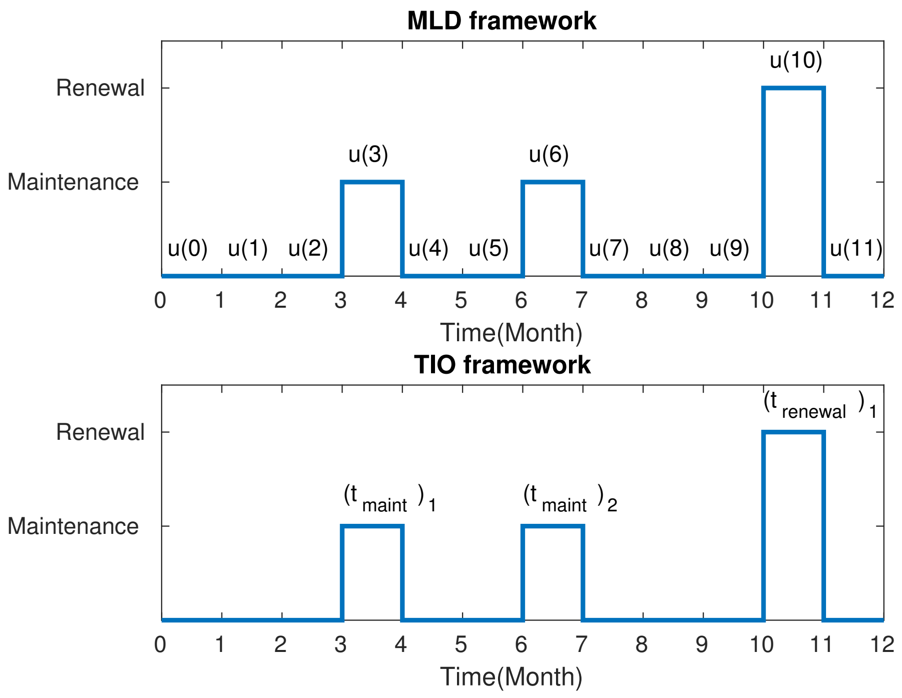


Figure 2.2: Comparisons between MLD and TIO framework illustrated by a small example, in which at most two maintenance actions and one renewal actions can be performed to an infrastructure within the one-year prediction window.

of hybrid systems with discrete control inputs, global constraints, and limited information sharing between local controllers. Recently, a practical approach for a class of networked hybrid MPC problems has been proposed in [101] where first the value of the binary decision variables in the local problem is determined, and then the Mixed Integer Quadratic Programming (MIQP) MPC optimization problem is transformed into a set of Quadratic Programming (QP) problems through distributed coordination. Although the solutions of both approaches are suboptimal, numerical experiments show that the loss of optimality is small for the corresponding application.
When distributed optimization scheme is adopted purely out of computational concerns, decomposition methods for large-scale Linear Programming (LP) and MIP problems can be used to divide the computational burden of the centralized MPC optimization problem among subproblems that are easier to solve. Benders decomposition [16], and Dantzig-Wolfe decomposition [38] are the most widely-used decomposition methods. The choice of decomposition method depends on the structure of the original problem. Benders decomposition is more suitable for problems coupled through common variables (complicating variables), while Dantzig-Wolfe decomposition is designed for problems coupled through common constraints (complicating constraints).

Benders Decomposition

Benders decomposition is initially designed for MILP problems in which the integer variables are the complicating variables, which, when temporarily fixed, yield an LP problem that allows for the use of strong duality to generate feasibility cuts and optimality cuts [109]. Interested readers are referred to [130] for an up-to-date review. Benders decomposition has already been applied to distributed MPC in [107] for temperature regulation in buildings. However, the MPC optimization problem in [107] is an LP problem. Because of the use of strong duality in classical Benders decomposition, the non-complicating variables, which appear in the subproblems, cannot be integers. Extensions have been made to apply Benders decomposition to address problems with MILP subproblems. The classical Benders decomposition framework has been extended to problems with a purely binary master problem and mixed binary subproblems in [150, 151] using reformulation linearization techniques and lift-and-project cutting plane scheme, to obtain a convex hull representation of the feasible region. Disjunctive programming is applied in [112, 146] to extend standard Benders decomposition to problems with mixed-binary subproblems using disjunctive decomposition. Reformulation linearization and disjunctive programming techniques are combined in [147] for problems with mixed binary master problems and subproblems. Recently, a branch-and-cut algorithm with local cuts has been developed in [46] for Benders decomposition with discrete subproblems. However, computational experiments show that the branch-and-cut algorithm cannot outperform state-of-the-art MILP solvers like Cplex.

Dantzig-Wolfe Decomposition

Dantzig-Wolfe decomposition is most suitable for large-scale problems with subproblems coupled through a small number of complicating constraints [18, 152]. Column generation technique [162] is usually used to increase tractability. Dantzig-Wolfe decomposition can be viewed as the dual of Benders decomposition, in the sense that applying Dantzig-Wolfe decomposition to an LP problem is equivalent to applying Benders decomposition to its dual. However, for MILP problems, Dantzig-Wolfe decomposition only solves an LP relaxation of the original problem. Exact solutions to the original problem can be found by combining branch-and-bound with column generation, known as the branch-and-price [7] algorithm. One typical application of Dantzig-Wolfe decomposition is the vehicle routing problem and its variants [50]. The maintenance scheduling and routing problem of offshore wind farms has been formulated as a Vehicle Routing Problem (VRP) with side constraints in [75] and solved efficiently using Dantzig-Wolfe decomposition. Dantzig-Wolfe decomposition has been used in [43] and [152] for distributed implementation of an LP- MPC optimization problem. Applications of Dantzig-Wolfe decomposition to MILP-MPC are relatively few. One example is [65], where a suboptimal solution of the MILP problem is obtained through column generation.

2.2.4 Chance-Constrained Optimization

Maintenance decision making can be influenced by different random factors like model uncertainties, missing data, measurement errors, etc. [98] Robust control [106], which guarantees good control performance and constraint satisfaction within a specific range of uncertainties, should then be considered. Popular robust control approaches, like the min-max approach [29, 63], are often conservative, as the worst-case scenario does not always occur. To avoid conservatism, chance constraints [127] can be considered, which guarantee that the constraints are satisfied with a probability no less than a given confidence level. Chance-constrained MPC, which replaces all constraints with uncertainties by chance constraints, and optimizes the expectation of the objective function, has been successfully applied to the management of drinking water networks [62] and stock management in hospital pharmacy

[81].
Let $(\Theta, \mathscr{B}(\Theta), \mathbb{P}_\theta)$ denote a probability space where Θ is a metric space with Borel σ-algebra $\mathscr{B}(\Theta)$ and probability distribution $\mathbb{P}_\theta$. We consider the following generic chance-constrained optimization problem

$$\min_{v \in \mathcal{V}} \mathbb{E}_\theta [J(v,\theta)] \tag{2.4}$$

$$\text{Subject to: } \mathbb{P}_\theta [g(v,\theta) \leq 0] \geq 1-\epsilon, \tag{2.5}$$

where $\theta \in \Theta \subseteq \mathbb{R}^{n_\theta}$ is a random vector containing all the uncertainties, and the parameter $\epsilon \in [0, 1]$ is the allowed violation level of the chance constraint. Moreover, we call a solution $v^* \in \mathcal{V}$ an ϵ-level solution if it is feasible for the chance constraint (2.5).
Depending on a priori knowledge on the probability distribution $\mathbb{P}_\theta$, we have the following classification of chance-constrained optimization problems:

- chance-constrained optimization problems with perfect information, where a given probability distribution, e.g. Gaussian, is assumed for the random vector θ;

- chance-constrained optimization problems with no information, where no assumption is made on $\mathbb{P}_\theta$.

An exact analytical deterministic representation of the chance constraint is in general unavailable for chance-constrained optimization problems with no information on the distribution of the uncertainty. Even with perfect information, an exact analytic representation is only available for very specific problems, e.g. linear chance constraints with Gaussian-distributed uncertainties [145]. This is why approximation methods, especially those with a probabilistic feasibility guarantees, have become the most prevailing solution approaches for chance-constrained optimization problem. The most notable approximation methods for chance-constrained optimization problems include analytical approximation approaches, sample averaging approaches, and scenario-based approaches[4].

Analytical Approximation

Analytical approximation methods, first proposed in [126], are most suitable for chance-constrained optimization problems with a known probability distribution. A review on safe[5] tractable approximation of chance constraints in provided in [110]. Notable analytical approximation methods for convex chance constraints include the quadratic approximation[14], the conditional value-at-risk approximation [135], and the Bernstein approximation [111]. These analytical approximation methods usually impose very strong assumptions. For instance, in [111] it is assumed that the convex chance constraint must be affine with respect to the uncertainty. A sequential convex approximation approach is developed in [74], which is applicable to general nonlinear convex chance-constrained problems. Another sequential analytical approximation based on the parametric function is developed in [56] for smooth non-convex chance constraints with general probability distributions.

[4]Also called scenario generation or scenario approximation approaches in literature.

[5]An approximation of the chance constraint is safe if the feasible region defined by the approximation is contained int the feasible region defined by the chance constraint.

Sample Average Approximation

Sample average approximation methods [148] replace the chance constraint (2.5) by the following relative-frequency count of constraint satisfaction on a randomized sample set $\mathscr{H}$:

$$\frac{1}{|\mathscr{H}|}\sum_{h\in\mathscr{H}} I_{g(v,\theta^{(h)})\leq 0} \geq 1-\epsilon, \tag{2.6}$$

where the indicator function I_X takes the value 1 if statement X is true, and 0 otherwise. The basic idea behind sample average approximation is Monte-Carlo simulation. The advantage of sample average approximation over a scenario-based approach is that it is valid without restrictive assumptions on the convexity of the chance-constrained problem. However, it requires an even larger sample size than the already computationally expensive scenario-based approach to obtain the same probabilistic feasibility bounds [19, 119]. The solution obtained by sample average approximation is also more conservative than the one obtained by a scenario-based approach because of the larger sample size.

Scenario-based Approach

The standard scenario-based approach [25] approximates the chance constraint with a finite number of randomized scenarios, i.e. replacing the chance constraint (2.5) by the following set of deterministic hard constraints:

$$g(v,\theta^{(h)}) \leq 0 \quad \forall h\in\mathscr{H}, \tag{2.7}$$

where $\theta^{(h)}$ denotes the realization of uncertainties of the h-th scenario in the scenario set $\mathscr{H}$. The optimal solution of the scenario-based optimization problem (2.4),(2.7) is also a random variable as the scenarios are generated randomly. A confidence level β is associated with the scenario-based optimization problem to provide probability bounds on its optimal solution. For a given β, the size of the scenario set $\mathscr{H}$ must be large enough to ensure that the optimal solution of (2.4),(2.7) is also a ϵ-level solution of the chance-constrained optimization problem (2.4)-(2.5) with a probability at least $1-\beta$.
The focus of scenario-based approaches is to find a lower bound on the size of the scenario set for a given violation and confidence level. Various scenario reduction techniques [28, 30, 72, 90, 91] have been applied to the standard scenario-based approach. A sampling-and-discarding approach is developed in [28] that quantifies the trade-off between performance and feasibility. An MIP problem is formulated in [90] for the optimal scenario reduction problem to minimize the probabilistic distance and performance difference between the original and the reduced scenario distributions. Recently, some sequential scenario reduction techniques [30, 91] have been developed for convex uncertain problems. The idea behind these sequential approaches is to verify the “temporary" scenario set at each time step against the given violation and confidence levels, and to increase the size of scenario set until it is validated.
Most proposed bounds on the size of the scenario set are only applicable to convex chance-constrained problems. For instance, it is assumed in [24, 25, 27, 172] that the chance constraint must be convex in the decision variable for any possible realization of the uncertainties. Even the scenario-based approach proposed in [59] for non-convex control design also requires the chance constraints to be convex. Performance and feasibility bounds for a class

of non-convex chance-constrained problems, including MIP problems with integer decision variables in the chance constraints, are provided in [44]. However, the feasibility bound is very conservative and not applicable to large-scale non-convex chance-constrained problems.

Robust Scenario-based Approach

A two-phase approach, which lies between a scenario-based method and a robust optimization approach, and can be applied to non-convex chance-constrained optimization problems,is proposed in [100] . This two-phase approach first solves a scenario-based optimization problem to obtain a set $\mathscr{B}^*$ covering a given fraction (determined by the violation level) of the probability mass of the uncertainty with a certain confidence, and then solves a robust version of the original chance-constrained optimization problem, where the uncertainty lies in the intersection of $\mathscr{B}^*$ and the uncertainty space Θ. As in our case $\mathscr{B}^*$ is a strict subset of Θ in most situations, the result of this two-phase approach is less conservative than the direct robust approach, where the whole uncertainty space Θ is considered.
Here we briefly summarize this two-phase approach, which will be used in Chapter 4 to approximate the chance-constrained MPC problem by a deterministic optimization problem. First we solve the following standard scenario-based problem (as described in Section 2.2.4) for a given violation level ϵ and confidence level β:

$$\min_{\{(\underline{\tau}_i,\overline{\tau}_i)\}_{i=1}^{n_\theta}} \sum_{i=1}^{n_\theta} \overline{\tau}_i - \underline{\tau}_i \tag{2.8}$$

$$\text{subject to: } (\theta)_i^{(h)} \in [\underline{\tau}_i, \overline{\tau}_i] \quad \forall h \in \mathscr{H}, \forall i \in \{1,\ldots,n_\theta\}, \tag{2.9}$$

where $(\theta)_i$ represents the i-th entry of the random vector θ, and $\mathscr{H}$ is the set of random scenarios. The size of $\mathscr{H}$ is chosen according to the following condition [3]:

$$|\mathscr{H}| \geq \left\lceil \frac{1}{\epsilon} \cdot \frac{e}{e-1} \left(2n_\theta - 1 + \ln \frac{1}{\beta} \right) \right\rceil, \tag{2.10}$$

which ensures that the chance constraint $\mathbb{P}_\theta \left[(\theta)_i \in [\underline{\tau}_i, \overline{\tau}_i] \right] \geq 1 - \epsilon$ is satisfied with a confidence β for each $i \in \{1,\ldots,n_\theta\}$.
Let $\{(\underline{\tau}_i^*, \overline{\tau}_i^*)\}_{i=1}^{n_\theta}$ denote the optimal solution of the scenario-based problem (2.8)-(2.9). We can then construct a hyperbox $\mathscr{B}^* = \times_{i=1}^{n_\theta} [\underline{\tau}_i^*, \overline{\tau}_i^*] \subset \Theta$, and solve the following robust optimization problem:

$$\min_{v \in \mathcal{V}} \frac{1}{|\mathscr{H}|} \sum_{h=1}^{|\mathscr{H}|} J(v, \theta^{(h)}) \tag{2.11}$$

$$\text{subject to: } \max_{\theta \in \mathscr{B}^*} g(v, \theta) \leq 0. \tag{2.12}$$

The optimal solution of this robust optimization problem is an ϵ-feasible solution of the chance-constrained problem (2.4)-(2.5) with probability at least $1 - \beta$.
As discussed in [100], although this multi-level approach does not require convexity in the decision variable or in the uncertainty to be valid, it is tractable only when the associated robust optimization problem (2.11)-(2.12) is tractable.

2.3 Optimal Scheduling of Maintenance Crews

Condition-based maintenance focuses on the time planning of maintenance interventions. How to optimally schedule the corresponding maintenance crews, including all necessary equipments and personnel, to perform the planned maintenance interventions on a railway network, taking into account the limited track possession time, is also of great concern for a maintenance contractor.

An integer linear programming problem is formulated in [39] to find the optimal railway track maitnenance schedule that minimizes the total maintenance cost in the planning horizon. An integer programming problem is formulated in [73] to schedule different types of routine maintenance activities for a single railway line, minimizing disruption to and from train traffic, and the weighted completion time of all maintenance activities. A mixed integer programming problem is formulated in [20] to minimize track possession cost and maintenance costs for a single railway line, considering both routine activities and projects like grinding or tamping. A metaheuristic using simulated annealing is developed in [141] to determine the optimal length of track to be treated by a tamping machine.

The maintenance crew scheduling problem over a railway network is usually formulated as a variant of vehicle routing problems. For example, in [70] the optimal scheduling of different maintenance tasks with various priorities over a railway network is formulated as a VRP with customer costs. In [125], the optimal clustering of small maintenance jobs into major projects is also recast as a VRP to minimize the total duration of all maintenance projects. Another popular approach for optimal scheduling of maintenance activities is the time-space network [124]. A comparison between the VRP approach and the time-space network model for the scheduling of maintenance activities can be found in [58]. Other approaches for scheduling maintenance activities over a railway network include the network-flow model proposed in [17], and the MINLP formulation developed in [171]. The maintenance schedule and the train timetable should be as compatible as possible to minimize the cost of traffic disruption. In most papers on maintenance scheduling, e.g. [20, 73], a timetable is already available, and the aim is to minimize disruption cost or timetable changes. On the other hand, [17] and [142] start with a given maintenance plan, and adjust the train schedule accordingly to maximize the traffic throughput. Recently, an integrated approach has been developed in [93] for the joint optimization of train timetabling and maintenance scheduling. Its focus is to optimally schedule the traffic-free maintenance time windows that are sufficient for the regular maintenance activities and the desired amount of train traffic. However, the maintenance time windows can only be chosen from a set of available options, limiting the flexibility of the proposed approach.

2.4 Multiple Traveling Salesman Problem

Although the travelling salesman problem (TSP) can be stated simply as *"Find the shortest route that connects all cities on a map"*, solving this problem has kept people busy for decades. The ongoing quest for faster algorithms for finding (an approximation of) the optimal solution of the TSP has led to a large amount of literature on the subject. Heuristic methods [71, 94] can be used to find solutions of large TSP instances quickly, but no guarantees can be given for finding the optimal solution. In this thesis we consider exact formulations that guarantee finding the globally optimal solution. A comprehensive discussion of the his-

tory and state-of-the-art of solving the TSP can be found in [4, 35].
The power of the TSP [37, 103, 116] does not only lie in finding tours of minimal distance along cities, but also in the fact that it forms the mathematical basis of many scheduling and routing problems. Extensions such as the vehicle routing problem [85, 120] and the pick-up and delivery problem [122, 123, 136, 143] are important problems in the fields of logistics and economics. Recent applications include optimal maintenance routing and scheduling for offshore wind farms [75], and optimal delivery or pickup of goods using hybrid electric vehicles [42]. In those problems one usually tries to minimize some "cost" (e.g. distance, time, money, or a combination) using multiple "salesmen" (e.g. people, trucks, air planes, vessels) that can visit the "cities" (e.g. shops, harbors, airports, or actual cities). The use of multiple salesmen to visit the cities makes the problems harder to solve due to the increase in the number of possible solutions. The multiple traveling salesmen problem (mTSP) is at the basis of the vehicle routing problem and the pick-up and delivery problem.

The essence of mTSP is to find the shortest total travel distance for multiple salesmen starting from and returning to a single depot/home city. Since certain problems require more than one depot (e.g. for delivering goods to shops that can be supplied from multiple storage facilities), an extension to the multi-depot multiple-salesmen TSP (MmTSP) has been made [9]. In this case the problem consists of finding the shortest distance such that several salesmen will start at a depot, they visit all the cities once (and only once), and return to a depot again. When it is not important at what depot the salesmen end their route, we talk about a *nonfixed-destination problem*; when the salesmen are supposed to return to their original depot we talk about a *fixed-destination problem* [82]. The work of [15] is also concerned with multi-depot TSPs, where the number of salesmen per depot is not limited and the travel distances are symmetric. In this thesis we will focus on problems with a fixed number of salesmen per depot, and asymmetric costs. The fixed-destination Multi-depot multiple-salesmen TSP (FMmTSP) is a restricted case of the nonfixed-destination problem, with the additional constraint that all salesmen should return to their original location. Therefore, the former is more difficult to solve than the latter; the solutions to the FMmTSP are a subset of the solutions to the MmTSP. In [82] an MILP description for the fixed-destination problem has been proposed using 3-index decision variables, resulting in a large increase in the number of binary variables for each added depot. Cycle (or subtour) elimination constraints (CECs) are used to ensure that no cycles exist within the set of city nodes. They have been a topic of active research over many decennia, starting with the use of loop constraints by **(author?)** [37] in 1954, the node potentials by **(author?)** [103] in 1960, and (multi)-commodity flow-based constraints in [54] starting from 1978. Loop conditions give strong LP relaxations, but the number of constraints grows exponentially with the problem size. The number of node-potential-based constraints only grows quadratically with the problem size, but they result in much weaker relaxations. Using multi-commodity flow formulations it is possible to obtain strong relaxations, but with a number of constraints growing cubically in the problem size. Cycle imposement constraints (CICs) can be used to ensure a (minimum) number of cycles in a set of nodes. Fixed-destination solutions for TSP-like problems can be created by enforcing that there should be at least D cycles in the combined set of depot and city nodes, while using CECs to ensure that no subtours exist in the set of city nodes; when D equals the number of depots this will result in exactly D cycles in the network; one for each of the depots. CICs have only recently been discussed in the literature, starting with the path elimination constraints of **(author?)** [11] in 2011, the multi-commodity flow-based constraints of **(author?)** [10] in 2012, and the node currents of [22] in 2014. Table 2.1 shows the order of the

Table 2.1: Overview of CECs and CICs and the order of their numbers

Order	Cycle elimination	Cycle imposement
$O(2^N)$	loop conditions [37]	path elimination [11]
$O(N^3)$	commodity flow [54]	commodity flow [10]
$O(N^2)$	node potentials [103]	**node currents** [22]

number of the discussed CECs and CICs. This thesis will introduce the node current-based CICs, which can be seen as the equivalent of the node potentials for cycle imposement.

2.5 Reverse Stackelberg Games with Incomplete Information

The Stackelberg game, a hierarchical leader-follower game first introduced in the 1930s in an economic context [165], has received growing recognition in the systems and control field since the 1970s [8, 80]. In a Stackelberg game, first the leader makes her decision; then the followers, informed of the leader's decision, make their decisions accordingly. The reverse Stackelberg game, in which the leader proposes a function mapping from the followers' decision spaces to the leader's decision space, instead of making a direct decision, can be viewed as a more general case of the original Stackelberg game. Reverse Stackelberg games have successfully been applied to many hierarchical decision making problems like nonlinear network pricing [149], optimal routing [60], and toll design [154].
We follow the "type" notation proposed in [68] for games with incomplete information, where at least one player possesses certain important attributes, the actual value of which is only known to himself. These attributes can be, e.g., the production cost or the risk attitude of the player. The type of a player is then characterized by a vector of these attributes. The actual type of a player is only known to himself, and his opponents only know the type space and the type distribution, i.e. all possible alternatives of each attribute and the probability of each possible combination. Moreover, we consider information asymmetry in reverse Stackelberg games with incomplete information, where the leader has no private information, thus no type. Compared with the situation of complete information, the leader's lack of information regarding the followers produces a less desirable result for her [149].
Solving the Stackelberg game is equivalent to solving a bilevel programming problem [33], and even the simplest linear bilevel programming problem has been proved to be NP-hard [67]. The more general reverse Stackelberg games are even more difficult to solve, especially when a wide class of leader functions are considered and the players have general, nonconcave utility functions. Most papers that discuss nonlinear leader functions often focus on deriving analytic solutions for problem-specific utility functions [115, 149, 170]. A systematic approach to compute optimal nonlinear leader functions for reverse Stackelberg games with general utility functions is proposed in [61], but under the restrictive assumption of complete information. Therefore, in this thesis, we focus on numerical solution approaches for the more realistic Stackelberg game with incomplete information, considering nonlinear leader functions and general utility functions.
Our key contribution is a systematic solution approach based on basis functions and semi-infinite programming for reverse Stackelberg games with incomplete information, considering nonlinear leader functions and general, nonconcave utility functions.

2.6 Summary

In this chapter we have first introduced typical railway track defects and the corresponding maintenance interventions. We then provided the preliminaries of the two aspects of maintenance optimization of railway infrastructures, namely, long-term condition-based maintenance planning and short-term maintenance crew scheduling. For long-term maintenance intervention planning, the generic formulation of the deterioration model is presented, as well as a brief introduction on model predictive control, distributed optimization, and chance-constrained optimization. We have then provided a brief survey on railway maintenance crew scheduling and the multiple-traveling salesman problem. Finally, we have briefly explained reverse Stackelberg games with incomplete information.

Chapter 3

Centralized Maintenance Optimization of Small-Scale Railway Networks

3.1 Problem Description

An optimization-based, multi-level approach is developed in this chapter for the optimal planning of maintenance interventions for railway infrastructures like rail and ballast. A schematic plot for the multi-level approach is provided in Figure 3.1. Three optimization problems, namely the intervention planning problem, the slot allocation problem, and the clustering problem, are solved at the high, middle, and low level, respectively. Based on the component-wise discrete-time prediction model of the condition of the infrastructure, at each time step the high-level intervention planning problem determines the optimal maintenance intervention for each component over a given prediction horizon. The sampling time, i.e. the length of each time step, is usually larger than one month because of the slow deterioration dynamics of a railway infrastructure. If a maintenance intervention is suggested at any time step at the high level, it should be performed within a traffic-free time slot (4-8 hours at night) to avoid any disruption to the train service. However, it is not always possible to complete an intervention within such a short time slot, and a new operation must then be scheduled into a new time slot to finish the required intervention, resulting in an additional setup cost including machinery, logistics, personnel, etc. This gives rise to the middle-level slot allocation problem, which determines the time slots that optimize the trade-off between traffic disruption and the total setup cost associated with each maintenance slot, while guaranteeing that the total duration of the resulting maintenance slots is no less than the estimated maintenance time. According to the intervention plan, the low-level clustering problem then groups the basic units into clusters that can be treated within the allocated time slots. If the resulting clusters cannot cover all the basic units that need to be treated according to the high-level intervention plan because of insufficient time slots, then the slot allocation problem is solved again with a longer estimated maintenance time. This iterative procedure between the slot allocation problem and the clustering problem repeats until all the basic units that need to be treated are covered by a cluster. This cluster-wise work plan is then applied to the infrastructure, and the condition of each component is then regularly measured or updated by estimation.
The motivation to adopt a multi-level scheme includes different time scales of the deterioration process and traffic schedule, and computational tractability. A flow chart is presented in Figure 3.2 to illustrate the proposed multi-level scheme. The condition of the railway in-

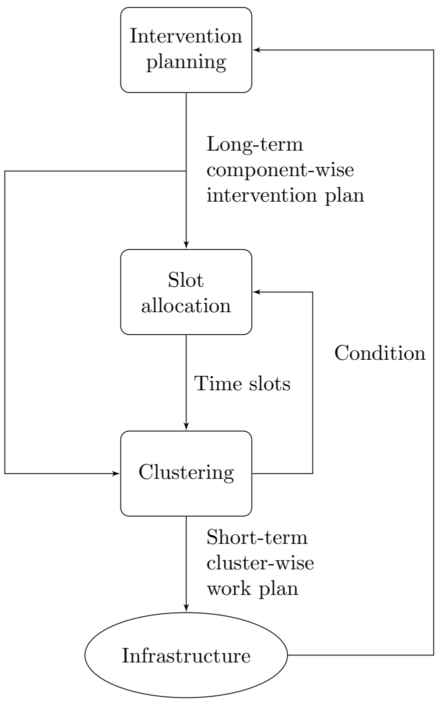


Figure 3.1: Schematic plot of the proposed multi-level approach for optimal condition-based maintenance planning.

frastructure is monitored at every time step. The measurements for each basic unit are collected and processed, then aggregated to represent the condition of each component. Based on the current condition of each component, the high-level MPC controller determines the maintenance plan for the current time step that optimizes the trade-off between condition deterioration and maintenance cost. If no intervention is suggested for any component for the current time step, the MPC controller moves to the next time step. The condition of the infrastructure is updated by new measurements, or estimates if no new measurements are available. The iterative procedure between the middle-level and low-level problems is triggered when an intervention, e.g. grinding or tamping, is suggested at the high level for at least one component for the current time step. Let $\hat{T}_{\text{Maint}}$ denote the estimated time to complete the suggested intervention. The optimal maintenance slots are allocated at the middle level, minimizing the trade-off between the total setup cost for the maintenance slots and the cost of disruption to the railway traffic, guaranteeing that the resulting maintenance time is no less than the estimated maintenance time $\hat{T}_{\text{Maint}}$. The resulting maintenance slots are then fed to the low level. Depending on their location and condition, the basic units inside the components that require the corresponding intervention are grouped into clusters that must be treated within the time slots determined at the middle level. If the resulting clusters at the low level cannot cover all the basic units, depending on the number and conditions of the remaining basic units, an evaluation is made to determine whether to solve the middle-level problem again with a larger estimated maintenance time in order to cover the remaining basic units. A value v_{rem} is calculated for the remaining basic units, which is the ratio of the accumulated condition of the remaining basic units and the accumulated condition of all the basic units that needed to be treated. An additional maintenance time $\Delta\hat{T}$ is also estimated, depending on the number of the remaining basic units. A factor μ is assigned to $\Delta\hat{T}$ to convert the additional maintenance time into a “cost". The detailed explanation on

how to compute v_{rem} and $\Delta\hat{T}$ is given in Section 3.4. If $v_{\text{rem}} - \mu\Delta\hat{T} \leq 0$, indicating that the benefit of covering the remaining basic units does not justify the cost of the required additional maintenance time, the iterative procedure terminates and the uncovered basic units remain uncovered. If $v_{\text{rem}} - \mu\Delta\hat{T} > 0$, indicating that it is worthwhile to cover the remaining basic units with additional maintenance time, the middle-level problem is solved again with a longer estimated maintenance time $\hat{T}_{\text{Maint}} + \Delta\hat{T}$.
The multi-level framework can be demonstrated by the pseudocode in Algorithm 3.1. Here, the vector ξ contains the positions of all basic units, e.g. kilometer positions of all the squats on a piece of rail. The conditions of all basic units at time step k are collected in the vector $w(k)$. In particular, $w(0)$ denotes the initial conditions of the basic units. Moreover, $t_{\text{sl}}(k)$ and $\phi(k)$ denote the solutions of the middle-level and the low-level problem, i.e. the resulting time slots and clusters, at time step k, respectively.

3.2 High-Level MPC Problem

In this section a scenario-based chance-constrained MPC controller is developed for the deterioration process described by (2.2),(2.3) to determine the optimal maintenance and renewal interventions for each component of the asset. A TIO approach is applied to transform the MPC optimization problem with both continuous and discrete decision variables into a continuous non-smooth optimization problem. To that aim we first recast the hybrid dynamic model (2.2),(2.3) into a TIO prediction model. Then we select a set of representative scenarios from the entire set of realizations of the uncertainties within the prediction period. Chance-constrained MPC is then applied to the scenario-based TIO prediction model.

3.2.1 TIO Prediction Model

A TIO prediction model is developed based on the original hybrid deterioration model (2.2)-(2.3), which contains both continuous and discrete dynamics. Let T_s denote the sampling time (usually in months for the deterioration process of railway infrastructures). Furthermore, let N_{P} and N_{C} denote the prediction and control horizons, respectively. Denote $\hat{x}(k+l|k)$ as the estimated state at time step $k+l$ based on the information available at time step k. The sequence of estimated states, control inputs, and uncertainties within the prediction period can be defined as:

$$\begin{aligned}
\tilde{x}(k) &= [\hat{x}^{\text{T}}(k+1|k)\ldots\hat{x}^{\text{T}}(k+N_{\text{P}}|k)]^{\text{T}}\\
\tilde{u}(k) &= [u^{\text{T}}(k)\ldots u^{\text{T}}(k+N_{\text{P}}-1)]^{\text{T}}\\
\tilde{\theta}(k) &= [\theta^{\text{T}}(k)\ldots\theta^{\text{T}}(k+N_{\text{P}}-1)]^{\text{T}}
\end{aligned}$$

The N_{P}-step prediction model can then be formulated as:

$$\tilde{x}(k) = \tilde{f}(x(k),\, \tilde{u}(k),\, \tilde{\theta}(k)) \tag{3.1}$$

$$\tilde{g}(x(k),\, \tilde{u}(k),\, \tilde{\theta}(k)) \leq 0 \tag{3.2}$$

where the function $\tilde{f}$ can be derived from recursive substitution of (2.2), as in standard MPC. The function $\tilde{g}$ can be derived similarly.

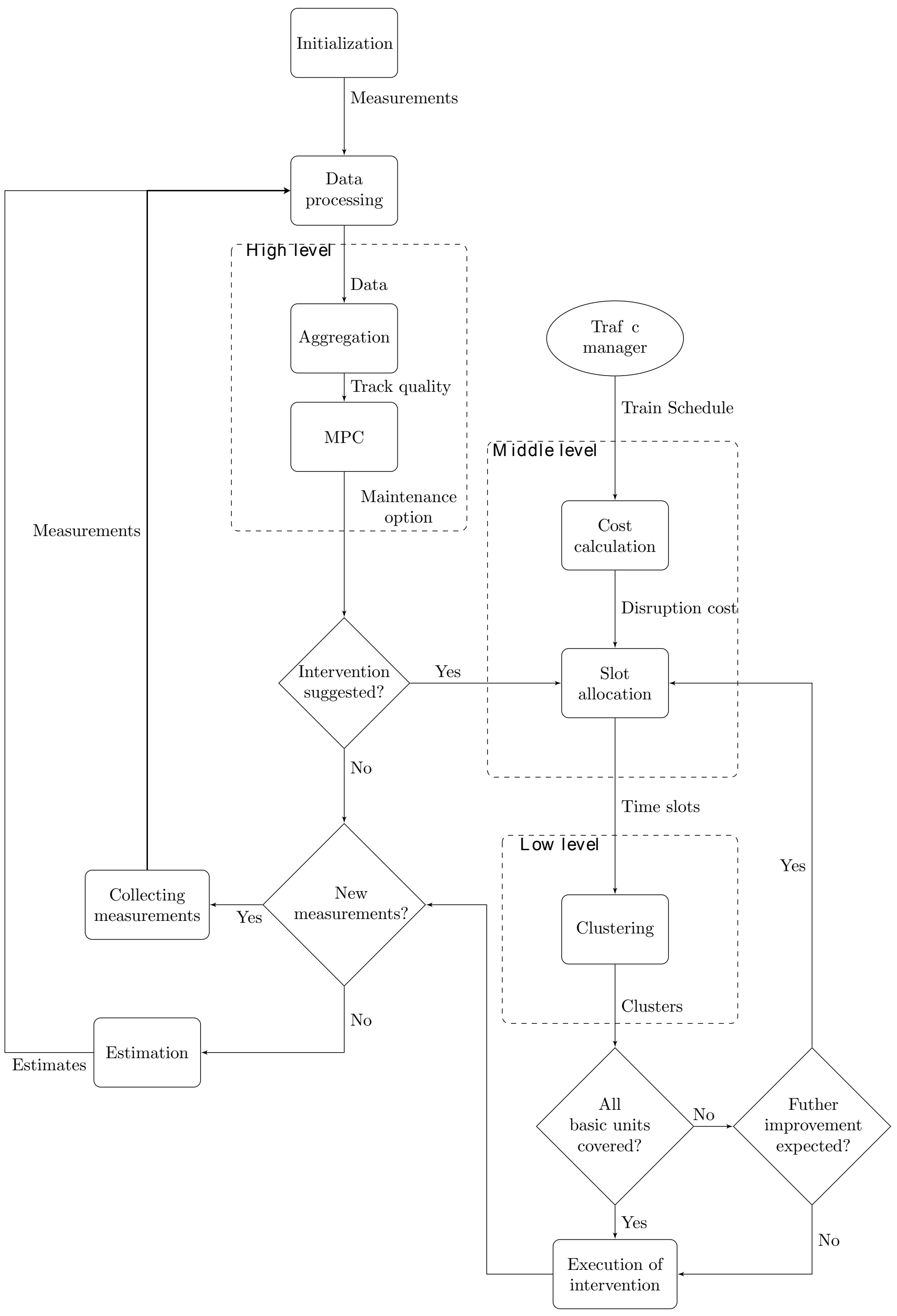


Figure 3.2: Flowchart of the proposed multi-level approach for online condition-based maintenance planning.

Algorithm 3.1 Procedure of the multi-level approach.

```
Function HighLevel_MPC($w(0), \xi$)
    // Initialize the condition of the components
    $k \leftarrow 1$
    $w(k) \leftarrow w(0)$
    $x(k) \leftarrow$ Aggregate($w(k), \xi$)
    while $k \leq k_{\text{end}}$ do
        // Solve the MPC optimization problem at time step $k$
        $u(k) \leftarrow$ MPC_Optimize($x(k)$)  /* Middle- and low-level problems are triggered whenever
            an intervention is suggested */
        for each intervention $a_l$ do
            if any $u_j(k) = a_l$ then
                /* Solve the middle-level problem to determine the time slots for
                    intervention $a_l$ */
                $t_{\text{sl}}(k) \leftarrow$ MiddleLevel($u(k), a_l, \hat{T}_{\text{Maint}}$)  /* Solve the low-level problem to determine
                    the clusters for intervention $a_l$ */
                $\phi(k) \leftarrow$ LowLevel($w(k), \xi, t_{\text{sl}}(k), a_l$)
                while a basic unit outside $\phi(k)$ do
                    Compute $v_{\text{rem}}$ and $\Delta\hat{T}$ using (3.35) and (3.36)
                    if $v_{\text{rem}} - \mu\Delta\hat{T} > 0$ then
                        $\hat{T}_{\text{Maint}} \leftarrow \hat{T}_{\text{Maint}} + \Delta\hat{T}$
                        $t_{\text{sl}}(k) \leftarrow$ MiddleLevel($u(k), a_l, \hat{T}_{\text{Maint}}$)
                        $\phi(k) \leftarrow$ LowLevel($w(k), \xi, t_{\text{sl}}(k), a_l$)
                    else
                        break
                end
                // Apply intervention $l$ to the infrastructure
                Intervention($\phi(k), a_l$)
            // Update conditions of basic units
            if New measurements available then
                $w(k+1) \leftarrow$ New measurements
            else
                $w(k+1) \leftarrow$ Simulate($w(k), \xi, \phi(k)$)
        end
        $x(k+1) \leftarrow$ Aggregate($w(k+1), \xi$)
        $k \leftarrow k+1$
    end
end
```

Recall that $u(k) \in \mathscr{U} = \{a_0, \ldots, a_N\}$, where the option a_0 indicates “no intervention" and the last intervention a_N is full renewal. TIO-MPC first fixes the maximum number of times that each of the N interventions can be performed within the prediction period, and optimizes the continuous time instants to perform the interventions. Formally, let $v_{j,q}$ denote the maximum number of occurrences of intervention q for component j with the prediction period. All the relative time instants at which intervention q occurs in the prediction horizon for component j at time step k are collected in the vector $t_{j,q,k}$ of length $v_{j,q}$. The sequence of time instants to be optimized at time step k can then be written as:

$$\tilde{t}(k) = [\underbrace{t_{1,1,k}^{\mathrm{T}} \cdots t_{1,N,k}^{\mathrm{T}}}_{t_1^{\mathrm{T}}(k)} \cdots \underbrace{t_{n,1,k}^{\mathrm{T}} \cdots t_{n,N,k}^{\mathrm{T}}}_{t_n^{\mathrm{T}}(k)}]^{\mathrm{T}} \tag{3.3}$$

The sequence of time instants $\tilde{t}(k)$ can be converted into the sequence of control inputs $\tilde{u}(k)$ by rounding to the nearest discrete time steps. The rounding function is always non-smooth, so TIO-MPC must solve a non-smooth optimization problem at each time step even for systems with linear dynamics.
For the situation of $N_{\mathrm{C}} < N_{\mathrm{P}}$, we have

$$u_j(k+l) = a_0 \quad \forall l \in \{N_{\mathrm{C}} \ldots N_{\mathrm{P}} - 1\}, \forall j \in \{1, \ldots, n\}$$

This means no intervention is applied beyond the control horizon.
An example to explain the sequence of continuous time instants $\tilde{t}(k)$ and how it is converted to the sequence of discrete control inputs $\tilde{u}(k)$ is given in Figure 3.3. In this example, the resulting sequence of time instants for component j at time step k is $t_j^{\mathrm{T}}(k) = [t_{j,1,k}^{\mathrm{T}}\ t_{j,2,k}^{\mathrm{T}}]^{\mathrm{T}}$. Among the two time instants for intervention a_1, only $(t_{j,1,k})_1$, which is located within the control period (indicated by the dashed vertical line), is rounded to the nearest relative time step l, indicating $u_j(k+l) = a_1$. The second time instant $(t_{j,1,k})_2$, is neglected as it is outside the control period $N_{\mathrm{C}} T_s$. Similarly, the first and only time instant for a_2 is within the control period, thus we have $u_j(k+m) = a_2$ as m is the time step closest to $(t_{j,2,k})_1$.
The following linear constraints should be considered for the time instants:

$$(t_{j,q,k})_1 \geq t_{j,q}^{\min} \tag{3.4}$$

$$(t_{j,q,k})_{v_{j,q}} \leq t_{j,q}^{\max} \tag{3.5}$$

$$(t_{j,q,k})_{i+1} - (t_{j,q,k})_i \geq \Delta t_{j,q}^{\min} \quad \forall i \in \{1, \ldots, v_{j,q} - 1\} \tag{3.6}$$

$$t_{j,q}^{\max} = N_{\mathrm{C}} T_s + v_{j,q} \Delta t_{j,q}^{\min} \tag{3.7}$$

$$\forall j \in \{1, \ldots, n\} \quad \forall q \in \{1, \ldots, N\}.$$

Constraints (3.4) and (3.5) specify the lower and upper bound of the time instants for intervention q on component j. The lower bound $t_{j,q}^{\min}$ is especially useful to address the issue of early planning. For example, if every maintenance intervention must be planned six months ahead, then we can simply set $t_{j,q}^{\min}$ to be equal to six months. The upper bound $t_{j,q}^{\max}$ is calculated in (3.7) to allow the situation of no intervention planned within the control period, where $\Delta t_{j,q}^{\min}$ is the minimal interval between two consecutive intervention of the same type, as specified in constraint (3.6).

The constraints (3.4)-(3.7) must be included in the optimization problem at each time step. Note that there is no stochasticity associated with these constraints, and they can be treated as normal linear constraints.

3.2.2 Representative Scenario-Based MPC

Now a representative scenario-based MPC controller is developed based on the TIO prediction model of Section 3.2.1. The optimization problem at each time step is illustrated by the schematic plot in Figure 3.4.
In practice, the set Θ containing all possible realizations of the uncertainties for a railway infrastructure might be very large. The set of all possible realizations of uncertainties over the whole prediction period, $\tilde{\Theta} = \Theta^{N_{\mathrm{P}}}$, might be huge for a long prediction horizon. For tractability, a relatively small number of representative scenarios is selected from the set $\tilde{\Theta}$. Let $\tilde{\mathscr{H}} \subset \tilde{\Theta}$ denote the set of representative scenarios. The following scenario-based TIO prediction model can then be derived for any $\tilde{h} \in \tilde{\mathscr{H}}$:

$$\tilde{x}(k) = \tilde{f}_{\mathrm{TIO}}(x(k), \tilde{t}(k), \tilde{h}) \tag{3.8}$$

$$\tilde{g}_{\mathrm{TIO}}(x(k), \tilde{t}(k), \tilde{h}) \leq 0. \tag{3.9}$$

Define

$$J(k) = J_{\mathrm{Deg}}(k) + \lambda\mu J_{\mathrm{Maint}}(k) \tag{3.10}$$

as the objective function that needs to be minimized at each time step k. The parameter μ is a scaling factor, and the parameter λ captures the trade-off between the condition of the infrastructure and the maintenance cost.
The first part

$$J_{\mathrm{Deg}}(k) = \sum_{j=1}^{n} \sum_{l=1}^{N_{\mathrm{P}}} |x_j^{\mathrm{con}}(k+l) - \underline{x}_j^{\mathrm{con}}|_q \tag{3.11}$$

minimizes the magnitude of condition degradation, where the norm $|\cdot|_q$ can be the 1-norm, 2-norm, or infinity norm, for $q = 1, 2, \infty$, respectively.
The second part in the objective function is the accumulated maintenance cost, which can be formulated as:

$$J_{\mathrm{Maint}}(k) = \sum_{j=1}^{n} \sum_{l=1}^{N_{\mathrm{P}}} \sum_{q=1}^{p} \gamma_{j,q} I_{u_j(k+l-1)=a_q} \tag{3.12}$$

where the binary indicator function I_X takes the value 1 if the statement X is true, otherwise it takes the value 0, and the parameter $\gamma_{j,q}$ represents the required cost if intervention a_q is applied to component j.
As shown in (3.10)-(3.12), the objective function $J(k)$ is a function of $\tilde{x}(k)$ and $\tilde{u}(k)$. Using the TIO converting rule, the value of the objective function at time step k can be rewritten as:

$$J(k) = f_{\mathrm{opt}}(x(k), \tilde{t}(k), \tilde{h}) \tag{3.13}$$

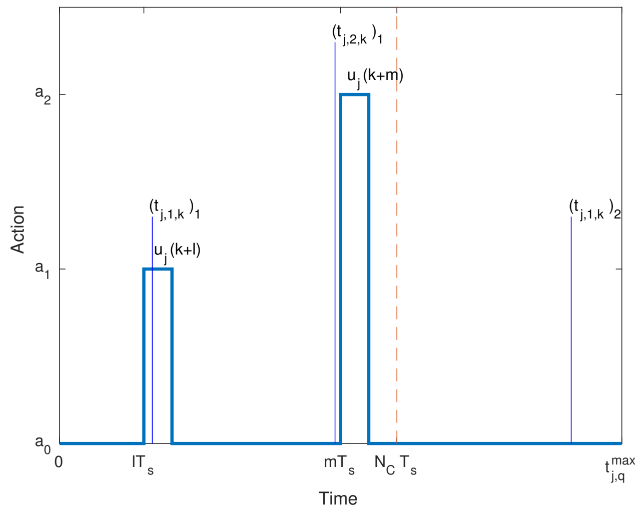


Figure 3.3: Example illustrating the time instants in TIO-MPC. In this example, we have $u_j \in \{a_0, a_1, a_2\}$. Interventions a_1 and a_2 can be applied to component j at most twice and once, respectively.

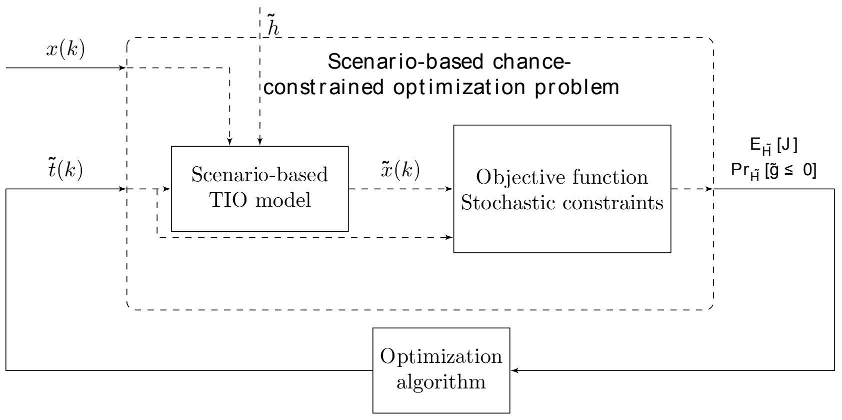

$\tilde{h}$
$x(k)$
Scenario-based chance-constrained optimization problem
$\tilde{t}(k)$
Scenario-based TIO model
$\tilde{x}(k)$
Objective function Stochastic constraints
$E_{\tilde{H}}[J]$
$Pr_{\tilde{H}}[\tilde{g} \leq 0]$
Optimization algorithm

Figure 3.4: Optimization scheme for scenario-based chance-constrained TIO-MPC. Unlike nominal MPC, chance-constrained MPC optimizes the expectation of the objective function, while guaranteeing a confidence level of the probability of constraint satisfaction.

Finally, the continuous, nonlinear optimization problem to be solved at each time step k by the scenario-based chance-constrained TIO-MPC controller can be formulated as:

$$\min_{\tilde{t}(k)} \mathrm{E}_{\tilde{\mathscr{H}}}[f_{\mathrm{opt}}(x(k), \tilde{t}(k), \tilde{h})] \tag{3.14}$$

$$\text{Subject to: } \mathbb{P}_{\tilde{\mathscr{H}}}[\tilde{g}_{\mathrm{TIO}}(x(k), \tilde{t}(k), \tilde{h}) \leq 0] \geq \eta \tag{3.15}$$

$$g_{\tilde{t}}(\tilde{t}(k)) \leq 0 \tag{3.16}$$

where constraint (3.16) is the compact expression of constraints (3.4)-(3.7), and $\eta \in (0, 1)$ is the confidence level of the chance constraints.
The goal is to minimize the expected value of the objective function f_{opt} over the set of all representative scenarios $\tilde{\mathscr{H}}$. Note that the expected value of the objective function f_{opt} can be computed as

$$\mathrm{E}_{\tilde{\mathscr{H}}}[f_{\mathrm{opt}}(x(k), \tilde{t}(k), \tilde{h})] = \sum_{\tilde{h} \in \tilde{\mathscr{H}}} p(\tilde{h}) f_{\mathrm{opt}}(x(k), \tilde{t}(k), \tilde{h}), \tag{3.17}$$

where $p(\tilde{h})$ is the probability of scenario $\tilde{h}$. In a similar way, the chance constraint (3.15) can be reformulated as

$$\sum_{\tilde{h} \in \tilde{\mathscr{H}}} p(\tilde{h}) I_{\tilde{g}_{\mathrm{TIO}}(x(k), \tilde{t}(k), \tilde{h}) \leq 0} \geq \eta. \tag{3.18}$$

Because of the rounding procedure in the TIO prediction model, (3.14)-(3.16) is a non-smooth optimization problem. Hence, derivative-free or direct search algorithms [89] like genetic algorithms or pattern search should be considered. Pattern search with multi-start is used in the case study in Section 3.5.

3.3 Middle-Level Slot Allocation Problem

The middle-level slot allocation problem is triggered at any time step at which an intervention is recommended at the high level. This problem is solved to determine the maintenance time slots for the corresponding intervention, optimizing the trade-off between traffic disruption and the total setup costs to complete the corresponding intervention, as stated in Section 3.1. The planning horizon of the slot allocation problem is from the current time step to the next time step (e.g. from October to November), as each intervention suggested at the high level should be completed within the sampling time (e.g. one month).
Let N_{sl} denote the number of time slots available at the current time step of the high-level controller. Based on the corresponding train schedule, we can evaluate the cost of traffic disruption as a piecewise-constant function of track possession time. Traffic-free time intervals are assigned a zero-cost, while other non-traffic-free intervals are associated with different costs of disruption. Let N_τ denote the number of intervals of this piecewise-constant function, and let c_j denote the cost of disruption of the j-th interval. An illustration of this piecewise-constant function is given in Figure 3.5.

Similar to the TIO technique used in the high-level MPC controller, we define t_i^{start} and t_i^{end} as the start and end time instants of the i-th maintenance slot at the current time step, respectively. Let the positive parameter $\Delta\tau_{\mathrm{min}}$ denote the minimal length of a time slot. Recall

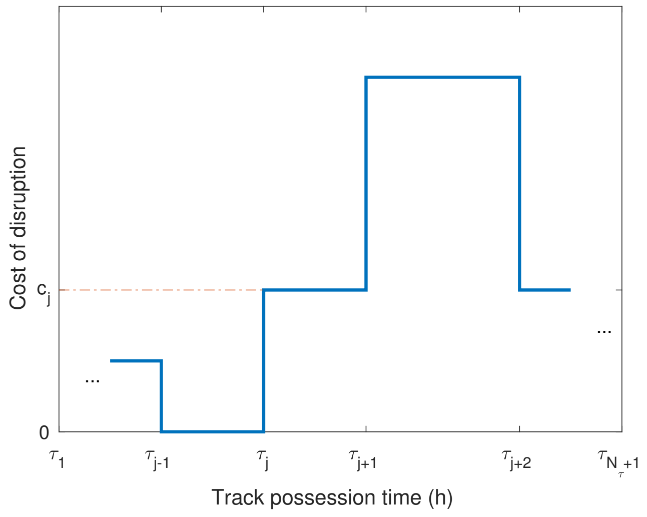


Figure 3.5: An example of the piecewise-constant function of the disruption cost, where τ_1 and $\tau_{N_\tau+1}$ coincide with the start of the current and next time steps, respectively. The cost of disruption is c_j if the j-th interval $[\tau_j, \tau_{j+1}]$ is occupied for maintenance.

that $\hat{T}_{\text{Maint}}$ is the estimated maintenance time, and let c_{sl} denote the setup cost associated with a maintenance slot. Moreover, a fixed setup time, denoted by T_{set}, is also associated with a maintenance operation. This setup time includes the time to prepare and finish a maintenance operation in a time slot. The middle-level optimization problem can then be formulated as:

$$\min_{\{(t_i^{\text{start}}, t_i^{\text{end}})\}_{i=1}^{N_{\text{sl}}}} \sum_{i=1}^{N_{\text{sl}}} \sum_{j=1}^{N_\tau} c_j \Big(\max(\min(\tau_{j+1}, t_i^{\text{end}}), \tau_j) - \min(\max(\tau_j, t_i^{\text{start}}), \tau_{j+1})\Big)$$
$$+\lambda_{\text{sl}} \sum_{i=1}^{N_{\text{sl}}} c_{\text{sl}} I_{t_i^{\text{end}} \le \tau_{N_\tau+1}} \tag{3.19}$$

subject to

$$t_1^{\text{start}} \ge \tau_1 \tag{3.20}$$
$$t_{N_{\text{sl}}}^{\text{end}} \le \tau_{\max} \tag{3.21}$$
$$t_i^{\text{end}} - t_i^{\text{start}} \ge \Delta\tau_{\min} \quad \forall i \in \{1, \dots, N_{\text{sl}}\} \tag{3.22}$$
$$t_i^{\text{end}} + \epsilon \le t_{i+1}^{\text{start}} \quad \forall i \in \{1, \dots, N_{\text{sl}} - 1\} \tag{3.23}$$
$$\tau_{\max} = \tau_{N_\tau+1} + 2N_{\text{sl}}\Delta\tau_{\min} \tag{3.24}$$
$$t_i^{\text{start}} \le \tau_{N_\tau+1} \implies t_i^{\text{end}} \le \tau_{N_\tau+1} \quad \forall i \in \{1, \dots, N_{\text{sl}}\} \tag{3.25}$$
$$\sum_{i=1}^{N_{\text{sl}}} I_{t_i^{\text{end}} \le \tau_{N_\tau+1}} \cdot \left(t_i^{\text{end}} - t_i^{\text{start}} - T_{\text{set}}\right) \ge \hat{T}_{\text{Maint}}. \tag{3.26}$$

The first term in the objective function (3.19) is the total cost of disruption, while the second term represents the number of *active* time slots, i.e. those located within the planning period. Constraints (3.20) and (3.21) correspond to the lower and upper bounds of the time slots, respectively. Similar to TIO, the upper bound $\tau_{\max}$ is given in (3.24) to allow for the

situation with no active time slots. Constraint (3.22) guarantees that each time slot is larger than the minimum length $\Delta\tau_{\min}$, while constraint (3.23) ensures that there is no overlap between the time slots. Constraint (3.25) excludes the situation of fractional time slots, where the starting instant is inside the planning period while the end instant is outside the planning period. Finally, constraint (3.26) guarantees that the resulting time slots are sufficient to perform all the maintenance interventions suggested at the high level.
The optimization problem (3.19)-(3.26) is a nonsmooth nonlinear programming problem. However, it can be converted into to an MILP problem by introducing new binary and auxiliary variables and linear constraints, following the procedure described in [12] for MLD systems. The number of binary variables in the transformed MILP problem is $2N_{\mathrm{sl}}(N_\tau+1)$. As the maximum number of time slots is in general small (say, less than 5) in practice, the resulting MILP problem can be solved exactly on a desktop PC when the number of intervals in the piecewise-constant function of disruption cost is not too large (say, less than 500, based on our computational experiments).

3.4 Low-Level Clustering Problem

The low-level problem is triggered whenever an intervention is suggested for any component by the high-level controller. A nonsmooth nonlinear programming problem is solved to determine the optimal execution plan for each active time slot determined at the middle level. The resulting execution plan groups various basic units into different clusters depending on their location and condition. Only the basic units located inside a cluster will be treated. The low-level problem determines the optimal start and end position of each cluster, trying to cover as many severely deteriorated basic units as possible inside a cluster, subject to the maintenance time slot. Similar to the middle-level problem, different low-level problems must be solved if different interventions are suggested at the high level. Moreover, for each intervention, we only consider the basic units inside the components where the corresponding intervention is prescribed. This further reduces the size of the optimization problem.
Let N^{Sq} denote the total number of basic units where the corresponding intervention is suggested by the high-level controller, while the vectors ξ and w contain the positions and conditions of these basic units respectively. Furthermore, let ξ_k and w_j denote the position and condition of the j-th basic unit. The basic units are all located inside the planning range $[\underline{\xi}, \overline{\xi}]$. Let T_s^{sl} denote the duration of the s-th resulting active time slot from the middle-level problem. Define $\mathscr{I}_1 = \{1, \ldots, N^{\mathrm{Sq}}\}$ as the set of indices that need to be treated at the first time slot. If the middle-level problem results in more than one time slot, let $\mathscr{I}_s$ denote the set of indices of the basic units still remained to be treated at time slot s for any $s > 1$. At each time slot s, the basic units are grouped into N^{cl} clusters to be treated by the given considered maintenance intervention, where $\phi_{i,s}^{\mathrm{start}}$ and $\phi_{i,s}^{\mathrm{end}}$ denote the start and end position of the i-th cluster. Let $\Delta\phi_{\min}$ and $\Delta\phi_{\max}$ denote the minimum and maximum size of a cluster, respectively. Only the basic units inside a cluster are processed by a specific machine. Let v_{on} and v_{off} represent the speed of the machine in working mode (e.g. tamping or grinding) and non-working mode (driving), respectively. Let T_{on} and T_{off} denote the switch-on and switch-off time of the machine. The following nonlinear optimization problem is formulated

to determine the clusters within the s-th time slot:

$$\max_{\{\phi_{i,s}^{\text{start}},\phi_{i,s}^{\text{end}}\}_{i=1}^{N^{\text{cl}}}} \sum_{i=1}^{N^{\text{cl}}} \sum_{j\in\mathcal{I}_s} w_j I_{\phi_{i,s}^{\text{start}}\le\xi_j\le\phi_{i,s}^{\text{end}}} + \lambda \sum_{i=1}^{N^{\text{cl}}} I_{\phi_{i,s}^{\text{end}}>\overline{\xi}} \tag{3.27}$$

subject to

$$\phi_{1,s}^{\text{start}} \ge \underline{\xi} \tag{3.28}$$

$$\phi_{N_{\text{cl}},s}^{\text{end}} \le \xi_{\max} \tag{3.29}$$

$$\Delta\phi_{\min} \le \phi_{i,s}^{\text{end}} - \phi_{i,s}^{\text{start}} \le \Delta\phi_{\max} \quad \forall i \in \{1,\dots,N_{\text{cl}}\} \tag{3.30}$$

$$\phi_{i+1,s}^{\text{start}} - \phi_{i,s}^{\text{end}} \ge \epsilon \quad \forall i \in \{1,\dots,N_{\text{cl}}-1\} \tag{3.31}$$

$$\xi_{\max} = \overline{\xi} + 2N_{\text{cl}}(\Delta\phi_{\min}+\epsilon) \tag{3.32}$$

$$\phi_{i,s}^{\text{start}} \le \overline{\xi} \iff \phi_{i,s}^{\text{end}} \le \overline{\xi} \quad \forall i \in \{1,\dots,N_{\text{cl}}\} \tag{3.33}$$

$$\sum_{i=1}^{N^{\text{cl}}} I_{\phi_{i,s}^{\text{end}}\le\overline{\xi}} \cdot \left(\frac{\phi_{i,s}^{\text{end}} - \phi_{i,s}^{\text{start}}}{v_{\text{on}}} + T_{\text{on}} + T_{\text{off}} \right) + \sum_{i=1}^{N^{\text{cl}}-1} I_{\phi_{i,s}^{\text{end}}\le\overline{\xi}} \cdot \frac{\phi_{i+1,s}^{\text{start}} - \phi_{i,s}^{\text{end}}}{v_{\text{off}}} + T_{\text{set}} \le T_s^{\text{sl}} \tag{3.34}$$

The first term in the objective function (3.27) strives to assign the most severely deteriorated basic units to a cluster, while the second term minimizes the total number of *active* clusters that are located within the planning range. Similar to constraints (3.20)-(3.25) of the middle-level problem, a TIO-like approach is used again to constraints (3.28)-(3.33). The only difference is that instead of time instants in a standard TIP approach, we consider spatial positions. The first term in constraint (3.34) calculates the time needed for the machine to treat the basic units inside all active clusters, including the switching on/off time. The second time computes the time need for the machine to drive between clusters. The summation of the two parts gives the total maintenance time, which should be less than the duration T_s^{sl}. The low-level problem (3.27)-(3.34) can be transformed into an MILP problem following the procedure described in [12]. The number of binary variables of the transformed MILP can be as large as $2N_{\text{cl}}N^{\text{Sq}} + 2N_{\text{cl}}$, which can be huge for a long track line with a large number of basic units. However, based on the computational experiments in the case study, for a short track line (e.g. 25 km) with a moderate number of basic units (e.g. less than 500 squats) and a small number of available clusters (e.g. less than 5), the resulting MILP is still tractable.
The resulting optimal clusters might not cover all the required basic units due to lack of maintenance time. Let $\mathcal{R}$ denote the set of the indices of all the remaining squats not covered by any cluster in any active time slot. Define

$$v_{\text{rem}} = \frac{\sum_{j\in\mathcal{R}} w_j}{\sum_{j=1}^{N^{\text{Sq}}} w_j} \tag{3.35}$$

$$\Delta\hat{T} = \frac{|\mathcal{R}|}{N^{\text{Sq}}} \cdot \frac{\overline{\xi} - \underline{\xi}}{v_{\text{on}}}, \tag{3.36}$$

where v_{rem} measures the ratio of the accumulated condition of the remaining basic units over the total condition of all the basic units, while $\Delta\hat{T}$ is an estimate of the additional

Table 3.1: Evolution of squats of different categories and with different uncertain growth rate. The length that a squat has evolved into after one month is calculated from its current length L (in mm).

Realization of uncertainties	Light squat ($L < 30$)	Medium squat ($30 \leq L \leq 50$)	Severe squat ($L > 50$)
Fast growth	$1.016 \cdot L + 1.2809$	$1.1029 \cdot L - 1.6725$	$L + 4.5$
Average growth	$1.0017 \cdot L + 0.9959$	$1.0699 \cdot L - 1.2165$	$1.0008 \cdot L + 2.6694$
Slow growth	$0.9915 \cdot L + 0.681$	$1.016 \cdot L - 0.0801$	$0.9949 \cdot L + 1.0127$

maintenance time to cover the remaining basic units. The values of v_{rem} and $\Delta\hat{T}$ determine whether to leave the remaining basic units uncovered, or treat them with additional maintenance time, as stated in Section 3.1.

3.5 Case Study

3.5.1 Setup

A case study on the optimal treatment of squats is performed for the Eindhoven-Weert line in the Dutch railway network. This line is approximately 25 km long, and we divide it into five sections of equal length, as shown in Figure 3.6. The rail of the Eindhoven-Weert line is considered as the infrastructure in this case study, and the five sections of rail are treated as components with independent deterioration dynamics. The basic units are the 454 individual squats located on the entire rail from historical data. A squat is a typical rail contact fatigue, and its evolution depends on the dynamic contact between wheels and rails. Squats are classified into different categories depending on their visual length[1], which can be detected automatically using techniques like axle box acceleration (ABA) systems [92, 105], eddy current testing [153], and ultrasonic surface waves [48]. In this thesis, squats with a visual length below 30 mm are considered as light squats, in which cracks have not appeared yet. Squats with a visual length ranging from 30 to 50 mm are considered to be at the medium stage of growth. The medium squats evolve into severe squats when the network of cracks spreads further. Squats with a visual length over 50 mm are considered as severe squats. They should be treated as early as possible because they might lead to hazards like derailment.

Simulation Model

We call the deterioration model of one individual squat the *simulation model*, which is a piecewise-affine function fitted using historical data of the Eindhoven-Weert line. The length of each individual squat is updated by the simulation model at each time step. Three realizations of the uncertainties are considered. They are fast, average, and slow growth, with a probability of occurrence of 0.3, 0.4, and 0.3, respectively. Let L denote the length of an individual squat; the natural evolution (without any maintenance interventions), which describes the length of the squat after one month, is given in Table 3.1.

A squat can be treated effectively by grinding only when its length is less than the effective

[1]The length noticeable at the surface of the rail.

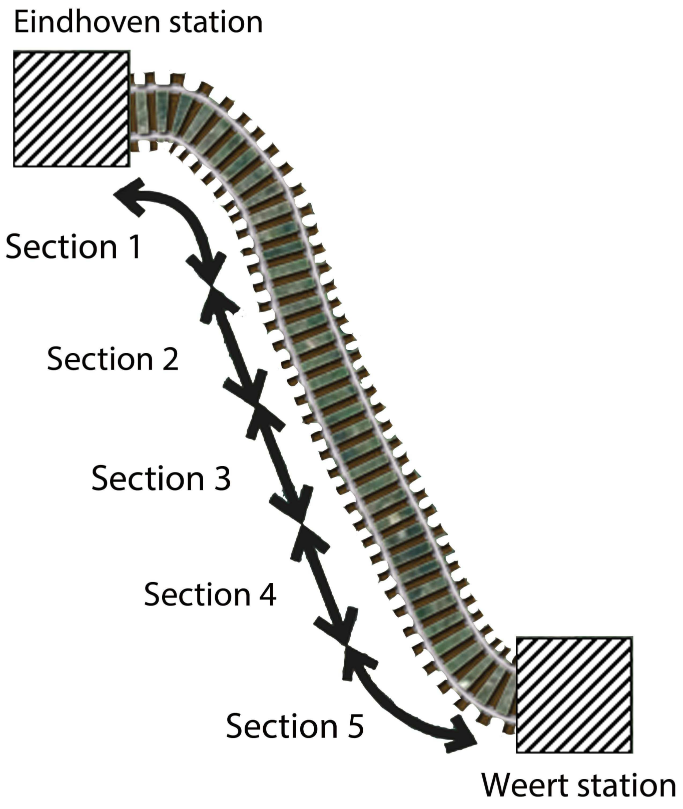
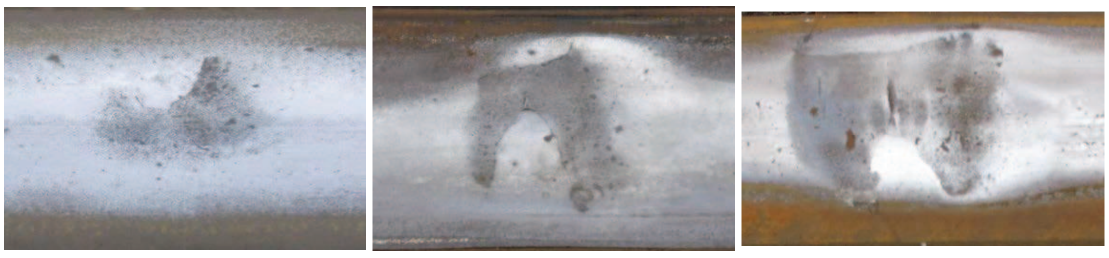


Figure 3.6: Eindhoven-Weert divided into five sections.



(a) Light squat. *(b) Medium squat.* *(c) Severe squat.*

Figure 3.7: Squats with different severities.

Table 3.2: Effect of grinding for squats of different categories and with different uncertain growth rates. The length of a squat after grindng can be calculted from its current length L (in mm).

Realization of uncertainties	$L \leq 16$	$L > 16$
Fast	0	$1.0028 \cdot (L - 16)$
Average	0	$1.0009 \cdot (L - 16)$
Slow	0	$0.9985 \cdot (L - 16)$

grinding length (16 mm). The length a squat can be reduced to by grinding is calculated using the formulas of Table 3.2.
We also consider the occurrence of new squats. At each time step (month), the number of new squats in the entire track is the rounded value of a random variable normally distributed with mean 3 and variance 1. The probability distribution of the location of a new squat is also a normal distribution with mean 12.5 km and standard deviation 4 km. New squats should always be early-stage squats when they first appear on the rail, and the initial length of a new squat follows a normal distribution with mean 15 mm and standard deviation 5 mm.

Prediction Model

The condition of one section is defined as the average length of all the squats within the section. The condition of each section is updated by aggregating the simulated squat lengths using the simulation model for each individual squat. The dynamics of the condition of one section is described by the *prediction model*, which can be obtained by piecewise-affine identification using simulated data obtained from the simulation model. The prediction model is used for long-term maintenance planning at the high level. The set $\mathscr{U} = \{0, 1, 2\}$ contains all possible maintenance actions that can be applied to one section, with 0, 1, 2 representing “performing no maintenance", “grinding", and “replacing", respectively. Note that these maintenance actions are applied to each squat in the section. The condition $x_{\text{con},j}$ is defined as the average length of squats in section j, while the auxiliary variable $x_{\text{aux},j}$ records the number of previous grinding operations on section j since the last replacement. Grinding cannot be applied an unlimited number of times, as it tries to remove a squat by reducing the thickness of the rail. We define $N_{\max}^{\text{Gr}}$ as the maximal number of grindings that can be applied to a section of rail. Three realizations of uncertainties are considered, which are collected in the set $\Theta_j = \{1, 2, 3\}$, with 1, 2, 3 representing fast, average, and slow deterioration. Following the notation of the generic deterioration model described in Section 2.2.1, the dynamics of the condition of section j can be described by the following scenario-based model:

$$\begin{aligned} x_j^{\text{con}}(k+1) &= f^{\text{con}}(x_j^{\text{con}}(k), u_j(k), \theta_j(k)) \\ &= \begin{cases} f_{\text{Deg}}(x_j^{\text{con}}(k), \theta_j(k)) & \text{if } u_j(k) = 0 \text{ (no maintenance)} \\ f_{\text{Gr}}(x_j^{\text{con}}(k), \theta_j(k)) & \text{if } u_j(k) = 1 \text{ (grinding)} \\ 0 & \text{if } u_j(k) = 2 \text{ (replacing)} \end{cases} \\ &\forall j \in \{1, \ldots, n\}. \end{aligned} \tag{3.37}$$

The natural degradation of one section is described by the function f_{Deg}, which is a piecewise-affine function in the form

$$f_{\mathrm{Deg}}(x_j^{\mathrm{con}}, \theta_j) = a_{q,\theta_j} x_j^{\mathrm{con}} + b_{q,\theta_j} \quad \text{if } x_j^{\mathrm{con}} \in \mathscr{X}_{j,q}^{\mathrm{con}} \subset \mathscr{X}_j^{\mathrm{con}}, \tag{3.38}$$

where the condition space of section j is partitioned as $\{\mathscr{X}_{j,q}\}_{q=1}^{3}$. As discussed in [77], light (early-stage), medium (middle-stage), and severe (late-stage) squats exhibit different deterioration dynamics, and a piecewise-affine function is able to capture the deterioration dynamics of the squats.
The function f_{Gr} captures the effect of grinding, which becomes less effective when the condition deteriorates more severely. It is also a piecewise-affine function of the form

$$f_{\mathrm{Gr}}(x_j,\theta_j) = \begin{cases} 0 & \text{if } x_j \le x_{\theta_j}^{\mathrm{eff}} \\ \psi_{\theta_j}(x_j - x_{\theta_j}^{\mathrm{eff}}) & \text{if } x_j > x_{\theta_j}^{\mathrm{eff}} \end{cases} \tag{3.39}$$

where $x_{\theta_j}^{\mathrm{eff}}$ represents the effective condition for grinding if θ_j is realized.
The sampling time T in this case study is one month. The prediction horizon N_{P} and control horizon N_{C} are both 6 months. The predicted condition of each section must be kept below a maintenance limit within the prediction window at every time step, which can be expressed by the following constraint:

$$x_j^{\mathrm{con}}(k+l) \le x_{\max} \quad \forall j \in \{1,\dots,n\}, \forall l \in \{1,\dots,N_{\mathrm{P}}\}. \tag{3.40}$$

The dynamics of the auxiliary variable x_j^{aux}, which is a counter for grinding, can be formally expressed as:

$$\begin{aligned} x_j^{\mathrm{aux}}(k+1) &= f^{\mathrm{aux}}(x_j^{\mathrm{aux}}(k), u_j(k)) \\ &= \begin{cases} x_j^{\mathrm{aux}} & \text{if } u_j(k) = 0 \text{ (no maintenance)} \\ x_j^{\mathrm{aux}} + 1 & \text{if } u_j(k) = 1 \text{ (grinding)} \\ 0 & \text{if } u_j(k) = 2 \text{ (replacing)} \end{cases} \\ &\forall j \in \{1,\dots,n\}. \end{aligned} \tag{3.41}$$

The number of grinding operations cannot exceed a maximum number $N_{\max}^{\mathrm{Gr}}$ within the prediction window, thus we have:

$$x_j^{\mathrm{aux}}(k+l) \le N_{\max}^{\mathrm{Gr}} \quad \forall j \in \{1,\dots,n\}, \forall l \in \{1,\dots,N_{\mathrm{P}}\}. \tag{3.42}$$

In summary, equations (3.37),(3.41), together with constraints (3.40),(3.42) form the stochastic deterioration model for this case study. A scenario-based approach is applied to this stochastic model. Three representative scenarios are selected from the set $\tilde{\Theta}$ containing all possible realizations of uncertainties within the prediction period. They correspond to fast, average, slow deterioration for all sections at every time step within the prediction period.
The same prediction model with the same parameters is used for each section. The condition space $\mathscr{X}_j^{\mathrm{con}} = [0, 70]$ for section j is partitioned into the following three intervals:

$$\mathscr{X}_{j,1}^{\mathrm{con}} = [0, 30), \quad \mathscr{X}_{j,2}^{\mathrm{con}} = [30, 50), \quad \mathscr{X}_{j,3}^{\mathrm{con}} = [50, 70].$$

The parameters for the piecewise-affine degradation function (3.38) are collected in the following two matrices:

$$A_j = (a_{q,\theta_j}) = \begin{bmatrix} 1.0037 & 1.0073 & 1.0120 \\ 1.0017 & 1.0053 & 1.0075 \\ 0.9992 & 1.0007 & 1.0008 \end{bmatrix}, B_j = (b_{q,\theta_j}) = \begin{bmatrix} 0.1954 & 0.1484 & 0 \\ 0.1438 & 0.0732 & 0 \\ 0.1041 & 0.0701 & 0.0743 \end{bmatrix}$$
$$\forall j \in \{1, \dots, n\}.$$

The parameters for the grinding model (3.39) are collected in the two matrices

$$\Psi_j = (\psi_{\theta_j}) = \begin{bmatrix} 0.9996 \\ 0.9996 \\ 0.9995 \end{bmatrix}, X_j^{\text{eff}} = (x_{\theta_j}^{\text{eff}}) = \begin{bmatrix} 11.8275 \\ 11.9586 \\ 11.9336 \end{bmatrix}$$
$$\forall j \in \{1, \dots, n\}.$$

The initial condition $x(0) = [(x^{\text{con}}(0))^{\text{T}} \, (x^{\text{aux}}(0))^{\text{T}}]^{\text{T}}$ is given by

$$x^{\text{con}}(0) = \begin{bmatrix} 23.8757 \\ 24.356 \\ 27.7457 \\ 26.0526 \\ 26.0487 \end{bmatrix}, \quad x^{\text{aux}}(0) = \begin{bmatrix} 7 \\ 8 \\ 7 \\ 7 \\ 8 \end{bmatrix}.$$

The operational limit $x_{\max}$ is 40 mm, and a maximum of 10 grinding operations since the last replacement is allowed for each section, i.e. $N_{\max}^{\text{Gr}} = 10$.
The trade-off between the condition and the cost is $\lambda = 10$ in the high-level objective function (3.14), and the scaling factor μ is 70. The 1-norm is taken for J_{Deg} in (3.11), and replacement is 30 times as expensive as grinding, i.e. $\gamma_{j,1} = 1$ and $\gamma_{j,2} = 30$ in (3.12). The maintenance limit is $x_{\max}^{\text{con}} = 40$ mm.

Parameters for Middle-Level and Low-Level Problems

The hourly cost of traffic disruption is determined by the number of passenger trains[2] from Eindhoven to Weert per hour. Only regular schedules (excluding special schedules for holidays) are considered. The train schedule is periodic with a period of one week. Moreover, all workdays (Monday-Fridy) have the same schedule, while Saturdays and Sundays have different schedules.
Three levels of hourly disruption cost, high cost ($c_{\text{H}} = 10$), medium cost ($c_{\text{M}} = 3$), and low cost ($c_{\text{L}} = 1$) are considered, while a zero cost is assigned to traffic-free intervals. The disruption function, which specifies the hourly disruption cost for any intervals in the planning horizon, can then be described by the lookup-table presented in Table 3.3. The planning horizon is the sampling time of the high-level controller (one month), and the parameters for the middle-level problem (3.19)-(3.26) are given in Table 3.4. The estimated time for mainte-

[2]Freight trains are not considered, as there is at most one freight train from Eindhoven to Weert every workday.

Table 3.3: Hourly disruption cost for a week. Interval with no trains passing are assigned a zero cost. Intervals with less than 2 trains per hour are assigned a low hourly cost c_L. Intervals with 2-3 trains per hour are assigned a medium hourly cost c_M. Intervals with 4-6 trains per hour are assigned a high hourly cost c_H.

Interval	Workday		Saturday		Sunday	
	Number of trains	disruption cost	Number of trains	Disruption cost	Number of trains	Disruption cost
0:00 -1:00	1	c_L	2	c_L	2	c_L
1:00 -6:00	0	0	0	0	0	0
6:00 -7:00	3	c_M	0	0	0	0
7:00 -8:00	6	c_H	4	c_H	1	c_L
8:00 -13:00	6	c_H	6	c_H	5	c_H
13:00 - 14:00	5	c_H	6	c_H	5	c_H
14:00 - 19:00	6	c_H	6	c_H	5	c_H
19:00 - 20:00	5	c_H	5	c_H	4	c_H
20:00 - 24:00	3	c_M	3	c_M	3	c_M

Table 3.4: Parameters for the middle-level optimization problem

Parameter	Explanation	Value
N_sl	Maximum number of time slots	2
c_sl	Setup cost of one time slot	1
λ_sl	Trade-off between disruption and setup cost	10 (representative run) 1 (supplementary run)
T_set	Setup time for one time slot	1 h
$\Delta\tau_\mathrm{min}$	Minimum size of a time slot	5 h

nance $\hat{T}_\mathrm{Maint}$ is calculated from the interventions suggested by the high-level controller, i.e.

$$\hat{T}_\mathrm{Maint} = T_\mathrm{Sec} \sum_{j=1}^{n} I_{u_j=1} \tag{3.43}$$

where $T_\mathrm{Sec} = 2.5$ is the estimated time (in hours) to grind one section.

As only squats located in the section where grinding is suggested by the high-level controller are considered in the low-level optimization problem, the number (N_Sq) and the locations (ξ) of the squats are not fixed. However, we have $N_\mathrm{Sq} \leq 454$ as there are 454 existing squats on the whole track. The vector w containing the lengths of the considered squats also changes at each time step. The parameters for the low-level problem (3.27)-(3.34) are given in Table 3.5. Only one slot is used in the middle-level problem, thus $s = 1$. The setup time T_set in constraint (3.34) is the same as in the middle-level problem, and the actual length of the time slot T_s^sl is the one obtained from the results of the middle-level problem (3.19)-(3.26).
The multi-level approach is implemented in Matlab R2016b, on a desktop computer with an Intel Xeon E5-1620 eight-core CPU and 64 GB of RAM, running a 64-bit version of SUSE Linux Enterprise Desktop 12. The nonlinear optimization problem at each time step of the

Table 3.5: Parameters for the low-level optimization problem

Parameter	Explanation	Value
N_{cl}	Maximum number of clusters	3
λ_{cl}	Penalty on the number of active clusters	1
$\Delta\phi_{\min}$	Minimum cluster size	1 km
$\Delta\phi_{\max}$	Maximum cluster size	25 km
T_{on}	Switch-on time of grinding machine	15 min
T_{off}	Switch-off time of grinding machine	15 min
v_{on}	Grinding speed	2.2 km/h (representative run) 1.6 km/h (supplementary run)
v_{off}	Driving speed of grinding machine	80 km/h

high-level MPC controller is solved using the function *patternsearch* of the Matlab Global Optimization Toolbox, with 100 random starting points. CPLEX 12.5 (called via Tomlab 8.0) is used as the MILP solver for the middle-level and the low-level problems.

3.5.2 Results & Discussion

The multi-level, scenario-based, chance-constrained approach is illustrated by a representative run with a five-year planning horizon. The optimal maintenance interventions suggested by the high-level MPC controller are shown in Figure 4.3b, and the simulated condition of each section is shown in Figure 4.3a. Note that the state at each time step is computed from the individual squat lengths simulated by the simulation model, and only the cluster-wise grinding plan from the low level is applied to the simulation model. Grinding is suggested when the condition is near its maintenance limit (40 mm), and the interval between two consecutive grinding turns out to be between 10 to 18 months for a section. It is interesting to note that the interval between two consecutive grindings becomes shorter over time, as shown in the resulting intervention plan of section 1, 3, and 4 in Figure 4.3b. This is because grinding becomes less effective the more it is applied to the same section of rail. When the maximum number of grindings (10 in the case study) is reached for any section, replacing is suggested. The mean and maximum CPU time to solve the MPC optimization problem at each time step is 47 s and 68 s, respectively, which is much faster than the sampling time (one month). Moreover, both the middle-level and the low-level problem can be solved to global optimality within 10 s. Thus we can claim that the proposed multi-level MPC approach is implementable in real-time.

The results of the middle-level and low-level problem are shown in Figure 3.9. The middle-level and low-level problems are triggered whenever grinding is suggested for any of the five sections. As shown in Table 3.3 in Appendix B, a 5-hour traffic free time slot (1:00-6:00) is available for workdays, and a 6-hour traffic free time slot (1:00-7:00) is available for weekends. As shown in Figure 3.9, only one section is to be ground at time step 9, 11, 12, 21, 25, 27, 28, 30, 35, 37, 42, according to the high-level MPC controller, and all the squats inside the single section can be covered in one cluster, using a 5-hour short time slot on a weekday (e.g. Monday). Two non-consecutive sections are suggested at time step 10 at the high level, and even with a 6-hour long time slot on weekends (e.g. Sunday), there is still one squat not covered by the resulting two clusters. This is because the benefit of covering this single medium-stage squat is less than the cost associated with the additional maintenance time to cover it.

3.5.3 Comparison with Alternative Approaches

Three alternative approaches for maintenance planning: nominal MPC, the cyclic approach, and the approach currently used in practice, are implemented for comparison with the proposed approach. The nominal MPC approach can be viewed as the deterministic counterpart of the scenario-based chance-constrained MPC approach, as it considers only the average deterioration dynamics in the optimization problem at each time step. The cyclic approach is a preventive maintenance strategy that performs grinding and replacement at regular intervals. Here we briefly explain the formulation of the offline optimization problem to obtain the optimal intervals for grinding and replacement for the cyclic approach. Let $t_{0,j}$ denote the time instant at which the first grinding is applied to the j-th section. Let $T_{\mathrm{Gr},j}$ denote the period of grinding for section j. Replacing is usually performed after a multiple of consecutive grindings, e.g. a section of rail is replaced after 5 grindings. We denote this multiple by a scalar r, which is the same for all sections. Define $t_0 = [t_{0,1} \dots t_{1,n}]^{\mathrm{T}}$ and $T_{\mathrm{Gr}} = [T_{\mathrm{Gr},1} \dots T_{\mathrm{Gr},n}]^{\mathrm{T}}$. Then the cyclic maintenance optimization problem can be formulated as:

$$\min_{t_0, T_{\mathrm{Gr}}, r} \sum_{k=1}^{k_{\mathrm{end}}} \sum_{j=1}^{n} x_j^{\mathrm{con}}(k) + \lambda(\gamma_{j,1} I_{u_j(k)=1} + \gamma_{j,2} I_{u_j(k)=2}) \tag{3.44}$$

subject to

$$x_j^{\mathrm{con}}(k+1) = f^{\mathrm{con}}(x_j^{\mathrm{con}}(k), u_j(k), 2) \tag{3.45}$$

$$x_j^{\mathrm{con}}(k) \leq x_{\max} \tag{3.46}$$

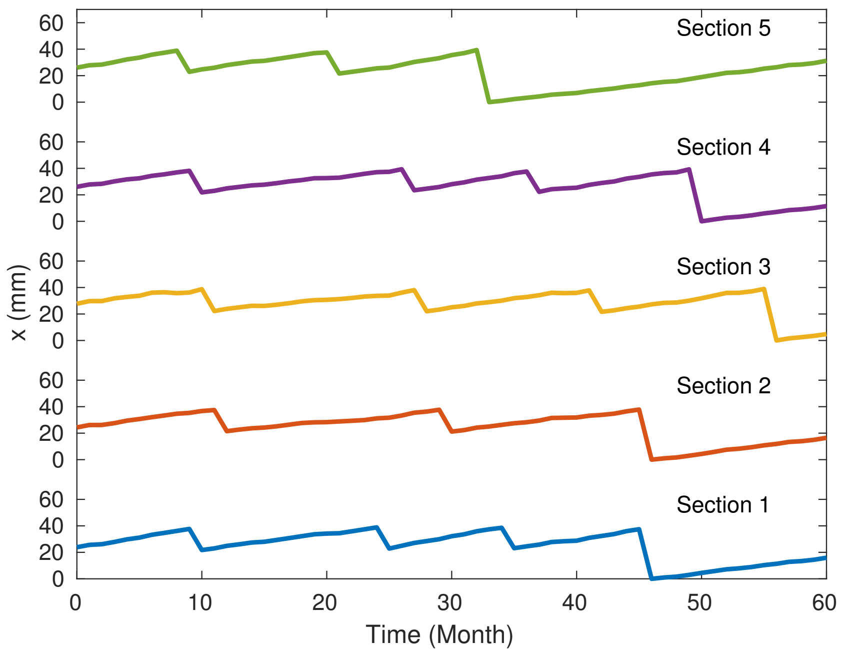
$$u_j(k) = \begin{cases} 1, \text{ if } k = t_{0,j} \text{ or } (k - t_{0,j}) \bmod \mathrm{round}(T_{\mathrm{Gr},j}) = 0 \\ 2, \text{ if } (k - t_{0,j}) \bmod \mathrm{round}(r T_{\mathrm{Gr},j}) = 0 \\ 0, \text{ otherwise} \end{cases} \tag{3.47}$$

$$t_0^{\min} \leq t_0 \leq t_0^{\max} \tag{3.48}$$

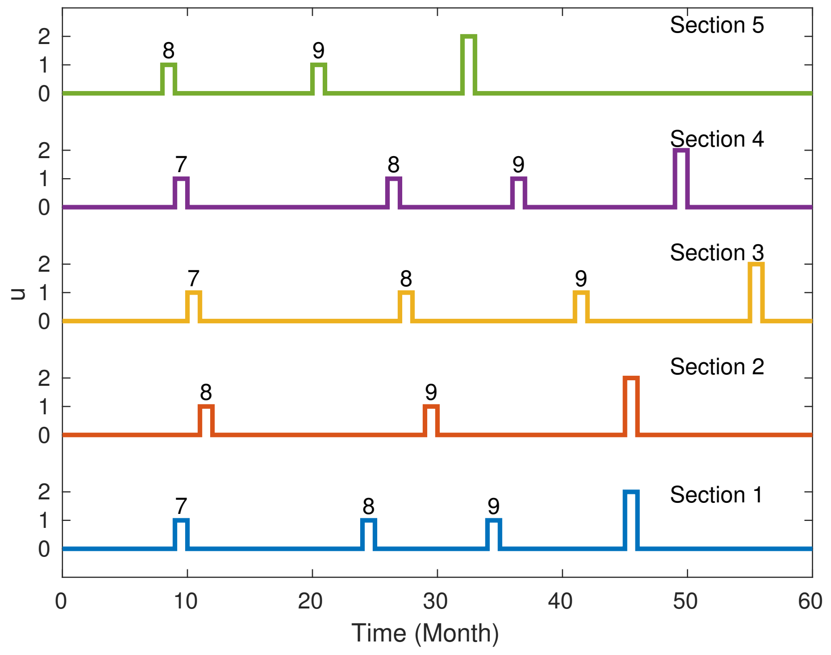
$$T_{\mathrm{Gr}}^{\min} \leq T_{\mathrm{Gr}} \leq T_{\mathrm{Gr}}^{\max} \tag{3.49}$$

$$2 \leq r \leq r_{\max} \tag{3.50}$$

for all $j \in \{1, \dots, n\}$ and $k \in \{1, \dots, k_{\mathrm{end}}\}$.
The objective (3.44) corresponds to minimizing the accumulated condition degradation and intervention cost for the entire track over the five-year planning horizon. Only the average (nominal) deterioration rate ($\theta(k) = 2$ for all k) is considered, as shown in (3.45). Constraint (3.46) guarantees that the nominal condition will not exceed the maintenance limit for the whole planning horizon. Equation (3.47) converts the grinding and replacing periods into interventions. Constraints (3.48) and (3.50) are upper and lower bounds for the decision variables. The offline cyclic maintenance optimization problem (3.44)-(3.50) is a nonsmooth optimization problem, which is solved using multi-start pattern search in the case study.
We also implement the approach currently used in the Netherlands. We refer to this ap-



(a) Average squat length per section calculated from the individual squat dynamics.



(b) Interventions suggested by the high-level MPC controller. The number above each grinding intervention indicates the number of previous grinding operations applied to the section since the last replacement.

Figure 3.8: Simulated average squat length and interventions suggested by the high-level MPC controller for each of the five sections of the entire track for a representative run.

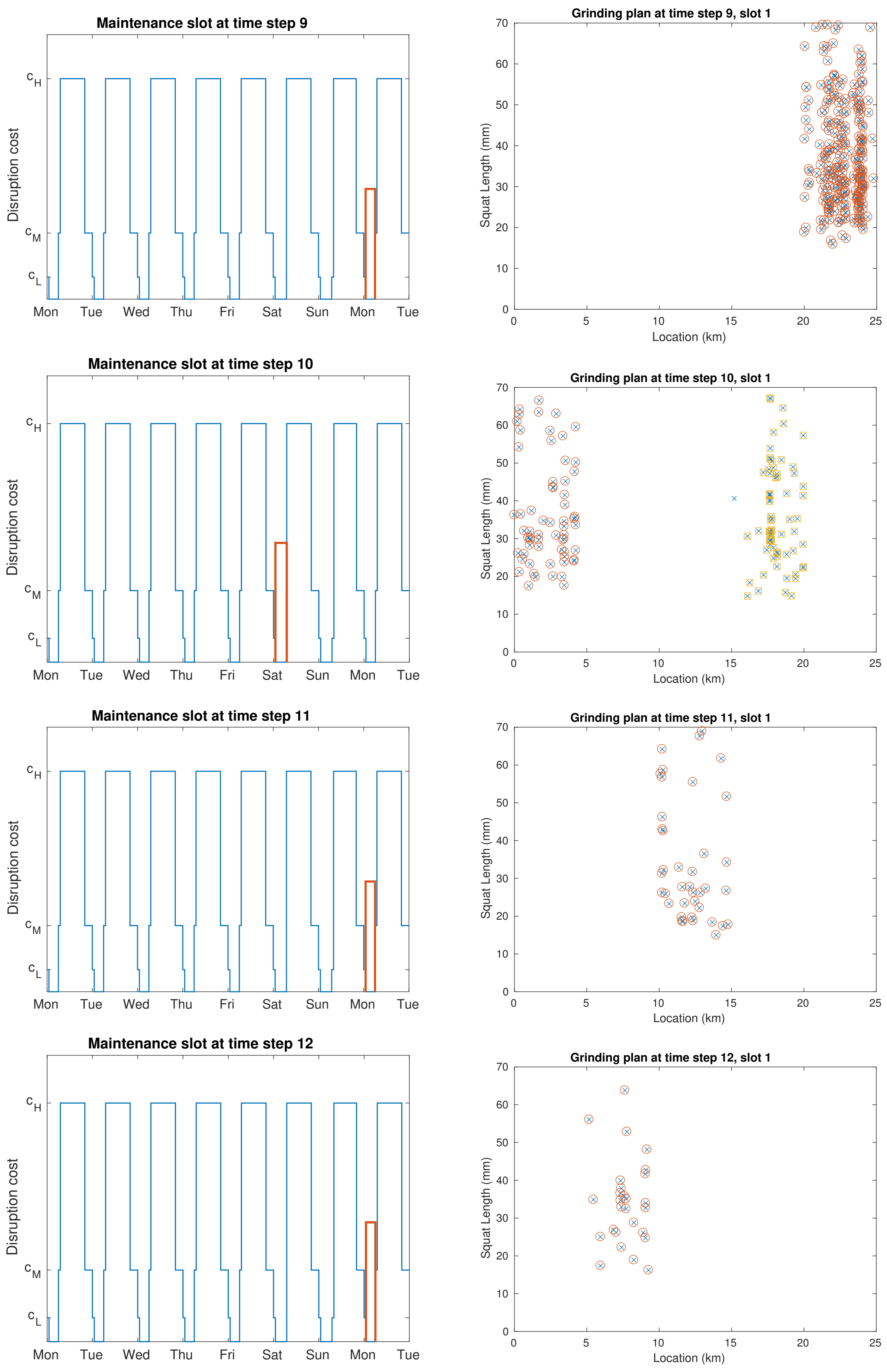


Figure 3.9: Results of the middle-level problem and the low-level problem at time step 9, 10, 11, and 12 of the high level. A squat (represented by a dot) is in a cluster if it is covered by a circle (first cluster) or square (second cluster).

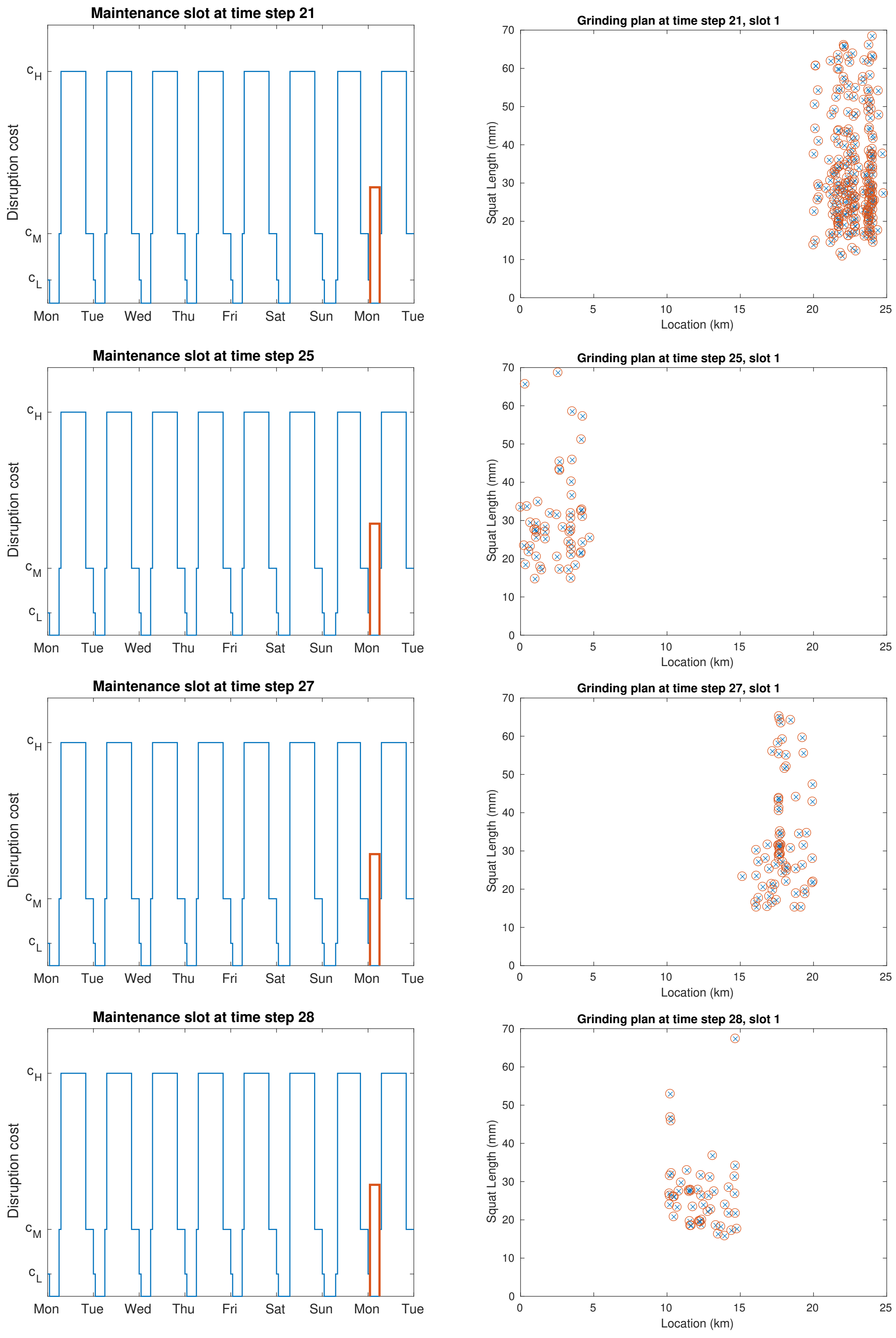


Figure 3.9: Results of the middle-level problem and the low-level problem at time step 21, 25, 27, and 28 of the high level. A squat (represented by a dot) is in a cluster if it is covered by a circle (first cluster) or square (second cluster).

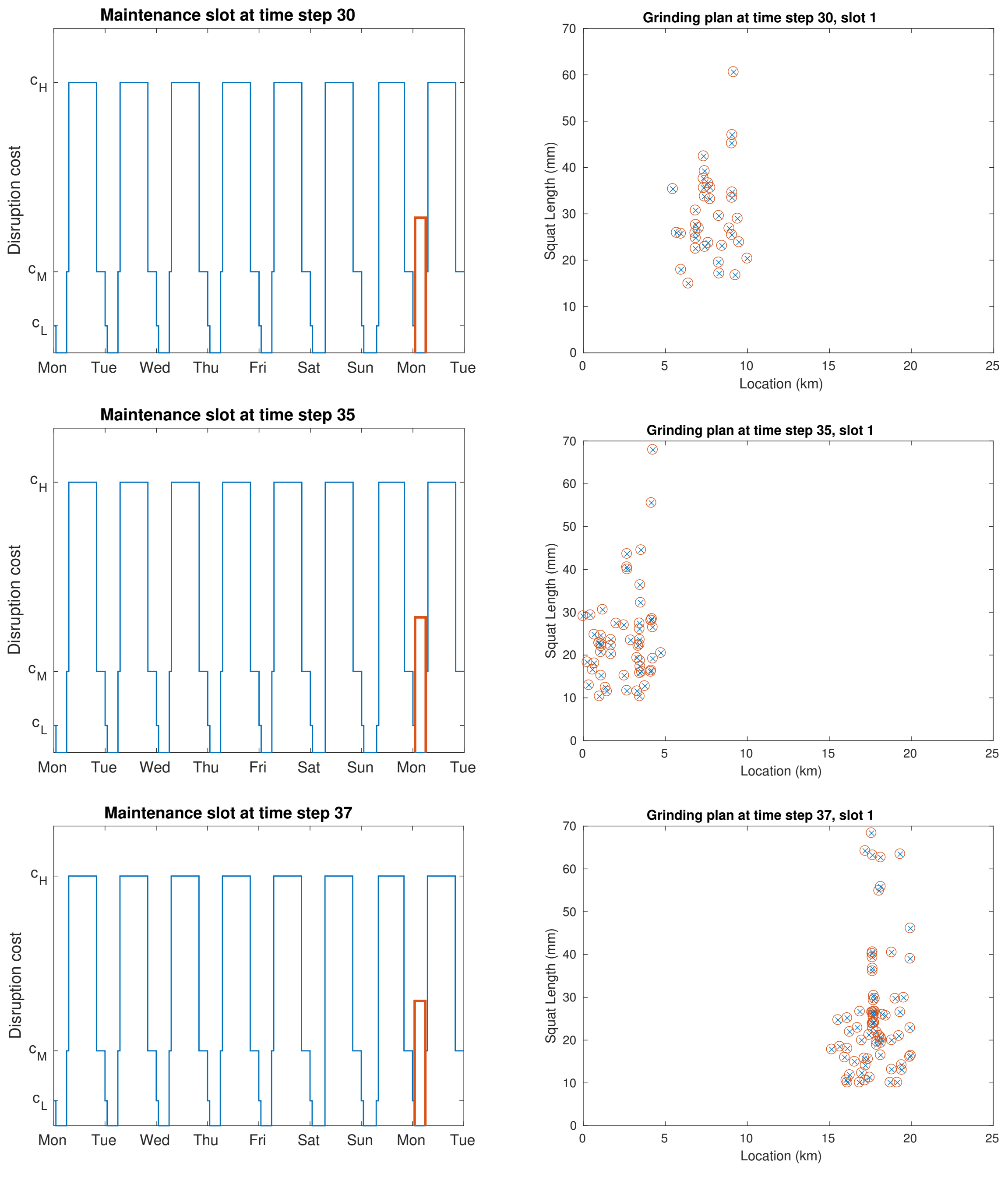


Figure 3.9: Results of the middle-level problem and the low-level problem at time step 30, 35, and 37 of the high level. A squat (represented by a dot) is in a cluster if it is covered by a circle (first cluster) and square (second cluster).

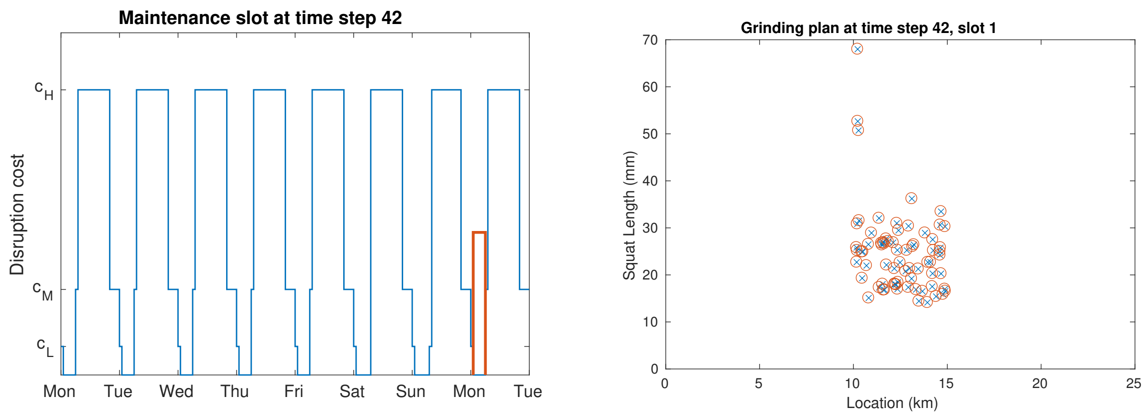


Figure 3.9: Results of the middle-level problem and the low-level problem at time step 42 of the high level. A squat (represented by a dot) is in a cluster if it is covered by a circle (first cluster) and square (second cluster).

Table 3.6: Sequences of the realizations of uncertainties, where 1, 2, 3 stands for fast, average, slow growth for every squat, respectively. For easy implementation only the first 10 entries of each sequence are given; the complete sequence for the entire five-year planning horizon can be obtained by repeating the 10 entires 6 times.

Run	Sequence of the realizations of uncertainties
1	1, 1, 1, 2, 2, 2, 2, 3, 3, 3, ...
2	1, 2, 2, 2, 2, 2, 2, 3, 3, 3, ...
3	1, 1, 1, 1, 2, 2, 3, 3, 3, 3, ...
4	1, 1, 2, 2, 3, 3, 3, 3, 3, 3, ...
5	1, 1, 1, 2, 2, 2, 3, 3, 3, 3, ...
6	3, 1, 1, 3, 2, 3, 2, 1, 2, 2, ...
7	2, 1, 3, 3, 2, 2, 1, 3, 2, 1, ...
8	1, 2, 3, 1, 3, 2, 2, 1, 3, 2, ...
9	1, 3, 2, 3, 2, 1, 2, 2, 3, 1, ...
10	1, 3, 2, 1, 3, 1, 2, 2, 1, 3, ...

proach as “current approach", which is a cyclic preventive maintenance approach that grinds the entire line every six months.

The four maintenance approaches are applied to ten test runs of a five-year planning horizon with different pre-defined sequences of realizations of uncertainties in the simulation model. Each sequence specifies the realization of uncertainties for each individual squat at each time step, i.e. each month, in the five-year planning period. For easy reproduction of our results, we set the realization of uncertainties to be the same for all the squats in the entire track at each time step. In this way, only one entry is needed at each time step to specify the realization of uncertainties for all the squats. We consider a total of ten runs, where for each run the 60 entries of the realization of the uncertainty are obtained by repeating the first 10 entries listed in Table 3.6.

We compare the constraint violation, the value of the closed-loop objective function for the three approaches, and CPU time of the two MPC controllers. The constraint violation is measured by:

$$v = \max_{\substack{j=1,\dots,n \\ k=1,\dots,k_{\text{end}}}} \left\{ \frac{x_j^{\text{con}}(k) - x_{\max}^{\text{con}}}{\overline{x}^{\text{con}}} \right\} \tag{3.51}$$

where $x_{\max}^{\text{con}} = 40$ mm is the maintenance limit, and $\overline{x}^{\text{con}} = 70$ mm is the range of the condition. Let v_{Nom}, v_{CC}, v_{Cyc}, and v_{Cur} denote the maximum constraint violation of the nominal MPC controller, the scenario-based chance-constrained MPC controller, the cyclic approach, and the current approach, respectively. The maximum constraint violation measures the robustness of each approach. Similarly, let J_{Nom}, J_{CC}, J_{Cyc}, and J_{Cur} denote the closed-loop objective function value[3] for the nominal MPC controller, the scenario-based chance-constrained MPC controller, the cyclic approach, and the current approach, respectively. The closed-loop objective function value measures the cost-efficiency of each approach. A lower value indicates higher cost-efficiency. Moreover, let T_{Nom} and T_{CC} denote the total CPU time[4] for the nominal and chance-constrained MPC controller, respectively. As our goal is a safe but non-conservative maintenance strategy that is tractable, robustness and cost-efficiency are more important evaluation criteria than CPU time. The performance and computational effort of the four maintenance approaches are compared in Table 3.7.

The current approach, i.e. grinding every six months, is the most conservative approach. Although it has no constraint violation for the ten test runs, the closed-loop objective function value is almost three time as much as that of the nominal MPC approach. The cyclic approach, which can be viewed as an improvement on the over-conservative current approach by optimizing the intervals for grinding and replacing, is more cost-efficient than the current approach, as its closed-loop objective function value is only slightly higher than that of the two MPC approaches, except in Run 4. The cyclic approach is not robust, as it results in constraint violations in seven out of the ten test runs. The advantage of the cyclic approach is that it is less computationally demanding than the MPC approaches, as only one optimization problem needs to be solved offline.

Theoretically, both MPC controllers can have constraint violations in the whole planning period. Although more robust than nominal MPC, scenario-based chance-constrained MPC only guarantees that the constraints are satisfied with a possibility higher than a given confidence level (90% in this case study). But there is no constraint violation for the scenario-based chance-constrained MPC in the ten runs of simulation, as shown in Table 3.7, while the constraints are violated for nominal MPC in seven out of the ten runs. The largest constraint violation for the nominal MPC approach is 0.78% for Run 3, which might lead to hazards like derailment. The closed-loop objective function values of the two MPC approaches are almost equal in every test run, and the two MPC approaches are the most cost-efficient among the four approaches. It is interesting to note that the closed-loop objective function values of the two MPC approaches in Run 4 are only half of those of the other runs, showing a significant advantage over the cyclic approach and the current approach in terms of cost-efficiency. This is because MPC is a flexible real-time decision making scheme that can adapt the intervention plan to the actual (or estimated) condition of the infrastructure, while the cyclic and the current approaches are both offline schemes that do not take the actual (or estimated) condition into consideration.

[3]This is obtained by evaluating (3.14) over the entire planning horizon.

[4]We measure only the CPU time to solve all the optimization problems within the planning horizon.

Table 3.7: Comparison between nominal MPC controller, the scenario-based chance-constrained MPC controller, the cyclic approach, and the current strategy for ten runs with a pre-defined sequence of uncertainties. The CPU time to solve the offline optimization problem for the cyclic approach is 7.5 mininutes.

Run	Constraint violation				Closed-loop performance				CPU time (h)	
	v_{Nom}	v_{CC}	v_{Cyc}	v_{Cur}	$\frac{J_{\text{Nom}}}{J_{\text{Cur}}}$	$\frac{J_{\text{CC}}}{J_{\text{Cur}}}$	$\frac{J_{\text{Cyc}}}{J_{\text{Cur}}}$	J_{Cur}	T_{Nom}	T_{CC}
1	0.62%	0	0.26%	0	35.37%	35.33%	36.12%	344400	0.39	0.87
2	0	0	0	0	35.53%	35.45%	36.00%	344320	0.40	0.89
3	0.78%	0	0.08%	0	35.39%	35.10%	36.12%	344420	0.38	0.89
4	0.27%	0	0	0	17.46%	23.17%	35.86%	344260	0.37	0.85
5	0.51%	0	0	0	35.48%	35.39%	36.05%	344380	0.38	0.88
6	0	0	0.61%	0	35.38%	35.33%	36.11%	344370	0.40	0.90
7	0.69%	0	0.69%	0	35.37%	35.33%	36.107%	344370	0.40	0.88
8	0	0	0.55%	0	35.12%	35.08%	36.10%	344390	0.38	0.89
9	0.68%	0	0.68%	0	35.17%	35.33%	36.11%	344360	0.39	0.88
10	0.51%	0	1.47%	0	35.34%	35.07%	36.19%	344390	0.42	0.89

The greatest advantage of the nominal MPC approach over the chance-constrained MPC approach is that it requires less than half of the CPU time of the chance-constrained approach. Despite being the most computationally demanding approach, from Table 3.7 we can still conclude that the chance-constrained MPC approach is the most promising one among the four alternative approaches, as robustness and cost-efficiency are more important evaluation criteria than the computational effort, since the time to compute an optimal maintenance plan is abundant for infrastructures with a slow deterioration process. Indeed, despite being the slowest one among the four approaches, the chance-constrained MPC approach is not only tractable but also real-time implementable, as the sampling time in the case study is one month, and the optimization problem at each time step of the chance-constrained MPC approach takes approximately one minute to solve.

3.6 Conclusions

In this chapter a multi-level approach for the optimal planning of maintenance interventions for railway infrastructures has been developed . A scenario-based, chance-constrained TIO-MPC controller is implemented at the high level for long-term, component-wise, condition-based planning of maintenance interventions. Both the middle-level and low-level problems are triggered whenever an intervention is suggested for any component by the high-level controller. When triggered, the middle-level problem allocates the time slots for the suggested interventions by optimizing the trade-off between total setup costs of maintenance operations and the disruption to the train traffic. The low-level problem then groups the basic units into clusters to execute the maintenance interventions suggested at the high level. This cluster-wise work plan must be performed within the time slot determined at the middle level. A case study on the optimal treatment of squats in the Eindhoven-Weert line in the

Dutch railway network is performed. The simulation results of a representative run with a five-year planning horizon show that the proposed multi-level approach is real-time implementable and provides a suitable maintenance plan. A comparison with the nominal approach further demonstrates the advantage of the proposed chance-constrained approach in keeping the condition below the maintenance limit.

Chapter 4

Distributed Maintenance Optimization of Large-Scale Railway Networks

4.1 Problem Description

We consider the optimal long-term condition-based maintenance planning and short-term maintenance crew scheduling of a railway network composed of multiple stations and lines, where a line is defined as the part of track between two stations. Each line is further divided into multiple sections. The degradation level of each section is represented by its condition, which is aggregated from measurements of individual defects, e.g. visual lengths and crack depths of a squat. Different types of maintenance interventions, with different effects and costs, can be applied to improve the condition of a section. In this chapter, for each section, a discrete-time deterioration model is developed to describe the deterioration process of its condition. The sampling time is usually long (at least one month), due to the slow deterioration dynamics of railway infrastructures. Various parameter uncertainties (e.g. random degradation rate) are taken into account in the stochastic deterioration model.

Each type of maintenance intervention is performed by a specific maintenance crew. We define a maintenance operation as a round tour of the maintenance crew departing from and returning to a maintenance base, where the heavy machineries, like a grinding machine, can be stored. We also consider a fixed setup cost, including the cost of machinery and personnel, for each maintenance operation. Furthermore, we define a time period, which usually ranges from one week to one month, as the smallest time unit a maintenance operation can be performed. Each type of maintenance intervention has also a time budget, which specifies the maximal track possession time allocated to this specific maintenance intervention per time period. We consider flexible time budgets, i.e. in addition to the given time budgets, the maintenance contractor can request extra maintenance time with additional costs from the infrastructure manager. We develop an integrated multi-level approach that covers both the long-term condition-based maintenance planning, and the short-term scheduling of maintenance crews, for a large-scale railway network. A maintenance intervention planning problem, and maintenance crew scheduling problems, are solved at the high level and low level, respectively. Based on the stochastic deterioration model, a Model Predictive Control (MPC) [26, 132] approach is developed at the high level to determine the optimal maintenance intervention plan for the whole network, minimizing condition deterioration and maintenance costs over a prediction horizon. The MPC optimization problem is formulated as a chance-constrained optimization problem to keep the condition deterioration under a given threshold with a probabilistic guarantee. To improve the scalability of the proposed approach, a distributed optimization scheme is applied to solve the MPC optimization problem containing both continuous and discrete decision variables. The low-level maintenance crew scheduling problem is triggered whenever the corresponding intervention is suggested by the high-level controller, and its planning horizon is equal to the high-level sampling time. The objective of the low-level problem is to minimize the total setup costs of maintenance operations, the total travel costs of the maintenance crew, and the penalty costs associated with additional maintenance time (if there is any), over the whole network, while ensuring the planned intervention is completed before the next sampling time step.

4.2 High-level Maintenance Intervention Planning

We consider the optimal maintenance intervention planning for a network of railway track divided into n sections over a given planning horizon. Each section is viewed as a subsystem. The subsystems are coupled by global resource constraints, e.g. limited track possession hours. We use the deterioration model of Section 2.2.1 to describe the deterioration dynamics of each section j.

4.2.1 Chance-Constrained MPC

In this section we present the chance-constrained MPC optimization problem for each subsystem, and apply the scenario-based robust approach developed in [100] to approximate each local chance-constrained MPC problem with a deterministic problem. Let N_p denote the prediction horizon, and define:

$$\begin{aligned}
\tilde{x}_{j,k} &= [\hat{x}^\mathrm{T}_{j,k+1|k} \ldots \hat{x}^\mathrm{T}_{j,k+N_\mathrm{p}|k}]^\mathrm{T}\\
\tilde{u}_{j,k} &= [u_{j,k} \ldots u_{j,k+N_\mathrm{p}-1}]^\mathrm{T}\\
\tilde{\theta}_{j,k} &= [\theta^\mathrm{T}_{j,k} \ldots \theta^\mathrm{T}_{j,k+N_\mathrm{p}-1}]^\mathrm{T},
\end{aligned}$$

where $\hat{x}_{j,k+l|k} = [\hat{x}^\mathrm{con}_{j,k+l|k}\, \hat{x}^\mathrm{aux}_{j,k+l|k}]^\mathrm{T}$ denotes the estimated state of subsystem j at time step $k+l$, based on information available at time step k. The vectors $\tilde{x}^\mathrm{con}_{j,k}$ and $\tilde{x}^\mathrm{aux}_{j,k}$ are defined similarly as $\tilde{x}_{j,k}$. The estimated state $\hat{x}_{j,k+l|k}$ can be calculated recursively using (2.1), and $\tilde{x}_{j,k}$ can be viewed as a function that depends on $\tilde{u}_{j,k}$, $\tilde{\theta}_{j,k}$ and that is parametrized by the current state $x_{j,k}$, i.e.

$$\tilde{x}_{j,k} = \tilde{f}_j(\tilde{u}_{j,k}, \tilde{\theta}_{j,k}; x_{j,k}). \tag{4.1}$$

The objective of each local MPC controller is to minimize the trade-off between condition deterioration and maintenance costs within the prediction window, i.e.

$$J_j(\tilde{x}_{j,k}, \tilde{u}_{j,k}) = J^\mathrm{Deg}_j(\tilde{x}_{j,k}) + \phi_j J^\mathrm{Maint}_j(\tilde{u}_{j,k}), \tag{4.2}$$

in which

$$J^\mathrm{Deg}_j(\tilde{x}_{j,k}) = \|P\tilde{x}_{j,k}\|_1, \tag{4.3}$$

and

$$J_j^{\text{Maint}}(\tilde{u}_{j,k}) = \sum_{l=0}^{N_\text{P}-1} \sum_{q=1}^{N} c_{q,j}^{\text{Maint}} I_{u_{j,k+l}=q}. \tag{4.4}$$

The parameter ϕ_j captures the trade-off between condition deterioration and maintenance cost in subsystem j. The notation $\|\cdot\|_1$ represents the 1-norm. The parameter $c_{q,j}^{\text{Maint}}$ is the cost of the q-th maintenance intervention in subsystem j. The chance-constrained optimization problem for subsystem j at time step k can then be formulated as:

$$\min_{\tilde{u}_{j,k}} \mathbb{E}_{\tilde{\theta}_{j,k}}[J_j(\tilde{x}_{j,k}, \tilde{u}_{j,k})] \tag{4.5}$$

$$\text{subject to: } \mathbb{P}_{\tilde{\theta}_{j,k}}\left[\max_{l=1,\dots,N_\text{P}} \hat{x}_{j,k+l|k}^{\text{con}}(\tilde{u}_{j,k}, \tilde{\theta}_{j,k}; x_{j,k}) \le x_{\max}^{\text{con}}\right] \ge 1-\epsilon_j \tag{4.6}$$

$$\tilde{x}_{j,k} = \tilde{f}_j(\tilde{u}_{j,k}, \tilde{\theta}_{j,k}; x_{j,k}), \tag{4.7}$$

where ϵ_j is the violation level of the chance constraint of subsystem j, and the function $\tilde{f}_j$ can be obtained by successive substitution of (2.1). The chance constraint (4.6) states that the probability that the worst estimated condition within the planning horizon does not exceed the maintenance threshold $x_{\max}^{\text{con}}$ is at least $1-\epsilon_j$.
We approximate the local chance-constrained problem (4.5)-(4.6) with a confidence level β_j using the two-phase scenario-based robust approach of [100] (see Section 2.2.4 for more details). Let $\mathscr{B}_j^*$ denote the hyperbox obtained by solving the scenario-based problem (2.8)-(2.9) for each dimension of $\tilde{\theta}_{j,k}$. Let $\mathscr{H}_j$ denote the set of random scenarios of subsystem j, and define:

$$\tilde{x}_{j,k}^{(h)} = \tilde{f}_j(\tilde{u}_{j,k}, \tilde{\theta}_{j,k}^{(h)}; x_{j,k}) \tag{4.8}$$

for any $h \in \mathscr{H}_j$. The resulting robust optimization problem can then be written as:

$$\min_{\tilde{u}_{j,k}, \tilde{x}_{j,k}^{(h)}} \frac{1}{|\mathscr{H}_j|} \sum_{h\in\mathscr{H}_j} J_j(\tilde{x}_{j,k}^{(h)}, \tilde{u}_{j,k}) \tag{4.9}$$

$$\text{subject to: } \max_{\tilde{\theta}_{j,k}\in\mathscr{B}_j^*\cap\tilde{\Theta}_j} \max_{l=1,\dots,N_\text{P}} \hat{x}_{j,k+l|k}^{\text{con}}(\tilde{u}_{j,k}, \tilde{\theta}_{j,k}; x_{j,k}) \le x_{\max}^{\text{con}} \tag{4.10}$$

$$\tilde{x}_{j,k}^{(h)} = \tilde{f}_j(\tilde{u}_{j,k}, \tilde{\theta}_{j,k}^{(h)}; x_{j,k}) \quad \forall h \in \mathscr{H}_j, \tag{4.11}$$

where (4.9) approximates the expectation of J_j. As proved by [100], any feasible solution of the robust optimization problem (4.9)-(4.10) is also an ϵ_j-solution of the chance-constrained MPC problem (4.5)-(4.7) with a probability of at least β_j.
We define the following worst-case scenario:

$$\tilde{\theta}_{j,k}^{(w)} \in \arg\max_{\tilde{\theta}_{j,k}\in\mathscr{B}_j^*\cap\tilde{\Theta}_j} \max_{l=1,\dots,N_\text{P}} \hat{x}_{j,k+l|k}^{\text{con}}(\tilde{u}_{j,k}, \tilde{\theta}_{j,k}; x_{j,k}). \tag{4.12}$$

The robust constraint (4.10) can then be replaced by:

$$P_j \tilde{x}_{j,k}^{(w)}(\tilde{u}_{j,k}, \tilde{\theta}_{j,k}^{(w)}; x_{j,k}) \le x_{\max}^{\text{con}}, \tag{4.13}$$

where P_j is a selection matrix satisfying $P_j \tilde{x}_{j,k} = \tilde{x}^{\text{con}}_{j,k}$. We then define $\mathscr{S}_j = \mathscr{H}_j \cup \{w\}$ as the set containing all scenarios that need to be considered to approximate the chance-constrained MPC optimization problem (4.5)-(4.7) by the deterministic optimization problem (4.9),(4.11), (4.13).
As the convexity of the estimated condition $\hat{x}^{\text{con}}_{j,k+l|k}$ is crucial in computing the worst-case scenario $\tilde{\theta}^{(w)}_{j,k}$, and $\hat{x}^{\text{con}}_{j,k+l|k}$ is obtained recursively using the system dynamics (2.1), we now provide a theorem to check the convexity of each $\hat{x}^{\text{con}}_{j,k+l|k}$ for a given deterioration model. For convenience, we rewrite the vector-valued multi-variable function f_j in the following form:

$$f_j(x_{j,k}, u_{j,k}, \theta_{j,k}) = \begin{bmatrix} f^{\text{con}}_j(x^{\text{con}}_{j,k}, x^{\text{aux}}_{j,k}, u_{j,k}, \theta_{j,k}) \\ f^{\text{aux}}_j(x^{\text{con}}_{j,k}, x^{\text{aux}}_{j,k}, u_{j,k}, \theta_{j,k}) \end{bmatrix}. \tag{4.14}$$

We have the following theorem on the convexity of $\hat{x}^{\text{con}}_{j,k+l|k}$ and $\hat{x}^{\text{aux}}_{j,k+l|k}$:

Theorem 4.1 *If f^{con}_j and f^{aux}_j are convex in $\theta_{j,k}$ and convex and non-decreasing in $x^{\text{con}}_{j,k}$ and $x^{\text{aux}}_{j,k}$, then the functions $\hat{x}^{\text{con}}_{j,k+l|k}$ and $\hat{x}^{\text{aux}}_{j,k+l|k}$ are both convex in $\tilde{\theta}_{j,k}$, for any $l \in \{2, \ldots, N_p\}$.*

Proof: We prove Theorem 1 by induction. First we prove that $x^{\text{con}}_{j,k+1|k}$ and $\hat{x}^{\text{aux}}_{j,k+1|k}$ are convex in $\tilde{\theta}_{j,k}$. By definition we have

$$\begin{aligned} x^{\text{con}}_{j,k+1|k}(\tilde{u}_{j,k}, \tilde{\theta}_{j,k}) &= f^{\text{con}}_j(x^{\text{con}}_{j,k}, x^{\text{aux}}_{j,k}, u_{j,k}, \theta_{j,k}) \\ \hat{x}^{\text{aux}}_{j,k+1|k}(\tilde{u}_{j,k}, \tilde{\theta}_{j,k}) &= f^{\text{aux}}_j(x^{\text{con}}_{j,k}, x^{\text{aux}}_{j,k}, u_{j,k}, \theta_{j,k}). \end{aligned}$$

Because f^{con}_j and f^{aux}_j are both convex in $\theta_{j,k}$, $\hat{x}^{\text{con}}_{j,k+1|k}$ and $\hat{x}^{\text{aux}}_{j,k+1|k}$ are also convex in $\theta_{j,k}$. Moreover, as $\hat{x}^{\text{con}}_{j,k+1|k}$ and $\hat{x}^{\text{aux}}_{j,k+1|k}$ have no dependence on $\theta_{j,k+2}, \ldots, \theta_{j,k+N_\text{P}}$, both $\hat{x}^{\text{con}}_{j,k+1|k}$ and $\hat{x}^{\text{aux}}_{j,k+1|k}$ are convex in $\tilde{\theta}_{j,k}$.
Now we prove that for any $p \leq l-1$, if $\hat{x}^{\text{con}}_{j,k+p|k}$ and $\hat{x}^{\text{aux}}_{j,k+p|k}$ are both convex on $\tilde{\theta}_{j,k}$, then $\hat{x}^{\text{con}}_{j,k+p+1|k}$ and $\hat{x}^{\text{aux}}_{j,k+p+1|k}$ are also convex on $\tilde{\theta}_{j,k}$.
Since $\hat{x}^{\text{con}}_{j,k+p|k}$ and $\hat{x}^{\text{aux}}_{j,k+p|k}$ are both convex on $\tilde{\theta}_{j,k}$, for any $w_1, w_2 \in \Theta^{N_\text{P}}$ and $\alpha \in [0,1]$, we have:

$$\begin{aligned} \hat{x}^{\text{con}}_{j,k+p|k}(\tilde{u}_{j,k}, \alpha w_1 + (1-\alpha) w_2) &\leq \alpha \hat{x}^{\text{con}}_{j,k+p|k}(\tilde{u}_{j,k}, w_1) + (1-\alpha)\hat{x}^{\text{con}}_{j,k+p|k}(\tilde{u}_{j,k}, w_2) \\ \hat{x}^{\text{aux}}_{j,k+p|k}(\tilde{u}_{j,k}, \alpha w_1 + (1-\alpha) w_2) &\leq \alpha \hat{x}^{\text{aux}}_{j,k+p|k}(\tilde{u}_{j,k}, w_1) + (1-\alpha)\hat{x}^{\text{con}}_{j,k+p|k}(\tilde{u}_{j,k}, w_2). \end{aligned}$$

Let $(v)_i$ denote the i-th entry of vector v. Because f^{con}_j is non-decreasing in $x^{\text{con}}_{j,k}$ and $x^{\text{aux}}_{j,k}$, we have:

$$\begin{aligned} &\hat{x}^{\text{con}}_{j,k+p+1|k}(\tilde{u}_{j,k}, \alpha w_1 + (1-\alpha) w_2) \\ &= f^{\text{con}}_j(\hat{x}^{\text{con}}_{j,k+p|k}(\tilde{u}_{j,k}, \alpha w_1 + (1-\alpha) w_2), \hat{x}^{\text{aux}}_{j,k+p|k}(\tilde{u}_{j,k}, \alpha w_1 + (1-\alpha) w_2), u_{j,k+p+1}, (\alpha w_1 + (1-\alpha) w_2)_{p+1}) \\ &\leq f^{\text{con}}_j\Big(\alpha \hat{x}^{\text{con}}_{j,k+p|k}(\tilde{u}_{j,k}, w_1) + (1-\alpha)\hat{x}^{\text{con}}_{j,k+p|k}(\tilde{u}_{j,k}, w_2), \alpha \hat{x}^{\text{aux}}_{j,k+p|k}(\tilde{u}_{j,k}, w_1) + (1-\alpha)\hat{x}^{\text{con}}_{j,k+p|k}(\tilde{u}_{j,k}, w_2), \\ &\qquad \tilde{u}_{j,k}, (\alpha w_1 + (1-\alpha) w_2)_{p+1}\Big) \end{aligned}$$

Moreover, because $f^{\text{con}}_j(\cdot, \cdot, \tilde{u}_{j,k}, \cdot)$ is convex for any $\tilde{u}_{j,k}$, and

$$(\alpha w_1 + (1-\alpha) w_2)_{p+1} = \alpha (w_1)_{p+1} + (1-\alpha)(w_2)_{p+1},$$

we have

$$
\begin{aligned}
f_j^{\rm con}\Big(&\alpha\hat{x}^{\rm con}_{j,k+p|k}(\tilde{u}_{j,k},w_1)+(1-\alpha)\hat{x}^{\rm con}_{j,k+p|k}(\tilde{u}_{j,k},w_2),\alpha\hat{x}^{\rm aux}_{j,k+p|k}(\tilde{u}_{j,k},w_1)+(1-\alpha)\hat{x}^{\rm con}_{j,k+p|k}(\tilde{u}_{j,k},w_2),\\
&\tilde{u}_{j,k},(\alpha w_1+(1-\alpha)w_2)_{p+1}\Big)\\
&\leq \alpha f_j^{\rm con}(\hat{x}^{\rm con}_{j,k+p|k}(\tilde{u}_{j,k},w_1),\hat{x}^{\rm aux}_{j,k+p|k}(\tilde{u}_{j,k},w_1),\tilde{u}_{j,k},(w_1)_{p+1})\\
&(1-\alpha)f_j^{\rm con}(\hat{x}^{\rm con}_{j,k+p|k}(\tilde{u}_{j,k},w_2),\hat{x}^{\rm aux}_{j,k+p|k}(\tilde{u}_{j,k},w_2),\tilde{u}_{j,k},(w_2)_{p+1})\\
&=\alpha\hat{x}^{\rm con}_{j,k+p+1|k}(\tilde{u}_{j,k},w_1)+(1-\alpha)\hat{x}^{\rm con}_{j,k+p+1|k}(\tilde{u}_{j,k},w_2) \qquad (4.15)
\end{aligned}
$$

Thus, $\hat{x}^{\rm con}_{j,k+p+1|k}$ is convex in $\tilde{\theta}_{j,k}$. Similarly, we can prove that $\hat{x}^{\rm aux}_{j,k+p+1|k}$ is also convex in $\tilde{\theta}_{j,k}$. □

4.2.2 Mixed Logical Dynamical Systems

We consider the following state dynamics for each scenario $s\in\mathscr{S}_j=\mathscr{H}_j\cup\{w\}$:

$$x^{(s)}_{j,k+1}=f_j(x^{(s)}_{j,k},u_{j,k},\theta^{(s)}_{j,k}). \qquad (4.16)$$

As stated in Section 2.2.1, for each maintenance option $q\in\{1,\dots,N\}$, the function $f_j^q(\cdot,\theta^{(s)}_{j,k})$ is in general nonlinear with respect to $x^{(s)}_{j,k}$. This nonlinear function can be approximated by a function $f_j^{q,\rm PWA}$ that is piecewise-affine with respect to $x^s_{j,k}$. In this way a piecewise-affine model $f_j^{\rm PWA}$ can be obtained to approximate (4.16). This approximate model can then be transformed into the following standard Mixed Logical Dynamical (MLD) model [12]:

$$
\begin{aligned}
x^{(s)}_{j,k+1}&=A^{(s)}_j x^{(s)}_{j,k}+B^{(s)}_{j,2}\delta^{(s)}_{j,k}+B^{(s)}_{j,3}z^{(s)}_{j,k} &(4.17)\\
E^{(s)}_{j,2}\delta^{(s)}_{j,k}+E^{(s)}_{j,3}z^{(s)}_{j,k}&\leq E^{(s)}_{j,4}x^{(s)}_{j,k}+E^{(s)}_{j,5}, &(4.18)
\end{aligned}
$$

where the vector $\delta^{(s)}_{j,k}$ contains all binary variables, and the vector $z^{(s)}_{j,k}$ contains all continuous auxiliary variables.

4.2.3 Centralized MLD-MPC Problem

Define $\tilde{\delta}^{(s)}_{j,k}=[(\delta^{(s)}_{j,k})^{\rm T}\dots(\delta^{(s)}_{j,k+N_{\rm P}-1})^{\rm T}]^{\rm T}$ and $\tilde{\delta}_{j,k}=[(\tilde{\delta}^{(1)}_{j,k})^{\rm T}\dots(\tilde{\delta}^{(|\mathscr{S}_j|)}_{j,k})^{\rm T}]^{\rm T}$. The vectors $\tilde{z}^{(s)}_{j,k}$ and $\tilde{z}_{j,k}$ are defined in the same way. The local robust scenario-based MPC optimization problem (4.9),(4.13),(4.11) for subsystem j can then be formulated as an MILP problem:

$$
\begin{aligned}
\min_{\tilde{\delta}_{j,k},\tilde{z}_{j,k}}&\left(\frac{1}{|\mathscr{H}_j|}\sum_{h\in\mathscr{H}_j}\|\tilde{x}^{(h)}_{j,k}\|_1\right)+w_j\|Q_j\tilde{\delta}_{j,k}\|_1 &(4.19)\\
\text{subject to: }\tilde{x}^{(s)}_{j,k}&=\tilde{A}^{(s)}_j x_{j,k}+\tilde{B}^{(s)}_{j,2}\tilde{\delta}_{j,k}+\tilde{B}^{(s)}_{j,3}z_{j,k}\quad\forall s\in\mathscr{S}_j &(4.20)\\
\tilde{E}^{(s)}_{j,2}\tilde{\delta}_{j,k}+\tilde{E}^{(s)}_{j,3}\tilde{z}_{j,k}&\leq\tilde{E}^{(s)}_{j,4}x_{j,k}+\tilde{E}^{(s)}_{j,5}\quad\forall s\in\mathscr{S}_j &(4.21)\\
\tilde{\delta}_{j,k}&\in\{0,1\}^{N_{\rm P}\cdot n_{\delta_j}} &(4.22)\\
\tilde{z}_{j,k}&\in\mathbb{R}^{N_{\rm P}\cdot n_{z_j}}, &(4.23)
\end{aligned}
$$

and constraint (4.13),

where Q_j is a matrix with nonnegative entries. The first term in the objective function (4.19) corresponds to the mean of the accumulated condition deterioration within the prediction window, while the second term corresponds to the total maintenance cost. Constraints (4.20) and (4.21) are the N_{P}-step-ahead prediction model derived from the MLD dynamics (4.17)-(4.18).
If we define $\tilde{\delta}_k = [\tilde{\delta}_{1,k} \dots \tilde{\delta}_{n,k}]^{\mathrm{T}}$ and $\tilde{z}_k = [\tilde{z}_{1,k} \dots \tilde{z}_{n,k}]^{\mathrm{T}}$, then the centralized MPC optimization problem can be formulated as:

$$\min_{\tilde{\delta}_k, \tilde{z}_k} \sum_{j=1}^{n} c_{j,1}\tilde{\delta}_{j,k} + c_{j,2}\tilde{z}_{j,k} \tag{4.24}$$

$$\text{subject to: } \sum_{j=1}^{n} R_j \tilde{\delta}_{j,k} \le r \tag{4.25}$$

$$F_{j,1}\tilde{\delta}_{j,k} + F_{j,2}\tilde{z}_{j,k} \le l_j \quad \forall j \in \{1, \dots, n\} \tag{4.26}$$

$$\tilde{\delta}_k \in \bigtimes_{j=1}^{n} \{0, 1\}^{N_{\mathrm{P}} n_{\delta_j}} \tag{4.27}$$

$$\tilde{z}_k \in \bigtimes_{j=1}^{n} \mathbb{R}^{N_{\mathrm{P}} \cdot n_{z_j}}. \tag{4.28}$$

Each cost vector in the objective function (4.24) can be obtained by substituting (4.20) into (4.19). Constraint (4.25) is the global constraint on the available resources, e.g. limited track possession time for maintenance, and constraints (4.26) summarize the local constraints (4.13),(4.20)-(4.21) for each subsystem.

4.2.4 Dantzig-Wolfe Decomposition

The centralized MPC problem (4.24)-(4.28) is intractable for large-scale railway network with many sections. Notice that without the global constraint (4.25), the centralized MPC problem can be solved by solving the n independent problems (4.19)-(4.23) for each subsystem $j \in \{1, \dots, n\}$. We apply Dantzig-Wolfe decomposition to improve the scalability of the proposed MPC approach. First we define

$$\mathscr{P}_{j,k} = \left\{ (\tilde{\delta}_{j,k}, \tilde{z}_{j,k}) \in \{0, 1\}^{N_{\mathrm{P}} n_{\delta_j}} \times \mathbb{R}^{N_{\mathrm{P}} n_{z_j}} : F_{j,1}\tilde{\delta}_{j,k} + F_{j,2}\tilde{z}_{j,k} \le l_j \right\}, \tag{4.29}$$

which is the feasible region of the local MPC optimization problem for subsystem j. Let $\mathscr{G}_{j,k}$ denote the extreme points of the convex hull of $\mathscr{P}_{j,k}$. We call $\mathscr{G}_{j,k}$ the *generating set* of subsystem j at time step k. Let $\tilde{\delta}_{j,k}^{[g]}$ and $\tilde{z}_{j,k}^{[g]}$ denote the values of $\tilde{\delta}_{j,k}$ and $\tilde{z}_{j,k}$ of the extreme point $g \in \mathscr{G}_{j,k}$, respectively. According to Minkowski's theorem, each point in a compact polyhedron can be represented by a convex combination of its extreme points, which are called columns. Let $\mu_{j,g}$ denote the weight of the column $g \in \mathscr{G}_{j,k}$, and let μ_j denote the vector containing all the weights for columns in the generating set $\mathscr{G}_{j,k}$. Furthermore, define $\mu = [\mu_1^{\mathrm{T}} \dots \mu_n^{\mathrm{T}}]^{\mathrm{T}}$. The Dantzig-Wolfe reformulation of the centralized MPC problem (4.24)-(4.28) can then be written as:

$$\min_{\mu} \sum_{j=1}^{n} \sum_{g \in \mathscr{G}_{j,k}} (c_{j,1}\tilde{\delta}_{j,k}^{[g]} + c_{j,2}\tilde{z}_{j,k}^{[g]})\mu_{j,g} \tag{4.30}$$

$$\text{subject to: } \sum_{j=1}^{n} \sum_{g \in \mathscr{G}_j} (R_j \tilde{\delta}_{j,k}^{[g]}) \mu_{j,g} \leq r \tag{4.31}$$

$$\sum_{g \in \mathscr{G}_j} \mu_{j,g} = 1 \quad \forall j \in \{1, \ldots, n\} \tag{4.32}$$

$$\mu_{j,g} \geq 0 \quad \forall g \in \mathscr{G}_{j,k}, \forall j \in \{1, \ldots, n\} \tag{4.33}$$

$$\tilde{\delta}_{j,k} = \sum_{g \in \mathscr{G}_{j,k}} \tilde{\delta}_{j,k}^{[g]} \mu_{j,g} \in \bigtimes_{j=1}^{n} \{0, 1\}^{N_{\mathrm{p}} n_{\delta_j}}, \forall j \in \{1, \ldots, n\}. \tag{4.34}$$

Problem (4.30)-(4.34) is called the Dantzig-Wolfe reformulation by convexification. The disadvantage of this formulation is that the binary condition is imposed on the old decision variable $\tilde{\delta}_{j,k}$, as shown in (4.34). However, as proved in [78], if the global constraint in the original binary MILP problem involves only binary variables, then the binary condition on the original variables can be directly transferred to the new variable μ. As the global constraint (4.25) contains only binary decision variables, we can then replace constraints (4.33)-(4.34) by

$$\mu_{j,g} \in \{0, 1\} \quad \forall g \in \mathscr{G}_{j,k}, \forall j \in \{1, \ldots, n\}. \tag{4.35}$$

The reformulated problem (4.30)-(4.32),(4.35) is still intractable, as the dimension of each generating set $\mathscr{G}_{j,k}$ grows exponentially with the dimension of the old variable $\tilde{\delta}_{j,k}$. We use column generation to tackle this difficulty.

Column generation

Column generation [162] solves the linear relaxation of the Dantzig-Wolfe reformulation (4.30)-(4.32), (4.35). We call the relaxed problem (4.30)-(4.33) the *master problem*. First we start with an initial partial generating set $\mathscr{G}_{j,k}^{\mathrm{s}} \subset \mathscr{G}_{j,k}$ for each subsystem j and solve the following *restricted master problem*:

$$\min_{\mu} \sum_{j=1}^{n} \sum_{g \in \mathscr{G}_{j,k}^{\mathrm{s}}} (c_{j,1} \tilde{\delta}_{j,k}^{[g]} + c_{j,2} \tilde{z}_{j,k}^{[g]}) \mu_{j,g} \tag{4.36}$$

$$\text{subject to: } \sum_{j=1}^{n} \sum_{g \in \mathscr{G}_{j,k}^{\mathrm{s}}} (R_j \tilde{\delta}_{j,k}^{[g]}) \mu_{j,g} \leq r \tag{4.37}$$

$$\sum_{g \in \mathscr{G}_{j,k}^{\mathrm{s}}} \mu_{j,q} = 1 \quad \forall j \in \{1, \ldots, n\} \tag{4.38}$$

$$\mu_{j,g} \geq 0 \quad \forall g \in \mathscr{G}_{j,k}^{\mathrm{s}}, \quad \forall j \in \{1, \ldots, n\}. \tag{4.39}$$

Each initial generating set $\mathscr{G}_{j,k}^{\mathrm{s}}$ should be chosen to ensure the feasibility of the restricted master problem. Let μ^* denote the optimal solution of this restricted master problem. The dual of problem (4.36)-(4.39) can be written as:

$$\max_{\lambda,\pi} -r\lambda + \sum_{j=1}^{n} \pi_j \tag{4.40}$$

$$\text{subject to: } \lambda(-R_j \tilde{\delta}_{j,k}^{[g]}) + \pi_j \leq c_{j,1} \tilde{\delta}_{j,k}^{[g]} + c_{j,2} \tilde{z}_{j,k}^{[g]} \tag{4.41}$$

$$\forall g \in \mathscr{G}_{j,k}^{\mathrm{s}}, \forall j \in \{1, \ldots, n\}$$

$$\lambda \geq 0 \tag{4.42}$$

$$\pi \in \mathbb{R}^n. \tag{4.43}$$

Let (λ^*, π^*) denote the optimal solution of the dual problem (4.40)-(4.43). We then define the following *pricing subproblem* for each subsystem j:

$$\begin{aligned} \rho_j &= \min_{g \in \mathcal{G}_{j,k}} c_{j,1}\tilde{\delta}^{[g]}_{j,k} + c_{j,2}\tilde{z}^{[g]}_{j,k} + \lambda^* (R_j \tilde{\delta}^{[g]}_{j,k}) - \pi^*_j \\ &= \min_{(\tilde{\delta}_{j,k}, \tilde{z}_{j,k}) \in \mathcal{P}_{j,k}} c_{j,1}\tilde{\delta}_{j,k} + c_{j,2}\tilde{z}_{j,k} + \lambda^* (R_j \tilde{\delta}_{j,k}) - \pi^*_j, \end{aligned} \tag{4.44}$$

where ρ_j is called the *reduced cost* of subproblem j. We add the new column, which is the optimal solution of (4.44), to the partial generating set $\mathcal{G}^s_{j,k}$ only if the corresponding reduced cost is negative. The restricted master problem (4.36)-(4.39) and its dual (4.40)-(4.43) are solved again including the new columns, and the new optimal dual solution (λ^*, π^*) is sent to each pricing subproblem. The iteration terminates when all reduced costs are 0, and the optimal solution of the restricted master problem corresponds to the optimal solution of the master problem.

Upper and lower bounds

Upper and lower bounds of the objective function value of the Dantzig-Wolfe reformulation can be implemented to the basic column generation algorithm to achieve faster convergence. Any binary solution of the restricted master problem encountered in the column generation procedure provides an upper bound of the objective function value of the Dantzig-Wolfe reformulation and the centralized MPC optimization problem. A lower bound can be obtained by the Lagrangian dual function of the centralized MPC problem [162]:

$$\begin{aligned} q(\lambda^*) &= \inf_{(\tilde{\delta}_k, \tilde{z}_k) \in \times_{j=1}^n \mathcal{P}_{j,k}} \sum_{j=1}^n (c_{j,1}\tilde{\delta}_{j,k} + c_{j,2}\tilde{z}_{j,k}) + \lambda^* \left(\sum_{j=1}^n R_j \tilde{\delta}_{j,k} - r \right) \\ &= -\lambda^* r + \sum_{j=1}^n (\rho_j + \pi^*_j). \end{aligned} \tag{4.45}$$

In addition to checking whether all reduced costs are 0, the upper and lower bounds provides another convergence criterion, i.e. whether the two bounds meet. The primal upper bounds are in general very weak, especially in the beginning of the column generation procedure, when the sets of columns are small. The dual bounds might oscillate, as the optimal solution of the dual of the restricted master problem might change drastically when a new column is added. Typical remedies for the erratic behaviour of the dual bounds include warm start e.g. [152], which provides a good dual bound at the beginning of the iteration, and stabilization techniques [64, 138], which add penalizing terms to (4.45) to avoid drastic change in the Lagrangian dual bounds. Another improvement of the standard column generation algorithm is the primal-dual column generation technique developed in [57], which uses suboptimal primal and dual solutions of the restricted master problem to improve the stability of the iteration.

Inexact method

If the optimal solution of the master problem (4.30)-(4.33) obtained by column generation is also binary, then we have found the optimal solution of the Dantzig-Wolfe reformulation and the original MILP problem. However, the solution obtained through column generation is in general fractional. As stated in [65], a feasible[1] suboptimal solution of the Dantzig-Wolfe reformulation can be obtained by solving the restricted master problem (4.36)-(4.39) as a binary MILP problem, using the sets of columns obtained at the end of column generation. Furthermore, a lower bound, and possibly an upper bound (depending on whether a binary solution is encountered during the iteration) are also provided by the column generation procedure. Exact solutions to the Dantzig-Wolfe formulation can be found by combining branch-and-bound with column generation, known as the branch-and-price [7] algorithm.

4.3 Low-Level Maintenance Crew Scheduling

The low-level problem is triggered whenever a maintenance intervention is suggested for at least one section of the whole network by the high-level MPC controller. Each type of maintenance intervention, e.g. grinding, is associated with a distinct low-level problem. Its goal is to find the optimal schedule to perform the planned maintenance activities, and the optimal route for the maintenance crew, minimizing the total setup costs of maintenance operations, the travelling costs of the maintenance crew, and the penalty cost on extra maintenance time (if any).

4.3.1 Arc Routing Problem

First we define the Capacitated Arc Routing Problem with Flexible Capacity (CARPFC), which is composed of:

- a connected undirected graph $G = (\mathcal{V}, \mathcal{E})$;
- a depot node $0 \in \mathcal{V}$;
- a cost matrix C defining the travel cost associated with each edge;
- a set of required edges $\mathcal{R} \subseteq \mathcal{E}$ that must be serviced by a vehicle;
- a demand q_{ij} for each required edge $\{i, j\} \in \mathcal{R}$;
- a set $\mathcal{T}$ (fleet) containing all available vehicles;
- a fixed setup cost c_{Setup} associated with each vehicle;
- a flexible capacity $Q_t \in [\underline{Q}, \overline{Q}]$ associated with each vehicle $t \in \mathcal{T}$;
- a capacity-related cost $Q_{\text{Extra},t} = \nu(Q_t - \underline{Q})$ for any vehicle $t \in \mathcal{T}$, where ν is a positive parameter.

[1]Feasibility can be guaranteed, as long as the restricted master problem is binary feasible with initial sets of columns, or a binary solution is encountered during the iteration.

The CARPFC can then be defined as finding an optimal set of routes of the fleet starting and ending at the same depot, and the optimal capacities of the vehicles, that minimizes the total setup costs and travel costs of vehicles, and the costs related to the extra capacity, while ensuring that each required edge is serviced exactly once by a vehicle, and the edge demand is satisfied without exceeding the vehicle capacity.
To recast the low-level maintenance crew scheduling problem into a CARPFC, we map the physical network into a virtual graph $G = (\mathcal{V}, \mathcal{E})$. The stations are mapped into nodes in $\mathcal{V}$, and the lines are mapped into edges in $\mathcal{E}$. In particular, the maintenance base is mapped into the depot node 0. The travel cost of each edge is proportional to the length of the line. Furthermore, the lines in which at least one section is to be maintained before the next time step, are mapped into the required edges in $\mathcal{R}$. The demand of a required edge is interpreted as the estimated time to complete the maintenance activity on the corresponding line. Each time period in the low-level planning horizon corresponds to a vehicle in the CARPF, and the maintenance time budget per time period is translated as the capacity of the vehicle. The maintenance time budget is considered to be flexible within a given range, e.g. typically $\underline{Q} = 6$ hours and $\overline{Q} = 10$ hours per time period for the Dutch railway network.

4.3.2 Node Routing Problem

We transform the arc routing problem described in Section 4.3.1 into an equivalent node routing problem using the approach developed in [6]. The transformed complete undirected graph is denoted by $\hat{G} = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$, with a new cost matrix $\hat{C}$. Each endpoint of a required edge in $\mathcal{R}$ of the original graph becomes a customer node in $\hat{\mathcal{V}}$ of the transformed graph, resulting in a node routing problem instance of $2|\mathcal{R}|$ customer nodes. We refer the readers to [6] for the detailed transformation procedure. Furthermore, we partition $\hat{\mathcal{V}}$ into the set of virtual depots $\mathcal{T}$, and the set of customer nodes $\hat{\mathcal{C}}$. Each virtual depot is a duplicate of the depot in the original graph, and corresponds to a vehicle $t \in \mathcal{T}$. We introduce these virtual depots to ensure that each tour is performed by one vehicle with a specific capacity. The demand of a customer node $i \in \hat{\mathcal{C}}$ in the transformed graph is denoted by $\hat{q}_i$. In this section we provide the MILP formulation of the Capacitated Vehicle Routing Problem with Flexible Capacity (CVRPFC), which is a node routing counterpart of the CARPFC described in Section 4.3.1.
We define the binary decision variables:

$$x_{ij} = \begin{cases} 1 & \text{if node } j \text{ is visited directly after node } i; \\ 0 & \text{otherwise} \end{cases} \tag{4.46}$$

for any $i, j \in \hat{\mathcal{V}}$, and

$$z_{it} = \begin{cases} 1 & \text{if customer } i \text{ is visited by a vehicle from depot } t \\ 0 & \text{otherwise} \end{cases} \tag{4.47}$$

for any node $i \in \hat{\mathcal{C}}$ and $t \in \mathcal{T}$.
We use the Miller-Tucker-Zemlin (MTZ) subtour elimination constraints [104], and define a continuous node potential variable u_i for each customer node $i \in \hat{\mathcal{C}}$. Because of the multiple virtual depots corresponding to heterogeneous vehicles, cycle imposement constraints are needed to ensure that each resulting route starts and ends at the same virtual depot, i.e. each round tour is made by the same vehicle. For this purpose, we choose the node-current based

cycle imposement constraints [23], and define the continuous node current variable k_i for each node $i \in \hat{\mathscr{V}}$.
The CVRPFC can then be expressed as:

$$\min \sum_{\{i,j\}\in\tilde{\mathscr{E}}} \tilde{c}_{ij} x_{ij} + \sum_{t\in\mathscr{T}} \sum_{j\in\tilde{\mathscr{V}}} \left(c_{\text{Setup}} + \nu(Q_t - \underline{Q})\right) x_{tj}, \tag{4.48}$$

where the first term corresponds to the travel costs, and the second term computes the total setup costs of the vehicles, including the costs related to the extra capacity, subject to the following:

- assignment constraints:

$$\sum_{j\in\tilde{\mathscr{V}}} x_{ij} = \sum_{j\in\tilde{\mathscr{V}}} x_{ji} = 1 \quad \forall i \in \hat{\mathscr{C}} \tag{4.49}$$

$$\sum_{i\in\tilde{\mathscr{V}}} x_{it} = \sum_{j\in\tilde{\mathscr{V}}} x_{tj} \le 1 \quad \forall t \in \mathscr{T}; \tag{4.50}$$

- path continuity constraints:

$$z_{it} - z_{jt} \le 1 - x_{ij} - x_{ji} \quad \forall t \in \mathscr{T}, i, j \in \hat{\mathscr{C}}, i \ne j \tag{4.51}$$

$$z_{jt} - z_{it} \le 1 - x_{ij} - x_{ji} \quad \forall t \in \mathscr{T}, i, j \in \hat{\mathscr{C}}, i \ne j \tag{4.52}$$

  that ensure any two consecutive customers on a resulting tour are visited by the same vehicle;

- labeling constraints:

$$x_{tj} + x_{jt} - z_{jt} \le 0 \quad \forall t \in \mathscr{T}, j \in \hat{\mathscr{C}} \tag{4.53}$$

  that ensure the first and last visited customer by a vehicle is associated with the corresponding virtual depot;

- the MTZ subtour elimination constraints:

$$u_i - u_j + \overline{Q} x_{ij} + (\overline{Q} - \hat{q}_i - \hat{q}_j) x_{ji} \le \overline{Q} - \hat{q}_i \quad \forall i, j \in \hat{\mathscr{C}}, i \ne j; \tag{4.54}$$

- the node-current based cycle imposement constraints:

$$k_t = t \quad \forall t \in \mathscr{T} \tag{4.55}$$

$$k_i - k_j \le (|\mathscr{T}| - 1)(1 - x_{ij}) \quad \forall i, j \in \tilde{\mathscr{V}}, i \ne j \tag{4.56}$$

$$k_j - k_i \le (|\mathscr{T}| - 1)(1 - x_{ij}) \quad \forall i, j \in \tilde{\mathscr{V}}, i \ne j; \tag{4.57}$$

- the bounds for the continuous decision variables:

$$\hat{q}_i \le u_i \le \sum_{t\in\mathscr{T}} Q_t z_{it} \quad \forall i \in \hat{\mathscr{C}} \tag{4.58}$$

$$1 \le k_i \le |\mathscr{T}| \quad \forall i \in \tilde{\mathscr{V}}; \tag{4.59}$$

- the integrality constraints on the binary decision variables:

$$x_{ij} \in \{0,1\} \quad \forall i, j \in \hat{\mathcal{V}} \tag{4.60}$$

$$z_{it} \in \{0,1\} \quad \forall i \in \hat{\mathcal{C}}, \forall t \in \mathcal{T}. \tag{4.61}$$

Note that the problem (4.48)-(4.61) is an MINLP, as the flexible capacity Q_t is also a decision variable, leading to the nonlinear terms $Q_t x_{t,j}$ in the objective (4.48) and $Q_t z_{it}$ in constraints (4.58). Following the procedure developed in [12, 167], we introduce the following continuous auxiliary decision variables:

$$y_{it} = z_{it} Q_t, \quad \phi_{tj} = x_{tj} Q_t \tag{4.62}$$

for $t \in \mathcal{T}$, $i \in \hat{\mathcal{C}}$, $j \in \tilde{\mathcal{V}}$, to eliminate the nonlinear terms. This results in the following equivalent linear constraints [12]:

$$y_{it} \le \overline{Q} z_{it}, \quad \phi_{tj} \le \overline{Q} x_{tj} \tag{4.63}$$

$$y_{it} \ge \underline{Q} z_{it}, \quad \phi_{tj} \ge \underline{Q} x_{tj} \tag{4.64}$$

$$y_{it} \le Q_t - \underline{Q}(1 - z_{it}), \quad \phi_{tj} \le Q_t - \underline{Q}(1 - x_{tj}) \tag{4.65}$$

$$y_{it} \ge Q_t - \overline{Q}(1 - z_{it}), \quad \phi_{tj} \ge Q_t - \overline{Q}(1 - x_{tj}) \tag{4.66}$$

that are equivalent to (4.62).
The MILP formulation of the CVRPFC can then be written as:

$$\min \sum_{\{i,j\} \in \tilde{\mathcal{E}}} \tilde{c}_{ij} x_{ij} + \sum_{t \in \mathcal{T}} \sum_{j \in \tilde{\mathcal{V}}} v f_{tj} + (c_{\text{Setup}} - v\underline{Q}) x_{tj} \tag{4.67}$$

$$\text{subject to: } \hat{q}_i \le u_i \le \sum_{t \in \mathcal{T}} y_{it} \quad \forall i \in \hat{\mathcal{C}} \tag{4.68}$$

and constraints (4.49)-(4.57), (4.59)-(4.61), (4.63)-(4.66).

4.4 Case Study

4.4.1 Setup

A numerical case study on the optimal treatment of squats is performed on a part of the Dutch railway network containing Randstad Zuid and the middle-south region. This network contains 10 stations[2] and 13 lines, which are divided into 53 sections of 5 km, as shown in the schematic plot in Figure 4.1. A squat is a type of rail contact fatigue, the evolution of which depends on the dynamic wheel-rail contact [45]. It first appears on the rail surface, and evolves into a network of cracks underneath the rail surface over time. If not treated properly, it can lead to hazards like rail breakage. According to field observation, grinding, which removes the irregularities on the rail surface, is effective for early-stage squats with visual length less than 20 mm, but for late-stage squat with visual length more than 50 mm [77] , replacement is the only option. Effectiveness is also related to the grinding depth to

[2]Intermediate stations are not included this case study.

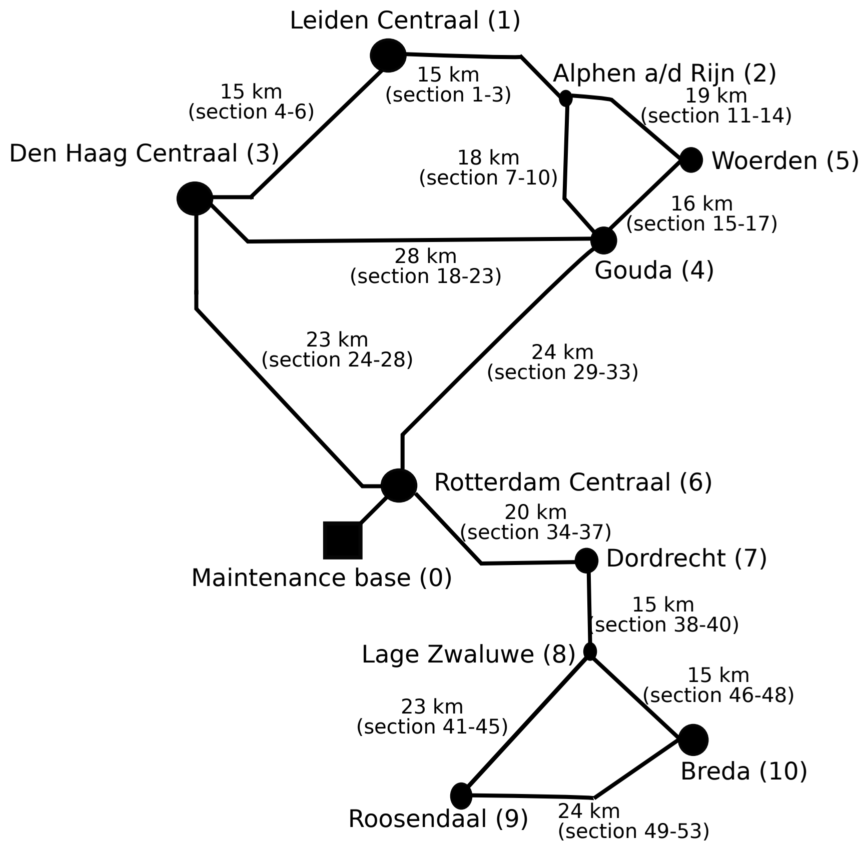


Figure 4.1: The part of the Dutch railway network including Randstad Zuid and the middle-south region considered in the case study. The number next to a station is its index, while the maintenance base indexed as "0". The sections of each track line is indexed starting from the the station with the smaller index.

reduce residual damages.

We adopt the big data analysis approach developed in [76] to calculate the failure probability of each squat. The failure probability, which is initially calculated from the visual length, estimates the probability that a given squat might lead to rail failure within the next million gross tons (MGT) step, which is 3 months in this case study. For this reason, the time step is also 3 months in the high-level MPC controller. The prediction horizon $N_\mathrm{p} = 3$, i.e. 9 months, and the planning horizon is 20, i.e. 5 years.

A *simulation model* is developed to describe the evolution of failure probability of each individual squat. This simulation model is based on the big data analysis approach developed in [76]. The probability that squat i at time step k might lead to rail failure can be calculated by:

$$\xi_{i,k} = (f_\mathrm{Prob} \circ f_\mathrm{Cr} \circ f_\mathrm{M})(L_{i,k-1}, L_{i,k}), \tag{4.69}$$

where $L_{i,k-1}$ and $L_{i,k}$ are two consecutive measurements on the visual length of squat i. The function f_M computes the estimated Mega Gross Tonage (MGT) from two consecutive measurements/simulated data on visual lengths, and the function f_Cr estimates the crack length growth from the estimated MGT. Finally, the function f_Prob calculates the failure probability from the crack growth length. We use the same functions as in the case study of [76].

The risk level, i.e. the condition, of a section of rail can then be calculated from the failure probabilities of all squats within the section[3]. By definition, the condition of each section is within the range [0, 1]. A *prediction model*, which describes the dynamics of the condition of a section, can be obtained by piecewise-affine identification based on the simulated data produced by the simulation model. Let $\mathcal{U} = \{1,2,3\}$ denote the set of all possible maintenance actions that can be applied to a section of rail, with 1, 2, 3 representing “no maintenance”, “grinding”, and “replacing", respectively. Replacing a section is 30 times as expensive as grinding. The parameter λ, which captures the trade-off between condition degradation and maintenance cost, takes a value of 100. Finally, we define the number of grinding operations on section j since the last replacement as the auxiliary variable $x^\mathrm{aux}_{j,k}$ in the prediction model.

The prediction model of section j, in accordance with the generic model (2.1), can then be expressed as:

$$\begin{aligned} x^\mathrm{con}_{j,k+1} &= f^\mathrm{con}_j(x^\mathrm{con}_{j,k}, u_{j,k}, \theta_{j,k}) \\ &= \begin{cases} f^\mathrm{Deg}_j(x^\mathrm{con}_{j,k}, \theta_{j,k}) & \text{if } u_{j,k} = 1 \text{ (no maintenance)} \\ f^\mathrm{Gr}_j(x^\mathrm{con}_{j,k}, \theta_{j,k}) & \text{if } u_{j,k} = 2 \text{ (grinding)} \\ 0 & \text{if } u_{j,k} = 3 \text{ (replacing)} \end{cases} \end{aligned} \tag{4.70}$$

and

$$\begin{aligned} x^\mathrm{aux}_{j,k+1} &= f^\mathrm{aux}_j(x^\mathrm{aux}_{j,k}, u_{j,k}) \\ &= \begin{cases} x^\mathrm{aux}_{j,k} & \text{if } u_{j,k} = 1 \text{ (no maintenance)} \\ x^\mathrm{aux}_{j,k} + 1 & \text{if } u_{j,k} = 2 \text{ (grinding)} \\ 0 & \text{if } u_{j,k} = 3 \text{ (replacing).} \end{cases} \end{aligned} \tag{4.71}$$

[3]We assume the failure of each squat is independent from that of other squats.

The threshold value $x^{\text{con}}_{\max}$ in the chance constraint (4.6) is 0.95. As each grinding operation removes a certain depth of rail (e.g. 2 mm), grinding can only be applied consecutively to the same section for a limited number of times. So we have the following deterministic constraints on the auxiliary variable:

$$x^{\text{aux}}_{j,k+l} \leq x^{\text{aux}}_{\max} \quad \forall j \in \{1,\dots,n\}, \forall l \in \{1,\dots,N_{\text{p}}\}, \tag{4.72}$$

where $x^{\text{aux}}_{\max} = 10$ in this case study.
The following global constraint is imposed on the maximal number of sections that can be ground at one time step:

$$\sum_{j=1}^{n} I_{u_{j,k}=1} \leq n^{\text{Gr}}_{\max} \quad \forall l \in \{1,\dots,N_{\text{p}}\}, \tag{4.73}$$

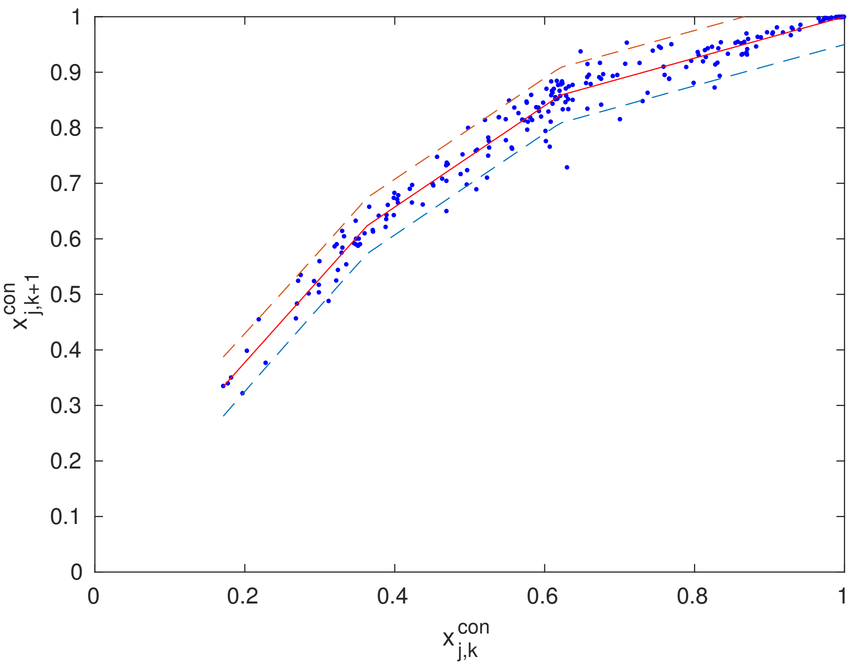
where $n^{\text{Gr}}_{\max} = 20$ in this case study.
The function f^{Deg}_j, which describes the natural degradation of the condition, and the function f^{Gr}_j, which describes the effect of grinding on section j, are both piecewise-affine functions, as shown in the example in Figure 4.2. To determine a piecewise-affine approximation of f^{Deg}_j, we partition the condition space of section j, i.e. $\mathscr{X}^{\text{con}}_j = [0, 1]$, into three intervals $\mathscr{X}^{\text{con}}_{j,1}$, $\mathscr{X}^{\text{con}}_{j,2}$, and $\mathscr{X}^{\text{con}}_{j,3}$. The natural degradation of the condition can then be expressed by the following piecewise-affine function:

$$f^{\text{Deg}}_j(x^{\text{con}}_{j,k}) = \begin{cases} y^{\text{int}}_{j,1} + \dfrac{y^{\text{int}}_{j,2} - y^{\text{int}}_{j,1}}{x^{\text{swi}}_{j,1}} x^{\text{con}}_{j,k} & \text{if } x^{\text{con}}_{j,k} \in \mathscr{X}^{\text{con}}_{j,1} = [0, x^{\text{swi}}_{j,1}) \\ y^{\text{int}}_{j,2} + \dfrac{y^{\text{int}}_{j,3} - y^{\text{int}}_{j,2}}{x^{\text{swi}}_{j,2} - x^{\text{swi}}_{j,1}} \left(x^{\text{con}}_{j,k} - x^{\text{swi}}_{j,1}\right) & \text{if } x^{\text{con}}_{j,k} \in \mathscr{X}^{\text{con}}_{j,2} = [x^{\text{swi}}_{j,1}, x^{\text{swi}}_{j,2}) \\ y^{\text{int}}_{j,3} + \dfrac{y^{\text{int}}_{j,4} - y^{\text{int}}_{j,3}}{1 - x^{\text{swi}}_{j,2}} \left(x^{\text{con}}_{j,k} - x^{\text{swi}}_{j,2}\right) & \text{if } x^{\text{con}}_{j,k} \in \mathscr{X}^{\text{con}}_{j,3} = [x^{\text{swi}}_{j,2}, 1], \end{cases} \tag{4.74}$$

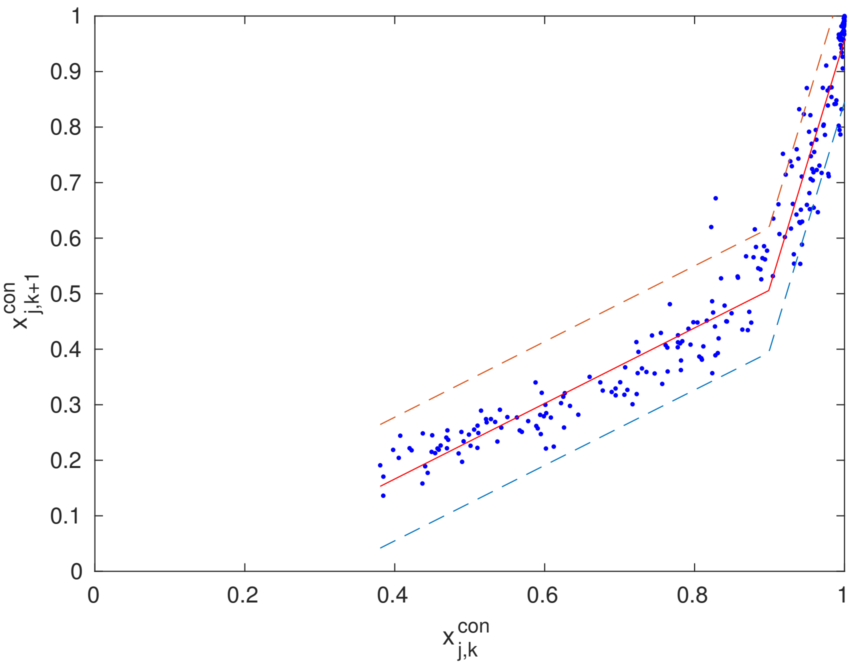
where $x^{\text{swi}}_{j,1}$ and $x^{\text{swi}}_{j,2}$ are the two switching points, and $y^{\text{int}}_{j,1}$, $y^{\text{int}}_{j,2}$, $y^{\text{int}}_{j,3}$, and $y^{\text{int}}_{j,4}$ are the four interpolation points.
The function f^{Gr}_j can also be represented by the following piecewise-affine function with three intervals:

$$f^{\text{Gr}}_j(x^{\text{con}}_{j,k}) = \begin{cases} 0 & \text{if } x^{\text{con}}_{j,k} \leq x^{\text{eff}}_j \\ \dfrac{y^{\text{sev}}_j}{x^{\text{sev}}_j - x^{\text{eff}}_j} \left(x^{\text{con}}_{j,k} - x^{\text{eff}}_j\right) & \text{if } x^{\text{eff}}_j < x^{\text{con}}_{j,k} \leq x^{\text{sev}}_j \\ y^{\text{sev}}_j + \dfrac{y^{\max}_j - y^{\text{sev}}_j}{1 - x^{\text{sev}}_j} \left(x^{\text{con}}_{j,k} - x^{\text{sev}}_j\right) & \text{if } x^{\text{con}}_{j,k} > x^{\text{sev}}_j. \end{cases} \tag{4.75}$$

For identification of the functions f^{Deg}_j and f^{Gr}_j, we create 200 fictional sections, where the number of squats within a section is a random number with a mean value of 10 and standard deviation of 2. Let N_{Sq} denote the number of squats in a section of rail. The failure



(a) Piecewise-affine identification of f_j^{Deg}.



(b) Piecewise-affine identification of f_j^{Gr}.

Figure 4.2: Piecewise-affine identification of one prediction model with 95% nonsimultaneous observation confidence bound (indicated by the blue and red dashed lines). The data points for the identification are generated by aggregating the simulated failure probabilities of individual squats using the simulation model.

Table 4.1: Parameters of the functions f_j^{Deg} and f_j^{Gr} for five different models. Both the nominal values and the 95% nonsimultaneous confidence bounds (given in the squar brackets) are provided for all uncertain parameters.

Parameter	Model				
	1	2	3	4	5
$x_{j,1}^{\text{swi}}$	0.512	0.526	0.543	0.363	0.563
$x_{j,2}^{\text{swi}}$	0.683	0.784	0.781	0.621	0.798
$y_{j,1}^{\text{int}}$	0.107 [0.086, 0.128]	0 [0,0]	0.051 [0.040, 0.063]	0.076 [0.036, 0.115]	0.058 [0.049,0.068]
$y_{j,2}^{\text{int}}$	0.783 [0.776, 0.790]	0.849 [0.845, 0.853]	0.815 [0.809. 0.821]	0.624 [0.615, 0.633]	0.805 [0.900, 0.809]
$y_{j,3}^{\text{int}}$	0.929 [0.924, 0.934]	0.975 [0.967, 0.983]	0.972 [0.966, 0.977]	0.859 [0.853, 0.865]	0.963 [0.958, 0.968]
$y_{j,4}^{\text{int}}$	1 [0.997, 1.003]	1 [0.997, 1.004]	1 [0.998, 1.002]	1 [0.994, 1.006]	1 [0.998, 1.002]
x_j^{eff}	0.156	0.177	0.172	0.141	0.106
x_j^{sev}	0.899	0.810	0.880	0.938	0.882
y_j^{sev}	0.506 [0.494, 0.518]	0.516 [0.505, 0.527]	0.502 [0.490, 0.514]	0.506 [0.490, 0.521]	0.443 [0.432, 0.455]
y_j^{max}	0.957 [0.944, 0.970]	0.991 [0.981, 1]	0.977 [0.965, 0.990]	0.922 [0.905, 0.939]	0.944 [0.931, 0.956]

probability of one section of rail can then be calculated by:

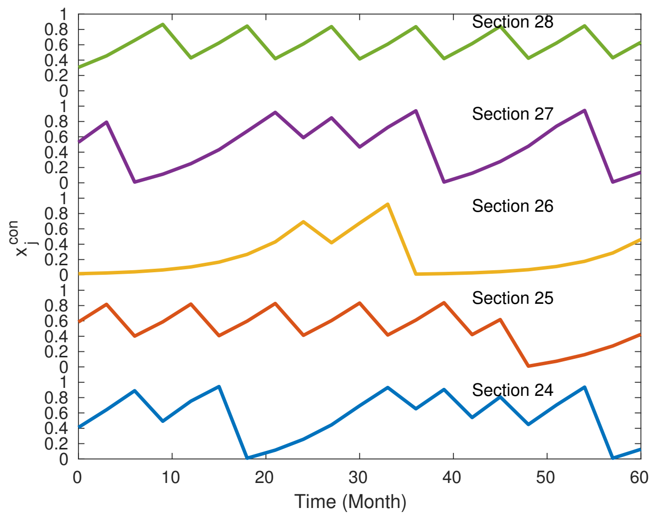
$$x_k^{\text{con}} = 1 - \prod_{i=1}^{N_{\text{Sq}}} (1 - \xi_{i,k}). \tag{4.76}$$

The following squat evolution model is used to simulate the dynamics of the visual length of an individual squat i:

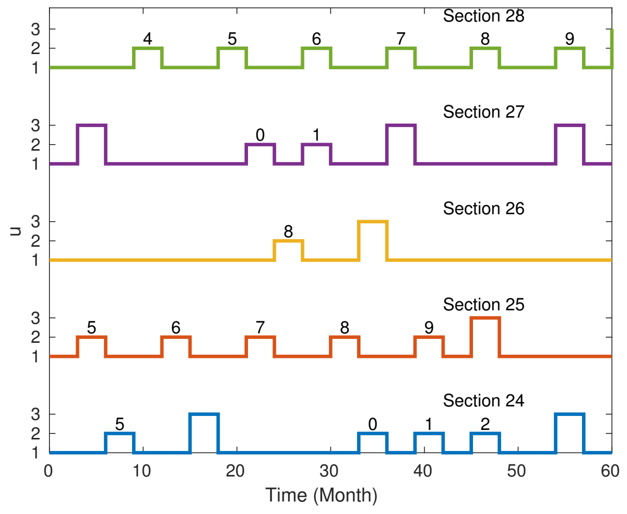
$$L_{i,k+1} = \begin{cases} aL_{i,k} + b & \text{if not treated} \\ \max(\phi(L_{i,k} - L_{\text{eff}}), 0) & \text{if ground} \end{cases}, \tag{4.77}$$

where a, b, and ϕ are all parameters generated from a normal distribution. For each squat i within a section, we can simulate three consecutive measurements of the visual length, i.e. $L_{i,k-1}$, $L_{i,k}$, and $L_{i,k+1}$, and calculate the squat's failure probability at time step k and $k+1$, namely, $\xi_{i,k}$ and $\xi_{i,k+1}$, respectively. The condition of the section at time step k and $k+1$ can then be calculated using (4.76). Five sets of squat evolution models are used, resulting in five different condition deterioration models, the parameters of which are presented in Table 4.1. These five prediction models are then randomly assigned to the 53 sections following a uniform discrete distribution in the case study. The uncertain parameters of the deterioration model (4.70)-(4.71) of section j are collected in the vector $\theta_{j,k} = [y_{j,1}^{\text{int}} \cdots y_{j,4}^{\text{int}}\ y_j^{\text{sev}}\ y_j^{\text{max}}]^{\text{T}}$. The confidence level β_j and violation level ϵ_j are both 0.1 for any section $j = 1, \ldots, n$, resulting in 591 random scenarios per section.

For illustration purpose we only trigger the low-level maintenance crew scheduling problem for grinding. The travel cost between any two station is 100€ per kilometer. Two 6-hour maintenance time slots are available for grinding within one time step (3 months). A fixed setup cost of 100k€ is associated with each maintenance time slot. Furthermore, extra time for grinding in addition to the given 6-hour time slot can be requested with an hourly cost of 10k€. However, a maximum length of 10 hours is imposed on each maintenance time slot. The minimum amount of maintenance time spent on a railway line corresponds to the



(a) Risk level per section calculated from the individual squat dynamics.



(b) Interventions suggested by the high-level MPC controller. The number above each grinding intervention indicates the number of previous grinding operations applied to the section since the last replacement.

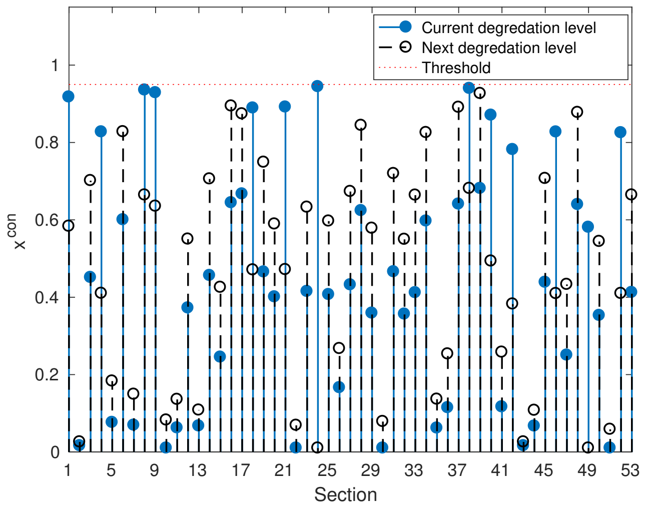
Figure 4.3: High-level simulation results for the line between Den Haag and Rotterdam.

number of sections to be ground before the next time step.

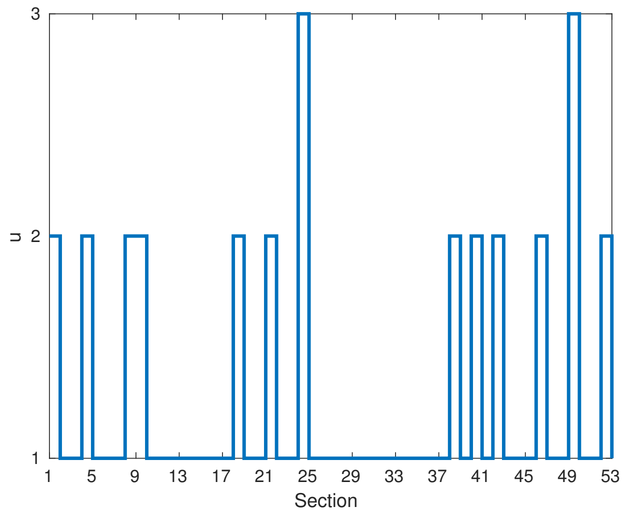
4.4.2 Discussion of Results

A representative run is conducted to demonstrate how the proposed multi-level approach works. The partial results of the high-level MPC controller from the line between Den Haag and Rotterdam are shown in Figure 4.3.

As the deterioration dynamics and initial risk level of each section is different, the resulting intervention plan is also very different for different sections. For example, replacing is suggested at the first time step for section 27, because its initial risk level is already very high



(a) Risk levels of the whole railway network at time step 6 (current time step) and time step 7 (next time step).



(b) Interventions suggested by the high-level MPC controller at time step 6 for the whole railway network.

Figure 4.4: High-level simulation results for the whole railway network at time step 6.

(almost 0.6). On the contrary, grinding is firstly suggested at time step 8, i.e. 24th month within the 5-year planning horizon for section 26, as its initial risk level is almost 0. The next grinding operation is suggested at least 18 months after a replacement, as the growth of the risk level is very slow for a newly replaced section of rail. A maintenance intervention is usually suggested when the risk level is sufficiently high, i.e. over 0.8, to justify the high cost of track maintenance operations. Unlike for the time-based cyclic maintenance approach, the interval between two consecutive interventions is flexible and ranges from 6 to 9 months.

The simulation results of the whole case study network at a representative time step ($k = 7$) are shown in Figure 4.4. From Figure 4.4a we can clearly see that no section in the whole network has a risk level exceeding the critical threshold at time step 7, indicating the network is safe at the current time step. Furthermore, the simulated risk levels of all the sections in the

next time step are also below the threshold, ensuring the safety of the whole network three months later.
The risk level at time step 7 is the outcome of the intervention at time step 6. As shown in Figure 4.4b, 11 sections are to be ground and 2 sections are to be replaced within the three months between time step 6 and 7. The results of the low-level crew scheduling problem to execute these planned grindings over the railway network are shown in Figure 4.5. The planned grindings are performed in two different operations. In the first grinding operation, the maintenance crew starts from the maintenance base and drives to Dordrecht. The maintenance crew then spends 2 hours grinding section 38 and 40 between Dordrecht and Lage Zwaluwe. It then traverses the “triangle" formed by Lage Zwaluwe, Roosendaal, and Breda, spending one hour grinding one section at each edge of the triangle. Finally, the maintenance crew drives the same way back from Lage Zwaluwe to the maintenance base. The total maintenance time in this grinding operation is 5 hours, which is less than the allocated 6-hour maintenance slot. No additional cost for extra maintenance time is incurred for this operation. Similarly, a second tour is made by the maintenance crew to grind the remaining 6 sections in the other part of the network, as shown by the dotted line in Figure 4.5. No additional cost for extra maintenance time is incurred for this operation neither.

4.4.3 Comparison with Centralized MPC

A computational comparison is performed between the centralized MPC approach and the distributed MPC approach. We generate 14 MPC optimization problems for 14 fictional railway networks with a number of sections ranging from 10 to 140. The current states and values of uncertain parameters are randomly generated following a normal distribution. The trend of the CPU time with an increasing number of sections is plotted in Figure 4.6. For the first 13 test instances, the distributed approach always takes a much shorter CPU time than the centralized approach. Moreover, the centralized approach fails when the number of sections becomes 140, due to memory related issues, while the distributed approach can still find a solution within 40 minutes.
As the distributed approach based on Dantzig-Wolfe decomposition is inexact, for each test instance we also check its relative loss of optimality, compared to the global optimum provided by the centralized approach. For the first 13 test instances, the distributed approach is able to achieve global optimality. As the centralized approach becomes intractable when the number of sections reaches 140, we cannot conclude whether the distributed approach finds the global optimum for the largest test instance.

4.4.4 Comparison with Alternative Approaches

In this section we compare the results of the proposed chance-constrained MPC approach to two alternative approaches, namely, the nominal MPC approach and the cyclic approach. The only difference between the nominal MPC approach and the chance-constrained MPC approach is that the nominal approach considers only the mean values of the uncertain parameters in the deterioration model. So it can be viewed as a deterministic counterpart of the chance-constrained MPC approach. The cyclic approach uses a time-based maintenance strategy, and performs grinding and replacing at a fixed optimal interval. Unlike the two

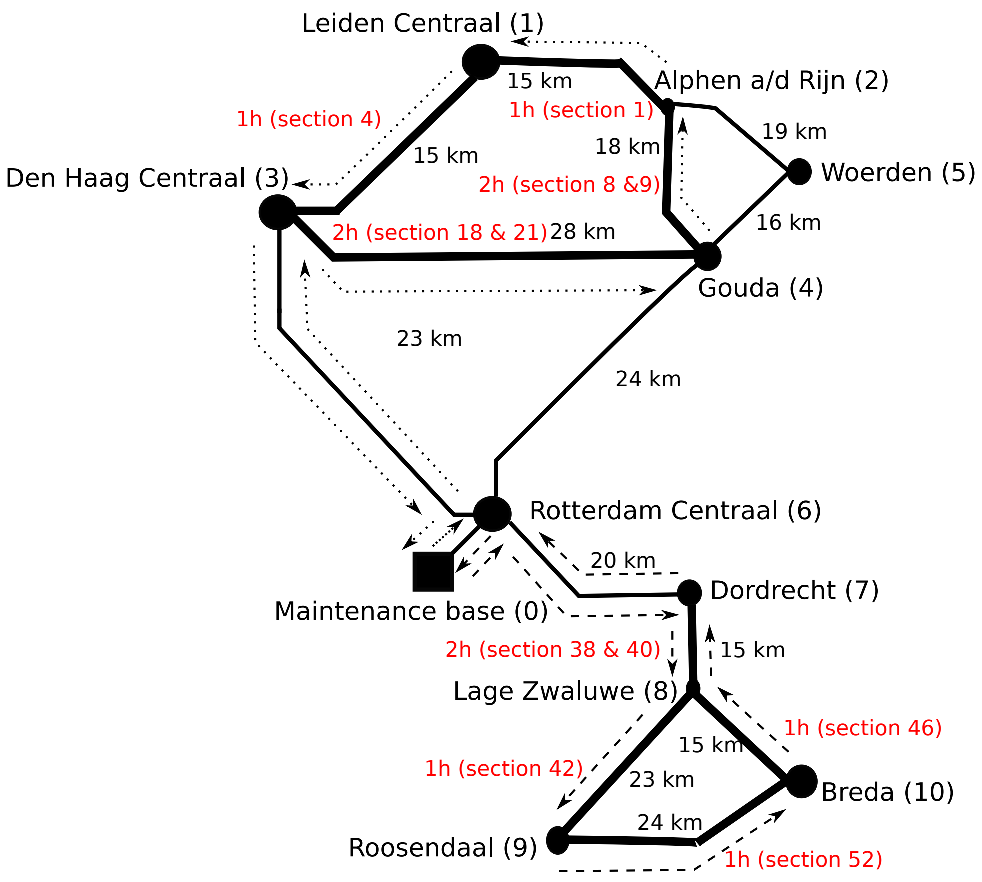


Figure 4.5: Result of the low-level maintenance crew scheduling problem at the 6th time step. The railway lines that must be ground within the next time step (3 months) are marked in bold. For each railway line that requires grinding, the exact sections to be ground, and the minimum time to grinding them also provided. The dashed and the dotted arrows show the resulting itinerary of the first and second tour of the maintenance crew, respectively.

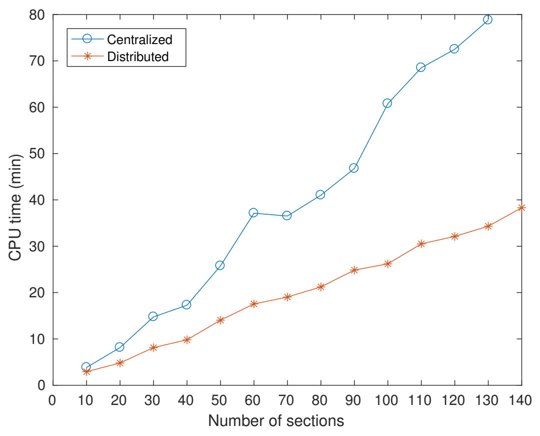


Figure 4.6: Computational comparison of centralized MPC and distributed MPC based on the Dantzig-Wolfe decomposition for a prediction horizon $N_p = 3$.

Table 4.2: A comparison between the proposed chance-constrained MPC approach (with subscript "CC"), the nominal approach (with subscript "Nom"), and the cyclic approach (with subscript "Cyc").

Run	Constraint violation			Closed-loop performance			CPU time (h)	
	v_{CC} (%)	v_{Nom}(%)	v_{Cyc} (%)	$\frac{J_{\text{CC}}}{J_{\text{Cyc}}}$ (%)	$\frac{J_{\text{Nom}}}{J_{\text{Cyc}}}$ (%)	J_{Cyc}	T_{CC}	T_{Nom}
1	0	0.063	0	39.335	34.148	670502	5.671	0.003
2	0	0.006	0	38.127	36.577	670504	5.075	0.003
3	0	0.353	0	37.635	35.043	670503	5.062	0.003
4	0	0.129	0	37.606	33.344	670502	5.703	0.003
5	0	0	0	36.354	34.536	670502	5.141	0.003
6	0	0.082	0	36.413	35.803	670502	5.802	0.003
7	0	0.021	0	39.425	36.250	670503	5.134	0.003
8	0	0.053	0	38.440	35.028	670500	5.126	0.003
9	0	0.0344	0	40.244	33.359	670503	5.088	0.003
10	0	0.172	0	38.902	34.656	670503	5.082	0.003

MPC approaches, the cyclic approach is an offline approach, i.e. an optimal maintenance intervention plan for the whole planning horizon is computed beforehand and applied to the infrastructure network without updating it using real-time measurements or simulation. The formulation and solution approach of the cyclic approach is presented in Appendix **??**. We created ten test instances in which the values of the uncertain parameters are randomly generated following a Gaussian distribution. Three criteria, safety, cost-effectiveness, and computational efficiency, are applied to evaluate the three approaches. Safety is measured by constraint violation v, which is calculated as following:

$$v = \max\left(\frac{x^{\text{con}}_{\text{worst}} - x^{\text{con}}_{\text{max}}}{x^{\text{con}}_{\text{max}}}, 0\right), \tag{4.78}$$

where $x^{\text{con}}_{\text{worst}}$ is the highest risk level for all sections within the entire planning horizon. Cost-effectiveness is measured by closed-loop performance, which can be calculated by the summation of all the n local objectives (4.2) evaluated over the entire 5-year planning horizon. Finally, computational efficiency is measured by the CPU time required to solve all the high-level[4] optimization problems at all time steps. We only compare the CPU time of the two MPC approaches, since the cyclic approach is an offline approach in which only one optimization problem must be solved for the entire planning horizon. The summary of the comparison between the three approaches is presented in Table 4.2.

According to Table 4.2, the proposed chance-constrained approach is safe, as it has no constraint violation for the 10 test runs. It is also cost-effective, as the closed-loop performance is less than 40% of the reference cyclic approach in almost every test run. However, the chance-constrained MPC approach is also the slowest in terms of CPU time. It is almost 1700 times slower than the nominal MPC approach. This is because a much larger MILP problem (571 times as large as that of the nominal MPC approach) must be solved at each time step due to the consideration of high-dimensional parameter uncertainty. However,

[4]This is because the same formulation for the low-level problems is used for all three approaches.

the long computation time does not impair its real-time implementability, as track degradation is a very slow process (3-month sampling time in the case study). The nominal MPC approach is fast and scores the best in closed-loop performance. However, as it does not take into account any uncertainty, the resulting intervention plan is unsafe, as shown by the constraint violations in the test runs. On the contrary, the cyclic maintenance approach results in very conservative intervention plans which tend to “over-maintain" the asset. The resulting intervention plans are safe (i.e. there is no constraint violation), but not cost-effective (i.e. they gave the worst closed-loop performance).
From the comparison with two alternative approaches, we can conclude that the proposed chance-constrained MPC approach is the most suitable one for track maintenance planning, as it is safe, cost-effective, and real-time implementable.

4.5 Conclusions

In this chapter we have developed an integrated approach for both long-term condition-based maintenance planning and short-term maintenance crew scheduling of a railway infrastructure network. Uncertainties in the deterioration dynamics are taken into account in condition-based maintenance planning, and distributed optimization scheme is adopted to improved the scalability of the proposed approach. An exact MILP formulation is proposed for the optimal scheduling and routing of maintenance crews with flexible maintenance time slot. This integrated approach can be applied to the optimal treatment of typical track defects like squats and ballast defects. The proposed approach has been illustrated by a numerical case study of the optimal treatment of squats for a regional Dutch railway network. Comparison with the centralized approach shows that the adopted distributed optimization scheme based on Dantzig-Wolfe decomposition is scalable. Comparison with two alternative approach shows that the proposed approach yields an excellent trade-off between safety and cost-effectiveness.

Chapter 5

Fixed-Destination Multi-Depot Multiple Traveling Salesman Problem

5.1 Problem Description

The Traveling Salesman Problem (TSP) forms a basis for many optimization problems in logistics, finance, and engineering. Several variants exist to accommodate for different problem types. In this chapter we discuss the Fixed-Destination Multi-Depot Multiple Traveling Salesman Problem (FmMTSP), where several salesmen will start from different depots, and they are required to return to the depot they originated from.
We propose a novel formulation for this problem using 2-index binary variables and node currents, and compare it to other 2-index formulations from the literature. This novel formulation requires less binary variables and continuous variables to formulate a problem, resulting in lower computation times. Using a large benchmark the effectiveness of the new formulation is demonstrated. Moreover, the node-current cycle imposement constraints can be applied to other node routing problems like vehicle routing problem, which is the framework for the maintenance crew scheduling problem in Chapter 4.
For multiple traveling salesman problem with one depot (and multiple salesmen) it is obvious that all salesmen should return to this single depot. However, when considering multiple depots, two situations can occur: either the salesmen may end their tour at any depot, or they are required to return to their original depot. The latter is a restricted case of the former and can be obtained by using additional constraints, as will be discussed next.

5.1.1 Description of Multi-depot Multiple Traveling Salesman Problem

We consider Multi-depot Multiple Traveling Salesman Problem (MmTSP), where there are D depots available from where C cities should be visited. Each depot d has a certain number of salesmen available, indicated by m_d. Each city should be visited by *one and only one* salesman. The cost to travel from city i to j is denoted by the constant c_{ij}. The costs can be asymmetric, i.e., c_{ij} and c_{ji} might be different. The decision variable x_{ij} indicates whether ($x_{ij} = 1$) or not ($x_{ij} = 0$) city j is visited *directly after* city i by a salesman.
The locations of both the cities and the salesmen can be taken into account in the modeling by denoting both the D locations of the depots (where the salesmen are) and the C locations of the cities as one set of $L = C + D$ locations. The sets $\mathscr{D}$, $\mathscr{C}$, and $\mathscr{L}$ —associated with the

depots, the cities, and the locations respectively— are defined as

$$\mathscr{D} = \{1, \ldots, D\}, \ \mathscr{C} = \{D+1, \ldots, L\}, \ \mathscr{L} = \mathscr{D} \cup \mathscr{C}. \tag{5.1}$$

5.1.2 Description of Fixed-Destination Multi-Depot Multiple Traveling Salesman Problem

For the fixed-destination problems, the additional restriction that all salesmen should return to their original depots makes the Fixed-Destination Multi-Depot Multiple Traveling Salesman Problem (FMmTSP) more difficult to solve. The reason is that compared with the non-fixed destination MmTSP, new auxiliary variables or decision variables of a higher index are required for the fixed-destination setting to impose the additional restriction that each salesman must return to his departing depot. In the literature often the (fixed-destination) multi-depot problems (or multi-vehicle problems when considering vehicle routing) are solved using a 3-index variant of the decision variables [9, 32, 120, 131], thereby drastically increasing the number of binary variables (computational complexity) with each depot added to the problem. The three indices represent 1) the origin, 2) the destination, and 3) the depot (vehicle) number. An example of an MILP description of both nonfixed- and fixed-destination MmTSP formulations using 3-index formulations is given in [82].
Recently, formulations of the MmTSP [114] and FMmTSP [10] using 2-index binary variables have been presented in the literature. Both methods make use of a copy $\mathscr{D}'$ of the set of depot nodes $\mathscr{D}$, where one depot serves as a starting point and the other depot serves as the end point of a tour for a salesman. For D depot nodes and C city nodes in the original problem, there will be $2D$ depot nodes and C city nodes in the extended problem. The resulting set of nodes in the graph is

$$\mathscr{L}' = \mathscr{D} \cup \mathscr{C} \cup \mathscr{D}', \tag{5.2}$$

where the copied set of the D depots in $\mathscr{D}$ is defined as

$$\mathscr{D}' = \{1', \ldots, D'\} = \{L+1, \ldots, L+D\}, \tag{5.3}$$

such that node i and node $i' = L + i$ represent the starting and end depot of depot $i \in \mathscr{D}$, respectively. This formulation results in $L' = 2D + C$ nodes in the graph[1].
A transformation of the MmTSP problem into an asymmetric TSP has been proposed by Oberlin et al. [113, 114] by using a copy of the depot nodes as in (5.3). More recently, Bektaş [10] has proposed a method to solve the FMmTSP using $\mathscr{D}'$ based on commodity flows. In Section 5.3 we will introduce a formulation that only requires $L = D + C$ nodes to represent the same problem.

[1]With an efficient implementation, where the starting node only has outgoing arcs and the end node only has incoming arcs, the number of arcs (and thereby the number of binary variables) remains the same.

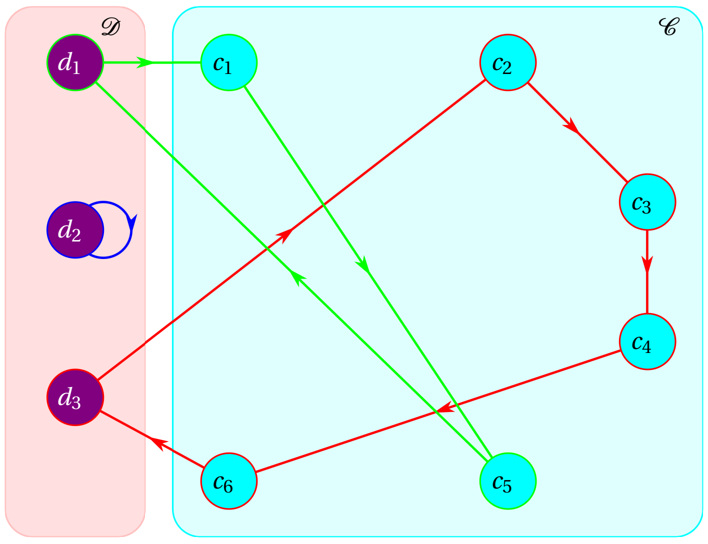


Figure 5.1: A solution to the FmMTSP. Cycles should be eliminated in the set $\mathscr{C}$, whereas they should be imposed in the set $\mathscr{L} = \mathscr{D} \cup \mathscr{C}$.

5.2 Mathematical Formulation

The FMmTSP can be described as a mathematical program consisting of the following components[2]:

minimize	*costs*	(5.4a)
subject to	*assignment constraints*	(5.4b)
	cycle elimination constraints	(5.4c)
	cycle imposement constraints	(5.4d)

In this section we provide a brief overview of the currently available types of constraints for each of the components. For a more thorough discussion on the available cycle (subtour) elimination constraints (including the relations of their linear relaxation strengths) we refer to [134] and the references therein.

For ease of notation we will use the index d' and the sets $\mathscr{D}'$ and $\mathscr{L}'$ in this section, where their definition will depend on the type of formulation that is used, as specified in Table 5.1.

Table 5.1: ***The definition of prime symbols for formulations with and without depot-node copies.***

	with copies of depots	without copies of depots
d'	$d+L$	d
$\mathscr{D}'$	$\{L+1,\ldots,L+D\}$	$\mathscr{D}$
$\mathscr{L}'$	$\{1,\ldots,L+D\}$	$\mathscr{L}$

[2]Constraints (5.4c) are commonly known as subtour elimination constraints (SECs). Since the term 'subtour' has the implicit property of being undesired, we will present the counterpart of the SECs as cycle imposement constraints, thereby using the modern term 'cycle' instead of 'subtour'. For consistency, we therefore also use the term cycle elimination constraints.

5.2.1 Costs

The cost in (FMm)TSPs is the distance the salesmen travel, resulting in minimising the *total travel distance*

$$J_{\text{td}} = \sum_{i \in \mathscr{L}'} \sum_{j \in \mathscr{L}'} c_{ij} x_{ij}, \tag{5.5}$$

where $c_{ij} \geq 0$ is the travel distance between cities i and j, and $x_{ij} \in \{0,1\}$ is a decision variable satisfying

$$x_{ij} = \begin{cases} 1 & \text{if city } j \text{ is visited directly after city } i, \\ 0 & \text{otherwise.} \end{cases} \tag{5.6}$$

When using 3-index decision variables x_{ijk} the total travel distance is given by

$$J_{\text{td}} = \sum_{i \in \mathscr{L}} \sum_{j \in \mathscr{L}} \sum_{k \in \mathscr{D}} c_{ij} x_{ijk}, \tag{5.7}$$

where $c_{ij} \geq 0$ is the travel distance between cities i and j, and $x_{ijk} \in \{0,1\}$ is a decision variable satisfying

$$x_{ijk} = \begin{cases} 1 & \text{if city } j \text{ is visited directly after city } i \text{ by a} \\ & \text{salesman originating from depot } k, \\ 0 & \text{otherwise.} \end{cases} \tag{5.8}$$

5.2.2 Assignment Constraints

The assignment constraints ensure that each node has exactly one incoming arc and one outgoing arc (see 5.1), thereby satisfying a necessary condition for visiting the cities *once and only once.*

Description of the Assignment Constraints

The assignment constraints for the (F)MmTSP are given by [9, 87]:

$$\sum_{j \in \mathscr{L}'} x_{dj} = m_d \ \forall\ d \in \mathscr{D} \tag{5.9a}$$

$$\sum_{j \in \mathscr{L}'} x_{ij} = 1 \qquad \forall\ i \in \mathscr{C} \tag{5.9b}$$

$$\sum_{i \in \mathscr{L}'} x_{ij} = 1 \qquad \forall\ j \in \mathscr{C} \tag{5.9c}$$

$$\sum_{i \in \mathscr{L}'} x_{id'} = m_d \ \forall\ d' \in \mathscr{D}' \tag{5.9d}$$

$$x_{ij} \in \{0,1\} \qquad \forall\ i,j \in \mathscr{L}' \tag{5.9e}$$

Due to (5.9a) all of the m_d salesmen will leave their depot d, and by (5.9b) each city i is succeeded by exactly one location (a salesman leaves the city). Furthermore, equations

(5.9c) ensure that each city j is preceded by exactly one location (a salesman enters the city), whereas (5.9d) ensures that m_d salesmen will return to depot d'. The set $\mathscr{D}'$ denotes —depending on the problem formulation— either a copy of the depot nodes, or the original set $\mathscr{D}$ of depot nodes, as defined in Table 5.1. Finally, (5.9e) ensures that the decision variable x_{ij} is treated as a binary variable.

Variants for the Assignment Constraints

The assignment constraints in (5.9) force all m_d salesmen to leave their depot, and also require m_d salesmen to return to depot d. The former is restrictive for both fixed-destination and nonfixed-destination problems, whereas the latter only restricts the solutions for the FMmTSP. Both restrictions can be loosened as shown next.

Idle salesmen: To allow salesmen to stay at the depot without visiting a city (hence some salesmen may be 'idle'), the equality constraints (5.9a) and (5.9d) can be substituted by (see [159])

$$\sum_{j\in\mathscr{L}'} x_{dj} \le m_d \qquad \forall\, d \in \mathscr{D} \tag{5.9a*}$$

$$\sum_{i\in\mathscr{L}'} x_{id'} = \sum_{j\in\mathscr{L}} x_{dj} \;\forall\, d' \in \mathscr{D}' \tag{5.9d*}$$

where (5.9a*) limits the number of salesmen that can leave depot d to m_d (which equals the number of salesmen present at depot d), whereas (5.9d*) ensures that the same number of salesmen that have left the depot, will also return to the depot.

Fixed-capacity depots: For nonfixed-destination problems the number of salesmen at a depot will in general be different before and after the salesmen traveled. To avoid solutions where certain depots will receive more salesmen than they can facilitate, an upper bound on the number of salesmen that are allowed to return to each specific depot should be set. To accomplish this we propose the following.
If the capacity of depot d (with d' the associated end depot) is $q_{d'}$ salesmen one could substitute (5.9d*) with

$$m_d + \sum_{i\in\mathscr{L}'} x_{id'} \le q_d + \sum_{j\in\mathscr{L}'} x_{dj} \qquad \forall\, d' \in \mathscr{D}' \tag{5.9d$_1^\star$}$$

$$\sum_{d'\in\mathscr{D}'}\sum_{j\in\mathscr{L}'} x_{d'j} = \sum_{d\in\mathscr{D}}\sum_{i\in\mathscr{L}'} x_{id} \tag{5.9d$_2^\star$}$$

Inequalities (5.9d$_1^\star$) ensure that no more than $q_{d'}$ salesmen end up in depot d',[3] whereas (5.9d$_2^\star$) ensures that all the salesmen that leave a depot will also return to a depot.

5.2.3 Cycle Elimination Constraints

Note that the constraints (5.9) do not avoid

(1) the existence of cycles (subtours) in $\mathscr{C}$, resulting in routes along cities that do not have a salesman associated with them,

[3]The number of salesman returning to depot d' is less or equal to the capacity q_d minus the number of salesmen m_d present at the start plus the number of salesmen that leave depot d.

(2) the existence of cycles in $\mathscr{L}'$ containing more than one node from $\mathscr{D}$, resulting in a schedule where salesmen end their tour at an arbitrary depot.

The former situation would result in a schedule where some cities will not be visited by a salesman (since no-one is assigned to do so), whereas the latter situation would result in a schedule where the salesmen do not have the guarantee that they return to their original depot. To assure that *each city is visited by a salesman*, solutions using *cycle elimination constraints* have been proposed in the literature [37, 52, 54, 103]. These constraints are based on different concepts, for which a brief description will be provided next; a more detailed description can be found in [117].

Loop Conditions

The loop conditions were introduced in the seminal work [37], which can be stated as

$$\sum_{i,j\in\mathscr{S}} x_{ij} \leq |\mathscr{S}| - 1 \quad \forall\, \mathscr{S} \subset \mathscr{L}', 2 \leq |\mathscr{S}| \leq L' - 1, \tag{5.10}$$

where $|\mathscr{S}|$ represents the cardinality of set $\mathscr{S}$. These constraints provide strong linear programming relaxations, but the number of constraints grows exponentially with the number of nodes.

Node Potentials

An approach for eliminating cycles was proposed in [103] by using additional variables u_i that represent node potentials. Using the strengthened formulation in [40], the C continuous variables u_i should satisfy

$$u_i - u_j + Cx_{ij} + (C-2)x_{ji} \leq C - 1 \quad \forall\, i, j \in \mathscr{C}, \tag{5.11}$$

resulting in C^2 inequality constraints.
The node potential representation has been extended by Kara and Bektaş [10, 82] to set *workload bounds*[4] on the number of cities a salesman should visit. Denoting $\underline{u}$ and $\overline{u}$ as the minimum and maximum number of cities the salesmen may visit respectively, the cycle imposement constraints

$$u_i - u_j + \overline{u}x_{ij} + (\overline{u}-2)x_{ij} \leq \overline{u} - 1 \qquad \forall\, i, j \in \mathscr{C} \tag{5.12a}$$

$$u_i + (\overline{u}-2)\sum_{d\in\mathscr{D}} x_{di} - \sum_{d'\in\mathscr{D}'} x_{id'} \leq \overline{u} - 1 \qquad \forall\, i \in \mathscr{C} \tag{5.12b}$$

$$u_i + \sum_{d\in\mathscr{D}} x_{di} + (2-\underline{u})\sum_{d'\in\mathscr{D}'} x_{id'} \geq 2 \qquad \forall\, i \in \mathscr{C} \tag{5.12c}$$

ensure that each salesman will be assigned between $\underline{u}$ and $\overline{u}$ cities to visit, and city i will be the u_i-th city a salesman visits[5]. Inequalities (5.12a) provide cycle elimination constraints. In both (5.12b) and (5.12c) the first summation is 1 if and only if node i represents the first

[4]Loop conditions representation with workload bound were first proposed in [82] for the mTSP (single depot), and were extended in [10] for the MmTSP (multidepot)

[5]In [10, 82] it is stated that these inequality constraints are only valid for $\underline{u} \geq 4$; this is only under the restriction that salesmen should at least visit two cities, and hence $x_{di} = x_{id'} = 1$ is not allowed. Lifting this restriction by allowing salesmen to visit zero or one city these inequality constraints are valid for all $\underline{u} \geq 0$.

city of a tour (and it is 0 otherwise), and the second summation is 1 if and only if node i represents the last city of a tour (and is 0 otherwise). Therefore, (5.12b) and (5.12c) ensure that $u_i \leq \overline{u}$ and $u_i \geq \underline{u}$ for the last cities in a tour, thereby setting the desired upper and lower bound respectively on the number of visited cities per salesman.
For the TSP with time windows [5] the constraints

$$t_i - t_j + \tau_{ij} + Tx_{ij} \leq T \tag{5.13}$$

can be seen as a variant of the node potential approach, where t_i is the time instant city i is visited, τ_{ij} is the potential difference between the nodes, and T is a sufficiently large constant.

Commodity Flows

Commodity flows are introduced in [54] as a means for eliminating undesired cycles. The $\not{L}^2$ continuous variables f_{ij} should satisfy the constraints

$$\sum_{j\in\mathscr{L}'} f_{ij} - \sum_{j\in\mathscr{L}'} f_{ji} = 1 \qquad \forall\, i \in \mathscr{C} \tag{5.14a}$$

$$f_{ij} \leq Cx_{ij} \qquad \forall\, i \in \mathscr{C}, j \in \mathscr{L}' \tag{5.14b}$$

$$f_{ij} \geq 0 \qquad \forall\, i, j \in \mathscr{L}' \tag{5.14c}$$

Extensions to two-commodity flows [51] and multi-commodity flows [31, 168] have been proposed, resulting in stronger linear programming relaxations [86, 118].

Time Periods

For a graph with $\not{L}$ nodes, a cycle elimination formulation was presented in [52], using a 3-index binary variable representation x_{ijt}, where

$$x_{ijt} = \begin{cases} 1 & \text{if } i \text{ precedes } j \text{ as the } t\text{-th node in the tour,} \\ 0 & \text{otherwise.} \end{cases} \tag{5.15}$$

The index $t \in \mathscr{T}$ represents the time period in which the salesman travels from city i to city j. To ensure that all cities are visited in some time period, and that in each time period only one city is visited, the $O(\not{L}^3)$ binary variables x_{ijt} should satisfy

$$\sum_{i\in\mathscr{L}'} \sum_{j\in\mathscr{L}'} \sum_{t\in\mathscr{T}} x_{ijt} = \not{L} \tag{5.16a}$$

$$\sum_{j\in\mathscr{L}'} \sum_{t\in\mathscr{T}\setminus\{1\}} tx_{ijt} - \sum_{j\in\mathscr{L}'} \sum_{t\in\mathscr{T}} tx_{jit} = 1 \qquad \forall\, i \in \mathscr{L}' \tag{5.16b}$$

$$x_{ijt} \in \{0,1\} \qquad \forall\, i, j \in \mathscr{L}' \qquad \forall t \in \mathscr{T} \tag{5.16c}$$

Constraints (5.16a) and (5.16c) replace the assignment constraints (5.9). Equality constraints (5.16b) ensure that in each time period t exactly one city i is visited.

5.2.4 Cycle Imposement Constraints

To ensure that *each salesman returns to the original depot*, additional constraints are needed to enforce cycles that start and end in the same depot (or paths leading from one start depot towards the associated end depot). Opposite to the cycle elimination constraints, the constraints that enforce the existence of a certain amount of cycles in a graph can be seen as *cycle imposement constraints*. To the author's best knowledge, currently only three approaches exist for obtaining fixed-destination solutions: using 3-index binary variables, using 2-index binary variables plus commodity flow variables [10, 82], or using the path elimination constraints [11], which introduces no new variables, but modifies the definition of the 2-index binary variable associated with each arc. We will introduce a fourth approach in Section 5.3 that is based on node currents, which can be seen as the dual of the node potentials introduced in [103] presented in (5.11).

3-Index Formulation

Using decision variables x_{ijd} that satisfy

$$x_{ijd} = \begin{cases} 1 & \text{if } i \text{ precedes } j \text{ directly in the tour of depot } d, \\ 0 & \text{otherwise,} \end{cases} \tag{5.17}$$

the existence of D cycles can be enforced [82] using

$$\sum_{j\in\mathscr{C}} x_{djd} = m_d \qquad \forall\, d \in \mathscr{D} \tag{5.18a}$$

$$\sum_{d\in\mathscr{D}} \left\{ x_{djd} + \sum_{i\in\mathscr{C}} x_{ijd} \right\} = 1 \qquad \forall\, j \in \mathscr{C} \tag{5.18b}$$

$$x_{djd} + \sum_{i\in\mathscr{C}} x_{ijd} = x_{jdd} + \sum_{i\in\mathscr{C}} x_{jid} \qquad \forall\, d \in \mathscr{D},\, j \in \mathscr{C} \tag{5.18c}$$

$$\sum_{j\in\mathscr{C}} x_{djd} = \sum_{j\in\mathscr{C}} x_{jdd} \qquad \forall\, d \in \mathscr{D} \tag{5.18d}$$

Constraints (5.18a) and (5.18d) ensure that exactly m_d salesmen depart and return to depot d. Constraints (5.18b) guarantee that each city are visited exactly once. Constraints (5.18c) ensure the path continuity. Together with the degree constraints (5.18b), constraints (5.18c) ensure that a salesman starts at depot d and visits city j first will either continue to another city i or return to the same depot. Note that this formulation uses $O(L^2D)$ binary variables, and the number of binary variables increases cubically with the number of depots (as opposed to the quadratic increase for 2-index formulations).

Multi-Commodity Flow Formulation

A multi-commodity flow problem is a network flow problem with multiple flows [129]. Besides applications for cycle elimination (as discussed in Section 5.2.3) it was shown by Bektaş [10] that this concept can also be used for enforcing fixed-destination solutions to the mTSP. In this context a commodity f^d represents the number of salesmen originating from depot

d. The constraints based on commodity flows [10] are given by

$$\sum_{j\in\mathscr{C}\cup\mathscr{D}'} f_{dj}^{d} - \sum_{j\in\mathscr{D}\cup\mathscr{C}} f_{jd}^{d} = m_d \qquad \forall\, d \in \mathscr{D} \tag{5.19a}$$

$$\sum_{j\in\mathscr{C}\cup\mathscr{D}'} f_{ij}^{d} - \sum_{j\in\mathscr{D}\cup\mathscr{C}} f_{ji}^{d} = 0 \qquad \forall\, i \in \mathscr{C}, d \in \mathscr{D} \tag{5.19b}$$

$$\sum_{j\in\mathscr{D}\cup\mathscr{C}} f_{jd'}^{d} - \sum_{j\in\mathscr{C}\cup\mathscr{D}'} f_{d'j}^{d} = m_d \qquad \forall\, d' \in \mathscr{D}' \tag{5.19c}$$

$$0 \le f_{ij}^{d} \le x_{ij} \qquad \forall\, i, j \in \mathscr{L}', d \in \mathscr{D} \tag{5.19d}$$

In this formulation each depot d in $\mathscr{D}$ acts as a source of commodity f^d, while each depot d' in $\mathscr{D}'$ acts as a sink where only commodity $d = d' - L$ is accepted. By (5.19a) exactly m_d units of commodity f^d will leave depot d (meaning that m_d salesmen will leave the depot). Constraints (5.19b) are flow-conservation constraints that guarantee that the same amount of commodity f^d entering a node i will also leave node i (meaning that each salesman that enters a city will also leave the city). By (5.19c) exactly m_d units of commodity f^d will reach depot d' (meaning that m_d salesmen will arrive at the duplicate depot node d'). Combined with the assignment constraints (5.9), the inequality constraints (5.19d) restrict the commodities to only flow along arcs that are part of the selected routes; if $x_{ij} = 0$ no commodity can flow from city i to city j. This formulation uses $L'^2 D$ commodity flow variables f_{ij}^d, where $L' = 2D + C$ is the number of nodes in the graph.

Path Elimination Constraints

The path elimination constraints, first proposed in [11], fix the destination of each salesman by eliminating paths that start and end in two different depots. The idea is inspired by the chain-baring constraints introduced in [88]. Although originally designed for location routing problems, path elimination constraints have been applied to many fixed-destination mTSP variants [15, 34, 157]. The decision variables are defined as follows:

$$x_{ij} = \begin{cases} 1 & \text{if } (i, j) \text{ is traversed exactly once,} \\ 0 & \text{otherwise,} \end{cases} \tag{5.20}$$

for $i, j \in \mathscr{L}$ and

$$w_{ij} = \begin{cases} 1 & \text{if } (i, j) \text{ is in a return trip,} \\ 0 & \text{otherwise,} \end{cases} \tag{5.21}$$

for all $i \in \mathscr{D}$, $j \in \mathscr{C}$.
The path elimination constraints given in [11] are

$$\sum_{p,q\in\mathscr{S}\cup\{i,j\}} x_{pq} + \sum_{d\in\mathscr{D}^\circ} x_{jd} + \sum_{d\in\mathscr{D}\backslash\mathscr{D}^\circ} x_{id} \le |\mathscr{S}| + 2 \tag{5.22}$$
$$\forall i, j \in \mathscr{C}, \forall \mathscr{S} \subseteq \mathscr{C}\backslash\{i, j\}, \forall \mathscr{D}^\circ \subset \mathscr{D}$$

If all cities in $\mathscr{S} \cup \{i, j\}$ are in a consecutive path, then the loop conditions (5.10) are satisfied with equality, i.e., $\sum_{p,q\in\mathscr{S}\cup\{i,j\}} = |S| + 1$. Because of constraints (5.22) we have $\sum_{d\in\mathscr{D}^\circ} x_{jd} + \sum_{d\in\mathscr{D}\backslash\mathscr{D}^\circ} x_{id} \le 1$. This indicates that a path connected to a depot in $\mathscr{D}^\circ$ cannot be connected to another depot in $\mathscr{D}\backslash\mathscr{D}^\circ$. Constraints (5.22) can eliminate all unwanted paths that visit at least two cities and start and end in different depots. The constraints

$$\sum_{d\in\mathscr{D}} x_{dj} + w_{dj} \leq 1 \quad \forall j \in \mathscr{C} \tag{5.23}$$

are needed to also eliminate undesired paths that visit only one city (and start and end in different depots). This formulation requires $O(L^2 + DC)$ binary variables, and the number of constraints ($O(2^C C^2 2^D)$ grows exponentially with the number of cities and depots. Just as for the loop conditions (5.10) these constraints are suitable for branch-and-cut implementations, but not to formulate the complete problem and use a MILP solver to obtain optimal solutions.

5.3 Novel Formulation of Fixed-Destination Multi-Depot Multiple Traveling Salesman Problem

In the previous section it has been discussed that the FMmTSP can be formulated using four components (as given in (5.4)). The *cost* that needs to be minimised is the total travel distance of the salesmen, for which a standard formulation is given in (5.5). For the *assignment constraints* the conventional constraints are given in (5.9), but variations can be used to e.g. allow some salesmen to be idle or to limit the number of salesmen that may end at a depot, as discussed in Section 5.2.2. The component that has the most variants in literature introduces the *cycle elimination constraints* (or subtour elimination constraints), discussed in Section 5.2.3.
The component that has received the least attention introduces the *cycle imposement constraints*. Until recently, fixed-destination solutions for mTSPs and its variants have been ensured by using 3-index formulations of the decision variables as discussed in Section 5.2.4. The formulation of Bektaş proposed in [10] is the first formulation that ensures fixed-destination solutions using 2-index binary variables and the multi-commodity flow constraints presented in Section 5.2.4. In the current section we will introduce a novel approach for cycle imposement. This approach is also based on 2-index binary variables, where one continuous variable per node is added to the formulation to ensure a fixed-destination solution. The new formulation needs a few less binary variables than the formulation of [10]; more importantly, it uses DL times less continuous variables than the multi-commodity flow approach.

5.3.1 Cycle Imposement Through Node Currents

Inspired by the node potentials of Miller, Tucker, and Zemlin [103] we propose an alternative formulation of the FMmTSP using *node currents* [22]. Similar to the commodity flow transported between cities over the arcs, the current in an electric circuit can also be considered as a flow in a directed graph. With the depots representing current sources, a proper electric circuit contains only cycles (if not, there would be an open circuit or short circuit), which corresponds to the cycle imposement constraint stating that every salesman must return to his departing depot. A flow conservation law combined with assignment constraints forces the current flowing into a node to be equal to the current flowing out of the node, so that nodes in the same cycle must have the same current. Thus, we can view the current k_i as a property of node i (instead of a property of the arc x_{ij}).

Node Current Formulation

For the newly proposed node current formulation there is no need to use copies of the depot nodes. Therefore, this formulation will use less binary variables as for the copy-based formulation, since there are D less nodes to represent the graph. Fixed-destination solutions can be obtained by using $L = D + C$ continuous variables k_i satisfying the cycle imposement constraints

$$k_d = d \quad \forall\, d \in \mathscr{D} \tag{5.24a}$$
$$k_i - k_j \leq (D-1)(1-x_{ij})\ \forall\ i,\, j \in \mathscr{L} \tag{5.24b}$$

resulting in $k_i \leq k_j$ if $x_{ij} = 1$ using $(D+C)^2 - C$ constraints. Additionally, one can obtain the tighter constraint $k_i = k_j$ if $x_{ij} = 1$ by adding

$$k_j - k_i \leq (D-1)(1-x_{ij})\ \forall\ i,\, j \in \mathscr{L}. \tag{5.24c}$$

to the constraints (5.24a)-(5.24b), resulting in stronger linear relaxations at the cost of using more constraints. If the minimum number of cities to visit is set to be at least two, one can replace (5.24b)-(5.24c) with

$$k_i - k_j \leq (D-1)(1-x_{ij}-x_{ji})\ \forall\ i,\, j \in \mathscr{L}. \tag{5.24d}$$

This enforces the equality $k_i = k_j$ using half the amount of inequality constraints. Note that constraints (5.24d) exclude solutions where a salesman visits only one node, since $x_{ij} + x_{ji} = 2$ is infeasible by (5.24d).
Figure 5.2 shows an example of a feasible solution for $D = 3$ depots and $C = 6$ cities. Note that within the set $\mathscr{L} = \mathscr{D} \cup \mathscr{C}$ the existence of three cycles has been imposed, whereas in the set $\mathscr{C}$ no cycles exist due to the cycle elimination constraints.

Theorem 5.3.1 (Cycle imposement) *The MILP consisting of (5.5), (5.9), any of the cycle elimination constraints, and the cycle imposement constraints (5.24) will result in a graph with exactly* $\sum_{d \in \mathscr{D}} m_d$ *cycles, where each node* $d \in \mathscr{D}$ *is contained in exactly* m_d *cycles.*

Proof: Let the directed graph $G = (\mathscr{L}, \mathscr{A})$ be the graph associated with a feasible solution of the given MILP. The node set of G coincides with the set of locations of the FMmTSP, and the arc set $\mathscr{A}$ is defined as

$$\forall i,\, j \in \mathscr{L},\ (i,\, j) \in \mathscr{A} \text{ if and only if } x_{ij} = 1$$

Define a cut $(\mathscr{D}, \mathscr{C})$ on G, and denote the subset of forward and backward arcs in the cut set as

$$\delta_+ = \{(i,j) \in \mathscr{A} \mid i \in \mathscr{D},\, j \in \mathscr{C}\}$$
$$\delta_- = \{(i,j) \in \mathscr{A} \mid i \in \mathscr{C},\, j \in \mathscr{D}\}$$

which represents all salesmen leaving the depots and all salesmen returning from the cities, respectively. By assignment constraints on the city nodes (5.9b)–(5.9c), the in-degree and out-degree of each node in $\mathscr{C}$ is one, and the cycle elimination constraints ensure that no cycles exist in $\mathscr{C}$, so no path can start or end in $\mathscr{C}$. Therefore $|\delta_+| = |\delta_-|$, indicating that any

salesman leaving a depot must also return to a depot (see Figure 5.2).
The above arguments actually show that the graph associated with a solution of the non-fixed destination MmTSP contains exactly $\sum_{d\in\mathscr{D}} m_d$ distinct paths starting and ending in $\mathscr{D}$.
Now we need to prove that by the additional cycle imposement constraints for the fixed-destination setting, the $\sum_{d\in\mathscr{D}} m_d$ distinct paths are all cycles, and each node $d \in \mathscr{D}$ is contained in exactly m_d cycles, i.e., each path starting in a depot node $d \in \mathscr{D}$ must also end in the same depot d. We prove this statement by induction.
For any path $P = \{d, c_{i1}\, c_{i2} \dots d^*\}$, where $d, d^* \in \mathscr{D}$ and $c_{i1}, c_{i2} \dots \in \mathscr{C}$, by constraint (5.24b) that the node current k_i is non-decreasing along a path, one has

$$k_d \le k_{c_{i1}} \le k_{c_{i2}} \le \dots \le k_{d^*}$$

In addition, by (5.24a) each depot node is assigned a unique node current, and hence

$$1 \le k_d \le D \quad \forall d \in \mathscr{D}.$$

We use the following inductive steps to prove that any path starting in depot d must also end in the same depot.

(1) By (5.9a) there will be m_D paths leaving depot D. For any path P starting in depot $d = D$, one has $D = k_d \le k_{d^*} \le D$, where the upper limit follows from the fact that a path must return to a depot, for which D is the highest value. Thus $k_{d^*} = D$, indicating $d^* = D = d$. Therefore, a path starting in node D can only end in node D. By (5.9a)-(5.9d) exactly m_D cycles start and end in depot D, i.e., depot D is contained in exactly m_D cycles, and can accept no more incoming arcs because the constraint (5.9d) on its in-degree is already satisfied.

(2) For any path P starting in depot $d = D-1$, similarly one has $D-1 = k_d \le k_{d^*} \le D$. So k_{d^*} can only take the value D or $D-1$, i.e., path P can only end in depot D or $D-1$. By the previous argument P cannot end in D because (5.9d) is already satisfied for $d = D$, so the m_{D-1} paths starting at depot $D-1$ can only end in depot $D-1$. Similarly, by the assignment constraints, depot $D-1$ is contained in exactly m_{D-1} cycles, and can accept no more incoming arcs.

(3) Continuing this argumentation for any path P starting in depot d, P can only end in the same depot d since the constraint (5.9d) is already satisfied for depots $d+1$ to D, and by the assignment constraints depot d is contained in exactly m_d cycles.

(4) Finally, it follows that any path P starting in depot 1 must end in depot 1.

□

By assigning a unique value to the node currents of the depots through (5.24a) and adding constraints (5.24b)–(5.24c) or (5.24d)—such that $k_i = k_j$ if there is a connection between nodes i and j— it is guaranteed that a tour starting at depot d will return to depot d without visiting another depot by Theorem 5.3.1.
Remark: For the optimal solution the node current variables will implicitly satisfy

$$1 \le k_i \le D \; \forall i \in \mathscr{L}, \tag{5.25}$$

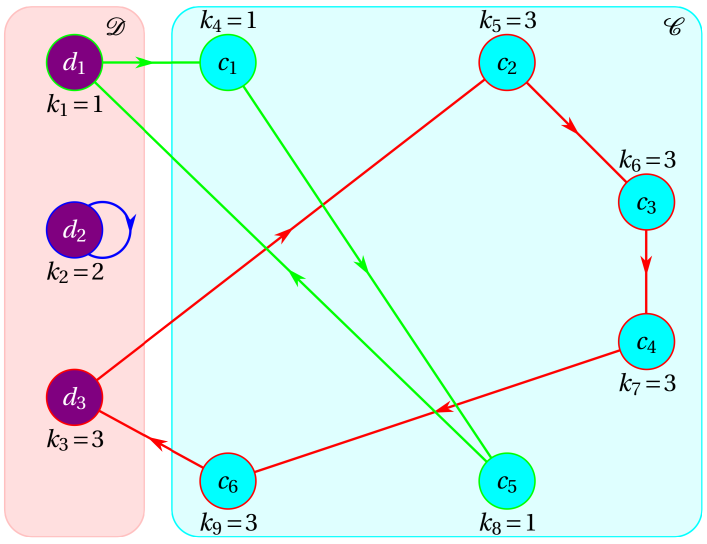


Figure 5.2: A solution without a copy of the set $\mathscr{D}$, using node currents to ensure a fixed-destination solution. Each of the three cycles has a unique 'current': the depot nodes act as current sources of 1, 2 and 3 Ampère respectively, and this current is flowing through the arcs and nodes. Since each node has exactly one incoming arc and one outgoing arc (due to the assignment constraints) this 'node current' uniquely defines to which depot a city is assigned.

and these bounds can be set explicitly in the MILP formulation without affecting the result.

MILP Formulation Using Node Currents

As an alternative to the FMmTSP formulation presented in [10] we propose a novel formulation of the problem based on node currents as cycle imposement constraints. It is based on 2-index decision variables using the cost function given by (5.5). The formulation will be presented using the standard assignment constraint given in (5.9), but the variant with idle salesmen may also be used. For a comparison with the formulation in [10] (which excludes the possibility of idle salesmen) it should be possible to set workload bounds for the salesmen, and therefore the cycle elimination constraints (5.12) are chosen. For the computational comparison in the next section the minimum number of cities to visit per salesman will be $\overline{u} = 1$, and therefore we use constraints (5.24a–5.24c) to impose $\sum\limits_{d\in\mathscr{D}} m_d$ cycles in the set $\mathscr{L}$; if $\overline{u} \geq 2$ it would be more efficient to use (5.24a) and (5.24d). The maximum number of cities per salesman can be set to

$$\underline{u} = C + \overline{u}(1 - \sum_{d\in\mathscr{D}} m_d) \tag{5.26}$$

where $\sum_{d\in\mathscr{D}} m_d$ gives the total number of salesmen, such that $\overline{u}(1-\sum_{d\in\mathscr{D}} m_d)$ becomes the minimum number of cities that need to be visited by other salesmen; a single salesman can visit at most all cities minus the minimum number of cities visited by the others.
The MILP formulation of the FMmTSP using workload bounds and node currents then be-

comes

$$\begin{aligned}
\min \sum_{i\in\mathscr{L}} \sum_{j\in\mathscr{L}} & c_{ij}x_{ij} && (5.27)\\
\text{s.t.} \sum_{i\in\mathscr{L}} x_{ic} &= \sum_{j\in\mathscr{L}} x_{cj} = 1 && \forall\, c\in\mathscr{C}\\
\sum_{i\in\mathscr{L}} x_{id} &= \sum_{j\in\mathscr{L}} x_{dj} = m_d && \forall\, d\in\mathscr{D}\\
u_i - u_j + \overline{u}x_{ij} + (\overline{u}-2)x_{ji} &\le \overline{u}-1 && \forall\, i,j\in\mathscr{C}\\
u_i + (\overline{u}-2)\sum_{d\in\mathscr{D}} x_{di} - \sum_{d\in\mathscr{D}} x_{id} &\le \overline{u}-1 && \forall\, i\in\mathscr{C}\\
u_i + \sum_{d\in\mathscr{D}} x_{di} + (2-\underline{u})\sum_{d\in\mathscr{D}} x_{id} &\ge 2 && \forall i\in\mathscr{C}\\
k_d &= d && \forall\, d\in\mathscr{D}\\
k_i - k_j &\le (D-1)(1-x_{ij}) && \forall\, i,j\in\mathscr{L}\\
k_j - k_i &\le (D-1)(1-x_{ij}) && \forall\, i,j\in\mathscr{L}\\
x_{ij}\in\{0,1\},\ 1\le u_i\le C,\ &1\le k_i\le D && \forall\, i,j\in\mathscr{L}
\end{aligned}$$

Remark: As opposed to the two alternative FMmTSP formulations [10, 113], this novel formulation does not use copies of the depot nodes; therefore, only $L = D + C$ nodes are needed for this formulation instead of $2D + C$. Furthermore, unlike the other two formulations, the costs of traveling between the nodes remain the same as for the nonfixed-destination problem; hence there is no need to build a new cost matrix; the original cost matrix C can be used without any modification.

5.3.2 Properties of Fixed-Destination Formulations

Adding more variables to an optimization problem in general results in larger computation times and a higher memory usage. Compared to continuous variables the number of binary variables used in a programming problem can significantly influence the computation times. Therefore, reducing the number of binary variables to represent a problem can result in a noticeable performance gain. Although in general the addition of a few continuous variables has little influence on the computation times, using many continuous variables can cause problems due to the larger memory use, and it can also result in larger computation times.
Table 5.2 shows the number of nodes, binary variables, continuous variables, equality constraints, and inequality constraints that are needed to represent the FMmTSP per formulation. In the following 'I' denotes the novel MILP formulation (5.27), 'II' denotes the extended formulation based on [114], and 'III' denotes the formulation from [10].

Note that besides less binary variables, the newly proposed formulation also uses less continuous variables compared to the formulation of [10]; there are $D + C$ node currents necessary to solve the fixed-destination problem, compared to $D(2D + C)^2$ commodity flow parameters needed to represent the D different commodities that could move along the $(2D + C)^2$ arcs in the extended network.

Table 5.2: ***Properties of the three formulation types, divided into the number of nodes (N), binary variables (BV), continuous variables (CV), equality constraints (EC), and inequality constraints (IC).***

	I	II	III
N	L=C+D	L'=C+2D	L'=C+2D
BV	$(C+D)^2$	$(C+D)^2$	$(C+D)^2$
CV	2C+2D	2C+4D	$(DL'+1)L'$
EC	2C+3D	2C+4D	2C+ $(4+L')D$
IC	$C^2+2C+2L^2$	$C^2+2C+2L'^2$	$C^2+2C+DL'^2$

5.4 Computational Comparison

In this section we compare three MILP formulations for solving FMmTSPs by a large number of test cases. Formulation I is the proposed formulation (5.27). Formulation II is composed of the MILP formulation originally designed for non-fixed destination MmTSP in [**?**], and the node-current cycle imposement constraint. Formulation III is the multi-commodity flow formulation in [**?**]. First we describe the benchmark that we use, followed by a discussion of the results. The formulations provide optimal solutions for the FMmTSP, and all computation times give the time it took to reach this optimum. When the optimum was not reached within 3 hours wall-clock time the test was marked as failed.

5.4.1 Description of Test Instances

To compare the three formulations of the FMmTSP we have chosen 32 symmetric and asymmetric TSP test cases with size ranging from 14 to 170 nodes from the library TSPLIB [133], where the numbers in the name of the test instance (e.g. `dantzig42`) represent the number of locations L in the problem. The first 16 test cases (`burma14` up to `ry48p`) are considered small problems, while the last 16 test cases (`hk48` up to `ftv170`) are included in the large problems. For each test case we have selected D cities to represent depots. Since e.g. the cities in `dantzig42` are given in the order of the optimal tour (and hence subsequent cities are close to each other), the depot nodes are selected as the i-th cities satisfying

$$i = 1 + (d-1)\left\lfloor \frac{C}{D} \right\rfloor \quad \forall d \in \mathcal{D}, \tag{5.28}$$

where $\lfloor a \rfloor$ represents the operator that returns the largest integer smaller than or equal to a. This approach is used to reduce the chance that the depots are close to each other. The number of depots D varies from 2 to 6 for each test case. We consider two scenarios, namely, one-salesman-at-each-depot and multiple-salesmen-at-each-depot. For the multiple-salesmen-at-each-depot scenario, the following three-step procedure is used to allocate the number of salesmen for each depot.
Step 1: Compute the number of cities $C = L - D$, and generate the total number of salesmen to be assigned to the D depots as

$$S = \min\left(\max\left(D+1, \left\lfloor \frac{L}{3} \right\rfloor\right), C-1\right)$$

Note that we choose $\lfloor \frac{L}{3} \rfloor$ for the total number of salesmen (as long as it lies in the interval $[D+1, C-1]$), since too few salesmen are insufficient to consider the multiple-salesmen-at-each-depot scenario, and too many salesmen can lead to idle salesmen in the solution.

Step 2: Assign $x = \left\lfloor \dfrac{S}{D} \right\rfloor$ salesmen to each depot, and calculate the number of the unassigned salesmen

$$r = S - x \cdot D$$

Step 3: Assign one salesman to the depots with indices

$$i = 1 + (k-1)\left\lfloor \frac{D}{r} \right\rfloor \qquad \forall k \in \{1, 2, \cdots r\}$$

Since the number of remaining salesmen calculated at *Step 2* is always less than D, all salesmen have been assigned to a depot after performing the three-step procedure. Moreover, the last step also ensures a fair allocation of the remaining salesmen.
In this way we create a benchmark of $32 \times 5 \times 2 = 320$ FMmTSP test instances, which are solved using three different formulations, and three MILP solvers.
Since the formulation using the commodity flows according to (5.19) requires that each salesman visits at least one city (due to (5.19a)), we set $\underline{u} = 1$ and $\overline{u} = C$ to obtain the same problem for each formulation. All computations are performed on a desktop computer with an Intel Xeon E5-1620 Quad Core CPU and 64 GB of RAM, running 64-bit versions of SUSE Linux Enterprise Desktop 11, and Matlab R2014b. Three state-of-the-art commercial and free MILP solvers are used, namely, CPLEX 12.5 (called via Tomlab 8.0), Gurobi Optimizer 5.6, and CBC 2.9.4 from COIN-OR (called via the OPTI-Toolbox).

5.4.2 Results of Computational Experiments

A time limit of 3 hours is imposed on each test run. The optimal values for all the test instances are provided in Table 5.3. These values are obtained by summarizing all instances successfully solved by three MILP solvers (CPLEX, Gurobi, and CBC) and three formulations within a 3-hour time limit.
The CPU times for the one-salesman-at-each-depot problem are reported in Tables 5.4–5.6, followed by the CPU times for the multiple-salesmen-at-each-depot problem in Tables 5.7–5.9. To reduce the chance that the outcome is affected by random events, we chose to run each test case a few times and take the average value of the computation times. Tables 5.4–5.9 contain the average CPU time to find the optimal solution over 10 runs for each small test case, and over 5 runs for each large test case, for each number of depots D. All reported times are in seconds. The fastest instances are indicated by the bold-faced numbers, and the symbol “-" is used to denote the failed test instances.

5.4.3 Comparison of Problem Formulations

When a test case is solved to optimality within 3 hours wall-clock time, we register this time. Otherwise, we mark the test case as failed. A comparison is made between the three formulations on both the average CPU times and the number of failed cases.

Table 5.3: ***Summary of optimal values obtained using three solvers and three formulations.***

test case	one-salesman-at-each-depot					multiple-salesmen-at-each-depot				
	D=2	*D*=3	*D*=4	*D*=5	*D*=6	*D*=2	*D*=3	*D*=4	*D*=5	*D*=6
burma14	3098	3033	2993	3480	3728	3253	3079	3039	3696	3944
ulysses16	6986	6326	6097	5809	8862	8324	6873	6101	5816	9774
gr17	2054	1819	1945	1684	1815	2374	2056	2182	1818	1953
br17	36	23	16	6	34	39	23	16	6	34
gr21	2716	2662	2674	2684	3228	3623	3454	3932	3004	3360
ulysses22	6445	6282	6435	6147	6855	12520	6769	7701	6386	6888
fri26	930	939	940	932	903	1442	1385	1143	1093	1061
bayg29	1596	1598	1641	1583	1608	1955	1972	1962	1867	1851
bays29	1988	1993	2018	1972	1966	2471	2469	2476	2369	2351
ftv33	1302	1291	1292	1237	1214	1831	1732	1536	1402	1490
ftv35	1457	1453	1429	1396	1421	2157	1999	1801	1918	1735
ftv38	1521	1510	1518	1455	1512	2297	2158	1962	1941	1966
dantzig42	661	633	645	611	631	1202	1016	977	795	813
swiss42	1272	1262	1274	1277	1257	2054	1982	1640	1583	1604
ftv44	1611	1602	1608	1569	1582	2744	2232	2546	2205	2171
ry48p	14097	14318	14366	14306	14146	23925	22637	19560	18506	19378
hk48	11439	11358	11222	11285	11310	18259	19258	14989	14697	13717
eil51	426.358	423.013	424.452	431.966	423.28	712.039	624.878	554.267	549.326	558.837
berlin52	7464.36	7591.94	7528.97	7501.83	7733.97	11896.5	14113.6	9754.51	9460.11	9444.27
ft53	6926	7029	6880	6842	6896	12211	12012	9376	8857	8943
ftv55	1590	1585	1594	1623	1584	2905	2715	2865	2666	2279
ftv64	1782	1835	1798	1770	1846	3052	3067	2722	2935	2601
st70	671.792	667.264	659.917	654.026	655.046	1423.55	1166.02	1023.21	1029.61	884.627
eil76	542.325	541.004	540.436	555.114	551.761	941.31	898.858	866.489	864.664	795.141
gr96	54795	55076	54047	54653	54999	130169	126797	101320	88968	92425
kroB100	21954.8	21698.8	21695.2	21680.6	21415.8	47224.9	40190.7	38024.8	32726.8	33430.7
kroC100	20504.7	20400.2	20308.2	20321.8	20434	49947.6	39621.9	40213.4	35719.9	32695.2
kroD100	21493	-	-	20792.4	-	60888.7	47247.5	36873.3	36597.8	37458.2
kroE100	21896.9	21932.2	21700.5	21865	21386.3	46588.1	42585.9	35477.3	33493	37063.9
eil101	641.711	637.213	636.541	641.933	640.554	1353.47	1414.26	1098.31	973.447	973.366
kro124p	36316	36244	36252	36041	36427	69844	67146	-	58912	53206
ftv170	2755	2740	-	2744	-	-	-	-	-	-

Table 5.4: ***Mean CPU time (in seconds) obtained from the CPLEX solver, for scenario one-salesman-at-each-depot.***

type	test case	*D*=2	*D*=3	*D*=4	*D*=5	*D*=6	test case	*D*=2	*D*=3	*D*=4	*D*=5	*D*=6
I		0.50	0.19	0.19	0.31	0.40		**4.30**	32.14	**1.34**	**0.45**	1.97
II	burma14	**0.21**	0.25	**0.16**	**0.23**	**0.15**	hk48	8.06	33.00	2.26	0.79	**1.54**
III		0.34	**0.18**	0.34	0.26	0.24		114.81	**25.21**	10.05	4.00	36.04
I		1.16	**0.30**	**0.17**	**0.23**	392.31		**1.86**	2.25	**1.88**	13.28	2.18
II	ulysses16	**0.57**	0.31	0.23	0.23	**0.32**	eil51	2.48	**1.69**	2.49	**9.05**	**1.54**
III		21.30	0.37	0.35	0.37	0.43		70.44	29.06	15.39	152.15	26.55
I		0.36	0.36	0.33	**0.20**	0.25		1.71	**5.58**	**1.22**	**1.65**	28.17
II	gr17	**0.32**	**0.27**	**0.29**	0.29	**0.21**	berlin52	**1.58**	5.98	1.85	2.68	**9.55**
III		1.39	0.35	0.32	0.26	0.34		264.08	27.24	14.69	12.67	110.63
I		**0.39**	**0.35**	0.25	0.30	0.30		**2.31**	67.92	**2.04**	**2.17**	**1.36**
II	br17	0.43	0.39	0.26	**0.24**	**0.28**	ft53	4.54	**20.50**	3.84	2.72	1.67
III		1.09	0.61	**0.25**	0.34	0.31		866.05	1024.84	52.29	46.70	18.25
I		0.23	0.25	0.24	0.25	1.79		**1.19**	**1.08**	**1.51**	5.87	12.48
II	gr21	**0.23**	**0.20**	**0.19**	**0.21**	**0.22**	ftv55	1.93	1.37	2.34	**2.37**	**0.85**
III		0.41	0.28	0.27	0.34	0.38		4.13	9.30	17.74	7.31	28.34
I		**0.44**	**0.28**	**0.51**	**0.30**	3.62		1.68	**7.01**	2.51	**1.30**	14.36
II	ulysses22	0.55	0.37	0.59	0.41	**0.38**	ftv64	**1.56**	13.07	**1.67**	3.09	**3.09**
III		1.86	0.40	0.92	0.64	0.73		21.04	59.80	22.06	9.54	1107.09
I		**0.54**	**0.45**	**0.41**	0.51	**0.33**		**107.87**	**45.59**	**146.17**	**106.91**	**25.82**
II	fri26	0.63	0.77	0.55	**0.48**	0.52	st70	253.25	82.69	171.05	122.70	35.93
III		4.68	0.92	1.17	1.34	1.35		527.36	92.28	698.98	448.71	131.62
I		**0.38**	**0.57**	3.01	0.36	1.15		**10.83**	**11.52**	**5.57**	3034.21	1060.30
II	bayg29	0.45	0.67	**2.72**	**0.33**	**0.86**	eil76	14.81	15.08	8.32	**25.54**	**6.06**
III		2.62	1.18	9.71	0.85	2.98		71.22	126.64	120.74	444.12	98.20
I		0.45	**0.39**	**1.30**	**0.30**	**0.46**		**52.71**	232.55	**23.88**	158.68	405.34
II	bays29	**0.40**	0.53	1.46	0.30	0.75	gr96	126.06	**129.70**	51.32	**97.39**	**181.88**
III		1.87	0.73	9.85	0.75	1.77		366.28	2506.87	342.34	-	-
I		**0.71**	2.76	**0.53**	**0.38**	**0.32**		**715.08**	253.79	**10249.36**	**159.69**	**418.96**
II	ftv33	0.91	**1.04**	0.73	0.40	0.32	kroB100	1100.80	**223.31**	-	265.20	915.75
III		73.28	1.52	1.43	1.77	1.33		-	2192.25	-	-	-
I		**0.49**	0.54	**0.37**	0.34	**0.31**		**1079.32**	**198.89**	**175.56**	759.76	2653.58
II	ftv35	0.57	**0.51**	0.44	**0.29**	0.37	kroC100	2032.30	299.03	199.45	**632.67**	**1007.79**
III		2.18	1.24	1.61	1.68	1.70		6054.43	7025.49	-	-	-
I		0.77	0.60	**0.47**	0.48	**0.62**		-	**-**	**-**	**8739.53**	**-**
II	ftv38	**0.62**	**0.59**	0.70	**0.34**	0.67	kroD100	**2995.81**	-	-	-	-
III		3.86	1.63	1.70	1.83	3.15		6718.39	-	-	-	-
I		**3.25**	**0.75**	**1.80**	**0.87**	**0.61**		**440.11**	**222.39**	254.65	**676.93**	**44.51**
II	dantzig42	3.96	1.40	2.25	1.55	0.66	kroE100	798.82	238.56	**163.79**	946.49	-
III		36.54	2.06	6.93	6.10	5.23		-	-	-	-	-
I		**1.50**	**1.13**	**1.42**	**1.75**	**1.27**		**78.90**	**26.07**	**133.80**	**71.34**	71.21
II	swiss42	1.84	1.65	2.41	1.92	1.87	eil101	338.65	33.88	143.56	117.90	**58.03**
III		33.78	6.40	6.61	16.46	9.48		6377.94	-	-	-	-
I		1.17	**0.67**	1.27	0.58	0.95		**25.41**	**34.76**	**22.02**	**11.13**	**33.28**
II	ftv44	**0.99**	0.96	**0.82**	**0.58**	**0.62**	kro124p	1154.00	2196.56	-	-	-
III		3.26	1.99	6.16	4.47	3.79		1151.96	2195.11	-	-	-
I		**1.52**	**7.30**	**8.02**	**5.60**	6.46		-	-	-	-	-
II	ry48p	2.32	13.40	11.75	9.48	**1.23**	ftv170	-	-	-	-	-
III		37.00	32.92	36.70	44.33	11.96		-	-	-	-	-

Table 5.5: ***Mean CPU time (in seconds) obtained from the Gurobi Optimizer, for the scenario one-salesman-at-each-depot.***

type	test case	*D*=2	*D*=3	*D*=4	*D*=5	*D*=6	test case	*D*=2	*D*=3	*D*=4	*D*=5	*D*=6
I		**0.07**	0.07	**0.04**	0.22	0.68		7.17	**86.31**	2.55	**0.23**	**3.10**
II	burma14	0.13	**0.06**	0.09	**0.10**	**0.04**	hk48	**3.66**	107.00	**2.22**	0.41	5.32
III		0.14	0.08	0.08	0.14	0.09		4.27	105.61	5.51	0.59	4.86
I		2.05	**0.15**	0.05	**0.07**	461.46		2.69	3.97	**0.99**	32.10	4.03
II	ulysses16	**1.02**	0.26	**0.04**	0.08	0.27	eil51	**1.86**	2.89	1.53	38.29	**2.56**
III		1.59	0.27	0.11	0.24	**0.26**		3.86	**2.78**	4.56	**21.79**	3.32
I		**0.19**	**0.14**	**0.18**	**0.03**	0.11		2.06	10.88	**1.94**	**1.84**	49.24
II	gr17	0.35	0.17	0.28	0.05	**0.10**	berlin52	**1.10**	**8.64**	2.86	3.90	**21.73**
III		1.08	0.27	0.40	0.08	0.18		3.50	29.66	6.74	5.95	29.06
I		**1.14**	0.78	**0.21**	**0.11**	0.33		28.29	56.11	**22.43**	19.16	36.85
II	br17	2.71	**0.43**	0.24	0.22	**0.32**	ft53	**7.13**	46.61	43.77	**11.04**	**6.97**
III		2.52	0.70	0.32	0.19	0.43		28.82	**39.73**	35.95	111.07	11.81
I		**0.13**	**0.06**	**0.16**	**0.07**	3.04		1.84	**1.69**	3.10	10.94	18.37
II	gr21	0.21	0.09	0.23	0.10	**0.07**	ftv55	**1.60**	2.28	**2.12**	**2.32**	**1.59**
III		0.20	0.20	0.29	0.45	0.11		4.15	3.98	4.17	4.12	4.33
I		**0.46**	**0.18**	**0.42**	**0.23**	3.19		1.84	**4.22**	2.56	**1.46**	21.49
II	ulysses22	0.67	0.22	0.47	0.55	**0.46**	ftv64	**1.62**	4.96	**1.68**	3.17	**5.62**
III		0.70	0.31	1.26	0.47	0.48		1.95	12.61	4.02	2.82	8.09
I		**0.40**	**0.72**	**0.97**	**0.51**	**0.38**		**636.50**	**241.88**	**1133.48**	**379.68**	**29.23**
II	fri26	0.76	0.98	1.01	0.67	0.48	st70	-	261.64	1506.71	475.47	84.12
III		0.87	1.24	1.30	1.36	0.80		1595.30	307.17	4683.77	1098.81	189.27
I		**0.73**	**0.55**	4.45	**0.47**	2.27		9.48	11.43	4.68	10547.85	2205.36
II	bayg29	0.88	0.99	**3.41**	1.23	**1.03**	eil76	**6.85**	**10.49**	**2.50**	**14.06**	**2.51**
III		0.95	1.13	5.70	1.18	2.07		13.21	27.92	4.30	48.19	8.06
I		**0.38**	**0.57**	**2.81**	0.42	1.32		349.41	957.70	**97.39**	**362.25**	**329.48**
II	bays29	0.54	1.09	2.92	0.40	**1.00**	gr96	**209.83**	**376.86**	128.38	443.95	666.62
III		0.69	0.96	3.84	**0.30**	1.26		259.31	1196.67	124.34	439.49	-
I		**0.55**	**0.85**	**0.91**	**0.37**	**0.17**		6013.72	**420.09**	**-**	1580.18	**2098.60**
II	ftv33	2.12	1.17	1.18	0.87	0.27	kroB100	**5341.60**	1115.09	-	**1071.68**	4772.04
III		1.85	1.24	3.19	0.65	0.46		-	1648.63	-	-	-
I		**0.64**	**0.63**	0.31	**0.23**	**0.18**		-	-	319.52	**2720.50**	**-**
II	ftv35	1.17	0.97	**0.26**	0.37	0.24	kroC100	-	**2370.29**	**308.45**	5469.71	-
III		1.27	0.71	0.54	0.63	0.49		**5304.34**	-	2094.30	-	-
I		**0.94**	0.79	0.62	0.27	**0.71**		-	**-**	**-**	**-**	**-**
II	ftv38	1.31	1.22	**0.61**	**0.26**	1.55	kroD100	**8377.61**	-	-	-	-
III		1.00	**0.64**	1.09	0.51	0.98		10581.25	-	-	-	-
I		**4.67**	**0.77**	2.39	1.84	**0.56**		-	1189.62	800.29	**6460.41**	**303.35**
II	dantzig42	4.98	2.17	**1.27**	**1.55**	0.63	kroE100	**10475.87**	**1143.61**	**675.25**	8845.54	525.64
III		4.77	2.44	1.52	1.59	0.98		-	3482.50	1186.41	-	-
I		2.37	**1.45**	**1.70**	**2.80**	**1.49**		229.58	317.55	242.76	**140.20**	189.96
II	swiss42	**2.33**	1.77	3.20	3.63	2.89	eil101	**125.33**	26.50	**126.40**	157.08	**47.72**
III		5.27	5.63	4.97	5.00	5.09		421.55	**17.71**	356.45	-	-
I		**0.61**	**0.81**	1.51	**0.32**	0.83		**30.17**	**254.38**	**24.80**	**16.82**	**269.64**
II	ftv44	1.23	0.97	**0.81**	0.85	**0.62**	kro124p	74.38	336.59	77.14	-	-
III		0.76	1.25	1.48	1.02	1.02		74.46	337.20	77.08	-	-
I		3.35	**8.76**	**14.18**	**6.82**	7.52		**32.24**	**26.90**	-	24.86	-
II	ry48p	**2.84**	14.54	77.48	14.98	**2.16**	ftv170	-	-	-	-	-
III		3.89	18.89	17.72	29.67	3.86		-	-	-	-	-

Table 5.6: ***Mean CPU time (in seconds) obtained from the CBC solver, for scenario one-salesman-at-each-depot. As the largest test case that CBC can solve for this scenario is `eil76`, we have truncated the table to make it more concise.***

type	test case	*D*=2	*D*=3	*D*=4	*D*=5	*D*=6	test case	*D*=2	*D*=3	*D*=4	*D*=5	*D*=6
I		0.79	**0.41**	**0.39**	1.55	49.48		**133.87**	19.19	**30.06**	**23.48**	**4.92**
II	burma14	**0.78**	0.44	0.57	0.70	0.75	dantzig42	356.50	**15.07**	89.85	128.51	5.17
III		2.55	0.72	0.99	**0.55**	**0.60**		-	75.43	339.87	126.71	70.08
I		33.17	1.97	**0.32**	**0.51**	6772.37		**27.77**	79.93	**21.82**	**41.73**	**16.61**
II	ulysses16	**11.18**	**0.86**	0.42	0.72	**1.35**	swiss42	45.40	**25.81**	60.22	48.75	27.37
III		28.21	1.67	0.35	1.78	2.42		598.24	68.51	72.12	132.23	102.62
I		**1.19**	**0.65**	**0.52**	**0.32**	**0.45**		23.38	**5.13**	237.46	**8.41**	20.77
II	gr17	1.68	1.04	1.05	0.50	0.92	ftv44	**9.08**	9.95	**24.70**	13.49	**11.77**
III		3.56	1.36	0.82	1.31	1.04		65.52	38.73	30.67	106.26	104.84
I		**599.43**	46.45	**0.54**	**0.58**	33.90		**20.75**	1775.21	**729.87**	**222.86**	1302.83
II	br17	618.82	**23.23**	0.83	0.89	**1.00**	ry48p	216.58	**321.03**	4956.45	348.75	**17.26**
III		1995.82	31.05	2.50	1.91	1.10		577.59	875.94	1188.08	1306.19	220.15
I		**0.64**	**0.54**	0.99	1.05	25.46		**333.89**	-	75.05	**6.73**	43.39
II	gr21	0.77	0.65	1.89	**1.00**	0.84	hk48	406.06	**7452.41**	152.95	11.88	**11.87**
III		1.32	4.39	**0.96**	3.52	**0.79**		2509.90	7621.84	**51.88**	33.55	174.17
I		**5.87**	**0.93**	5.31	**1.64**	70.65		51.23	34.59	44.43	935.99	247.47
II	ulysses22	6.23	1.34	**2.27**	2.25	**5.73**	eil51	**47.27**	**21.94**	**36.56**	853.15	**14.74**
III		31.50	4.18	16.83	7.09	6.54		2116.50	70.25	76.43	**490.93**	101.66
I		**1.99**	**6.20**	**7.97**	9.51	**3.93**		13.95	**129.52**	**27.39**	106.86	3666.44
II	fri26	7.10	8.37	14.70	**7.04**	7.99	berlin52	**11.10**	362.80	53.40	**86.09**	548.08
III		140.05	11.24	11.38	12.23	16.64		9137.46	1977.97	135.74	355.46	**423.07**
I		**5.31**	**8.11**	67.46	3.75	**15.35**		159.03	-	9468.16	**121.40**	85.57
II	bayg29	8.51	15.64	**67.00**	**2.06**	23.53	ft53	**152.91**	**527.96**	-	130.30	**24.68**
III		29.43	13.70	104.49	8.47	24.26		-	1926.62	**1004.57**	439.18	85.58
I		**4.46**	11.88	42.83	3.24	78.02		**16.37**	**14.44**	36.75	259.52	614.58
II	bays29	7.22	16.14	**22.97**	**2.20**	11.73	ftv55	116.94	27.70	**22.57**	**31.31**	**15.38**
III		24.90	**10.34**	37.78	12.36	**9.43**		2539.82	86.93	86.52	337.31	314.63
I		**12.61**	**10.90**	**7.60**	**2.83**	**1.52**		**36.75**	**192.80**	627.11	**27.18**	879.96
II	ftv33	14.44	15.71	17.40	9.78	3.04	ftv64	93.52	225.63	**104.97**	34.65	**98.39**
III		319.31	15.58	22.57	59.12	64.11		2311.47	211.36	318.15	650.50	770.92
I		**6.44**	**3.34**	**4.68**	**2.33**	**3.18**		**-**	**-**	**-**	**-**	**-**
II	ftv35	21.41	5.17	5.55	3.45	3.83	st70	-	-	-	-	-
III		177.13	13.57	17.46	17.31	30.32		-	-	-	-	-
I		**7.81**	**4.58**	**6.90**	**3.13**	9.90		**1246.47**	**402.44**	**69.14**	**-**	**-**
II	ftv38	15.28	8.56	8.37	4.55	**6.24**	eil76	-	-	-	-	-
III		39.18	39.36	8.82	40.58	12.55		-	-	-	-	-

Table 5.7: ***Mean CPU time (in seconds) obtained from the CPLEX solver, for the scenario multiple-salesmen-at-each-depot.***

type	test case	*D*=2	*D*=3	*D*=4	*D*=5	*D*=6	test case	*D*=2	*D*=3	*D*=4	*D*=5	*D*=6
I		0.26	**0.16**	**0.21**	0.48	0.54		**1.92**	-	**0.66**	1.28	1.97
II	burma14	**0.22**	0.29	0.23	**0.22**	**0.16**	hk48	2.06	**3.36**	1.10	**1.16**	**1.85**
III		0.29	0.27	0.26	0.28	0.21		4.26	9.11	5.00	11.19	19.29
I		6.91	**0.23**	**0.14**	0.33	1037.06		**1.01**	**0.82**	0.49	7.12	83.13
II	ulysses16	**0.20**	0.27	0.17	**0.24**	**0.25**	eil51	2.22	2.01	**0.43**	**3.84**	**1.45**
III		0.26	0.33	0.23	0.32	0.39		4.44	11.14	2.04	27.18	14.75
I		**0.36**	0.33	**0.33**	**0.18**	**0.23**		1.15	3.85	**2.05**	**1.63**	588.83
II	gr17	0.42	**0.29**	0.35	0.23	0.23	berlin52	**0.83**	**1.69**	2.57	1.65	**2.68**
III		0.44	0.56	0.40	0.37	0.31		6.12	16.82	13.62	11.57	37.65
I		0.35	**0.28**	0.27	**0.20**	0.28		-	2305.90	**2.07**	96.97	**2.40**
II	br17	**0.32**	0.29	**0.21**	0.24	**0.25**	ft53	**1.93**	336.30	4.82	**5.75**	2.66
III		0.45	0.33	0.27	0.37	0.39		44.95	**31.36**	49.45	16.60	10.19
I		0.40	0.45	91.74	0.29	1.30		4.16	-	2627.80	-	1158.37
II	gr21	**0.22**	**0.27**	**0.21**	**0.20**	**0.17**	ftv55	**0.98**	**2.57**	**5.96**	**19.94**	**1.93**
III		0.35	0.27	0.25	0.28	0.28		10.74	23.80	7.58	21.99	9.09
I		0.31	**0.25**	**0.42**	0.49	2.12		1.20	**1.10**	91.39	2.79	-
II	ulysses22	**0.29**	0.31	0.44	**0.36**	**0.31**	ftv64	**0.56**	1.69	**1.14**	**0.78**	**4.01**
III		0.43	0.31	0.82	0.64	0.63		1.70	5.86	4.56	5.50	38.52
I		**0.38**	0.73	**0.42**	**0.45**	0.44		-	-	**27.42**	-	120.53
II	fri26	0.40	**0.34**	0.44	0.59	**0.42**	st70	**22.42**	**10.68**	27.66	**8.85**	**49.54**
III		0.59	0.69	1.06	0.89	1.04		50.52	101.39	159.51	144.87	343.32
I		**0.28**	**0.27**	**0.38**	0.66	**0.49**		**3.66**	51.39	**2.50**	-	-
II	bayg29	0.40	0.37	0.64	**0.31**	0.65	eil76	4.72	**1.38**	2.74	**2.37**	**2.96**
III		0.42	0.41	1.28	0.82	1.45		32.96	7.01	67.98	45.81	43.07
I		0.25	0.33	**0.55**	1.04	0.55		136.12	-	-	-	-
II	bays29	**0.21**	**0.32**	0.61	**0.46**	**0.41**	gr96	**70.37**	**32.37**	5271.15	**216.89**	**400.03**
III		0.33	0.62	1.36	0.74	1.54		430.93	739.18	**3902.65**	-	-
I		5.32	**0.55**	**0.39**	**0.19**	0.67		**11.80**	42.46	-	**154.68**	-
II	ftv33	**0.38**	0.74	0.44	0.27	**0.42**	kroB100	19.18	**29.77**	**324.27**	232.42	**1647.09**
III		0.88	1.19	0.98	0.53	2.17		85.15	477.31	-	-	-
I		**0.34**	0.69	0.43	1.38	0.28		35.14	-	5360.70	-	-
II	ftv35	0.43	**0.48**	**0.41**	**0.34**	**0.26**	kroC100	**32.06**	**575.64**	**178.00**	**70.52**	**253.15**
III		0.51	0.88	1.03	1.11	0.79		340.68	1199.03	-	-	-
I		88.85	**0.64**	5.76	0.55	3.32		-	-	33.37	-	-
II	ftv38	**0.28**	0.73	**0.47**	**0.54**	**0.55**	kroD100	**62.94**	**226.68**	**18.18**	**6993.22**	**157.58**
III		0.59	1.95	2.90	1.27	3.07		312.03	923.29	-	-	-
I		0.47	0.43	**1.23**	**0.86**	0.72		271.89	-	6675.83	-	-
II	dantzig42	**0.41**	**0.34**	2.64	1.26	**0.57**	kroE100	**143.93**	**138.07**	**242.76**	**8398.49**	**58.45**
III		1.13	1.21	11.39	2.73	11.66		479.70	858.16	-	-	-
I		1.41	554.87	1.73	10.63	13.95		-	-	23.64	-	-
II	swiss42	**0.49**	**2.11**	**1.61**	**1.05**	**1.49**	eil101	**45.14**	**19.24**	**16.01**	**52.81**	**18.28**
III		1.20	11.93	10.22	2.41	21.49		350.63	357.35	-	-	-
I		1233.89	1.32	1968.36	**0.54**	1.11		**4.03**	**5.51**	-	**2768.97**	**39.11**
II	ftv44	**0.35**	**0.51**	**1.06**	0.64	**0.40**	kro124p	185.08	1410.72	-	-	-
III		1.29	1.21	4.36	2.84	1.44		184.42	1412.88	-	-	-
I		**1.25**	3691.25	16.56	52.54	-		-	-	-	-	-
II	ry48p	2.06	**3.36**	**4.29**	**1.34**	**1.21**	ftv170	-	-	-	-	-
III		7.43	12.42	28.10	8.19	11.64		-	-	-	-	-

Table 5.8: ***Mean CPU time (in seconds) obtained from Gurobi Optimizer, for the scenario multiple-salesmen-at-each-depot.***

type	test case	*D*=2	*D*=3	*D*=4	*D*=5	*D*=6	test case	*D*=2	*D*=3	*D*=4	*D*=5	*D*=6
I		**0.04**	**0.04**	**0.04**	0.52	0.61		2.21	-	**0.48**	**0.99**	**2.71**
II	burma14	0.04	0.05	0.05	**0.05**	**0.04**	hk48	**1.23**	**1.52**	0.68	2.05	2.84
III		0.05	0.07	0.09	0.09	0.08		3.14	5.46	1.74	2.10	3.04
I		6.14	0.14	**0.02**	**0.11**	1054.08		0.83	2.58	0.50	17.94	164.84
II	ulysses16	0.08	**0.09**	0.04	0.17	**0.24**	eil51	**0.79**	**0.95**	**0.30**	**2.86**	**1.70**
III		**0.06**	0.23	0.07	0.27	0.30		2.27	2.03	0.63	6.36	2.04
I		**0.16**	**0.13**	**0.21**	**0.03**	0.12		1.70	3.96	3.41	**2.09**	1013.67
II	gr17	0.25	0.25	0.30	0.05	**0.07**	berlin52	**1.48**	**1.61**	**3.19**	2.79	**4.77**
III		0.79	0.39	0.37	0.12	0.16		13.82	2.04	8.20	3.48	14.19
I		**0.30**	**0.07**	**0.11**	**0.03**	0.27		6739.40	-	**20.23**	133.20	**16.10**
II	br17	0.52	0.07	0.13	0.04	**0.11**	ft53	6.66	65.09	55.45	**9.69**	21.73
III		1.20	0.11	0.17	0.10	0.26		**3.40**	**33.89**	106.06	64.09	16.45
I		**0.12**	0.36	105.00	0.18	1.38		6.28	-	3558.18	-	5136.91
II	gr21	0.22	**0.31**	0.17	**0.11**	**0.07**	ftv55	**1.04**	**3.00**	6.88	32.69	5.11
III		0.35	0.36	**0.14**	0.22	0.17		1.88	3.56	**2.48**	**6.14**	**2.65**
I		0.26	0.13	**0.33**	**0.34**	3.75		0.31	1.59	67.29	3.23	-
II	ulysses22	**0.14**	**0.08**	0.65	0.35	**0.29**	ftv64	**0.26**	**0.68**	**1.10**	2.24	**5.07**
III		0.20	0.12	0.83	0.55	0.51		0.79	1.35	1.89	**1.27**	6.26
I		**0.31**	1.55	0.72	**0.48**	**0.35**		-	-	**97.57**	-	**878.29**
II	fri26	0.43	**0.63**	**0.56**	0.78	0.91	st70	56.77	**16.84**	223.73	**29.98**	2060.49
III		0.41	1.00	0.68	0.65	0.51		**54.24**	45.69	161.56	135.73	1274.02
I		**0.07**	**0.21**	**0.45**	0.57	**0.77**		**1.82**	50.45	7.30	-	-
II	bayg29	0.11	0.22	0.90	**0.49**	1.23	eil76	2.82	1.30	**1.68**	**2.06**	**1.40**
III		0.21	0.39	1.45	0.81	0.91		5.60	**0.79**	6.57	5.28	2.98
I		0.14	**0.24**	**0.60**	1.67	1.35		**266.89**	-	-	-	-
II	bays29	**0.13**	0.35	1.01	0.71	1.01	gr96	560.08	**32.93**	-	1685.22	**344.50**
III		0.29	0.43	1.15	**0.68**	**0.92**		277.99	50.03	**658.95**	**1055.28**	-
I		3.85	**0.57**	0.57	**0.11**	0.99		40.80	**68.72**	-	**546.46**	-
II	ftv33	**0.65**	0.87	**0.46**	0.19	1.08	kroB100	**17.19**	147.15	**805.84**	1425.71	-
III		1.51	1.37	0.75	0.29	**0.69**		39.94	256.84	-	-	-
I		0.52	0.74	0.68	1.86	0.30		141.21	-	-	-	-
II	ftv35	0.65	**0.49**	**0.47**	**0.25**	**0.23**	kroC100	**46.60**	**675.18**	**697.69**	**245.39**	**2182.16**
III		**0.39**	0.58	0.83	0.50	0.33		103.36	933.91	848.71	-	-
I		89.25	1.20	11.33	0.57	3.15		-	-	57.62	-	-
II	ftv38	**0.59**	**0.66**	1.43	**0.26**	1.01	kroD100	**125.71**	1042.29	**33.18**	-	**262.00**
III		0.66	0.81	**1.14**	0.46	**0.66**		388.16	**903.60**	67.51	-	-
I		**0.33**	**0.25**	**1.03**	1.62	**1.38**		810.50	-	-	-	-
II	dantzig42	0.39	0.68	1.68	**0.90**	1.94	kroE100	**322.32**	**373.75**	732.20	-	**84.27**
III		0.64	0.89	1.06	1.44	1.40		526.81	421.44	**387.38**	-	-
I		1.00	750.93	**2.02**	17.79	15.17		-	-	19.28	-	-
II	swiss42	**0.40**	**3.11**	2.93	**0.78**	**3.01**	eil101	**31.80**	**13.56**	**3.77**	**48.96**	**11.67**
III		1.15	9.78	4.58	2.55	4.63		41.80	66.04	5.72	-	-
I		2981.10	3.50	2019.71	1.48	3.72		10.60	11.34	-	-	**192.54**
II	ftv44	**0.34**	0.84	1.86	1.21	**0.47**	kro124p	**9.69**	10.08	81.48	-	-
III		0.91	**0.61**	**1.05**	**0.93**	0.55		9.73	**10.02**	**81.07**	-	-
I		**1.85**	3081.24	22.57	112.09	-		-	-	-	-	-
II	ry48p	3.87	**2.89**	**3.07**	**1.71**	**1.08**	ftv170	-	-	-	-	-
III		2.57	3.14	4.28	2.26	1.80		-	-	-	-	-

Table 5.9: ***Mean CPU time (in seconds) obtained from the CBC solver, for the scenario multiple-salesmen-at-each-depot. As the largest test case that CBC can solve for this scenario is `swiss42`, we have truncated the table to make it more concise.***

type	test case	D=2	D=3	D=4	D=5	D=6	test case	D=2	D=3	D=4	D=5	D=6
I		0.40	**0.42**	**0.40**	2.45	15.72		**0.60**	**1.03**	**5.67**	76.38	26.24
II	burma14	**0.37**	0.45	0.53	**0.52**	**0.43**	bayg29	0.70	1.29	7.53	**2.39**	**10.80**
III		0.80	0.48	0.95	0.54	0.71		5.24	8.82	8.73	19.08	22.07
I		96.08	**0.64**	**0.32**	**0.46**	-		**0.64**	1.89	**3.03**	66.69	12.45
II	ulysses16	**0.71**	0.80	0.44	0.60	1.62	bays29	1.58	**1.42**	6.97	**1.70**	**3.29**
III		0.92	1.38	0.48	0.74	**0.93**		4.26	2.09	10.62	11.07	17.07
I		**1.63**	**0.82**	6.61	**0.33**	**0.50**		121.84	**3.58**	8.75	1.61	39.19
II	gr17	2.60	0.93	2.42	0.54	0.86	ftv33	**2.57**	5.49	**3.86**	2.17	**7.21**
III		13.86	2.66	**1.29**	1.28	1.30		17.29	16.61	9.76	**1.39**	29.50
I		**4.84**	0.79	**0.53**	**0.33**	6.96		**1.97**	67.43	8.85	86.78	1.45
II	br17	5.41	**0.68**	1.31	0.50	**0.75**	ftv35	3.51	**4.45**	**6.10**	**4.80**	2.28
III		39.25	1.35	1.62	0.50	1.83		3.03	20.41	26.28	52.22	**1.21**
I		**0.87**	2.79	2147.21	1.95	50.15		2896.58	**15.63**	277.51	3.42	439.83
II	gr21	1.07	1.35	1.30	**0.92**	0.76	ftv38	**5.05**	18.83	**12.75**	4.86	**6.74**
III		3.46	**1.18**	**1.19**	2.35	**0.49**		9.85	25.90	62.36	**2.68**	25.37
I		1.26	**1.18**	4.80	**1.47**	96.28		**3.84**	6.43	**20.49**	44.62	**15.31**
II	ulysses22	**1.00**	1.71	**3.96**	1.51	**6.83**	dantzig42	7.19	**5.12**	24.19	**13.97**	18.96
III		6.85	3.00	6.94	2.65	12.51		17.89	5.83	54.73	120.24	82.66
I		3.18	17.85	11.20	**5.32**	5.82		35.59	-	73.03	765.07	305.13
II	fri26	**1.88**	**5.37**	11.99	6.44	**2.94**	swiss42	**6.65**	**78.95**	**37.61**	**8.37**	239.81
III		37.25	18.56	**8.25**	8.83	11.83		30.72	96.86	85.50	10.98	**98.13**

Table 5.10: ***The average relative difference for the mean CPU times compared with formulation I. The average is taken over all test instances that are successfully computed by the corresponding formulation and solver. A positive result means a longer computation time, indicating worse performance.***

	Solver	Small & single	Large & single	Small & multi	Large & multi
II	CPLEX	7%	192%	−27%	455%
	Gurobi	51%	7%	−13%	−9%
	CBC	46%	33%	−18%	-
III	CPLEX	535%	1880%	104%	720%
	Gurobi	113%	78%	37%	24%
	CBC	621%	1676%	130%	-

Comparison of Average CPU Times

To compare the three problem formulations, we have split the benchmark into four sets:

- Small problems with a single salesman per depot
- Large problems with a single salesman per depot
- Small problems with multiple salesmen per depot
- Large problems with multiple salesmen per depot

Table 5.10 shows the relative increase in CPU time needed to compute the solution compared to formulation I. For the FMmTSP with a single salesman per depot, formulation I was the fastest on average; for the variant with multiple salesmen per depot formulation II outperformed the other two. Although formulation II uses a few more binary values than formulation I, *it cannot be concluded from our results that the use of more binary variables results in larger computation times.*

Note that the difference between formulation I and II is small (I is less than 1.5 times faster than II for all averages), but formulation III is significantly slower on average when using CPLEX or CBC, even for the small instances (where memory use is not yet expected to be a problem); for Gurobi the differences are smaller. Nevertheless, we conclude that *node current formulations are expected to result in faster computations than multi-commodity-based formulations for fixed-destination problems.*

Comparison of Failed Test Cases

Next, we compare how often a test case did not reach an optimal solution in time. We distinguish between the results for a single salesman per depot and for multiple salesmen per depot. For each formulation we provide the number of failed cases (per solver) in Table 5.11. From Table 5.11, it is clear that formulation II demonstrates stronger ability to solve large test cases. Formulation III also performs rather well in solving large test cases when there are multiple salesmen at each depot, but it has problems for cases with a single salesman per depot. CPLEX and Gurobi seem to perform equally well, but also here it becomes clear that CBC cannot match the other two solvers.

Table 5.11: ***The number of failed test instances ('failed') and the size (i.e., the number of nodes) of the largest instance successfully solved ('largest') for each formulation (I, II, and III) solved per solver type. 'Single' means one-salesman-at-each-depot, and 'multiple' means multiple-salesmen-at-each-depot. The number before '/' is the number of failed test instances out of the 160 that were performed.***

	Solver	Single		Multiple	
		Failed	Largest	Failed	Largest
I	CPLEX	9/160	124	37/160	124
	Gurobi	12/160	170	40/160	124
	CBC	49/160	76	92/160	42
II	CPLEX	14/160	124	8/160	124
	Gurobi	15/160	124	11/160	124
	CBC	51/160	64	90/160	42
III	CPLEX	30/160	124	25/160	124
	Gurobi	24/160	124	19/160	124
	CBC	52/160	64	90/160	42

5.5 Conclusions

In this chapter we have provided a brief overview of cycle elimination and imposement constraints, and 2-index formulations for the fixed-destination multi-depot travelling salesman problem. A novel cycle imposement constraint formulation has been proposed based on node currents, which can be seen as the dual of the node potentials of Miller, Tucker, and Zemlin [103]. The main advantage of the novel formulation over the existing formulations is the reduced number of binary and continuous variables needed to formulate the problem. Computational experiments on a large benchmark show that the proposed formulation performs well with respect to average CPU times and ability to solve large instances. Furthermore, the novel formulation can be used to find solutions where several salesmen can be idle.

Chapter 6

Optimal Nonlinear Solutions for Reverse Stackelberg Games with Incomplete Information

6.1 Game Formulation

The reverse Stackelberg game provides a suitable decision-making framework for hierarchical decision making problems like network pricing and maintenance contract design. We propose a novel numerical solution approach for systematic computation of optimal nonlinear leader functions, also known as incentives, for reverse Stackelberg games with incomplete information and general, nonconcave utility functions. In particular, we apply basis function approximation to the class of nonlinear leader functions, and treat the incentive design problem as a standard semi-infinite programming problem. A worked example is provided to illustrate the proposed solution approach and to demonstrate its efficiency.

We consider a two-person reverse Stackelberg game with player set $\{\mathrm{L}, \mathrm{F}\}$, where L denotes the leader and F denotes the follower. The leader's decision is $d_\mathrm{L} \in D_\mathrm{L} \subset \mathbb{R}^{n_\mathrm{L}}$ and the follower's decision is $d_\mathrm{F} \in D_\mathrm{F} \subset \mathbb{R}^{n_\mathrm{F}}$, where the decision spaces D_L and D_F are both continuous and compact. The follower's type, which contains all his private information like preference, is denoted by $t \in T$, with the type space T a discrete and compact set. The follower's type is only known to himself, but the type distribution $P : T \to [0, 1]$ is known to both players. The leader's utility function is $U_\mathrm{L} : D_\mathrm{L} \times D_\mathrm{F} \to \mathbb{R}$, and the follower's utility function is $U_\mathrm{F} : D_\mathrm{L} \times D_\mathrm{F} \times T \to \mathbb{R}$. Let $\underline{U}_{\mathrm{F},t}$ be the reservation utility of the follower of type t, which specifies the minimum utility the follower requires to participate in the game.

In a reverse Stackelberg game, the leader moves first by announcing a leader function $\gamma_\mathrm{L} : D_\mathrm{F} \to D_\mathrm{L}$. The set of admissible leader functions is denoted by Γ_L. The follower then decides his best response d_F^BR to the announced leader function. If the best response gives a utility strictly lower than his reservation utility, the follower will quit and the game terminates. Otherwise, the follower executes d_F^BR and the game ends by the leader executing the promised decision $\gamma_\mathrm{L}(d_\mathrm{F}^\mathrm{BR})$.

In a reverse Stackelberg game with incomplete information, the leader knows only the probability distribution of the follower's type. The leader's objective is to maximizes her expected utility over all possible follower's types, which is achieved by announcing a leader function that maximizes her expected utility, considering all possible responses from the follower. As proposed in [62, 149], we decompose the problem of designing the optimal leader function into two sequential optimization problems: the leader's global optimization problem, which yields the global optimum (if it exists), and the incentive design problem, which induces the follower to adopt the global optimum, under the assumption of full rationality[1].

Let $t \in T$ denote the follower's type. Define $d_{\mathrm{L},t}$ as the leader's decision when the follower reports type t, and define $d_{\mathrm{F},t}$ as the decision of the follower of type t. Let $\underline{U}_{\mathrm{F},t}$ denote the reservation utility of a follower of type t. The optimization problem to find the global optimum, which is called the desired point (also called team solution in literature [149]), that maximize the leader's expected utility can be formulated as:

$$\text{find} \quad \{(d^*_{\mathrm{L},t}, d^*_{\mathrm{F},t})\}_{t\in T} \in \underset{\{(d_{\mathrm{L},t}, d_{\mathrm{F},t})\in D_{\mathrm{L}}\times D_{\mathrm{F}}\}_{t\in T}}{\arg\max} \sum_{t\in T} P(t) U_{\mathrm{L}}(d_{\mathrm{L},t}, d_{\mathrm{F},t}) \tag{6.1}$$

$$\text{subject to:} \quad U_{\mathrm{F}}(d^*_{\mathrm{L},t}, d^*_{\mathrm{F},t}, t) \geq \underline{U}_{\mathrm{F},t} \quad \forall t \in T \tag{6.2}$$

$$U_{\mathrm{F}}(d^*_{\mathrm{L},t}, d^*_{\mathrm{F},t}, t) \geq U_{\mathrm{F}}(d^*_{\mathrm{L},\hat{t}}, d^*_{\mathrm{F},\hat{t}}, t) \quad \forall t, \hat{t} \in T. \tag{6.3}$$

Constraint (6.2) is the *participation constraint*, which garantees the participation of the follower, and constraint (6.3) is the *incentive compatibility* constraint, which ensures that the follower has no incentive to pretend to be of any type other than his true type.

Assume that the leader's global optimum $\{(d^*_{\mathrm{L},t}, d^*_{\mathrm{F},t})\}_{t\in T}$, which is also her desired point, exists, then the incentive design problem is to find a leader function $\gamma_{\mathrm{L}} \in \Gamma_{\mathrm{L}}$ that induces the follower to adopt the team solution, i.e.

$$\text{find} \quad \gamma_{\mathrm{L}} \in \Gamma_{\mathrm{L}} \tag{6.4}$$

$$\text{subject to:} \quad d^*_{\mathrm{F},t} \in \underset{d_{\mathrm{F}}\in D_{\mathrm{F}}}{\arg\max}\, U_{\mathrm{F}}(\gamma_{\mathrm{L}}(d_{\mathrm{F}}), d_{\mathrm{F}}, t) \quad \forall t \in T \tag{6.5}$$

$$\gamma_{\mathrm{L}}(d^*_{\mathrm{F},t}) = d^*_{\mathrm{L},t} \quad \forall t \in T. \tag{6.6}$$

Constraint (6.5) ensures that the follower has no incentive to deviate from the leader's global optimum, regardless of his type. Constraint (6.6) ensures that the optimal leader function passes through the desired point for any type of follower. The desired point $\{(d^*_{\mathrm{L},t}, d^*_{\mathrm{F},t})\}_{t\in T}$ is called *incentive controllable*, if the feasibility program (6.4)-(6.6) has a solution. A leader function is called game-optimal, if it is a solution to (6.4)-(6.6) for the leader's global optimum[2].

In summary, an optimal leader function should pass through the desired point for any type. Moreover, it should not intersect with the 0-level curve of the function

$$g_{\inf}(d_{\mathrm{L}}, d_{\mathrm{F}}, t) := U_{\mathrm{F}}(d_{\mathrm{L}}, d_{\mathrm{F}}, t) - U_{\mathrm{F}}(d^*_{\mathrm{L},t}, d^*_{\mathrm{F},t}, t) \tag{6.7}$$

and it should remain inside the sublevel set

$$\Lambda := \{(d_{\mathrm{L}}, d_{\mathrm{F}}) \in D_{\mathrm{L}} \times D_{\mathrm{F}} | g_{\inf}(d_{\mathrm{L}}, d_{\mathrm{F}}, t) \leq 0\} \tag{6.8}$$

for any $t \in T$.

[1]In game theory, a player is said to have full rationality if he always acts in a way to maximize his utility.

[2]When the leader has multiple global optima, a leader function is called game-optimal if it is the solution to (6.4)-(6.6) for at least one global optimum.

6.2 Two-Step Solution Approach

Assume the leader's global optimum $\{(d^*_{\mathrm{L},t}, d^*_{\mathrm{F},t})\}_{t\in T}$ exists and is unique[3]. An analytic solution for a general nonlinear leader function to the incentive design problem (6.4)-(6.6) is difficult to obtain, especially for general, nonconcave utility functions. The difficulties in solving the incentive design problem (6.4)-(6.6) lie in the fact that the decision space Γ_L is a function space of infinite dimensions, and that the equilibrium constraint (6.5) is also numerically challenging for general nonconvex utility functions, as it involves solving a global optimization problem. To solve this challenge problem, we develop a two-step solution approach involving basis function approximation and semi-infinite programming.

6.2.1 Basis Function Approximation

Basis functions are universal approximators that can approximate any given function with arbitrary accuracy when the set of selected basis functions is rich enough [158]. A finite set of basis functions $\mathscr{B} = \{b_i : \mathbb{R}^{n_\mathrm{F}} \to \mathbb{R}^{n_\mathrm{L}}\}_{i=1}^n$ is used to approximate the leader function γ_L. Each of these basis functions is further denoted by $b_i(\cdot;\xi_i)$, $i = 1,\dots,n$, to emphasize its dependence on the parameter vector ξ_i, which contains information regarding the location and the shape of each basis function (e.g. the center and the width of a radial basis function). The leader function γ_L is represented by a linear combination of the selected basis functions[4]

$$\gamma_\mathrm{L}(\cdot) = \sum_{i=1}^n \alpha_i \odot b_i(\cdot;\xi_i) \tag{6.9}$$

with weights $\alpha_i \in \mathbb{R}^{n_\mathrm{L}}$ and parameter vectors $\xi_i \in \Xi$. We denote the leader function represented by basis function approximation (6.9) as $\gamma_\mathrm{L}(\cdot;\alpha,\xi)$, to highlight its dependence on the parameters and weight of each basis function, with $\alpha = [\alpha_1^\mathrm{T}\cdots\alpha_n^\mathrm{T}]^\mathrm{T}$ and $\xi = [\xi_1^\mathrm{T}\cdots\xi_n^\mathrm{T}]^\mathrm{T}$. Then we can approximate the incentive design problem (6.4)-(6.6) by the following feasibility program:

$$\text{find:}\quad (\alpha,\xi) \in \mathbb{R}^{n_\mathrm{L}\times n} \times \Xi^n \tag{6.10}$$

$$\text{subject to:}\quad g_{\inf}(\gamma_\mathrm{L}(d_\mathrm{F}), d_\mathrm{F}, t) = U_\mathrm{F}\left(\sum_{i=1}^n \alpha_i \odot b_i(d_\mathrm{F};\xi_i), d_\mathrm{F}, t\right) - U_\mathrm{F}\left(d^*_{\mathrm{L},t}, d^*_{\mathrm{F},t}, t\right) \le 0 \tag{6.11}$$

$$\forall d_\mathrm{F} \in D_\mathrm{F}, \forall t \in T$$

$$\sum_{i=1}^n \alpha_i \odot b_i(d^*_{\mathrm{F},t};\xi_i) = d^*_{\mathrm{L},t} \quad \forall t \in T \tag{6.12}$$

$$\sum_{i=1}^n \alpha_i \odot b_i(d_\mathrm{F};\xi_i) \in D_\mathrm{L} \quad \forall d_\mathrm{F} \in D_\mathrm{F}. \tag{6.13}$$

Constraints (6.11) and (6.12) correspond to constraints (6.5) and (6.6), respectively. Constraint (6.13) guarantees that the resulting leader function indeed maps the follower's decision space to the leader's decision space.

[3]If there are multiple global optima, we can repeat (6.4)-(6.6) for each global optimum, and choose the leader function γ_L that gives the highest expected follower utility over his type space.

[4]The operator $\odot$ represents the elementwise (Schur) product.

Constraint (6.11) and (6.13) are difficult to address, as they must be satisfied on a continuous domain D_{F}. Constraint (6.13) can be replaced by a finite linear constraint if a stricter rule is applied to the selection of basis functions. Instead of $\mathscr{B} = \{b_i : \mathbb{R}^{n_{\mathrm{F}}} \to \mathbb{R}^{n_{\mathrm{L}}}\}_{i=1}^{n}$, we can choose $\tilde{\mathscr{B}} = \{b_i : D_{\mathrm{F}} \to D_{\mathrm{L}}\}_{i=1}^{n}$ as the set of selected basis functions. Note that a convex combination of the basis functions in $\tilde{\mathscr{B}}$ also maps D_{F} to D_{L}. Then constraint (6.13) can be replaced by the following two linear constraints:

$$\alpha \in [0,1]^{n_{\mathrm{L}}} \tag{6.14}$$

$$\sum_{i=1}^{n} \alpha_i = 1 \tag{6.15}$$

In this way constraint (6.13) is satisfied by construction. The feasibility program for the incentive design problem thus becomes (6.10)-(6.12),(6.14),(6.15), with the new set $\tilde{\mathscr{B}}$ of selected basis functions.

6.2.2 Semi-Infinite Programming

The incentive design problem (6.10)-(6.12),(6.14),(6.15) still cannot be solved directly, as constraint (6.11) must be satisfied on a continuous domain D_{F}. Mathematical programming problems with a finite number of decision variables but an infinite number of constraints are called Semi-Infinite Programming (SIP) problems [96]. Standard SIP problems can be represented by the following general form[5]:

$$\min_{x \in X} f(x) \tag{6.16}$$

subject to:

$$g_i(x, y) \leq 0 \quad \forall y \in Y_i, \forall i \in \{1, \ldots, p\} \tag{6.17}$$

where X and Y_i are continuous, compact subsets of $\mathbb{R}^{n_x}$ and $\mathbb{R}^{n_y}$, respectively, and the functions $f : \mathbb{R}^{n_x} \to \mathbb{R}$ and $g_i : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}$ are real-valued and continuous on their respective domains, for all i. For clarity we call x the decision variable and y the index variable. Furthermore, we call the continuous set Y_i the *index set* of each infinite constraint g_i. The intractable feasibility program (6.10)-(6.12),(6.14),(6.15) can then be transformed into a standard SIP problem (6.16),(6.17) with $|T|$ infinite constraints, by treating the parameters of the basis function (α, ξ) as the decision variable x, and d_{F} as the index variable y of the infinite constraints.

A comprehensive survey on numerical methods for semi-infinite programming problems is given in [96]. The major challenge in solving a semi-infinite programming problem lies in the fact that to check the feasibility of a point $\bar{x} \in X$, the following lower-level optimization problem

$$\max_{y \in Y_i} g_i(\bar{x}, y) \tag{6.18}$$

[5]For clarity we omit all finite constraints in Section 6.2.2, as they can be easily added to the resulting finite programming problem.

must be solved to global optimality for each $i \in \{1,\ldots,p\}$ [155]. Let y_i^* denote the global optimum for the i-th lower-level problem (6.18); then $\bar{x}$ is feasible as long as:

$$\max_{i\in\{1,\ldots,p\}} g_i(\bar{x}, y_i^*) \leq 0.$$

The difficulty of solving a semi-infinite programming problem depends on whether the lower-level problems are convex. As the convexity of the lower-level problem is so crucial in solving a semi-infinite programming problem, we now provide several sufficient conditions to check the convexity of the lower-level problem for the feasibility program (6.10)-(6.12),(6.14),(6.15).

Theorem 6.1 *Let U_F and γ_L (in the form of (6.9)) be continuous and twice differentiable on their respective domains; then the lower-level problem of the feasibility program (6.10)-(6.12),(6.14),(6.15) is convex if any of the following conditions is satisfied:*

(1) $U_F(\cdot,\cdot,t)$ is linear in d_L and d_F, and non-decreasing in d_L for all $t \in T$ and γ_L is concave;

(2) $U_F(\cdot,\cdot,t)$ is linear in d_L and d_F, and non-increasing in d_L for all $t \in T$ and γ_L is convex;

(3) $U_F(\cdot,\cdot,t)$ is concave in d_L and d_F, and non-decreasing in d_L for all $t \in T$, and γ_L is non-decreasing and concave;

(4) $U_F(\cdot,\cdot,t)$ is concave in d_L and d_F, and non-increasing in d_L for all $t \in T$, and γ_L is non-decreasing and convex;

Proof: The key to determine the convexity of the lower-level problem is to determine the concavity of the functions $g_{\text{inf}}(\gamma_\text{L}(\cdot),\cdot,t)$ for all $t \in T$. The second-order derivative of g_{inf} w.r.t. d_F is given by:

$$\frac{\partial^2 g_{\text{inf}}}{\partial d_\text{F}^2} = \underbrace{\frac{\partial^2 U_\text{F}}{\partial d_\text{L}^2}\left(\frac{\text{d}\gamma_\text{L}}{\text{d}d_\text{F}}\right)^2}_{\text{term 1}} + \underbrace{\frac{\partial U_\text{F}}{\partial d_\text{L}}\frac{\text{d}^2\gamma_\text{L}}{\text{d}d_\text{F}^2}}_{\text{term 2}} + \underbrace{2\frac{\partial^2 U_\text{F}}{\partial d_\text{L} d_\text{F}}\frac{\text{d}\gamma_\text{L}}{\text{d}d_\text{F}}}_{\text{term 3}} + \underbrace{\frac{\partial^2 U_\text{F}}{\partial d_\text{F}^2}}_{\text{term 4}}. \tag{6.19}$$

First we prove the theorem for conditions (1) and (2). When $U_\text{F}(\cdot,\cdot,t)$ is linear, all its second-order derivatives become 0, so only term 2 remains in (6.19). If U_F is non-decreasing in d_L and γ_L is concave, then $\frac{\partial U_\text{F}}{\partial d_\text{L}} \geq 0$ and $\frac{\text{d}^2\gamma_\text{L}}{\text{d}d_\text{F}^2} \leq 0$; thus $\frac{\partial^2 g_{\text{inf}}}{\partial d_\text{F}^2} \leq 0$, and therefore g_{inf} is concave in d_F. So now condition (1) is proved; condition (2) can be proved following similar arguments. Now we prove the theorem for conditions (3) and (4). When $U_\text{F}(\cdot,\cdot,t)$ is concave, then its Hessian is negative semi-definite, so term 1 and term 4 are both less than or equal to 0. Since γ_L is non-decreasing, $\frac{\text{d}\gamma_\text{L}}{\text{d}d_\text{L}} \geq 0$, so term 3 is also less than or equal to 0. Moreover, term 2 is non-positive if U_F is non-decreasing and γ_L is concave, so condition (3) is proved; or U_F is non-increasing and γ_L is convex, so condition (4) is proved. □

Remark: Theorem 6.1 can also be applied to select proper basis functions when U_F is concave. As both convexity and monotonicity are preserved by convex combination, Theorem 6.1 also holds if we replace γ_L by "each basis function" in condition (1)-(4).
The importance of the convexity of the lower-level problems is that it allows for the use of equivalent reformulation methods to transform the semi-infinite programming problem

into a finite programming problem. When the lower-level problems are convex, the semi-infinite programming problem (6.16)-(6.17) can be equivalently transformed into a finite programming problem through bilevel reformulations [155], like the Mathematical Program with Complementary Constraints (MPCC) reformulation [156], and the reformulation based on lower-level Wolfe duality [41]. However, such equivalent reformulation methods cannot be directly applied when at least one lower-level problem is nonconvex. Some numerical methods have been developed for semi-infinite programming problems with general, non-convex lower-level problems. We refer the interested readers to [96, 155] for a comprehensive survey. Moreover, some numerical solvers have also been developed for semi-infinite programming problems, like *fseminf* in the Matlab Optimization Toolbox, and the AMPL-coded NSIPS solver [163, 164] available in the NEOS server [36].

6.3 Numerical Example

In this section we present a numerical example to illustrate the procedure of a systematic computation of the optimal non-linear leader function for reverse Stackelberg games with incomplete information and general, nonconcave utility functions.

6.3.1 Setup

Let the leader and the follower's decision spaces be $D_\mathrm{L} = [-5, 5]$ and $D_\mathrm{F} = [-2, 2]$, respectively. We denote by $\underline{d}_k$ and $\overline{d}_k$ the lower and upper bounds of D_k for $k \in \{\mathrm{L}, \mathrm{F}\}$, respectively. The follower's type space is given by $T = \{t_1, t_2\}$ where $t_1 = 1$ and $t_2 = 5$, and the type distribution is $P(t_1) = 0.75$ and $P(t_2) = 0.25$. The Rosenbrock function [137], a popular valley-shaped non-convex testing function for optimization algorithms, is selected as the utility functions[6] for both players. In particular, we let the utility functions of the leader and the follower to be:

$$U_\mathrm{L} = -(1 + d_\mathrm{F})^2 - 100(d_\mathrm{L} + d_\mathrm{F}^2)^2 \tag{6.20}$$

$$U_\mathrm{F} = -(1 - d_\mathrm{F})^2 - 100(t d_\mathrm{L} - d_\mathrm{F}^2)^2. \tag{6.21}$$

The type t can be viewed as a parameter that influences the shape of the follower's utility u_F. Two radial basis function [21] families, the Gaussian radial basis functions and the inverse multiquadric functions, are selected to approximate the leader function. The Gaussian radial basis functions are defined by:

$$\phi(r) = \exp(-\frac{r^2}{\Delta^2}) \tag{6.22}$$

and the inverse multiquadric functions are defined by:

$$\phi(r) = \frac{1}{\sqrt{r^2 + \Delta^2}} \tag{6.23}$$

[6]The signs are reversed as the Rosenbrock function is designed for minimization problems.

Then each basis function can be represented by:

$$b_i(d_\mathrm{F}) = \frac{\overline{d}_\mathrm{L} - \underline{d}_\mathrm{L}}{\overline{\phi} - \underline{\phi}}\phi(r) + \underline{d}_\mathrm{L} \tag{6.24}$$

where $r = \| d_\mathrm{F} - c_i \|_2$ is the Euclidean distance to the center of the $i-$th radial basis function, and $\overline{\phi}$ and $\underline{\phi}$ are the upper and lower bounds of the selected radial basis function. The centers $\{c_i\}_{i=1}^n$ are equidistantly placed on D_F, and the width is fixed to $\Delta = \frac{\overline{d}_\mathrm{F} - \underline{d}_\mathrm{F}}{n-1}$.
Since both the centers and widths are fixed for the radial basis functions, the parameter vector ξ is empty as we only optimize the weights α. Since we have the freedom to choose a well-behaved objective function to the feasibility program (6.10)-(6.12),(6.14),(6.15), we add the following quadratic objective:

$$\min \sum_{i=1}^{n} \alpha_i^2. \tag{6.25}$$

The leader's global optimization problem (6.1)-(6.3) is now solved by the nonlinear programming solver SNOPT from Tomlab 8.0 with multi-start, and the standard semi-infinite programming problem (6.25)(6.10)-(6.12),(6.14),(6.15) for the incentive design problem is solved by *fseminf* from the Matlab Optimization Toolbox. All simulations are performed on a desktop computer with an Intel i5-3470 Quad core and 16 GB of RAM, running Matlab R2015b on a 64-bit version of SUSE Linux Enterprise Desktop 11.
As the semi-infinite programming solver *fseminf* uses a discretization method, which does not guarantee feasibility of each iteration, we will measure the violation of the infinite constraint (6.11) after a leader function is obtained. A fine uniform grid $\tilde{D}_\mathrm{L} \times \tilde{D}_\mathrm{F}$ (with 101×101 grid points) is generated for $D_\mathrm{L} \times D_\mathrm{F}$ for post-evaluation of the infinite constraint (6.12). The following measure, which is a non-negative value and should be kept as small as possible, is used to evaluate the constraint violation:

$$v_t = \frac{\max\limits_{\tilde{d}_\mathrm{F} \in \tilde{D}_\mathrm{F}} g_\mathrm{inf}(\gamma_\mathrm{L}(\tilde{d}_\mathrm{F}), \tilde{d}_\mathrm{F}, t)}{\max\limits_{(\tilde{d}_\mathrm{L}, \tilde{d}_\mathrm{F}) \in \tilde{D}_\mathrm{L} \times \tilde{D}_\mathrm{F}} g_\mathrm{inf}(\tilde{d}_\mathrm{L}, \tilde{d}_\mathrm{F}, t)} \quad \forall t \in T. \tag{6.26}$$

The denominator represents the maximal violation of a given constraint on the whole evaluation grid, and the numerator calculates the maximal constraint violation when the resulting γ_L is implemented. In this way we can have a quantitative measure on the performance of each leader function.

6.3.2 Discussion of Results

The team solution computed by SNOPT is $(-0.7450, -0.8634)$ for t_1 and $(-0.7445, -0.8637)$ for t_2. The leader functions obtained from different numbers of Gaussian radial basis functions and inverse multiquadric functions are shown in Figure 6.1. An optimal leader function should pass through the leader's desired points and should not intersect with the 0-level curves of g_inf, so that the follower cannot obtain a strictly higher utility if he deviates from the leader's desired points, regardless of his type. As we can see, all the leader functions are continuous and lie in the leader's decision space D_L, and all of them pass through the desired points[7]. Thus constraint (6.12) is satisfied for all of them. However, not all resulting leader functions satisfy the infinite constraint (6.11). For example, as shown in Figure 6.1b, the leader functions obtained using 10 and 15 inverse multiquadric basis functions both intersect with the 0-level curve of $g_{\inf}$ for t_2. Gaussian radial basis functions yield a better performance in comparison, as shown in Figure 6.1a, as even the leader function resulting from only 10 basis functions has no obvious intersection with either 0-level curve.

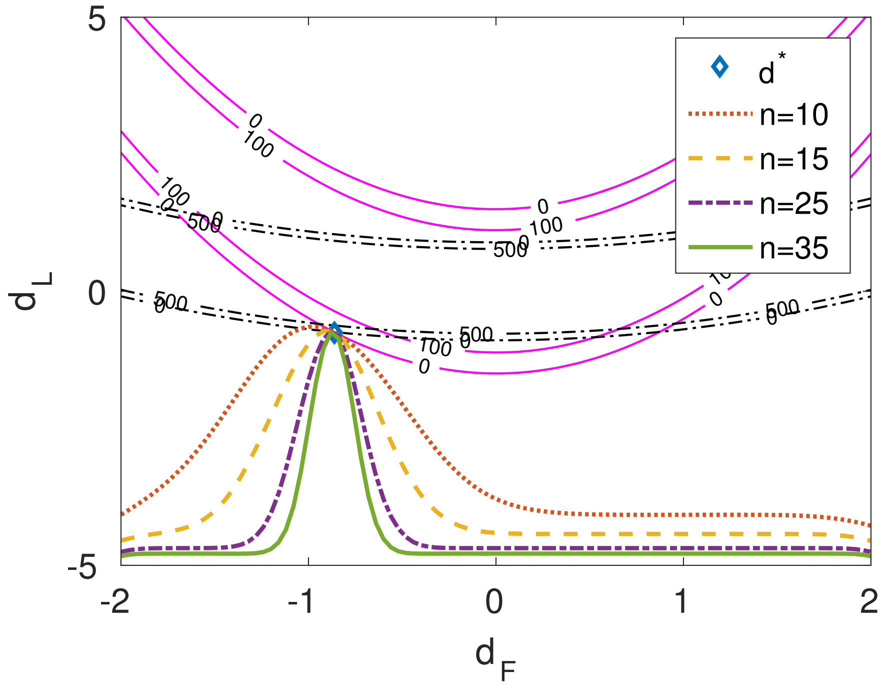
The performance of the two basis function families, quantified by the constraint violation (6.26), is visualized in Figure 6.2. Both basis function families show an improvement of performance as the number of basis functions increases. Gaussian radial basis functions obviously perform better than inverse multiquadric functions, as the infinite constraint for t_1 is never violated (v_{t_1} remains 0 in Figure 6.2a), and the maximum violation of $g_{\inf}$ for t_1 is only 0.14, compared to 1 for the inverse multiquadric case. Moreover, the performance using 10 Gaussian basis functions is better than the performance using 25 inverse multiquadratic functions, and with 30 Gaussian radial basis functions we can already find a “perfect" leader function with no constraint violations. From Figure 6.2 we can conclude that a selection of 30 basis functions is already sufficient to obtain a well-performing leader function, as the maximal constraint violation is no more than 0.1 for both Gaussian radial basis functions and inverse multiquadric basis functions.

The mean CPU time to solve the incentive design problem using *fseminf* with different numbers of Gaussian and inverse multiquadric radial basis functions is shown in Figure 6.3. We can see that neither choice of basis functions is computationally very demanding, as the largest problems ($n = 40$) can be computed within 1.6 seconds, and within 1 second we can already obtain a satisfactory leader function (for $n = 30$).

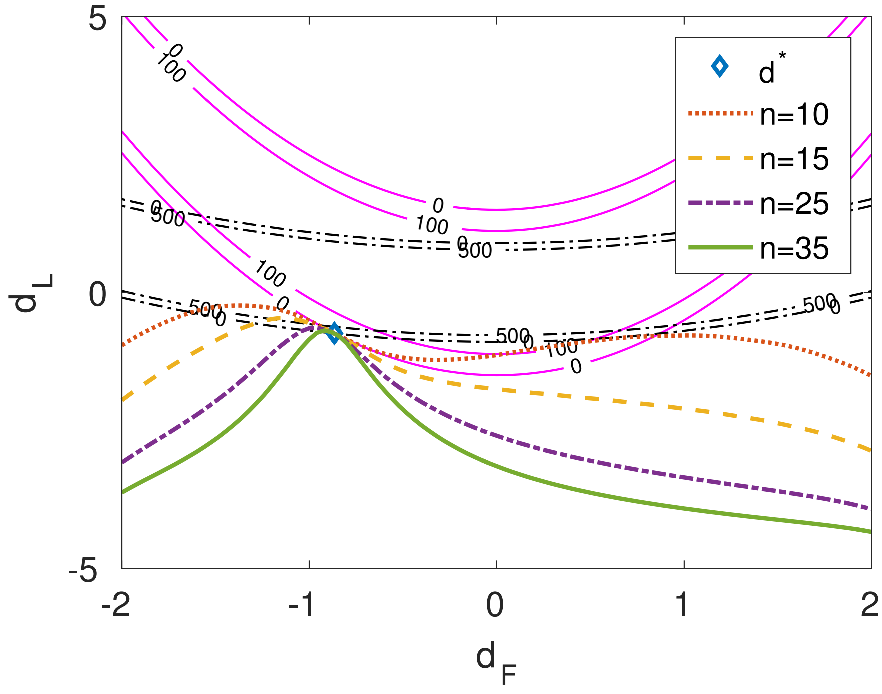
6.4 Conclusions

A structured numerical solution approach has been developed for the class of nonlinear leader functions for reverse Stackelberg games with incomplete information and general utility functions. Basis functions are used to approximate the nonlinear leader function, transforming the incentive design problem into a standard semi-infinite programming problem, which has been extensively studied in literature. The worked example shows that the proposed solution approach provides satisfactory results in a relatively short CPU time using typical basis functions like Gaussian radial basis functions, and standard semi-infinite programming solvers.

[7]Note that the global optimum in general includes two different points for different types, but they are very close to each other in this example.

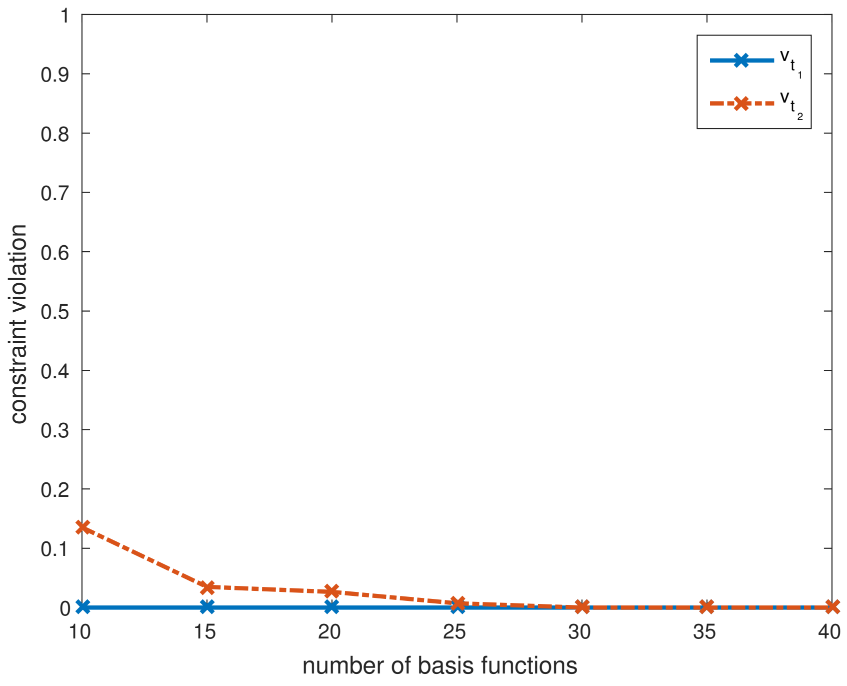


(a) Gaussian radial basis functions

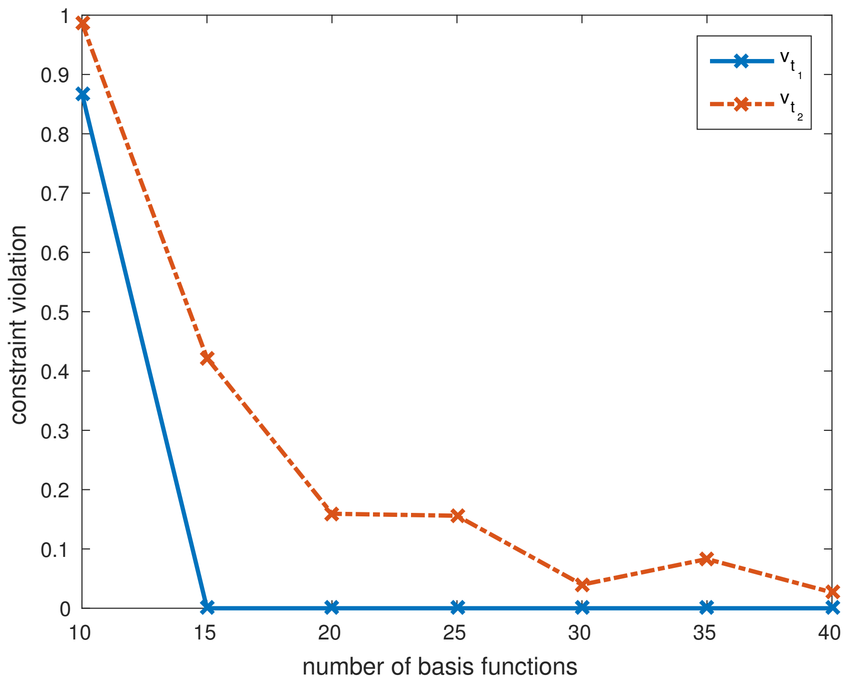


(b) Inverse multiquadric basis functions

Figure 6.1: Leader functions obtained from the indirect approach based on basis function approximation and semi-infinite programming. The solid and dashed contour lines are the 0-level curves of g_{inf} for t_1 and t_2, respectively.



(a) Gaussian radial basis functions



(b) Inverse multiquadric basis functions

Figure 6.2: Measure of constraint violation of the infinite constraint g_{inf} for both types of the follower.

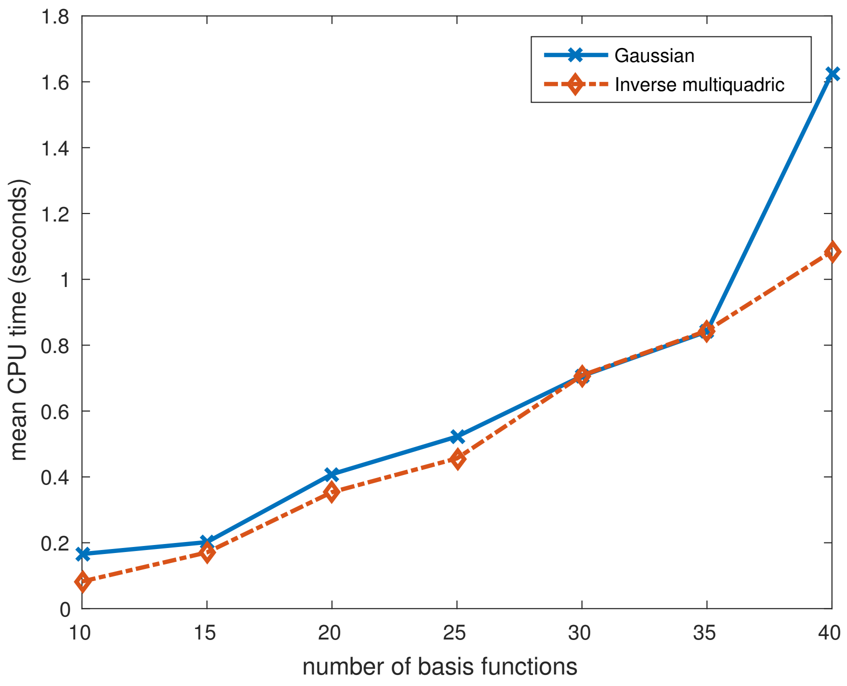


Figure 6.3: Mean CPU time for the compuation of the incentive design problem using Gaussian radial basis functions and inverse multiquadratic basis functions. The mean is taken over 10 runs.

Chapter 7

Conclusions and Future Research

7.1 Conclusions

In this thesis we have developed robust and tractable model-based approaches for maintenance optimization of railway infrastructure networks. In addition, we have developed a compact formulation for a variant of the multiple traveling salesman problem, and a systematic solution method for reverse Stackelberg games with incomplete information.

7.1.1 Multi-Level Maintenance Optimization

In the first part of the thesis, we have developed a multi-level model-based, optimization-based approach covering both the long-term and short-term perspectives in maintenance decision making for railway infrastructure networks.
First, we have developed a centralized multi-level approach for maintenance planning of a single line of track divided into a small number of sections. Representative scenario-based Model Predictive Control (MPC) is used at the high level to determine an optimal long-term component-wise intervention plan for the entire line, and the Time Instant Optimization (TIO) approach is applied to transform the MPC optimization problem with both continuous and integer decision variables into a nonlinear continuous optimization problem. The middle-level problem considers the optimal allocation of time slots for the maintenance interventions suggested at the high level to optimize the trade-off between traffic disruption and the setup cost of maintenance slots. Based on the high-level intervention plan, the low-level problem optimizes the clustering of individual defects to be treated by a maintenance agent, subject to the time limit imposed by the maintenance time slots. A case study with historical data on the optimal treatment of squats in the Eindhoven-Weert line in the Dutch railway network has been performed. Simulation results show that the developed approach is real-time implementable and provides suitable maintenance plans.
Subsequently, we have develop a distributed multi-level approach for maintenance planning of large-scale networks, i.e. a large complex network of tracks with a huge number of sections. The Mixed Logical Dynamical (MLD) framework has been adopted to model the hybrid deterioration dynamics, and a robust scenario-based MLD-MPC controller has been developed at the high level to determine the section-wise intervention at each time step, considering various sources of uncertainties. Dantzig-Wolfe decomposition is used to divide the computational burden of the centralized MPC optimization problem among subproblems that are easier to solve. The optimal schedule of the suggested maintenance in-

terventions that aims to minimize the total setup costs, disruption costs, and travel costs of maintenance crews over the whole network, has then been formulated as a capacitated arc routing problem and transformed into a node routing problem. The approach has been demonstrated by a numerical case study of the optimal treatment of squats in a regional Dutch railway network. Computational experiments show that the proposed approach is scalable. Comparison with alternative approaches shows that the proposed approach yields an excellent trade-off between safety and cost-effectiveness.

7.1.2 Fixed-Destination Multi-Depot Traveling Salesman Problem

In the second part of the thesis, we have considered the fixed-destination, multi-depot traveling salesman problem, where several salesmen will start from different depots, and they are required to return to the depot they originated from. We have proposed a novel formulation for this problem using 2-index binary variables and node currents. Compared to other 2-index formulations from the literature, this novel formulation requires less binary variables and continuous variables to formulate the problem, resulting in lower computation times. Using a large benchmark the effectiveness of the new formulation has been demonstrated.

7.1.3 Reverse Stackelberg Games

In the third part of the thesis, we have proposed a novel numerical solution approach for systematic computation of optimal nonlinear leader functions, also known as incentives, for reverse Stackelberg games with incomplete information and general, nonconcave utility functions. We have applied basis function approximation to the class of nonlinear leader functions, and have treated the incentive design problem as a standard semi-infinite programming problem. A numerical example has been provided to illustrate the proposed solution approach and to demonstrate its efficiency.

7.2 Future Research

From a practical perspective, there are several directions to extend this thesis:

- Instead of a simulation-based case study, a business case study can be performed, where the actual maintenance costs and degradation levels of the railway infrastructure are compared to the maintenance costs and degradation levels resulting from the multi-level maintenance optimization approach developed in this dissertation.

- Several practical aspects, e.g. the seasonal changes in the deterioration model of the railway infrastructure, and the interruption of planned maintenance operations caused by misty or rainy weather, can be included in the deterioration model.

- Condition-based maintenance planning and train timetabling for a railway network can be formulated as a joint optimization problem. In this thesis the length of the maintenance time slot, which corresponds to the traffic-free slot in the timetable, is optimized based on an existing timetable. However, a more optimal maintenance plan and timetable can be obtained by formulating the two optimization problem as a joint problem.

- Heterogeneous components, e.g. rail and switches, can be considered in the maintenance optimization problem.
- The model-based maintenance optimization approach in this thesis is specifically developed for railway infrastructure networks. It is interesting to investigate whether the developed approach can be adapted to other infrastructure networks like water and gas pipeline networks.

The following extensions can be made from a theoretical perspective:

- As a railway infrastructure is directly exposed to the environment, its deterioration dynamics demonstrates significantly difference in different seasons because of the difference in temperature, humidity, etc. Such seasonal changes in deterioration dynamics can be incorporated by a time-varying model.
- Most distributed optimization methods in literature are designed for continuous optimization problems. As most of the optimization problems in this thesis are mixed integer programming problems, it is worthwhile to develop distributed optimization methods that offer an optimality guarantee, or heuristics/metaheuristics with a performance guarantee.
- Typical track defects, like squats and ballast degradation, are caused by wear and tear. In this case, first-principles modeling of the degradation dynamics of railway track infrastructure is promising as it provides more insight into the failure mechanism with less data. However, the resulting first-principles model might be too complex to be used for efficient control and optimization algorithms. How to model the degradation dynamics that covers the important physical characteristics without adding too much complexity is a challenge. Regression techniques like piecewise-affine approximation can be used for complex nonlinear degradation dynamics to obtain a tractable problem.
- The node-current cycle imposement constraints developed for FmMTSP can be extended to other scheduling and routing problems like the multiple Vehicle Routing Problem (mVRP). However, the MILP formulation for mVRP using node-current cycle imposement constraints is only capable of solving relatively small test instances (less than 50 nodes). In the future, decomposition methods like Benders decomposition can be applied to the proposed MILP formulation to solve larger mVRP instances.

Bibliography

[1] R. Ahmad and S. Kamaruddin. An overview of time-based and condition-based maintenance in industrial application. *Computers & Industrial Engineering*, 63(1):135–149, 2012.

[2] Y.K. Al-Douri, P. Tretten, and R. Karim. Improvement of railway performance: a study of Swedish railway infrastructure. *Journal of Modern Transportation*, 24(1):22–37.

[3] T. Alamo, R. Tempo, and A. Luque. On the sample complexity of randomized approaches to the analysis and design under uncertainty. In *American Control Conference (ACC), 2010*, pages 4671–4676. IEEE, 2010.

[4] D.L. Applegate, R.E. Bixby, V. Chvatal, and W.J. Cook. *The Traveling Salesman Problem - A Computational Study*. Princeton University Press, 2006.

[5] N. Ascheuer, M. Fischetti, and M. Grötschel. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming*, 90(3):475–506, 2001.

[6] R. Baldacci and V. Maniezzo. Exact methods based on node-routing formulations for undirected arc-routing problems. *Networks*, 47(1):52–60, 2006.

[7] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

[8] T. Başar and H. Selbuz. Closed-loop Stackelberg strategies with applications in the optimal control of multilevel systems. *IEEE Transactions on Automatic Control*, 24(2):166–179, 1979.

[9] T. Bektaş. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209 – 219, 2006.

[10] T. Bektaş. Formulations and Benders decomposition algorithms for multidepot salesmen problems with load balancing. *European Journal of Operational Research*, 216(1):83 – 93, 2012.

[11] J.M. Belenguer, E. Benavent, C. Prins, C. Prodhon, and R.W. Calvo. A branch-and-cut method for the capacitated location-routing problem. *Computers & Operations Research*, 38(6):931 – 941, 2011.

[12] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[13] M. Ben-Daya, U. Kumar, and D.N. Murthy. Condition-based maintenance. *Introduction to Maintenance Engineering: Modeling, Optimization, and Management*, pages 355–387, 2016.

[14] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical programming*, 88(3):411–424, 2000.

[15] E. Benavent and A. Martínez. Multi-depot multiple TSP: a polyhedral study and computational results. *Annals of Operations Research*, 207(1):7–25, 2013.

[16] Jacques F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

[17] N. Boland, T. Kalinowski, H. Waterer, and L. Zheng. Mixed integer programming based maintenance scheduling for the Hunter valley coal chain. *Journal of Scheduling*, 16(6):649–659, 2013.

[18] R. Bourdais, J. Buisson, D. Dumur, H. Guéguen, and P.D. Moroşan. Distributed MPC under coupled constraints based on dantzig-wolfe decomposition. In *Distributed Model Predictive Control Made Easy*, pages 101–114. Springer, 2014.

[19] M. Branda. Sample approximation technique for mixed-integer stochastic programming problems with several chance constraints. *Operations Research Letters*, 40(3):207–211, 2012.

[20] G. Budai, D. Huisman, and R. Dekker. Scheduling preventive railway maintenance activities. *Journal of the Operational Research Society*, 57(9):1035–1044, 2006.

[21] M.D. Buhmann. Radial basis functions. *Acta Numerica 2000*, 9:1–38, 2000.

[22] M. Burger. *Exact and Compact Formulation of the Fixed-Destination Travelling Salesman Problem by Cycle Imposement Through Node Currents*, pages 83–88. Springer International Publishing, Cham, 2014.

[23] M. Burger, Z. Su, and B. De Schutter. A node current-based 2-index formulation for the fixed-destination multi-depot travelling salesman problem. *European Journal of Operational Research*, 265(2):463–477, 2018.

[24] G.C. Calafiore. Random convex programs. *SIAM Journal on Optimization*, 20(6):3427–3464, 2010.

[25] G.C. Calafiore and M.C. Campi. The scenario approach to robust control design. *IEEE Transactions on Automatic Control*, 51(5):742–753, 2006.

[26] E.F. Camacho and C.B. Alba. *Model Predictive Control*. Springer Science & Business Media, 2013.

[27] M.C. Campi and S. Garatti. The exact feasibility of randomized solutions of uncertain convex programs. *SIAM Journal on Optimization*, 19(3):1211–1230, 2008.

[28] M.C. Campi and S. Garatti. A sampling-and-discarding approach to chance-constrained optimization: feasibility and optimality. *Journal of Optimization Theory and Applications*, 148(2):257–280, 2011.

[29] P.J. Campo and M. Morari. Robust model predictive control. In *American Control Conference*, pages 1021–1026, 1987.

[30] M. Chamanbaz, F. Dabbene, R. Tempo, V. Venkataramanan, and Q. Wang. Sequential randomized algorithms for convex optimization in the presence of uncertainty. *IEEE Transactions on Automatic Control*, 61(9):2565–2571, 2016.

[31] A. Claus. A new formulation for the travelling salesman problem. *SIAM Journal on Algebraic Discrete Methods*, 5(1):21–25, 1984.

[32] L.C. Coelho and G. Laporte. The exact solution of several classes of inventory-routing problems. *Computers & Operations Research*, 40(2):558 – 565, 2013.

[33] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153(1):235–256, 2007.

[34] C. Contardo and R. Martinelli. A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optimization*, 12:129 – 146, 2014.

[35] W.J. Cook. *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation.* Princeton University Press, 2012.

[36] J. Czyzyk, M.P. Mesnier, and J.J. Moré. The NEOS server. *Computing in Science & Engineering*, 5(3):68–75, 1998.

[37] G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410, 1954.

[38] G.B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.

[39] C. Dao, R. Basten, and A. Hartmann. Maintenance scheduling for railway tracks under limited possession time. *Journal of Transportation Engineering, Part A: Systems*, 144(8):04018039, 2018.

[40] M. Desrochers and G. Laporte. Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters*, 10(1):27 – 36, 1991.

[41] 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, 2013.

[42] C. Doppstadt, A. Koberstein, and D. Vigo. The hybrid electric vehicle–traveling salesman problem. *European Journal of Operational Research*, 253(3):825–842, 2016.

[43] K. Edlund, J.D. Bendtsen, and J.B. Jørgensen. Hierarchical model-based predictive control of a power plant portfolio. *Control Engineering Practice*, 19(10):1126–1136, 2011.

[44] P.M. Esfahani, T. Sutter, and J. Lygeros. Performance bounds for the scenario approach and an extension to a class of non-convex programs. *IEEE Transactions on Automatic Control*, 60(1):46–58, 2015.

[45] C. Esveld. *Modern Railway Track.* MRT-Productions Zaltbommel, 2001.

[46] A. Fakhri, M. Ghatee, A. Fragkogios, and G.K.D. Saharidis. Benders decomposition with integer subproblem. *Expert Systems with Applications*, 89:20–30, 2017.

[47] S.M. Famurewa, T. Xin, M. Rantatalo, and U. Kumar. Optimisation of maintenance track possession time: A tamping case study. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, 229(1):12–22, 2015.

[48] Y. Fan, S. Dixon, R.S. Edwards, and X. Jian. Ultrasonic surface wave propagation and interaction with surface defects on rail track head. *NDT & E International*, 40(6):471–477, 2007.

[49] S. Fararooy and J. Allan. Condition-based maintenance of railway signalling equipment. In *International Conference on Electric Railways in a United Europe*, pages 33–37. IET, 1995.

[50] D. Feillet. A tutorial on column generation and branch-and-price for vehicle routing problems. *4OR: A Quarterly Journal of Operations Research*, 8(4):407–424, 2010.

[51] G. Finke, A. Claus, and E. Gunn. A two-commodity network flow arroach to the traveling salesman problem. *Congressus Numerantium*, 41:167–178, May 1984.

[52] K.R. Fox, B. Gavish, and S.C. Graves. An n-constraint formulation of the time-dependent traveling salesman problem. *Operations Research*, 28(4):1018–1021, August 1980.

[53] D.M. Frangopol, M.J. Kallen, and J.M. Van Noortwijk. Probabilistic models for life-cycle performance of deteriorating structures: review and future directions. *Progress in Structural Engineering and Materials*, 6(4):197–212, 2004.

[54] B. Gavish and S.C. Graves. The travelling salesman problem and related problems. Technical report, Massachusetts Institute of Technology, 1978.

[55] N.Z. Gebraeel, M.A. Lawley, R. Li, and J.K. Ryan. Residual-life distributions from component degradation signals: a Bayesian approach. *IIE Transactions*, 37(6):543–557, 2005.

[56] A. Geletu, M. Klöppel, A. Hoffmann, and P. Li. A tractable approximation of non-convex chance constrained optimization with non-gaussian uncertainties. *Engineering Optimization*, 47(4):495–520, 2015.

[57] J. Gondzio, P. González-Brevis, and P. Munari. New developments in the primal–dual column generation technique. *European Journal of Operational Research*, 224(1):41–51, 2013.

[58] M.F. Gorman and J.J. Kanet. Formulation and solution approaches to the rail maintenance production gang scheduling problem. *Journal of Transportation Engineering*, 136(8):701–708, 2010.

[59] S. Grammatico, X. Zhang, K. Margellos, P. Goulart, and J. Lygeros. A scenario approach for non-convex control design. *IEEE Transactions on Automatic Control*, 61(2):334–345, 2016.

[60] N. Groot, B. De Schutter, and H. Hellendoorn. Dynamic optimal routing based on a reverse stackelberg game approach. In *2012 15th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 782–787. IEEE, 2012.

[61] N. Groot, B. De Schutter, and H. Hellendoorn. On systematic computation of optimal nonlinear solutions for the reverse stackelberg game. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(10):1315–1327, 2014.

[62] J.M. Grosso, C. Ocampo-Martínez, V. Puig, and B. Joseph. Chance-constrained model predictive control for drinking water networks. *Journal of Process Control*, 24(5):504–516, 2014.

[63] J.K. Gruber, D.R. Ramirez, D. Limon, and T. Alamo. Computationally efficient nonlinear min-max model predictive control based on Volterra series models –application to a pilot plant. *Journal of Process Control*, 23(4):543–560, 2013.

[64] T. Gschwind and S. Irnich. Dual inequalities for stabilized column generation revisited. *INFORMS Journal on Computing*, 28(1):175–194, 2016.

[65] V. Gunnerud and B. Foss. Oil production optimization—a piecewise linear model, solved with two decomposition strategies. *Computers & Chemical Engineering*, 34(11):1803–1812, 2010.

[66] E. Gustavsson. Scheduling tamping operations on railway tracks using mixed integer linear programming. *EURO Journal on Transportation and Logistics*, 4(1):97–112, 2015.

[67] P. Hansen, B. Jaumard, and G. Savard. New branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing*, 13(5):1194–1217, 1992.

[68] J.C. Harsanyi. Games with incomplete information played by "Bayesian" players, i–iii. *Management Science*, 14(3, 5, 7):159–182, 320–334, 486–502, 1968.

[69] Q. He, H. Li, D. Bhattacharjya, D.P. Parikh, and A. Hampapur. Track geometry defect rectification based on track deterioration modelling and derailment risk assessment. *Journal of the Operational Research Society*, 66(3):392–404, 2015.

[70] F. Heinicke, A. Simroth, G. Scheithauer, and A. Fischer. A railway maintenance scheduling problem with customer costs. *EURO Journal on Transportation and Logistics*, 4(1):113–137, 2015.

[71] K. Helsgaun. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126:106–130, 2000.

[72] R. Henrion, C. Küchler, and W. Römisch. Scenario reduction in stochastic programming with respect to discrepancy distances. *Computational Optimization and Applications*, 43(1):67–93, 2009.

[73] A. Higgins. Scheduling of railway track maintenance activities and crews. *Journal of the Operational Research Society*, 49(10):1026–1033, 1998.

[74] J.L. Hong, Y. Yang, and L. Zhang. Sequential convex approximations to joint chance constrained programs: A Monte Carlo approach. *Operations Research*, 59(3):617–630, 2011.

[75] C.A. Irawan, D. Ouelhadj, D. Jones, M. Stålhane, and I.B. Sperstad. Optimisation of maintenance routing and scheduling for offshore wind farms. *European Journal of Operational Research*, 256(1):76–89, 2017.

[76] A. Jamshidi, S. Faghih-Roohi, S. Hajizadeh, A. Núñez, R. Dollevoet, Z. Li, and B. De Schutter. A big data analysis approach for rail failure risk assessment. *Risk Analysis*, 37(8):1495–1507, 2017.

[77] A. Jamshidi, A. Núñez, R. Dollevoet, and Z. Li. Robust and predictive fuzzy key performance indicators for condition-based treatment of squats in railway infrastructures. *Journal of Infrastructure Systems*, 23(3):04017006, 2017.

[78] R. Jans. Classification of Dantzig–Wolfe reformulations for binary mixed integer programming problems. *European Journal of Operational Research*, 204(2):251–254, 2010.

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

[80] J.B. Jose. Leader-follower strategies for multilevel systems. 1978.

[81] I. Jurado, J.M. Maestre, P. Velarde, C. Ocampo-Martinez, I. Fernández, B. Isla Tejera, and J.R. del Prado. Stock management in hospital pharmacy using chance-constrained model predictive control. *Computers in Biology and Medicine*, 72:248–255, 2016.

[82] I. Kara and T. Bektaş. Integer linear programming formulations of multiple salesman problems and its variations. *European Journal of Operational Research*, 174(3):1449–1458, 2006.

[83] M.C.A. Olde Keizer, S.D.P. Flapper, and R.H. Teunter. Condition-based maintenance policies for systems with multiple dependent components: A review. *European Journal of Operational Research*, 261(2):405–420, 2017.

[84] K.A.H. Kobbacy and D.N.P. Murthy. *Complex System Maintenance Handbook*. Springer Science & Business Media, 2008.

[85] R.V. Kulkarni and P.R. Bhave. Integer programming formulations of vehicle routing problems. *European Journal of Operational Research*, 20(1):58 – 67, 1985.

[86] A. Langevin, F. Soumis, and J. Desrosiers. Classification of travelling salesman problem formulations. *Operations Research Letters*, 9(2):127 – 132, 1990.

[87] G. Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247, June 1992.

[88] G. Laporte, Y. Nobert, and D. Arpin. An exact algorithm for solving a capacitated location-routing problem. *Annals of Operations Research*, 6(9):291–310, 1986.

[89] R.M. Lewis, V. Torczon, and M.W. Trosset. Direct search methods: then and now. *Journal of Computational and Applied Mathematics*, 124(1):191–207, 2000.

[90] Z. Li and C.A. Floudas. Optimal scenario reduction framework based on distance of uncertainty distribution and output performance: I. single reduction via mixed integer linear optimization. *Computers & Chemical Engineering*, 70:50–66, 2014.

[91] Z. Li and C.A. Floudas. Optimal scenario reduction framework based on distance of uncertainty distribution and output performance: II. sequential reduction. *Computers & Chemical Engineering*, 84:599–610, 2016.

[92] Z. Li, M. Molodova, A. Núñez, and R. Dollevoet. Improvements in axle box acceleration measurements for the detection of light squats in railway infrastructure. *IEEE Transactions on Industrial Electronics*, 62(7):4385–4397, 2015.

[93] T. Lidén and M. Joborn. An optimization model for integrated planning of railway traffic and network maintenance. *Transportation Research Part C: Emerging Technologies*, 74:327–347, 2017.

[94] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.

[95] L. Ling, X. Xiao, and X. Jin. Development of a simulation model for dynamic derailment analysis of high-speed trains. *Acta Mechanica Sinica*, 30(6):860–875, 2014.

[96] M. López and G. Still. Semi-infinite programming. *European Journal of Operational Research*, 180(2):491–518, 2007.

[97] R. Luo, R. Bourdais, T.J.J. van den Boom, and B. De Schutter. Multi-agent model predictive control based on resource allocation coordination for a class of hybrid systems with limited information sharing. *Engineering Applications of Artificial Intelligence*, 58:123–133, 2017.

[98] S. Madanat. Optimal infrastructure management decisions under uncertainty. *Transportation Research Part C: Emerging Technologies*, 1(1):77–88, 1993.

[99] J.M. Maestre and R.R. Negenborn. *Distributed model predictive control made easy*, volume 69. Springer Science & Business Media, 2013.

[100] K. Margellos, P. Goulart, and J. Lygeros. On the road between robust optimization and the scenario approach for chance constrained optimization problems. *IEEE Transactions on Automatic Control*, 59(8):2258–2263, 2014.

[101] P.R.C. Mendes, J.M. Maestre, C. Bordons, and J.E. Normey-Rico. A practical approach for hybrid distributed mpc. *Journal of Process Control*, 55:30–41, 2017.

[102] Sophie Mercier, Carolina Meier-Hirmer, and Michel Roussignol. Bivariate Gamma wear processes for track geometry modelling, with application to intervention scheduling. *Structure and Infrastructure Engineering*, 8(4):357–366, 2012.

[103] C.E. Miller, A.W. Tucker, and R.A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the Association for Computing Machinery*, 7(4):326–329, October 1960.

[104] C.E. Miller, A.W. Tucker, and R.A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the Association for Computing Machinery*, 7(4):326–329, October 1960.

[105] M. Molodova, Z. Li, A. Núñez, and R. Dollevoet. Automatic detection of squats in railway infrastructure. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):1980–1990, 2014.

[106] M. Morari and E. Zafiriou. *Robust Process Control*, volume 488. Prentice Hall Englewood Cliffs, NJ, 1989.

[107] P.D. Moroşan, R. Bourdais, D. Dumur, and J. Buisson. A distributed MPC strategy based on Benders' decomposition applied to multi-source multi-zone temperature regulation. *Journal of Process Control*, 21(5):729–737, 2011.

[108] N.N. Nandola and D.E. Rivera. An improved formulation of hybrid model predictive control with application to production-inventory systems. *IEEE Transactions on Control Systems Technology*, 21(1):121–135, 2013.

[109] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, 1999.

[110] A. Nemirovski. On safe tractable approximations of chance constraints. *European Journal of Operational Research*, 219(3):707–718, 2012.

[111] A. Nemirovski and A. Shapiro. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17(4):969–996, 2006.

[112] L. Ntaimo. Disjunctive decomposition for two-stage stochastic mixed-binary programs with random recourse. *Operations research*, 58(1):229–243, 2010.

[113] P. Oberlin, S. Rathinam, and S. Darbha. A transformation for a heterogeneous, multiple depot, multiple traveling salesman problem. In *2009 American Control Conference*, pages 1292–1297, June 2009.

[114] P. Oberlin, S. Rathinam, and S. Darbha. A transformation for a multiple depot, multiple traveling salesman problem. In *2009 American Control Conference*, pages 2636–2641, June 2009.

[115] G.J. Olsder. Phenomena in inverse Stackelberg games, part 1: Static problems. *Journal of Optimization Theory and Applications*, 143(3):589–600, 2009.

[116] T. Öncan, Ï.K. Altinel, and G. Laporte. A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research*, 36(3):637 – 654, 2009.

[117] A.J. Orman and H.P. Williams. *A Survey of Different Integer Programming Formulations of the Travelling Salesman Problem*, volume 9 of *Advances in Computational Management Science*, pages 91–104. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[118] M. Padberg and T.Y. Sung. An analytical comparison of different formulations of the travelling salesman problem. *Mathematical Programming*, 52(1):315–357, May 1991.

[119] B.K. Pagnoncelli, S. Ahmed, and A. Shapiro. Sample average approximation method for chance constrained programming: theory and applications. *Journal of optimization theory and applications*, 142(2):399–416, 2009.

[120] T. Paolo and D. Vigo, editors. *The Vehicle Routing Problem*. Monographs on Discrete Methematics and Applications. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.

[121] F. Pargar, O. Kauppila, and J. Kujala. Integrated scheduling of preventive maintenance and renewal projects for multi-unit systems with grouping and balancing. *Computers & Industrial Engineering*, 110:43–58, 2017.

[122] S.N. Parragh, K.F. Doerner, and R.F. Hartl. A survey on pickup and delivery problems part I: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008.

[123] S.N. Parragh, K.F. Doerner, and R.F. Hartl. A survey on pickup and delivery problems part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 2008.

[124] F. Peng and Y. Ouyang. Track maintenance production team scheduling in railroad networks. *Transportation Research Part B: Methodological*, 46(10):1474–1488, 2012.

[125] F. Peng and Y. Ouyang. Optimal clustering of railroad track maintenance jobs. *Computer-Aided Civil and Infrastructure Engineering*, 29(4):235–247, 2014.

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

[127] A. Prekopa. On probabilistic constrained programming. In *Proceedings of the Princeton Symposium on Mathematical Programming*, pages 113–138. Princeton University Press Princeton, NJ, 1970.

[128] L.M. Quiroga and E. Schnieder. Monte Carlo simulation of railway track geometry deterioration and restoration. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 226(3):274–282, 2012.

[129] D.J. Rader. *Deterministic Operations Research: Models and Methods in Linear Optimization*. John Wiley & Sons, Inc., 2010.

[130] R. Rahmaniani, T.G. Crainic, M. Gendreau, and W. Rei. The Benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817, 2016.

[131] N. Ramkumar, P. Subramanian, T.T. Narendran, and K. Ganesh. Mixed integer linear programming model for multi-commodity multi-depot inventory routing problem. *OPSEARCH*, 49(4):413–429, December 2012.

[132] J.B. Rawlings and D.Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, Madison, Wisconsin, 2009.

[133] G. Reinelt. TSPLIB - a traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, 1991.

[134] R. Roberti and P. Toth. Models and algorithms for the asymmetric traveling salesman problem: an experimental comparison. *EURO Journal on Transportation and Logistics*, 1(1):113–133, 2012.

[135] R.T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–42, 2000.

[136] S. Røpke and J.F. Cordeau. Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3):267–286, 2009.

[137] HoHo Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, 1960.

[138] L.M. Rousseau, M. Gendreau, and D. Feillet. Interior point stabilization for column generation. *Operations Research Letters*, 35(5):660–668, 2007.

[139] A. Sadowska, P.J. van Overloop, C. Burt, and B. De Schutter. Hierarchical operation of water level controllers: formal analysis and application on a large scale irrigation canal. *Water Resources Management*, 28(14):4999–5019, 2014.

[140] J. Sandström and A. Ekberg. Predicting crack growth and risks of rail breaks due to wheel flat impacts in heavy haul operations. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, 223(2):153–161, 2009.

[141] R. Santos and P.F. Teixeira. Heuristic analysis of the effective range of a track tamping machine. *Journal of Infrastructure Systems*, 18(4):314–322, 2012.

[142] M. Savelsbergh, H. Waterer, M. Dall, and C. Moffiet. Possession assessment and capacity evaluation of the central Queensland coal network. *EURO Journal on Transportation and Logistics*, 4(1):139–173, 2015.

[143] M. W. P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):17–29, February 1995.

[144] G. Schildbach and M. Morari. Scenario-based model predictive control for multi-echelon supply chain management. *European Journal of Operational Research*, 252(2):540–549, 2016.

[145] A.T. Schwarm and M.Nikolaou. Chance-constrained model predictive control. *AIChE Journal*, 45(8):1743–1752, 1999.

[146] S. Sen and J.L. Higle. The c 3 theorem and a d 2 algorithm for large scale stochastic mixed-integer programming: set convexification. *Mathematical Programming*, 104(1):1–20, 2005.

[147] S. Sen and H.D. Sherali. Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106(2):203–223, 2006.

[148] A. Shapiro. Sample average approximation. In *Encyclopedia of Operations Research and Management Science*, pages 1350–1355. Springer, 2013.

[149] H. Shen and T. Başar. Optimal nonlinear pricing for a monopolistic network service provider with complete and incomplete information. *IEEE Journal on Selected Areas in Communications*, 25(6):1216–1223, 2007.

[150] H.D. Sherali and B.M.P. Fraticelli. A modification of Benders' decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse. *Journal of Global Optimization*, 22(1-4):319–342, 2002.

[151] H.D. Sherali and X. Zhu. On solving discrete two-stage stochastic programs having mixed-integer first-and second-stage variables. *Mathematical Programming*, 108(2):597–616, 2006.

[152] L.E. Sokoler, L. Standardi, K. Edlund, N.K. Poulsen, H. Madsen, and J.B. Jørgensen. A Dantzig–Wolfe decomposition algorithm for linear economic model predictive control of dynamically decoupled subsystems. *Journal of Process Control*, 24(8):1225–1236, 2014.

[153] Z. Song, T. Yamada, H. Shitara, and Y. Takemura. Detection of damage and crack in railhead by using eddy current testing. *Journal of Electromagnetic Analysis and Applications*, 3(12):546, 2011.

[154] K. Staňková, G.J. Olsder, and M.C.J. Bliemer. Bilevel optimal toll design problem solved by the inverse Stackelberg games approach. *Urban Transport and the Environment in the 21st Century (C.A. Brebbia, V. Dolezel, eds.)*, 89:871–880, 2006.

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

[156] O. Stein. *Bi-level Strategies in Semi-Infinite Programming*, volume 71. Springer Science & Business Media, 2013.

[157] K. Sundar and S. Rathinam. Generalized multiple depot traveling salesmen problem—polyhedral study and exact algorithm. *Computers & Operations Research*, 70:39 – 55, 2016.

[158] A.F. Timan. *Theory of Approximation of Functions of a Real Variable: International Series of Monographs on Pure and Applied Mathematics*, volume 34. Elsevier, 2014.

[159] P. Toth and D. Vigo. Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics*, 123(1–3):487–512, nov 2002.

[160] C. Vale and I.M. Ribeiro. Railway condition-based maintenance model with stochastic deterioration. *Journal of Civil Engineering and Management*, 20(5):686–692, 2014.

[161] H. van Ekeren, R.R. Negenborn, P.J. van Overloop, and B. De Schutter. Time-instant optimization for hybrid model predictive control of the Rhine–Meuse delta. *Journal of Hydroinformatics*, 15(2):271–292, 2013.

[162] F. Vanderbeck and L.A. Wolsey. Reformulation and decomposition of integer programs. In M. Jünger, T.M. Liebling, D. Naddef, G.L. Nemhauser, W.R. Pulleyblank, G. Reinelt, G. Rinaldi, and L.A. Wolsey, editors, *50 Years of Integer Programming 1958-2008*, chapter 13, pages 431–502. Springer, 2010.

[163] A. Vaz, E. Fernandes, and M. Gomes. A sequential quadratic programming with a dual parametrization approach to nonlinear semi-infinite programming. *Top*, 11(1):109–130, 2003.

[164] A. Vaz, E. Fernandes, and M. Gomes. SIPAMPL: Semi-infinite programming with AMPL. *ACM Transactions on Mathematical Software (TOMS)*, 30(1):47–61, 2004.

[165] H. von Stackelberg. *Marktform und Gleichgewicht*. J. Springer, 1934.

[166] M. Wen, R. Li, and K.B. Salling. Optimization of preventive condition-based tamping for railway tracks. *European Journal of Operational Research*, 252(2):455–465, 2016.

[167] H.P. Williams. *Model Building in Mathematical Programming*. John Wiley & Sons, 1993.

[168] R. Wong. Integer programming formulations of the traveling salesman problem. In *IEEE International Conference of Cirtuits and Computers*, pages 149–152, New York, USA, 1980.

[169] A. Zafra-Cabeza, J.M. Maestre, M.A. Ridao, E.F. Camacho, and L. Sánchez. Hierarchical distributed model predictive control for risk mitigation: An irrigation canal case study. *Journal of Process Control*, pages 787–799, 2011.

[170] S.-Y. Zhang. A nonlinear incentive strategy for multi-stage Stackelberg games with partial information. In *1986 25th IEEE Conference on Decision and Control*, pages 1352–1357. IEEE, 1986.

[171] T. Zhang, J. Andrews, and R. Wang. Optimal scheduling of track maintenance on a railway network. *Quality and Reliability Engineering International*, 29(2):285–297, 2013.

[172] X. Zhang, S. Grammatico, G. Schildbach, P. Goulart, and J. Lygeros. On the sample size of random convex programs with structured dependence on the uncertainty. *Automatica*, 60:182–188, 2015.

[173] Q. Zhu, H. Peng, and G.J. van Houtum. A condition-based maintenance policy for multi-component systems with a high maintenance setup cost. *OR Spectrum*, 37(4):1007–1035, 2015.

[174] A. Zoeteman. Life cycle cost analysis for managing rail infrastructure. *European Journal of Transport and Infrastructure Research EJTIR, 1 (4),* 2001.

[175] A. Zoeteman, Z. Li, and R. Dollevoet. Dutch research results in wheel rail interface management: 2001-2013 and beyond. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, 228(6):642–651, 2014.

TRAIL Thesis Series

The following list contains the most recent dissertations in the TRAIL Thesis Series. For a complete overview of more than 200 titles see the TRAIL website: www.rsTRAIL.nl.

The TRAIL Thesis Series is a series of the Netherlands TRAIL Research School on transport, infrastructure and logistics.

intermodal transport operations, T2017/7, June 2017, TRAIL Thesis Series, the Netherlands

Jamshidnejad, A., *Efficient Predictive Model-Based and Fuzzy Control for Green Urban Mobility*, T2017/6, June 2017, TRAIL Thesis Series, the Netherlands

Araghi, Y., *Consumer Heterogeneity, Transport and the Environment*, T2017/5, May 2017, TRAIL Thesis Series, the Netherlands

Kasraian Moghaddam, D.,*Transport Networks, Land Use and Travel Behaviour: A long term investigation*, T2017/4, May 2017, TRAIL Thesis Series, the Netherlands

Smits, E.-S., *Strategic Network Modelling for Passenger Transport Pricing*, T2017/3, May 2017, TRAIL Thesis Series, the Netherlands

Tasseron, G., *Bottom-Up Information Provision in Urban Parking: An in-depth analysis of impacts on parking dynamics*, T2017/2, March 2017, TRAIL Thesis Series, the Netherlands

Halim, R.A., *Strategic Modeling of Global Container Transport Networks: Exploring the future of port-hinterland and maritime container transport networks*, T2017/1, March 2017, TRAIL Thesis Series, the Netherlands

Olde Keizer, M.C.A., *Condition-Based Maintenance for Complex Systems: Coordinating maintenance and logistics planning for the process industries*,T2016/26, December 2016, TRAIL Thesis Series, the Netherlands

Zheng, H., *Coordination of Waterborn AGVs*, T2016/25, December 2016, TRAIL Thesis Series, the Netherlands

Yuan, K., *Capacity Drop on Freeways: Traffic dynamics, theory and Modeling*, T2016/24, December 2016, TRAIL Thesis Series, the Netherlands

Li, S., *Coordinated Planning of Inland Vessels for Large Seaports*, T2016/23, December 2016, TRAIL Thesis Series, the Netherlands

Berg, M. van den, *The Influence of Herding on Departure Choice in Case of Evacuation: Design and analysis of a serious gaming experimental set-up*, T2016/22, December 2016, TRAIL Thesis Series, the Netherlands

Luo, R., *Multi-Agent Control of urban Transportation Networks and of Hybrid Systems with Limited Information Sharing*, T2016/21, November 2016, TRAIL Thesis Series, the Netherlands

Campanella, M., *Microscopic Modelling of Walking Behavior*, T2016/20, November 2016, TRAIL Thesis Series, the Netherlands

Horst, M. van der, *Coordination in Hinterland Chains: An institutional analysis of port-related transport*, T2016/19, November 2016, TRAIL Thesis Series, the Netherlands

Beukenkamp, W., *Securing Safety: Resilience time as a hidden critical factor*, T2016/18, October 2016, TRAIL Thesis Series, the Netherlands

Summary

Maintenance Optimization for Railway Infrastructure Networks

Maintenance is crucial for the proper functioning and lifetime extension of a railway infrastructure network, which is composed of various infrastructures with different functions. In this thesis we develop robust and tractable model-based approaches for maintenance optimization of railway infrastructure networks. In addition, we develop a compact formulation for a variant of the multiple Traveling Salesman Problem (TSP), that can be applied to optimal scheduling of maintenance crews for a railway network, and a systematic numerical solution method for reverse Stackelberg game with incomplete information, which can be viewed as the framework for optimal maintenance contract design.

In the first part of the thesis, we consider condition-based maintenance planning for railway infrastructure networks. We focus on railway track defects, e.g. squats or ballast defects, and our aim is to develop a model-based, optimization-based approach that is tractable, scalable, robust, and non-conservative. First, we develop a centralized multi-level approach for maintenance planning of a single line of track divided into a small number of sections. Representative-scenario-based Model Predictive Control (MPC) is used at the high level to determine an optimal long-term component-wise intervention plan for the entire track line, and the Time Instant Optimization (TIO) approach is applied to transform the MPC optimization problem with both continuous and integer decision variables into a nonlinear continuous optimization problem. The middle-level problem considers the optimal allocation of time slots for the maintenance interventions determined at the high level to optimize the trade-off between traffic disruption and the setup cost of maintenance slots. Based on the high-level intervention plan, the low-level problem optimally groups individual defects to be treated by a maintenance agent, subject to the time limit imposed by the maintenance time slots.

Subsequently we develop a distributed multi-level approach for maintenance planning of large-scale railway networks. The Mixed Logical Dynamical (MLD) framework is adopted to model the hybrid deterioration dynamics, and a robust scenario-based MLD-MPC approach is developed at the high level to determine the section-wise interventions at each time step, considering various sources of uncertainty. Decomposition methods like Dantzig-Wolfe decomposition are used to divide the computational burden of the centralized MPC optimization problem among subproblems that are easier to solve. The optimal scheduling of the planned maintenance interventions to minimize the total setup costs, disruption costs, and travel costs of maintenance crews over the whole network is then formulated as a capacitated arc routing problem with flexible capacity at the low level and transformed into an equivalent node routing problem.

In the second part of the thesis, we consider the fixed-destination, multi-depot traveling

salesman problem, where several salesmen will start from different depots, and they are required to return to the depot they originated from. We propose a novel formulation for this problem using 2-index binary variables and node currents, and compare it to other 2-index formulations from the literature. This novel formulation requires less binary variables and continuous variables to formulate a problem, resulting in lower computation times. Using a large benchmark the effectiveness of the new formulation is demonstrated.
In the third part of the thesis, we propose a novel numerical solution approach for systematic computation of optimal nonlinear leader functions, also known as incentives, for reverse Stackelberg games with incomplete information and general, nonconcave utility functions. We apply basis function approximation to the class of nonlinear leader functions, and treat the incentive design problem as a standard semi-infinite programming problem. A numerical example is provided to illustrate the proposed solution approach and to demonstrate its efficiency.

Zhou Su

Samenvatting

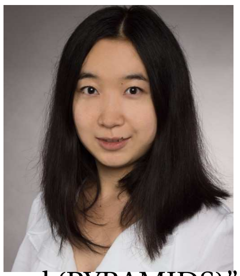
Onderhoudsoptimalisatie voor spoorwegnetwerken

Onderhoud is cruciaal voor het correct functioneren en het verlengen van de levensduur van een spoorwegnetwerk. Een spoorwerknetwerk bestaat uit verschillende onderdelen met verschillende functies. In deze dissertatie ontwikkelen wij een robuste en handelbare modelgebaseerde aanpak voor het optimaliseren van het onderhoud aan het spoorwegnetwerk. Tevens ontwikkelen wij een compacte formulering voor een variant op het handelsreizigersprobleem. Dit kan worden toegepast in het optimaal roosteren van onderhoudsteams voor spoorwegnetwerken. Verder ontwikkelen wij een systematische numerieke oplossingsmethode voor het omgekeerde Stackelberg spel met onvolledige informatie. Dit kan worden gezien als een raamwerk voor optimaal onderhoudscontractontwerp. In het eerste gedeelte van deze dissertatie bestuderen wij onderhoudsplanning voor spoorwegnetwerken. Wij richten ons op spoorwegdefecten zoals bijvoorbeeld druk of ballastdefecten. Ons doel is het ontwikkelen van een model- en optimalisatie-gebaseerde aanpak die handelbaar, schaalbaar, robuust en niet conservatief is. Eerst ontwikkelen wij een gecentraliseerde meerlaagsaanpak voor onderhoudsplanning van een enkele spoorweglijn die opgedeeld is in verscheidene kleinere sectoren. Representatieve-scenario-gebaseerde *Model Predictive Control (MPC)* wordt gebruikt op een hoger niveau om het lange termijn componentsgewijze interventieplan voor een gehele spoorlijn te optimaliseren. De tijdinstantie-optimalisatie aanpak wordt toegepast om *MCP* met continue en gehele beslissingsvariabelen te transformeren naar een niet-lineair continu optimalisatieprobleem. Het middennieveau probleem neemt de optimale allocatie van tijdsloten voor de onderhoudsinterventies in acht die op het hogere niveau worden bepaald om een afweging tussen verstoring in het verkeer en de onderhoudskosten te optimaliseren.
Vervolgens hebben wij een gedistribueerde meer-laags aanpak voor onderhoudsplanning voor grootschalige spoorwegnetwerken ontwikkelt. Het *Mixed Logical Dynamical (MLD)* raamwerk is aangepast om de hybriede verslechteringdynamica te modelleren en een robuuste scenario-gebaseerde MLD-MPC aanpak is ontwikkeld om, op een hoog niveau, de sectiegewijze interventie op elk tijdstap te bepalen terwijl verschillende soorten onzekerheid in acht worden genomen. Decompositiemethoden zoals de Dantzig-Wolfe decompositie zijn gebruikt om de berekeningskosten van het centrale MPC probleem te verdelen over sub problemen die gemakkelijker op te lossen zijn. De optimale roostering van de geplande onderhoudsinterventies die 1) de totale opstellingskosten 2) de verstoringskosten en 3) de reiskosten van het onderhoudsteam minimaliseren is daarna geformuleerd als een capacitated arc routing probleem met flexibele capaciteit op het lagere niveau. Daarna is deze formulering getransformeerd naar een gelijkwaardig *node routing* probleem.
In het tweede gedeelte van het proefschrift bestuderen wij het *fixed-destination, multi-depot*

traveling salesman probleem. In dit probleem starten meerdere handelsreizigers vanuit verschillende depots en zijn de handelsreizigers verplicht om terug te keren naar hun vertrekdepot. Hiervoor stellen wij een probleem formulering voor. Hierin maken wij gebruik van 2-index binaire variabelen en knoopstromen en wij vergelijken onze methode met andere 2-index formuleringen uit de literatuur. De voorgestelde formulering gebruikt minder binaire en continue variabelen om het probleem te formuleren. Dit resulteert in lagere berekeningstijden. De effectiviteit van de nieuwe aanpak is aangetoond op een groot *benchmark* probleem.
In het derde gedeelte van dit proefschrift stellen wij een nieuwe numerieke oplossingsmethode voor om systematisch de optimale niet-lineaire leider functies (ook bekend als *insentives*) te berekenen voor *Reverse Stackelberg Games* met incomplete informatie en in zijn algemeenheid niet concave gebruiksfuncties. Wij hebben basisfuncties benaderingen toegepast voor een klasse van niet-lineaire leiders functies en hebben het *insentive design* probleem als een standaard semi-oneindig programmeringsprobleem behandeld. Middels een numeriek voorbeeld is de voorgestelde aanpak aangetoond en is de efficiëntie van de methode aangetoond.

Zhou Su

Curriculum Vitae



Zhou Su was born in May 1989, Chongqing, China. She obtained her B.Eng degree in Automation in June 2010, from Huazhong University of Science and Technology, China. In September 2013, she received her M.Sc degree in Automation & Robotics from Technische Universität Dortmund, Germany.
In April 2014, She joined the Delft Center for Systems and Control as a PhD candidate, working on the NWO/ProRail project "Multi-party risk management and key performance indicator design at the whole system level (PYRAMIDS)", under the supervision of Prof.dr.ir. Bart De Schutter and Dr. Simone Baldi. She was a member of Unit DISC, the student representative body of Dutch Institute of Systems and Control, between 2015 and 2017.
Her PhD research has been focused on developing model-based approach for maintenance optimization for railway infrastructure networks. Her research interests include model predictive control, mixed-integer programming, and condition-based maintenance.