Local4Global

SYSTEM-OF-SYSTEMS THAT ACT LOCALLY FOR OPTIMIZING GLOBALLY

611538, FP7-ICT-2013.3.4

Deliverable D3.3

Extending Control System Theoretical Results to TSoS (2nd version)

Deliverable Version:	D3.3, v.2.1
Document Identifier:	local4global_wp3_d3.3_ETH_v2.1.pdf
Preparation Date:	August 15, 2015
Document Status:	Final
Author(s):	Paul Beuchat, Angelos Georghiou, John Lygeros and the Local4Global consortium
Dissemination Level:	PU - Public

Project funded by the European Community in the 7th Framework Programme

EU FP7 - SMALL/MEDIUM-SCALE FOCUSED RESEARCH PROJECT (STREP), FP7-ICT-2013.3.4: ADVANCED COMPUTING, EMBEDDED AND CONTROL SYSTEMS

D) FROM ANALYZING TO CONTROLLING BEHAVIOUR OF SYSTEM OF SYSTEMS (SOS)

Deliverable SUMMARY SHEET

Deliverable Details	
Type of Document:	Deliverable
Document Reference #:	D3.3
Title:	Extending Control System Theoretical Results to TSoS (2nd version)
Version Number:	2.1
Preparation Date:	August 15, 2015
Delivery Date:	August 15, 2015
Author(s):	Paul Beuchat, Angelos Georghiou, John Lygeros and the Local4Global consortium
Document Identifier:	local4global_wp3_d3.3_ETH_v2.1.pdf
Document Status:	Final
Dissemination Level:	PU - Public

Project Details	
Project Acronym:	Local4Global
Project Title:	SYSTEM-OF-SYSTEMS THAT ACT LOCALLY FOR OPTIMIZING GLOBALLY
Project Number:	611538
Call Identifier:	FP7-ICT-2013.3.4
Call Theme:	ADVANCED COMPUTING, EMBEDDED AND CONTROL SYSTEMS
Project Coordinator:	CERTH – Centre for Research and Technology – Hellas
Participating Partners:	CERTH – Centre for Research and Technology – Hellas; ETHZ – Eidgenössische Technische Hochschule Zürich; RWTH – RWTH Aachen University; IK4 – IK4 TEKNIKER; TRV – TRANSVER GmbH; TUC – Technical University of Crete; TUM – Technische Universtität Muenchen;
Instrument:	STREP
Contract Start Date:	October 1, 2013
Duration:	36 Months

Deliverable D3.3: Short Description

This document presents the theorical tools which are going to be used in the L4G project. The combine aim of this deliverable is to summarise and extend notions from optimization theory that will be later used to address the two case-studies. All the theory presented are in line with the work carried out by the consortium partners in the Centre for Research and Technology and the Technical University of Crete.

Keywords: TSoS, L4G platform, optimization problems, dynamic programming, decision making under uncertainty, decision rules, distributed optimization algorithms

Deliverable D3.3: Revision History

Version:	Date:	Status:	Comments
0.1	15/03/2014	Draft	Initial draft, Binary decision rules
0.2	15/05/2014	Draft	Second draft, Approximated dynamic programming
0.3	10/07/2014	Draft	Third draft, (Sofia Antipolis review meeting) Distributed optimization under uncertainty
0.4	15/09/2014	Draft	Completed draft
0.5	08/10/2014	Draft	Internal review (Eibar meeting)
1.0	28/10/2014	Submitted	ETHZ final vesion
2.0	28/07/2015	Draft	Internal review
2.1	15/08/2015	Final	Final version, revised based on Reviewer and Commission comments

Table Of Contents

Executive Summary

Technical System of Systems (TSoS) is a term used to characterise the structure of the underline system. In this setting, a collection of individual systems are pooled together, sharing resources and capabilities, to create a new more complex entity offering greater functionality and performance than individual constituent systems. The aim of this deliverable, is to provide the mathematical underpinning that will allow to rigorously formulate such Technical System of Systems, and demonstrate their capabilities in subsequent deliverables in two practical applications: building control and traffic management. In this deliverable we will demonstrate that optimization can provide the mathematical framework to achieve this goal.

Optimization algorithms have been successfully applied in a variety of areas such as engineering, operations management and finance. However, most algorithms need to be executed centrally and do not take into account the structure of the problem in hand. In recent years, decentralized optimization has gained great interest, due to its wide applicability in areas such as distributed control, wireless communications and power systems. In this setting, the problem has a network structure, where each node can be seen as an individual agent. The agents have limited access to information about their neighbours, but collectively try to optimize a global objective in the absence of a central authority. These problems can be addressed through distributed optimization algorithms where each player solves its own optimization problem relying on local information, and communicates only minimal information with the neighbouring nodes.

The current theory of distributed optimization is limited to static problem where all parameters are precisely known and involve only continuous decisions. In contrast, most practical problems are highly uncertain and dynamic in nature, involving both continuous and discrete decisions. Moreover, current distributed optimization algorithms require full knowledge of the underline mathematical model, which is rarely available in practice. These drawbacks restrict the use of distributed optimization in practical applications. In this deliverable, we aim to summarise and expand the applicability of distributed optimization algorithms to cover dynamic decision problems involving both real-valued and integer

decisions, and model-free distributed control.

The most efficient and effective control design algorithms require a mathematical formulation for the physical model in hand. These mathematical formulations allows to incorporate efficient optimization algorithms in the control design process. If the mathematical formulation exactly matches the physical system, then optimal control designs can be achieved. On the other hand, it is in general a difficult task to extract the exact mathematical model from the underline dynamics of the physical system. To this end, a large effort is made by the control community in developing model identification techniques. Even though the mathematical model extracted from the identification of the physical model might not be 100% accurate, in many practical cases the, designing a controller based on this approximate mathematical model can perform adequately in practice.

A different avenue in control design is to use model-free optimization techniques. These techniques combine the identification process and the control design in the same algorithm, avoiding the need to create a mathematical model. These algorithms iteratively learn the underline dynamics by implementing a sequence of control policies, many of which can be suboptimal. Nevertheless, this has the advantage that it can be used in cases where the dynamics of the problem cannot be easily described and suboptimal control designs can be tolerated until a near optimal design is achieved. Of course, this flexibility cames with a substantial cost: implement a sequence of suboptimal controls will lead to substantial losses in performance until a near optimal controller is achieved.

To bridge the gap between model-based and model-free control design, we propose to use the two techniques in a unified framework. To this end, we will use model-based techniques to design control inputs for the two case studies, by first formulating simple models for the dynamics of the physical system. This will provide easy to compute control designs using optimization techniques. The control design can then be further improved by using online model-free techniques. The proposed framework benefits from the efficient near-optimal designs of model-based techniques and the flexibility of model-free techniques, while avoiding the substantial losses in performance until a near optimal controller is achieved.

In this deliverable, we address the following:

- We give a precise characterization of the optimization problems considered. This includes the characterization of the general class of the optimization problems and a detailed description of the problem statement. Emphasis is put on the distinction between Local versus Global control, and the structural difference this induces to the underline optimization problems.

- We discuss concepts from approximate dynamic programming, which will constitute the main tool for model-free control. This is done in two parts: (i) We give an overview of approximate dynamic programming techniques in problem instances where the underline mathematical formulation of the problem is given, and (ii), we give an overview of model-free approximate dynamic programming techniques typically used when the dynamics of the underline problem are not precisely known or the parameters of the mathematical model cannot be accurately calibrated. This work is in line with the Cognitive-based Adaptive Optimization (CAO), and (PCAO) algorithms proposed by the consortium partners in the Centre for Research and Technology [3, 38, 46].

- We discuss and develop concepts for multistage decision making under uncertainty involving both real-valued and discrete decisions. To this end, we extend the framework of the functional approximation known as *decision rules* to cover integer recourse decisions. This is a novel research in the area of adaptive control that promises to deliver fast and reliable control designs for discrete decisions that can be used in applications such as active traffic management and building control.

- We discuss concepts from distributed optimization algorithms with special emphasis on the Alternating Direction Method of Multipliers. We present the basic concepts, and we demonstrate how this can be used in conjunction with the decision rule approximation to solve distributed optimization problems under uncertainty.

The combine aim of this deliverable is to summarise and extend notions from optimization theory that will be later used to address the two case-studies. All the theory presented are in line with the work carried out by the consortium partners in the Centre for Research and Technology and the Technical University of Crete.

1 Introduction

1.1 Description of Deliverable 3.3

Deliverable 3.3 of the Local4Global project is the direct result of completing Task 3.3, which is a part of Work Package 3. The title of Work Package 3 is “TSoS Analysis and Modelling tools" and it has the objective to provide the tools and components of the Local4Global. Within this work package Task 3.3 is specifically aimed at providing tools for analysing the properties of generic TSoS, captured in the title “Optimality, Certification, and Stability in TSoS".

This deliverable provides a description of a methodology that can be used to extend control system analysis to TSoS. When adopted within the Local4Global framework, this methodology will provide analysis tools that quantitatively measure the stability, robustness and optimality of TSoS.

1.2 Description of Task 3.3

The aim of Task 3.3 is two-fold:

1. To extend notions from standard control theory such as stability, feasibility and the degree of sub-optimality to TSoS. Extension of notions of optimal control, Lyapunov functions, certification, etc to hybrid systems and networked control systems developed by Local4Global. The past work of of some Local4Global partners serves as one starting point from which to perform such an extension.

2. As a second objective in this task will be to develop a mechanism for determining when action by the optimisation algorithm is needed in the TSoS context. We will develop a methodology of event-based optimisation, where the optimisation algorithms interacting with the low level controllers take the “back seat" and intervene only whenever necessary. Motivated by the Use Case studies, we will investigate criteria for generating the events to trigger the actions of the optimisation algorithms.

The combined aim of this task is to minimise the communication and coordination necessary but still preserve fundamental system theoretic properties of the closed loop system, in particular ensure stability and provide bounds on the sub-optimality of the resulting solution when compared with running the optimisation continuously (as is done, for example, in traditional model predictive control).

2 Problem Statement

2.1 Introduction

This chapter states the different problem classes that are commonly encountered in the control and optimization literature. Methods and algorithms developed usually only apply to one specific problem class and it can be non-trivial to extend them to a different problem class [44]. Section 2.2 presents the most common problem classes, defining the notation and terminology that is used throughout the report.

Experience has shown that successful control design is usually achieved by matching the method to the problem at hand [45] [48]. In the context of Local4Global, Section 2.3 discusses how the goal of designing *Systems-of-Systems that act locally for optimizing globally* is expressed rigourously in control and optimisation terminology. This allows us to specify the problem class for a particular TSoS and focus on extending existing algorithms for that problem class to be applicable in a TSoS setting.

2.2 Problem Classes

This report considers the following general problem statement, given in discrete time,

$$\begin{aligned} \min_{\pi_t} \quad & L(x_1, \ldots, x_T, u_1, \ldots, u_T) \\ \text{s.t.} \quad & (x_t, u_t) \in (\mathcal{X}_t \times \mathcal{U}_t), \quad \forall \xi \in \Xi, \quad \forall\, t \\ & x_{t+1} = f_t(x_t, u_t, \xi_t), \quad \forall t = 1, \ldots, T \\ & u_t = \pi_t(x_t) \end{aligned} \tag{1}$$

where x_t and u_t are the state and input respectively at time t, f_t is the dynamics of the system describing the evolution of the state, $(\mathcal{X}_t \times \mathcal{U}_t)$ describes the state-by-input constraint sets, $L(x_1, \ldots, x_T, u_1, \ldots, u_T)$ is the objective function to be minimised and ξ is an uncertain parameter that can influence the evolution of the system and it is assumed to come from an uncertainty set described by Ξ. The optimisation variable, π_t, is the control policy for computing the input u_t, at a given time t, as a function of the state x_t. For the

dynamics, constraints and control policy, the subscript t indicates that they can be different at each time step.

Equation (1) is a very general optimisation formulation and many problems can be written in this form, but it is intractable to solve. Each part of the problem statement (i.e. objective function, constraints, dynamics, uncertainty description) can take on a variety of restricted forms, which are defined in the following subsection. A specific problem class is specified by the restriction applied to each part of the problem statement.

2.2.1 Objective Function Types

The general problem statement, equation (1), is a multistage problem where the time horizon T, could be either finite or infinite. In both cases the total cost for the time horizon is commonly assumed to be cumulative and takes one of the following forms:

- Finite horizon

$$\sum_{t=0}^{T} \mathbb{E}_{\xi} \left[l_t(x_t, u_t) \right] \tag{2}$$

  where $l_t(x_t, u_t)$ is the stage cost at time t for being in state x_t and applying input u_t, T is the finite time horizon length over which the cost it to be minimised. Solving the optimisation with this objective function aims to find the state-input trajectory that minimises the total cost over the time horizon.

- Discounted infinite horizon

$$\sum_{t=0}^{\infty} \mathbb{E}_{\xi} \left[\gamma^t l(x_t, u_t) \right] \tag{3}$$

  where $\gamma \in [0, 1)$ is the discount factor. Solving the optimisation problem with this objective function usually requires the assumption that the dynamics, stage cost, constraints, and policy are time-invariant (i.e. $f_t = f$, $(\mathcal{X}_t \times \mathcal{U}_t) = (\mathcal{X} \times \mathcal{U})$, and $\pi_t = \pi \; \forall \, t$), and aims to minimises near term costs, dependent on the choice of γ.

- Average Cost

$$\lim_{T \to \infty} \frac{1}{T+1} \sum_{t=0}^{T} \mathbb{E}_{\xi} \left[\gamma^t l(x_t, u_t) \right] \tag{4}$$

where solving the optimisation problem with this cost fucntion requires the same assumption of time-invariant dynamics, stages cost, constraints and policy. This formulation aims to the minimise the cost at every step of the future.

The expectation $\mathbb{E}_\xi[\cdot]$ appearing in the three objective function forms is taken with respect to the random parameter ξ and therefore describes an objective of minimising the average cost in the face of uncertainty. It is possible to replace the expectation with other functions, such as a worse case (min-max") formulation which aims to minimise the maximum case cost that can occur for all possible realisation of the uncertainty $\xi \in \Xi$. We consider only the expected cost formulation in this report.

Many optimisation algorithms also require the stage cost $l_t(x_t, u_t)$ to have a certain function form. The most common forms are linear, quadratic or polynomial stage costs.

2.2.2 State, Input and Distrubance Spaces and Constraint Types

The first constraint in equation (1) is the most general robust constraint where at every time step $(x_t, u_t) \in (\mathcal{X}_t \times \mathcal{U}_t)$ is required to hold true for all uncertainty realisations ξ that could occur from the specified uncertainty set Ξ. It is also reasonable to consider a further restriction where the constraint set is separable, i.e. $x_t \in \mathcal{X}_t$, $\forall \xi \in \Xi$ and $u_t \in \mathcal{U}_t$, $\forall \xi \in \Xi$. It is also common to let either one or both the state and input spaces be unconstrained.

The robustness of the stochastic constraint can be relaxed to a chance constraints where the constraint of the form

$$\mathbb{P}_\xi \left[(x_t, u_t) \in (\mathcal{X}_t \times \mathcal{U}_t)\right] > 1 - \epsilon \tag{5}$$

where ϵ is a predefined probability criteria.

The state, input and disturbance space can be either finite or continuous. In the finite case, the set of allowable states and inputs is specified at each time step. The constraints are encoded in the definition of allowable sets.

In the case of continuous spaces, the form of the constraint and disturbance sets influences the optimisation algorithms applicable for solving the problem. For example, the constraint

and disturbance set could be any combination of a polytopic description, an elliptical description, a hyper-rectangle set or a box set, each of which should be treated differently when solving the problem.

2.2.3 Dynamics

If a model of the dynamics is required for a particular algorithm, then they are required to have a certain functional form. Similar to the stage cost, the most common forms are linear, quadratic or polynomial dynamics.

2.3 System-of-systems Problem Class

The goal of the Local4Global algorithm has 3 main points: it is applied to *System-of-Sytems*, each system only *acts locally*, and the cumulative effect of the control actions take by each participant is harmonised in such a way that it *optimises globally*. These 3 points are discussed in the following subsections. They are important to understand for making fair comparisons between theoretical results and algorithms that have been extended to System-of-Systems.

2.3.1 A System-of-Systems

There is not yet a unified agreement on the definition of a *System-of-Systems* (SoS), but the Description of Work and deliverable D2.1 of the Local4Global project gives the required clarity. The distinction is made between a Technical SoS and a Natural SoS, where the former is the one for which the Local4Global project is researching.

A TSoS is defined as been comprised of a large collection of synthetic, human-designed, constituent components, which have the capability to make decisions and act autonomously. To keep the language precise in this report, each constituent component will be referred to as a sub-system and the collection makes up the TSoS.

The coupling between sub-systems of a TSoS is one of the primary reasons that the dynamics of a TSoS are complex and the reason for emergent behaviours. A TSoS is usually coupled in the physical dynamics, but could also be coupled via the cost function. In the absence

of any coupling, then each sub-system can be considered independently, optimised locally, and it is not a TSoS.

2.3.2 Control - Local vs. Global

The control policy $\pi_t(x_t)$, specifies the action u_t, to be taken as a function of the available information x_t. Considering the combined TSoS, the full set of information is the combined set of all information available to all sub-systems. By definition of a TSoS, each individual control action is local to the sub-system, i.e. in the traffic management use case, each intersection is a sub-systems with the cycle times as its control action, it is clear that one intersection cannot set the cycle times of another.

It is possible that the control policy could be designed to depend on the full set of information available from all sub-systems, or any subset of the information. A truly local controller is one for which the actions taken by the sub-system are only a function of the information collected by that sub-system. If TSoS is controlled by only local controllers, then this is commonly referred to as *decentralised* control.

When there is a communication network for transferring information between sub-systems, this means the actions taken by the sub-system can be a function of the additional information it receives from those sub-systems it is connected to. This is referred to as *distributed* control and every arrangement of the communication network represents a different problem. When every sub-system has access to the full set of information then it is referred to as *centralised control*.

It is important to distinguish the control regime from how the controller was designed. It is possible to perform a global optimisation centrally where a single controller is designed for the TSoS, but the optimisation forces the controller to have a structure that can be implemented in a decentralised fashion.

2.3.3 Objective Function - Global vs. Local

In order to *optimise globally* the objective function needs to be set globally. As the global objective function will generally be a function of the states of all the participants

For the building control use case, the typical trade off is between energy consumption and occupancy comfort level. For the same technology installed, the comfort level can be improved at the cost of increase energy usage, or the energy consumption can be decreased at the expense of reducing the comfort level. As single value objective functions are consider for this project, this trade-off is expressed as a weighting factor specifying the relative importance of energy consumption versus comfort level.

In contrast, if each local system is allowed to choose its own objective function then the problem changes significantly. For example consider an apartment building where each sub-system pays for its own energy consumption and can choose its own desired comfort level. A reasonable global objective in this case is to minimise the total energy consumption of the building, with the only significant control action been the energy prices. This formulation is a game: how should the energy pricing be set so that the occupants of each apartment act in the desired way.

2.3.4 Optimisation - Local vs. Global

Similar to description above for controllers, optimisation algorithms can be broadly classified as centralised, or decentralised. In centralised optimisation, all the available problem data is collected into one problem formulation and this is solved via an applicable algorithm depending on the problem type. An algorithm can be considered a decentralised optimisation if the bulk of the optimisation work can be perform by solving local optimisation problem on each sub-system and coordinating the results on a global level. Chapter 5 expands on this definition and explains some of the system requirements for decentralised optimisation methods to be applicable.

2.3.5 Model-free versus Model-based

A model based approach means that the objective function, dynamics equations and constraint sets in the problem statement are known and are utilised as part of solving for a controller. In model-free approach all or some of this detail is not known and needs to be inferred from collected data. The data in a model-free approach could come for

a simulation-model of the system, in which case we refer to it as a simulation-based approach. Alternatively, the data for driving a model-free approach can be collected from the real system as time progresses.

The benefit of model-based or simulation-based approaches is that the computations for finding a controller can be completed offline. However in both approaches if the real system does not match the model or the simulation then the controller can perform quiet poorly. Further, building and verification of a simulation-model is labour intensive and requires constant tuning to maintain accuracy.

The Local4Global project aims at developing model-free algorithms that used data collected from the system in real-time. The potential benefit of a model-free algorithm is that it can be applied in a “plug-and-play" fashion to any system, thus reducing significantly the development, commissioning and maintenance costs. A model-free approach could either implicitly learn the unknown dynamics, cost functions, constraints of the system, in which case it learns the mapping from the available measurement data to the controller actions. Alternatively, the unknown parts could be explicitly learned from the measurement data, for example, with a “plug-and-play" estimator, and then model-based approaches can be used to design the controller based on the estimated elements.

2.4 Comparisons

The descriptions above explain that comparing the performance of a decentralised versus centralised controller, both solved for via a centralised optimisation, will give an indication of the objective value improvement that can be gained by enabling communication. In contrast, if the controller is fixed to be decentralised and a comparison is made between the performance when the problem is solved via a decentralised versus centralised optimisation method, this will give an indication of the sub-optimality of the decentralised optimisation method.

As the Local4Global approach aims for no additional modelling to be performed or infrastructure to be installed, the allowable controller structure is inherently defined as part of the problem data. The focus for comparing algorithms and theoretical results will be on

comparing optimisation techniques.

The *"P Cognative-based Adaptive Optimisation"* (PCAO) algorithm was developed by the consortium partners in the Centre for Research and Technology [3]. The PCAO algorithm is aimed at finding a centralised controller via a centralised optimisation algorithm without any knowledge of the system dynamics, i.e. it is "model-free". This report takes some of the core principles behind the PCAO algorithm and reviews how they can be extended to TSoS where the controller is inherently decentralised or distributed.

3 Approximate Dynamic Programming

3.1 Introduction

The name Dynamic Programming was coined by Richard Bellman [6] and is a field of study that has been active since the 1950's. Dynamic programming is an optimal control algorithm for solving multi-stage problems. It is based on the principle of optimality as stated by Bellman [5]:

> *"An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."*

Phrased another way, if a policy is optimal from time t forward, then its sub-policy from time $t+1$ forward must also be optimal for the problem starting from $t+1$. The dynamic programming algorithms to find such an optimal policy suffer from the curse of dimensionality (explained in section 3.2 below) and are hence not applicable to the large-scale type of problems to be considered in the Local4Global framework.

Approximate Dynamic Programming (ADP) is a field of research where methods and algorithms are adapted from the principles of Dynamic Programming to remove the curse of dimensionality by making some approximations. This tractability generally comes at the expense of introducing a degree of sub-optimality into the solution.

A wide range of approaches fall under the umbrella of ADP, a few of which as discussed in detail in the subsequent sections. Accompanying the range of approaches there is also a range of names which can be considered synonymous to ADP. Wang et al. [59] and the references therein provide an extensive list for the synonyms of ADP, and the subtleties of each.

Model-based APD approaches are described first, in Section 3.3, because it simplifies the introduction of ADP as a solution method. Section 3.4 then explains a few of the model-free ADP approaches that have been investigated.

3.2 The Curse of Dimensionality

Consider the problem classes presented in Chapter 2, when the problem is in discrete time then the key step of any dynamic programming algorithm is to solve Bellman's Equation. For a finite horizon formulation, Bellman's equation is

$$V_t(x) = \left(\min_{u_t \in \mathcal{U}_t} l_t(x, u_t) + \mathbb{E}_\xi \Big[V_{t+1}\big(f_t(x, u_t, \xi)\big) \Big] \right), \quad \forall x \in \mathcal{X}_t \tag{6}$$

where V_t is the optimal cost-to-go function (a.k.a. the Value function) at time t and all other symbols were defined in the general problem statement presented in Chapter 2. Evaluated for a given state, $V_t(x)$ is the expected cost from time t until the end of the horizon T if the optimal control policy, given by the arg min of the RHS, is played at each time step. A DP approach to solve the finite horizon problem thus requires recursively solving Bellman's equation from $t = T - 1$ back to $t = 0$. This gives the optimal cost-to-go from the current time $V_0(x)$, and the optimal control policy to play at each step of the horizon.

For an infinite horizon formulation, with invariant dynamics, constraints and stage-cost, Bellman's equation is

$$V(x) = \left(\min_{u \in \mathcal{U}} l(x, u) + \gamma \mathbb{E}_\xi \Big[V\big(f(x, u, \xi)\big) \Big] \right), \quad \forall x \in \mathcal{X} \tag{7}$$

where the main difference is that the Value function $V(x)$ is the same on both side of the equation, the policy is also time-invariant, and the discount factor γ has appeared on the RHS.

The RHS of Bellman's equation is often notationally simplified to the operator $\mathcal{T}_t$ in the finite horizon case and $\mathcal{T}$ in the infinite horizon case. Both are referred to as the *Bellman Operator*.

Although easy to state, the Bellman equation becomes intractable to solve for large, complex systems. To see this consider a problem with finite state, input and disturbance spaces, where N_x, N_u and N_ξ denotes the cardinality of each space respectively. A naive implementation of DP would enumerate over all possible state-input-disturbance combinations which is of order $\mathcal{O}(N_x N_u N_\xi)$ per time step. With today's computational standard, this

brute force approach is tractable up about $\mathcal{O}(N_x N_u N_\xi) \approx 10^8$ [61]. Beyond this magnitude, or if any of the spaces are continuous, solving the Bellman equation is intractable.

The exact Dynamic Programming algorithm can be formulated as a linear program (LP), a connection made by Manne [40], but this LP still suffers from the curse of dimensionality. Some other algorithms can solve the exact problem more efficiently, but none so far have overcome the curse of dimensionality. The well studied exception is the Linear Quadratic Regulator (LQR) with Linear dynamics function f, Quadratic stage cost l, and unconstrained, continuous state and input spaces, and infinite time horizon.

The systems being considered for the Local4Global project have complex dynamics and constraint functions and hence fall into the category of being intractable to solve with exact DP methods.

3.3 Model-Based Value Function Approximation

Considering a discrete time, finite horizon problem where the optimal value function has been solved for via the Bellman Equation, (7) above, the optimal input to apply at time t based on knowledge of the current state x_t is found by the following

$$u_t^\star = \arg\min_{u_t \in \mathcal{U}_t} l(x_t, u_t) + \mathbb{E}_\xi \Big[V_{t+1}\big(f(x_t, u_t, \xi)\big) \Big] \tag{8}$$

where the model-information, f and l, is explicitly required to evaluate the solution of this optimisation problem. As the optimisation above enforces the input constraints explicitly, V can be replaced by any Approximate value function $\hat{V}$ and the above optimisation will still give a feasible control action u_t, though likely sub-optimal. This concept motivates the idea behind the term *Value Function Approximation*, which aims to formulate an optimisation problem to solve for a $\hat{V}$ that closely approximates the true optimal Value function V.

A common approach, introduced by Schweitzer and Seidmann [51], is to constructed an approximate Value function as a linear combination of a set of basis functions

$$\hat{V}(x) = \sum_{i=1}^{K} \alpha_i \, \hat{V}^{(i)}(x) \tag{9}$$

where the $\hat{V}^{(i)}(x)$ are the set of K basis functions that are chosen in advance and $\alpha_i \in \mathbb{R}$ are the weightings. In applying this basis function approach to solve the Bellman equation approximately, Schweitzer and Seidmann [51] formulate an LP to solve for the "best" weightings α_i. This reduces the dimensionality of the problem from the number of state combinations down to the number of basis functions, thus significantly improving the tractability.

The following subsection discuss: providing bounds on how close the solution will be to the true optimal, defining "best" in terms of formulating the LP for the weightings, the choice of basis functions and complications with enforcing state constraints.

3.3.1 Lower Bounding Approximations

Between Schweitzer and Seidmann [1985] [51] and de Farias and Roy [2003] [29] there were many papers published and examples of successful ADP implementations. A particularly iconic example being the backgammon playing algorithm by Tesauro [1992] [56] that achieved a strong intermediate level of performance, surpassing other backgammon playing algorithms at the time. The variety of methods generally include tuning parameters that require extensive trail and error tuning to achieve good results for each particular application.

In the work of de Farias and Roy [29] and Wang et al. [61] a slightly modified approach is used to solve for a basis function approximation which is guaranteed to lower bound the optimal solution. de Farias and Roy [29] first improved the theoretical guarantees of the basis function approach by leveraging the monotinicity property of the Bellman operator. The authors start from a formulation equivalent to the following

$$\begin{aligned} \max_{\alpha} \quad & \int_{\mathcal{X}} c(x)\hat{V}(x) \\ \text{s.t.} \quad & \hat{V}(x) \leq (\mathcal{T}\hat{V})(x), \qquad \forall\, x \in \mathcal{X} \\ & \hat{V}(x) = \sum_{i=1}^{K} \alpha_i \, \hat{V}^{(i)}(x) \end{aligned} \tag{10}$$

where $\alpha = [\alpha_1, \ldots, \alpha_K]^T \in \mathbb{R}^K$ is the vector of basis function weightings and is the decision variable in the optimisation problem. The function $c(\cdot)$ is called the *state relevance weighting*

and it is explained in more detail in the subsection below, it is a design choice of this ADP method.

The constraint in this formulation is referred to as *Bellman's Inequality* and any feasible $\hat{V}$ satisfies $\hat{V} \leq V$, thus giving a lower bound to (under estimator of) the optimal cost-to-go function. The goal is to find the best under estimator $\hat{V} \leq V$, but the constraint in equation (10) excludes many feasible under estimators. Hence, in Wang et al. [61] the authors increase the set of feasible $\hat{V}$ which are under estimators, allowing them to give an improved lower bound. This increase in the feasible set is achieved by replacing Bellman's Inequality with an iterated version

$$\hat{V}(x) \leq (\mathcal{T}^M \hat{V})(x), \qquad \forall\, x \in \mathcal{X} \tag{11}$$

where M is the number of iterations.

The iterated bellman inequality approach can be applied to both infinite and finite horizon problems and cannot perform worse than the non-iterated approach. In the infinite case the choice of the number of iterations is essentially a case of increasing M until either no more improvement is gained or computational limits are reached.

To provide some intuition, each basis function $\hat{V}^{(i)}$ can be thought of as a feature [48], and its weighting α_i gives an indication of how important that feature is to representing the optimal V.

In the examples of [61], they consider the subspace of quadratic functions as a basis. The work of Summers et al. [55] leverages the same iterated bellman inequality and use polynomials of a given degree as basis functions. When the systems has a polynomial description for the dynamics, cost function, and constraints, then optimisation problem (10) can be reformulated as a Semi-Definite Program (SDP) and solved for the co-efficient of the polynomial Value Function approximation. In this way the authors have reformulated the problem to consider all quadratic functions (resp. all polynomials of a given degree) as candidates for the "best" Value function approximation.

This formulation gives a lower bound for the optimal objective function from the original problem, finding an upper bound for the optimal solution requires Monte Carlo simulation.

This involves using the "best" Value function approximation that was solved for via equation (10), selecting a large number of initial conditions x_0, evolving the system from each x_0 for a number of different scenarios and computing the expected value of the resulting costs.

The relevance of a lower bounding approach for the Local4Global concept, is that is can be used as part of an event based methodology. Using the lower bound, the degree of sub-optimality can be computed for the current operating point and when the sub-optimality exceeds a certain limit this can trigger an event to re-optimize the low level controllers.

3.3.2 Formulating for the "best" weightings

The "best" weightings are the α_i's that achieve the optimal solution of problem (10) for a given set of basis functions. The state relevance weighting, $c(x)$, in the objective of (10) is used in both [29] and [55], and it plays a crucial role in determining the weightings.

All $\hat{V}$'s that are feasible for problem (10) are under estimators of the optimal V. In the objective, $\max \int_{\mathcal{X}} c(x)\hat{V}(x)$, the state relevance weighting $c(x)$ specifies which region of the state space is important for maximising $\hat{V}(x)$. Hence the approximate Value function $\hat{V}$ found by optimisation problem (10) is the best fit to the optimal V on that region. Outside of the region encoded by $c(x)$ the fit could be quiet poor. If $c(x)$ is chosen to give a high relevance to a region of the state space that is rarely visited, then the quality of the solution could be quiet poor.

For a thorough discussion on the importance of the state relevance weighting, see [29, Section 3]. There the authors provide a theorem which is interpreted to say that when the value function approximation $\hat{V} = \sum_{i=1}^{K} \alpha_i \hat{V}^{(i)}$ is close to the optimal Value function, then the policy generated by using $\hat{V}$ is also close the the optimal policy.

In Wang et al. [61] the state relevance weighting is not explicitly used. Instead the authors choose $\mathbb{E}_x[\hat{V}(x)]$ as the objective function of (10), which essentially corresponds to a particular choice of c. The tightest lower bounds presented in the numerical results of Wang et al. [61] actually consider solving problem 10 for a modest number of L different state relevance weighting, giving a set of L lower bounding Approximate Value functions. The Approximate Value function to be used is then computed as the following point-wise

maximum

$$\hat{V}(x) = \max_{i=1,\dots,L} \hat{V}_j(x) \tag{12}$$

where each $\hat{V}_j$ is the solution to problem (10) with differing state relevance weighting and possibly even different basis function. This $\hat{V}$ can be readily used in Monte Carlo simulation to evaluate $\mathbb{E}_x[\hat{V}(x)]$ because the point-wise maximum is relatively cheap to compute.

Note, when solving the exact Dynamic Programming problem, the state-relevance weighting has no influence on the results. To see this, consider that exact Dynamic Programming problem takes the form

$$\begin{aligned} \max_{V} \quad & \int_{\mathcal{X}} c(x)V(x) \\ \text{s.t.} \quad & V(x) = (\mathcal{T}V)(x), \qquad \forall\, x \in \mathcal{X} \end{aligned} \tag{13}$$

where the maximisation is over all general functions $V(x)$, and the constraint is exactly Bellman's equation which is know to have a unique solution. Therefore the only feasible Value function for this problem is the optimal Value function and hence the state relevance weighting does not affect the solution.

3.3.3 Choosing Basis Functions

The approach described above for approximating the Value Function by a linear combination of basis functions assumes that a set of K basis functions ($\{\hat{V}^{(i)}\}_{i=1}^{K}$) are already given. Solving for the best value function approximation by this approach can be considered as a projection of the optimal value function onto the space spanned by the basis functions. Generally, the closer the span of the basis function is to the true optimal value function, the higher quality the solution will be. All the references in this section indicate that the set of basis functions is a design choice and needs to be made appropriately for the individual problem at hand.

In [37] the authors consider systems with input-affine dynamics, convex stage costs and convex constraints. They use quadratic basis function as a basis for their Value Function approximations. Although no theoretical guarantees are given, it is suggested that good

performance can be achieved in practice by using quadratic value function approximations. One numerical example they present in [37] is for control of an input-constrained linear-quadratic system. The optimal solution for such a system is known to be piecewise quadratic (TODO - find a reference stating this), thus a quadratic basis for $\hat{V}$ seems a reasonable choice.

In general it would seem appropriate to use any available knowledge about the system in designing the set of basis functions.

3.3.4 Imposing State Constraints

Equation (8) states how the input action u_t should be computed for a given state x_t and approximate Value Function $\hat{V}$ substituted in for the optimal V. It is clear that the input constraint $u_t \in \mathcal{U}_t$ is enforced by the optimisation and the computed input action will be feasible. Consider a system that also has state constraints. Even for the simplest case of time-invariant state constraints that are decoupled from the input constraints, it is not clear how to enforce the state constraint in the ADP setting so that the state-constraint is respected at every step of the time horizon.

In the exact DP approach, the stage cost is defined to be infinite for states lying outside the state constraint. Thus in evaluating the DP recursion steps, the true optimal Value Function also takes on infinite values for those states for which there does not exist an input that can bring the system into the feasible state-space for all realisations of the uncertainty:

$$V_t(x_t) = \infty \qquad \text{if} \qquad \nexists u_i \in \mathcal{U}_t \; : \; f_t(x_t, u_t, \xi) \in \mathcal{X}, \quad \forall \xi \in \Xi \tag{14}$$

In the ADP approach presented in Wang and Boyd [60] the authors fit a quadratic under estimator to the stage cost function where the fitting is performed over the constraint set $(\mathcal{X} \times \mathcal{U})$. Hence there will be significant error outside of this fitting region where the actual extended value stage cost function is infinity. Fitting a quadratic value function using this quadratic approximated stage cost, will also not go to infinity for the regions where the state constraint cannot be respected at the next time step.

This means that a quadratic basis function approach for ADP should not be used when there are state constraints that must be respected for operation critical reasons.

3.4 Model-Free Approximate Dynamic Programming Approaches

3.4.1 Temporal Difference (TD) Methods

In section 3.3 the approximate value function $\hat{V}$ was approximated as a linear combination of basis functions, taking the following form:

$$\hat{V}_\alpha(x) = \sum_{i=1}^{K} \alpha_i \hat{V}^{(i)}(x) \tag{15}$$

The methods described above for determining the weightings, α_i, require knowledge of the dynamics, stage costs and constraints. For problems of the same class, Temporal Differencing is also a method for determining the weightings α_i, for combining the basis functions. A Temporal Difference method updates the α_i weightings based on observations of how the system evolves. It uses the error in the approximated cost-to-go, as measured after each time step has evolved, to update the approximate value function $\hat{V}$.

Roy [48] presents temporal difference methods in a discrete time, infinite horizon discounted cost, continuous state space setting, while Bertsekas and Tsitsiklis [13] presents the methods in a discrete time, finite state space setting. We introduce Temporal Differencing (TD) methods in a similar fashion to [48], by first considering an autonomous system, i.e. a system with no input.

Recalling that in equation (15), the $\hat{V}^{(i)}$'s are the set of basis functions chosen in advance and the current approximation is specified by the vector $\alpha \in \mathbb{R}^K$. The temporal difference d_t between time t and $t+1$, for a sequence of states x_0, x_1, $x_2, \ldots$, is defined as

$$d_t = \left[l(x_t) + \gamma \hat{V}_\alpha(x_{t+1})\right] - \hat{V}_\alpha(x_t) \tag{16}$$

where γ is the discount factor. The last term represents the estimate of the cost-to-go at time t given state x_t, while the first two terms represent the updated estimate of the same

cost based on new information of:

- what state x_{t+1} the system actually progressed
- the stage cost $l(x_t)$ that was actually incurred

The temporal difference is then used to update the basis function weighting vector α according to

$$\alpha_{t+1} = \alpha_t + \beta_t \, d_t \, z_t \tag{17}$$

where β_t is a scalar step size, and $z_t \in \mathbb{R}^K$ is called the eligibility vector and is defined as

$$z_t = \sum_{\tau=0}^{t} (\gamma\lambda)^{t-\tau} \, \hat{V}(x_\tau) = \gamma \, \lambda \, z_{t-1} + \hat{V}(x_\tau) \tag{18}$$

where $\hat{V}(x) = [\hat{V}^{(1)}(x), \ldots, \hat{V}^{(K)}(x)]^T \in \mathbb{R}^K$ is a vector of each basis function evaluated at the point x and $\lambda \in [0, 1]$ is a scalar factor that is a design choice for the Temporal Difference method. The choice of the parameter λ influences the behaviour of how the Temporal Difference method updates the approximated value function. The method is often denoted TD(λ) to highlight the importance of the λ parameter.

Considering now a controlled system, the algorithm is referred to a Controlled TD and the input to apply based on the current approximate of the value function is computed as

$$u_t = \arg\min_{u \in \mathcal{U}} \, l(x_t, u) + \gamma \, \mathbb{E}_\xi \Big[\hat{V}_{\alpha_t} \big(f(x_t, u, \xi) \big) \Big] \tag{19}$$

analogous to the Bellman equation (7). The system then progresses to the next state, x_{t+1} and the temporal difference d_t can be computed as

$$d_t = \big[l(x_t, u_t) + \gamma \hat{V}_\alpha(x_{t+1}) \big] \; - \; \hat{V}_\alpha(x_t) \tag{20}$$

similar to above. The temporal difference d_t is used to update weighting vector α_{t+1} in a similar fashion to that described above for an autonomous system.

From a model-free perspective, the immediate barrier to using a Temporal Differencing method is that computing the input to be applied as per equation (19), requires a model

(or simulator) of the dynamics, the stage costs function and the ability to compute the expectation over the uncertainty. The algorithm in the next subsection outcomes these requirement.

3.4.2 Q-Learning

In the thesis of Watkins [62], the Q-Learning method was introduced as an *"alternative algorithm for performing dynamic programming"*. The Q-learning method denotes the full objective function of equation (19) as $Q(x, u)$, referred to as *the Q function*,

$$u_t = \arg\min_{u \in \mathcal{U}} \underbrace{l(x_t, u) + \gamma \, \mathbb{E}_\xi \Big[\hat{V}_{\alpha_t} \big(f(x_t, u, \xi) \big) \Big]}_{Q(x_t, u)} \tag{21}$$

This shows that the Q function contains all the information about the model dynamics, stage costs and uncertainty expectation. Distinct from the value function, the Q function maps $\mathcal{X} \times \mathcal{U} \to \mathbb{R}$.

The goal of the Q-Learning method is to obtain a good estimate of the Q function by learning an approximation $\hat{Q}$ using data obtained from the actual system. A Q-learning method can also be based on a basis function approximation

$$\hat{Q}_\alpha(x, u) = \sum_{i=1}^{K} \alpha_i \hat{Q}^{(i)}(x, u) \tag{22}$$

where $\hat{Q}^{(i)}$ are the set of basis function chosen in advance and the current approximation is specified by the vector $\alpha \in \mathbb{R}^K$. Given a current estimate $\hat{Q}$, it is then possible to compute the input to apply as

$$u_t = \arg\min_{u \in \mathcal{U}} \hat{Q}(x_t, u) \tag{23}$$

Under the application of this input the system will progress to state x_{t+1}, stage cost $l(x_t, u_t)$ will be incurred and the next next input to be applied, u_{t+1}, can be computed based only on knowledge of x_{t+1}. This allows the temporal difference d_t between time t to $t+1$ to be computed as

$$d_t = \big[l(x_t, u_t) + \gamma \hat{Q}_\alpha(x_{t+1}, u_{t+1}) \big] \; - \; \hat{Q}_\alpha(x_t, u_t) \tag{24}$$

where the only other parameter that needs to be specified is the discount factor γ.

In Watkins and Dayan [63] the authors consider a discrete time, infinite horizon discounted cost, finite state-input space system and provided a proof that *"Q-learning converges to the optimum action-values with probabilty 1 so long as all actions are repeatedly sampled in all states and the action-values are represented discretely".*

Although a strong theoretical guarantee, the requirement for this proof of convergence eludes at one of the main difficulties with learning algorithms, often referred to as *exploration versus exploitation* and discussed further in the next section.

3.4.3 Exploration versus Exploitation

In the temporal difference methods described above, the control input to be applied is always chosen to minimise the current estimate of the approximate Value function or approximate Q function, and the function is then updated based on the resulting state and stage cost incurred. It is possible for such algorithms to get stuck in a deadlock of cyclicly visiting the same subset of the state space [48]. When stuck in a deadlock it is likely that only a portion of the full state space $\mathcal{X}$, is visited and if the true optimal is outside of this subset, then it will never be "learned". This is a case of exploitation only.

In Watkins [62] the author gives this informal description of why exploitation combined with exploration should lead to optimality:

> *"According to the optimality argument, if an animal has many opportunities to practise a critical skill, and if it is able to try out some alternative strategies without disaster, then the animal should ultimately acquire a capacity for maximally efficient performance."*

In the work of De Farias and Van Roy [28], a framework is proposed for including exploration in a Temporal Difference algorithm. The authors suggest randomising the choice to control action. Using the probability distribution proposed, they proved that there exists a fixed point of the Temporal Difference recursion. However, the fixed point is not necessarily unique or optimal. Their results extend to Q-Learning by applying a similar randonising

method for the actions [48].

3.4.4 Bellman Residual Elimination

In the work of Bethke and How [19] the authors introduce an approximate dynamic programming approach they refer to as Bellman Residual Elimination (BRE). The BRE method is developed for infinite horizon discounted cost, discrete time and finite state-input space where the transition probabilities are known, i.e. the full model is know. In their subsequent work [20] the author extend the BRE approach to be applicable for model-free analysis and control where neither the stage cost function nor the transition probabilities are known.

The Bellman equation for infinite horizon, discounted cost, finite state-input space can be written as

$$V(i) = \min_{u \in \mathcal{U}} l(i, u_t) + \sum_{j \in \mathcal{X}} \Big[P(i, j, u) V(j) \Big] = \mathcal{T} V(i), \quad \forall\, i \in \mathcal{X} \tag{25}$$

where i, j are elements of the finite state space $\mathcal{X}$ and u is an element of the finite action space $\mathcal{U}$. The expectation over the uncertainty is replaced by the probability transition $P(i, j, u)$ which is the probability of the system transitioning to state j given that it was in state i and action u was applied.

For a finite state-input space, it is common, similar to Schweitzer and Seidmann [51], to choose a representative subset of the state space $\hat{\mathcal{X}} \subseteq \mathcal{X}$ and assess the quality of the approximate value function $\hat{V}$ by the following

$$\left(\sum_{i \in \hat{\mathcal{X}}} \left| \hat{V}(i) - \mathcal{T} \hat{V}(i) \right|^p \right)^{1/p} = \left(\sum_{i \in \hat{\mathcal{X}}} |BR(i)|^p \right)^{1/p} \tag{26}$$

with $p \geq 1$. This is the p-norm of the residuals from the Bellman equation, (25), evaluated at each state from the representative set. Thus algorithms based on this metric are often referred to as *Bellman Residual* methods.

The BRE algorithm proposed in [19] formulates a regression problem that use a kernel-based regression scheme to simultaneously force all the Bellman Residuals $BR(i)$ to zero over the representative state space, $i \in \hat{\mathcal{X}}$. A standard kernel-based regression scheme can

then be used to solve the problem for the approximate Value Function and control policy, for which equation (26) will be identically zero.

The choice of $\hat{\mathcal{X}}$ is similar to the choice of the state relevance vector c described in section 3.3.2. It plays a key role in determining the quality of the solution over the whole state space. As the Bellman residual is zero at each $i \in \mathcal{X}$, it is proven in [19] that the BRE algorithm gives the exact solution in the limit of sampling the entire space, $\hat{\mathcal{X}} \to \mathcal{X}$.

In the model-free variant of BRE [20], stochastic approximations of stage cost function and the transition model are trained using a reinforcement learning algorithm. The input data required for the model-free BRE method is a number of state trajectories collected from data of the actual system in operation. The required state trajectory data could also be generated using a simulator of the system, but driving the algorithm by data collected form the actual system in more inline with the Local4Global goals.

The model-free BRE algorithm requires m trajectories, each of length n transitions, starting from each of the $n_s = |\hat{\mathcal{X}}|$ states of the representative state-space set, i.e. $m\,n_s$ trajectories are required in total. The total complexity of the algorithm is

$$\mathcal{O}(m^2\, n_s^2 + n_s^3) \tag{27}$$

The theoretical guarantee provided in [20] is that as the number of sample trajectories goes to infinity, the approximate Value Function will converge the same are model-based BRE for the same $\hat{\mathcal{X}}$. Therefore as $\hat{\mathcal{X}} \to \mathcal{X}$ model-free BRE will also converge to the optimal Value Function. Although a strong guarantee, without any information about the rate of convergence, the method could perform quiet poorly in practice if it takes a prohibitively large number of samples before converging to a near optimal solution.

3.5 Summary

The theory of Approximate Dynamic Programming covers a broad range of topics that show potential for adapting to the Local4Global framework. The Value function approximation techniques explained are powerful because they provide theoretical guarantees of being

under-estimators of the true value function. Such guarantees are important for developing certificates on the degree of sub-optimality of a control policy or current state. These certificates are be aimed at providing a flag for when a global coordinator should intervene to re-optimise the local controllers.

The Temporal Difference and Q-Learning techniques also stem from value function approximation and the latter has the advantage of being model free. These techniques "learn" an approximation for the Value function based on measurements of the state as the system evolves. The PCAO algorithm [3] and AOC (Adaptive Optimal Control) algorithm [42] developed in the previous work of the consortium partners has a connection with these techniques because their work also aims to build the best quadratic value function approximation based on measurements taken as the system evolves.

In Warren Powell's overview of ADP [45] he makes the following generalisation *"the richer message of approximate dynamic programming is learning what to learn and how to learn it, to make better decisions over time"*, which syncs with the goals of Local4Global.

4 Robust Decision Rules

4.1 Introduction

In this chapter, we use robust optimization techniques to develop efficient solution methods for the class of multistage adaptive mixed-integer optimization problems. The one-stage variant of the problem can be described as follows: Given matrices $A \in \mathbb{R}^{m\times k}$, $B \in \mathbb{R}^{m\times q}$, $C \in \mathbb{R}^{n\times k}$, $D \in \mathbb{R}^{q\times k}$, $H \in \mathbb{R}^{m\times k}$ and a probability measure $\mathbb{P}_\xi$ supported on set Ξ for the uncertain vector $\boldsymbol{\xi} \in \mathbb{R}^k$, we are interested in choosing n real-valued functions $\boldsymbol{x}(\cdot) \in \mathcal{R}_{k,n}$ and q binary functions $\boldsymbol{y}(\cdot) \in \mathcal{B}_{k,q}$ in order to solve:

$$\left.\begin{array}{ll} \text{minimize} & \mathbb{E}_\xi\Big((C\boldsymbol{\xi})^\top \boldsymbol{x}(\boldsymbol{\xi}) + (D\boldsymbol{\xi})^\top \boldsymbol{y}(\boldsymbol{\xi})\Big) \\ \text{subject to} & \boldsymbol{x}(\cdot) \in \mathcal{R}_{k,n}, \boldsymbol{y}(\cdot) \in \mathcal{B}_{k,q}, \\ & A\boldsymbol{x}(\boldsymbol{\xi}) + B\boldsymbol{y}(\boldsymbol{\xi}) \leq H\boldsymbol{\xi}, \end{array}\right\} \quad \forall \boldsymbol{\xi} \in \Xi. \tag{28}$$

Here, $\mathcal{R}_{k,n}$ denotes the space of all real-valued functions from $\mathbb{R}^k$ to $\mathbb{R}^n$, and $\mathcal{B}_{k,q}$ the space of binary functions from $\mathbb{R}^k$ to $\{0,1\}^q$. Problem (28) has a long history both from the theoretical and practical point of view, with applications in many fields such as engineering [34, 54], operations management [11, 52], and finance [22, 23]. From a theoretical point of view, Dyer and Stougie [30] have shown that computing the optimal solution for the class of Problems (28) involving only real-valued decisions, is #P-hard, while their multi-stage variants are believed to be *"computationally intractable already when medium-accuracy solutions are sought"* [53]. In view of these complexity results, there is a need for computationally efficient solution methods that possibly sacrifices optimality for tractability. To quote Shapiro and Nemirovski [53], *"... in actual applications it is better to pose a modest and achievable goal rather than an ambitious goal which we do not know how to achieve."*

A drastic simplification that achieves this goal is to use *decision rules*. This functional approximation restricts the infinite space of the adaptive decisions $\boldsymbol{x}(\cdot)$ and $\boldsymbol{y}(\cdot)$ to admit pre-specified structures, allowing the use of numerical solution methods. Decisions rules for real-valued functions have been used since 1974 [31]. Nevertheless, their potential was not fully exploited until recently, when new advances in robust optimization provided the tools

to reformulate the decision rule problem as tractable convex optimization problems [10, 8]. Robust optimization techniques were first used by Ben-Tal *et al.* [9] to reformulate the *linear decision rule problem*. In this work, real-valued functions are parameterised as linear function of the random variables, i.e., $x(\boldsymbol{\xi}) = \boldsymbol{x}^\top \boldsymbol{\xi}$ for $\boldsymbol{x} \in \mathbb{R}^k$. The simple structure of linear decision rules offers the scalability needed to tackle multistage adaptive problems. Even though linear decision rules are known to be optimal in a number of problem instances [2, 17, 35], their simple structure generically sacrifices a significant amount of optimality in return for scalability. To gain back some degree of optimality, attention was focused on the construction of non-linear decision rules. Inspired by the linear decision rule structure, the real-valued adaptive decisions are parameterised as $x(\boldsymbol{\xi}) = \boldsymbol{x}^\top L(\boldsymbol{\xi})$, $\boldsymbol{x} \in \mathbb{R}^{k'}$, where $L : \mathbb{R}^k \to \mathbb{R}^{k'}$, $k' \geq k$, is a non-linear operator defining the structure of the decision rule. This approximation provides significant improvements in solution quality, while retaining in large parts the favourable scalability properties of the linear decision rules. Table 1 summarizes non-linear decision rules from the literature that are parameterized as $x(\boldsymbol{\xi}) = \boldsymbol{x}^\top L(\boldsymbol{\xi})$, together with the complexity of the induced convex optimization problems. A notable exception of real-valued decision rules that are alternatively parameterized as

Real-Valued Decision Rule Structures

$x(\boldsymbol{\xi}) = \boldsymbol{x}^\top L(\boldsymbol{\xi})$	**Computational Burden**	**Reference**
Linear	LOP	[9, 34, 39, 54]
Piecewise linear	LOP	[26, 27, 32, 33]
Multilinear	LOP	[32]
Quadratic	SOCOP	[7, 32]
Power, Monomial, Inverse monomial	SOCOP	[32]
Polynomial	SDOP	[4, 18]

Table 1: *The table summarizes the literature of real-valued decision rule structures for which the resulting semi-infinite optimization problem can be reformulated exactly using robust optimization techniques. The computation burden refers to the structure of the resulting optimization problems for the case where the uncertainty set is described by a polyhedral set, with Linear Optimization Problems denoted by LOP, Second-Order Cone Optimization Problems denoted by SOCOP, and Semi-Definite Optimization Problems denoted by SDOP.*

$x(\boldsymbol{\xi}) = \max\{\overline{\boldsymbol{x}}_1^\top \boldsymbol{\xi}, \ldots, \overline{\boldsymbol{x}}_P^\top \boldsymbol{\xi}\} - \max\{\underline{\boldsymbol{x}}_1^\top \boldsymbol{\xi}, \ldots, \underline{\boldsymbol{x}}_P^\top \boldsymbol{\xi}\}$, $\overline{\boldsymbol{x}}_p, \underline{\boldsymbol{x}}_p \in \mathbb{R}^k$, is proposed by Bertsimas and Georghiou [16] in the framework of worst-case adaptive optimization. This structure offers near-optimal designs but requires the solution of mixed-integer optimization problems.

In contrast to the plethora of decision rule structures for real-valued decisions, the literature

on discrete decision rules is somewhat limited. There are, however, three notable exceptions that deal with the case of worst-case adaptive optimization. The first one is the work of Bertsimas and Caramanis [14], where integer decision rules are parameterized as $y(\boldsymbol{\xi}) = \boldsymbol{y}^\top \lceil \boldsymbol{\xi} \rceil$, $\boldsymbol{y} \in \mathbb{Z}^k$, where $\lceil \cdot \rceil$ is the component-wise ceiling function. In this work, the resulting semi-infinite optimization problem is further approximated and solved using a randomized algorithm [24], providing only probabilistic guaranties on the feasibility of the solution. The second binary decision rule structure was proposed by Bertsimas and Georghiou [16], where the decision rule is parameterized as

$$y(\boldsymbol{\xi}) = \begin{cases} 1, & \max\{\overline{\boldsymbol{y}}_1^\top \boldsymbol{\xi}, \ldots, \overline{\boldsymbol{y}}_P^\top \boldsymbol{\xi}\} - \max\{\underline{\boldsymbol{y}}_1^\top \boldsymbol{\xi}, \ldots, \underline{\boldsymbol{y}}_P^\top \boldsymbol{\xi}\} \leq 0, \\ 0, & \text{otherwise}, \end{cases} \tag{29}$$

with $\overline{\boldsymbol{y}}_p, \underline{\boldsymbol{y}}_p \in \mathbb{R}^k$, $p = 1, \ldots, P$. This binary decision rule structure offers near-optimal designs at the expense of scalability. Finally, in the recent work by Hanasusanto *et al.* [36], the binary decision rule is restricted to the so called "K-adaptable structure", resulting to binary adaptable policies with K contingency plans. This approximation is an adaption of the work presented in Bertsimas and Caramanis [15], and its use is limited to two-stage adaptive problems.

The goal of this chapter is to develop binary decision rules structures that can be used together with the real-valued decision rules listed in Table 1 for solving multistage adaptive mixed-integer optimization problems. The proposed methodology is inspired by the work of Bertsimas and Caramanis [14], and uses the tools developed in Georghiou *et al.* [32] to robustly reformulate the problem into a finite dimensional mixed-integer linear optimization problem. The main contributions of this chapter can be summarized as follows.

1. We propose a linear parameterized binary decision rule structure, and derive the exact reformulation of the decision rule problem that achieves the best binary decision rule. We prove that for general polyhedra uncertainty sets and arbitrary decision rule structures, the decision rule problem is computationally intractable, resulting in mixed-integer linear optimization problems whose size can grow exponentially with respect to the problem data. To remedy this exponential growth, we use similar ideas

as those discussed in Georghiou *et al.* [32], to provide a systematic trade-off between scalability and optimality, resulting to practical binary decision rules.

2. We apply the proposed binary decision rules to the class of problems with random-recourse, i.e., to problem instances where the technology matrix $B(\boldsymbol{\xi})$ is a given function of the uncertain vector $\boldsymbol{\xi}$. We provide the exact reformulation of the binary decision rule problem, and we show that the resulting mixed-integer linear optimization problem shares similar complexity as in the fixed-recourse case.

The rest of this chapter is organized as follows. In Section 4.2, we outline our approach for binary decision rules in the context of one-stage adaptive optimization problems with fixed-recourse, and in Section 4.3 we discuss the extension to random-recourse problems. In Section 4.4, we extend the proposed approach to multistage adaptive optimization problems. Throughout the chapter, our work is focused on problem instances involving only binary decision rules for ease of exposition. The extension to problem instances involving both real-valued and binary recourse decisions can be easily extrapolated, and thus omitted.

Notation. We denote scalar quantities by lowercase, non-bold face symbols and vector quantities by lowercase, boldface symbols, e.g., $x \in \mathbb{R}$ and $\boldsymbol{x} \in \mathbb{R}^n$, respectively. Similarly, scalar and vector valued functions will be denoted by, $x(\cdot) \in \mathbb{R}$ and $\boldsymbol{x}(\cdot) \in \mathbb{R}^n$, respectively. Matrices are denoted by uppercase symbols , e.g., $A \in \mathbb{R}^{n \times m}$. We model uncertainty by a probability space $\left(\mathbb{R}^k, \mathcal{B}\left(\mathbb{R}^k\right), \mathbb{P}_\xi\right)$ and denote the elements of the sample space $\mathbb{R}^k$ by $\boldsymbol{\xi}$. The Borel σ-algebra $\mathcal{B}\left(\mathbb{R}^k\right)$ is the set of events that are assigned probabilities by the probability measure $\mathbb{P}_\xi$. The support Ξ of $\mathbb{P}_\xi$ represents the smallest closed subset of $\mathbb{R}^k$ which has probability 1, and $\mathbb{E}_\xi(\cdot)$ denotes the expectation operator with respect to $\mathbb{P}_\xi$. $\mathrm{Tr}(A)$ denotes the trace of a square matrix $A \in \mathbb{R}^{n \times n}$ and $\mathbf{1}(\cdot)$ is the indicator function. Finally, $\boldsymbol{e}_k$ the kth canonical basis vector, while $\boldsymbol{e}$ denotes the vector whose components are all ones. In both cases, the dimension will be clear from the context.

4.2 Binary Decision Rules for Fixed-Recourse Problems

In this section, we present our approach for one-stage adaptive optimization problems with fixed recourse, involving only binary decisions. Given matrices $B \in \mathbb{R}^{m\times q}$, $D \in \mathbb{R}^{q\times k}$, $H \in \mathbb{R}^{m\times k}$ and a probability measure $\mathbb{P}_\xi$ supported on set Ξ for the uncertain vector $\boldsymbol{\xi} \in \mathbb{R}^k$, we are interested in choosing binary functions $\boldsymbol{y}(\cdot) \in \mathcal{B}_{k,q}$ in order to solve:

$$\left.\begin{array}{ll} \text{minimize} & \mathbb{E}_\xi\Big((D\boldsymbol{\xi})^\top \boldsymbol{y}(\boldsymbol{\xi})\Big) \\ \text{subject to} & \boldsymbol{y}(\cdot) \in \mathcal{B}_{k,q}, \\ & B\boldsymbol{y}(\boldsymbol{\xi}) \leq H\boldsymbol{\xi}, \end{array}\right\} \quad \forall \boldsymbol{\xi} \in \Xi. \tag{30}$$

Here, we assume that the uncertainty set Ξ is a non-empty, convex and compact polyhedron

$$\Xi = \Big\{\boldsymbol{\xi} \in \mathbb{R}^k : \exists \boldsymbol{\zeta} \in \mathbb{R}^v \text{ such that } W\boldsymbol{\xi} + U\boldsymbol{\zeta} \geq \boldsymbol{h},\ \xi_1 = 1\Big\}, \tag{31}$$

where $W \in \mathbb{R}^{l\times k}, U \in \mathbb{R}^{l\times v}$ and $\boldsymbol{h} \in \mathbb{R}^l$. Moreover, we assume that the linear hull of Ξ spans $\mathbb{R}^k$. The parameter ξ_1 is set equal to 1 without loss of generality as it allows us to represent affine functions of the non-degenerate outcomes $(\xi_2, \ldots, \xi_k)$ in a compact manner as linear functions of $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_k)$.

If the distribution governing the uncertainty $\boldsymbol{\xi}$ is unknown or if the decision maker is very risk-averse, then one might choose to alternatively minimize the worst-case costs with respect to all possible scenarios $\boldsymbol{\xi} \in \Xi$. This can be achieved, by replacing the objective function in Problem (30) with the worst-case objective $\max_{\xi\in\Xi}\Big((D\boldsymbol{\xi})^\top \boldsymbol{y}(\boldsymbol{\xi})\Big)$. By introducing the auxiliary variable $\tau \in \mathbb{R}$ and an epigraph formulation, we can equivalently write the worst-case problem as the following adaptive robust optimization problem.

$$\left.\begin{array}{ll} \text{minimize} & \tau \\ \text{subject to} & \tau \in \mathbb{R},\ \boldsymbol{y}(\cdot) \in \mathcal{B}_{k,q}, \\ & (D\boldsymbol{\xi})^\top \boldsymbol{y}(\boldsymbol{\xi}) \leq \tau, \\ & B\boldsymbol{y}(\boldsymbol{\xi}) \leq H\boldsymbol{\xi}, \end{array}\right\} \quad \forall \boldsymbol{\xi} \in \Xi. \tag{32}$$

Notice that $\boldsymbol{\xi}$ appears in the left hand side of the constraint $(D\boldsymbol{\xi})^\top \boldsymbol{y}(\boldsymbol{\xi}) \leq \tau$, multiplying the binary decisions $\boldsymbol{y}(\boldsymbol{\xi})$. Therefore, Problem (32) is an instance of the class of problems with random-recourse, which will be investigated in Section 4.3.

Problem (30) involves a continuum of decision variables and inequality constraints. Therefore, in order to make the problem amenable to numerical solutions, there is a need for suitable functional approximations for $\boldsymbol{y}(\cdot)$.

4.2.1 Structure of Binary Decision Rules

In this section, we present the structure of the binary decision rules. We restrict the feasible region of the binary functions $\boldsymbol{y}(\cdot) \in \mathcal{B}_{k,q}$ to admit the following piecewise constant structure:

$$\left.\begin{array}{l} \boldsymbol{y}(\boldsymbol{\xi}) = YG(\boldsymbol{\xi}),\ Y \in \mathbb{Z}^{q\times g}, \\ 0 \leq YG(\boldsymbol{\xi}) \leq \boldsymbol{e}, \end{array}\right\} \quad \forall \boldsymbol{\xi} \in \Xi, \tag{33a}$$

where $G : \mathbb{R}^k \to \{0,1\}^g$ is a piecewise constant function:

$$G_1(\boldsymbol{\xi}) := 1, \quad G_i(\boldsymbol{\xi}) := \mathbf{1}\left(\boldsymbol{\alpha}_i^\top \boldsymbol{\xi} \geq \beta_i\right), \quad i = 2,\ldots,g, \tag{33b}$$

for given $\boldsymbol{\alpha}_i \in \mathbb{R}^k$ and $\beta_i \in \mathbb{R}$, $i = 2,\ldots,g$. Here, we assume that both the set $\{\boldsymbol{\xi} \in \Xi : \boldsymbol{\alpha}_i^\top \boldsymbol{\xi} \geq \beta_i\}$ and $\{\boldsymbol{\xi} \in \Xi : \boldsymbol{\alpha}_i^\top \boldsymbol{\xi} \leq \beta_i\}$ have non-empty interior for all $i = 2,\ldots,g$, and $\boldsymbol{\alpha}_i^\top \boldsymbol{\xi} - \beta_i \neq \boldsymbol{\alpha}_j^\top \boldsymbol{\xi} - \beta_j$ for all $\boldsymbol{\xi} \in \Xi$ where $i,j \in \{1,\ldots,g\}$, $i \neq j$. Since, $G(\cdot)$ is a piecewise constant function, $\boldsymbol{y}(\boldsymbol{\xi}) = YG(\boldsymbol{\xi})$, $Y \in \mathbb{Z}^{q\times g}$ can give rise to an arbitrary integer decision rule. By imposing the additional constraint $0 \leq YG(\boldsymbol{\xi}) \leq \boldsymbol{e}$, for all $\boldsymbol{\xi} \in \Xi$, $\boldsymbol{y}(\cdot)$ is restricted to the space of binary decision rules.

Applying decision rules (33) to Problem (30) yields the following semi-infinite problem, which involves a finite number of decision variables $Y \in \mathbb{Z}^{q\times g}$, and an infinite number of

constraints:

$$\left.\begin{array}{ll} \text{minimize} & \mathbb{E}_\xi\Big((D\boldsymbol{\xi})^\top Y G(\boldsymbol{\xi})\Big) \\ \text{subject to} & Y \in \mathbb{Z}^{q\times g}, \\ & BYG(\boldsymbol{\xi}) \leq H\boldsymbol{\xi}, \\ & 0 \leq YG(\boldsymbol{\xi}) \leq \boldsymbol{e}, \end{array}\right\} \quad \forall \boldsymbol{\xi} \in \Xi. \tag{34}$$

Notice that all decision variables appear linearly in Problem (34). Nevertheless, the objective and constraints are non-linear functions of $\boldsymbol{\xi}$.

The following example illustrates the use of the binary decision rule (34) in a simple instance of Problem (30), motivating the choice of vectors $\boldsymbol{\alpha}_i \in \mathbb{R}^k$ and $\beta_i \in \mathbb{R}$, $i = 2, \ldots, g$, and showing the relationship between Problems (30) and (34).

Example 1 *Consider the following instance of Problem* (30)*:*

$$\left.\begin{array}{ll} \textit{minimize} & \mathbb{E}_\xi\big(\boldsymbol{y}(\boldsymbol{\xi})\big) \\ \textit{subject to} & y(\cdot) \in \mathcal{B}_{2,1}, \\ & y(\boldsymbol{\xi}) \geq \xi_2, \end{array}\right\} \quad \forall \boldsymbol{\xi} \in \Xi, \tag{35}$$

where $\mathbb{P}_\xi$ *is a uniform distribution supported on* $\Xi = \big\{(\xi_1, \xi_2) \in \mathbb{R}^2 \,:\, \xi_1 = 1,\, \xi_2 \in [-1, 1]\big\}$. *The optimal solution of Problem* (35) *is* $y^*(\boldsymbol{\xi}) = \mathbf{1}(\xi_2 \geq 0)$ *achieving an optimal value of* $\frac{1}{2}$. *One can attain the same solution by solving the following semi-infinite problem where* $G(\cdot)$ *is defined to be* $G_1(\boldsymbol{\xi}) = 1$, $G_2(\boldsymbol{\xi}) = \mathbf{1}(\xi_2 \geq 0)$, *i.e.,* $\boldsymbol{\alpha}_2 = (0, 1)^\top$, $\beta_2 = 0$ *and* $g = 2$, *see Figure 1.*

$$\left.\begin{array}{ll} \textit{minimize} & \mathbb{E}_\xi\big(\boldsymbol{y}^\top G(\boldsymbol{\xi})\big) \\ \textit{subject to} & \boldsymbol{y} \in \mathbb{Z}^2, \\ & \boldsymbol{y}^\top G(\boldsymbol{\xi}) \geq \xi_2, \\ & 0 \leq \boldsymbol{y}^\top G(\boldsymbol{\xi}) \leq 1, \end{array}\right\} \quad \forall \boldsymbol{\xi} \in \Xi. \tag{36}$$

The optimal solution of Problem (36) *is* $\boldsymbol{y}^* = (0, 1)^\top$ *achieving an optimal value of* $\frac{1}{2}$, *and is equivalent to the optimal binary decision rule in Problem* (35).

In the following section, we present robust reformulations of the semi-infinite Problem (34).

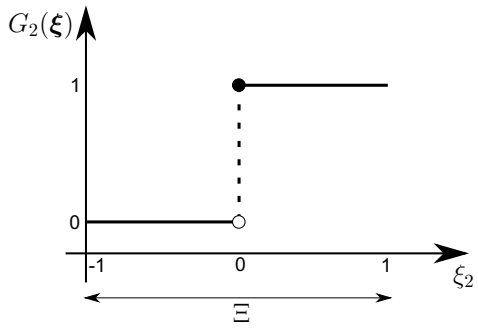

Figure 1: *Plot of piecewise constant function $G_2(\xi) = \mathbf{1}(\xi_2 \geq 0)$. This structure of $G(\cdot)$ will give rise of piecewise constant decision rules $y(\xi) = y^\top G(\xi)$, for some $y \in \mathbb{R}^2$. If one imposes constraints $y \in \mathbb{Z}^2$ and $0 \leq y^\top G(\xi) \leq 1$, $y(\cdot)$ is restricted to the space of binary decision rules induced by G.*

4.2.2 Computing the Best Binary Decision Rule

In this section, we present an exact reformulation of Problem (34). The solution of the reformulated problem will achieve the best binary decision rule associated with structure (33). The main idea used in this section is to map Problem (34) to an equivalent lifted adaptive optimization problem on a higher-dimensional probability space. This will allow to represent the non-convex constraints with respect to $\boldsymbol{\xi}$ in Problem (34), as linear constraints in the lifted adaptive optimization problem. The relation between the uncertain parameters in the original and the lifted problems is determined through the piecewise constant operator $G(\cdot)$. By taking advantage of the linear structure of the lifted problem, we will employ linear duality arguments to reformulate the semi-infinite structure of the lifted problem into a finite dimensional mixed-integer linear optimization problem. The work presented in this section uses the tools developed by Georghiou *et al.* [32, Section 3].

We define the non-linear operator

$$L : \mathbb{R}^k \to \mathbb{R}^{k'}, \qquad L(\boldsymbol{\xi}) = \begin{pmatrix} \boldsymbol{\xi} \\ G(\boldsymbol{\xi}) \end{pmatrix}, \tag{37}$$

where $k' = k + g$, and define the projection matrices $R_\xi \in \mathbb{R}^{k \times k'}$, $R_G \in \mathbb{R}^{g \times k'}$ such that

$$R_\xi L(\boldsymbol{\xi}) = \boldsymbol{\xi}, \qquad R_G L(\boldsymbol{\xi}) = G(\boldsymbol{\xi}).$$

We will refer to $L(\cdot)$ as the *lifting operator*. Using $L(\cdot)$ and R_ξ, R_G, we can rewrite Problem (34) into the following optimization problem:

$$\left.\begin{array}{ll} \text{minimize} & \mathbb{E}_\xi\Big((DR_\xi L(\boldsymbol{\xi}))^\top Y R_G L(\boldsymbol{\xi})\Big) \\ \text{subject to} & Y \in \mathbb{Z}^{q\times g}, \\ & BYR_G L(\boldsymbol{\xi}) \leq HR_\xi L(\boldsymbol{\xi}), \\ & 0 \leq YR_G L(\boldsymbol{\xi}) \leq \boldsymbol{e}, \end{array}\right\} \quad \forall \boldsymbol{\xi} \in \Xi. \tag{38}$$

Notice that this simple reformulation still has infinite number of constraints. Nevertheless, due to the structure of R_ξ and R_G, the constraints are now linear with respect to $L(\boldsymbol{\xi})$. The objective function of Problem (38) can be further reformulated to

$$\mathbb{E}_\xi\Big((DR_\xi L(\boldsymbol{\xi}))^\top Y R_G L(\boldsymbol{\xi})\Big) = \mathrm{Tr}\Big(MR_\xi^\top D^\top Y R_G\Big),$$

where $M \in \mathbb{R}^{k'\times k'}$, $M = \mathbb{E}_\xi(L(\boldsymbol{\xi})L(\boldsymbol{\xi})^\top)$ is the second order moment matrix of the uncertain vector $L(\boldsymbol{\xi})$. In some special cases where the $\mathbb{P}_\xi$ has a simple structure, e.g., $\mathbb{P}_\xi$ is a uniform distribution, and $G(\cdot)$ is not too complicated, M can be calculated analytically. If this is not the case, an arbitrarily good approximation of M can be calculated using Monte Carlo simulations. This is not computationally demanding as it does not does not involve an optimization problem, and can be done offline.

We now define the uncertain vector $\boldsymbol{\xi}' = (\boldsymbol{\xi}_1'^\top, \boldsymbol{\xi}_2'^\top)^\top \in \mathbb{R}^{k'}$, such that $\boldsymbol{\xi}' := L(\boldsymbol{\xi}) = (\boldsymbol{\xi}^\top, G(\boldsymbol{\xi})^\top)^\top$. $\boldsymbol{\xi}'$ has probability measure $\mathbb{P}_{\xi'}$ which is defined on the space $\big(\mathbb{R}^{k'}, \mathcal{B}(\mathbb{R}^{k'})\big)$ and is completely determined by the probability measure $\mathbb{P}_\xi$ through the relation

$$\mathbb{P}_{\xi'}\left(B'\right) := \mathbb{P}_\xi\left(\{\boldsymbol{\xi} \in \mathbb{R}^k :\ L(\boldsymbol{\xi}) \in B'\}\right) \qquad \forall B' \in \mathcal{B}(\mathbb{R}^{k'}). \tag{39}$$

We also introduce the probability measure support

$$\begin{aligned} \Xi' := L(\Xi) &= \Big\{\boldsymbol{\xi}' \in \mathbb{R}^{k'} :\ \exists \boldsymbol{\xi} \in \Xi \ \text{ such that } \ L(\boldsymbol{\xi}) = \boldsymbol{\xi}'\Big\}, \\ &= \Big\{\boldsymbol{\xi}' \in \mathbb{R}^{k'} :\ R_\xi \boldsymbol{\xi}' \in \Xi, \ \ L(R_G \boldsymbol{\xi}') = \boldsymbol{\xi}'\Big\}, \end{aligned} \tag{40}$$

and the expectation operator $\mathbb{E}_{\xi'}(\cdot)$ with respect to the probability measure $\mathbb{P}_{\xi'}$. We will

refer to $\mathbb{P}_{\xi'}$ and Ξ' as the *lifted* probability measure and uncertainty set, respectively. Notice that although Ξ is a convex polyhedron, Ξ' can be highly non-convex due to the non-linear nature of $L(\cdot)$. Using the definition of $\boldsymbol{\xi}'$ we introduce the following *lifted adaptive optimization problem*:

$$\left.\begin{array}{ll} \text{minimize} & \mathrm{Tr}\Big(M' R_\xi^\top D^\top Y R_G\Big) \\ \text{subject to} & Y \in \mathbb{Z}^{q\times g}, \\ & B Y R_G \boldsymbol{\xi}' \leq H R_\xi \boldsymbol{\xi}', \\ & 0 \leq Y R_G \boldsymbol{\xi}' \leq \boldsymbol{e}, \end{array}\right\} \quad \forall \boldsymbol{\xi}' \in \Xi', \tag{41}$$

where $M' \in \mathbb{R}^{k'\times k'}$, $M' = \mathbb{E}_{\xi'}(\boldsymbol{\xi}'\boldsymbol{\xi}'^\top)$ is the second order moment matrix associated with $\boldsymbol{\xi}'$. Since there is an one-to-one mapping between $\mathbb{P}_\xi$ and $\mathbb{P}_{\xi'}$, $M = M'$.

Proposition 1 *Problems* (38) *and* (41) *are equivalent in the following sense: both problems have the same optimal value, and there is a one-to-one mapping between feasible and optimal solutions in both problems.*

Proof See [32, Proposition 3.6 (i)]. ∎

The lifted uncertainty set Ξ' is an open set due to the discontinuous nature of $G(\cdot)$. This can be problematic in an optimization framework. We now define $\overline{\Xi}' := \mathrm{cl}(\Xi')$ to be the closure of set Ξ', and introduce the following variant of Problem (41).

$$\left.\begin{array}{ll} \text{minimize} & \mathrm{Tr}\Big(M' R_\xi^\top D^\top Y R_G\Big) \\ \text{subject to} & Y \in \mathbb{Z}^{q\times g}, \\ & B Y R_G \boldsymbol{\xi}' \leq H R_\xi \boldsymbol{\xi}', \\ & 0 \leq Y R_G \boldsymbol{\xi}' \leq \boldsymbol{e}, \end{array}\right\} \quad \forall \boldsymbol{\xi}' \in \overline{\Xi}', \tag{42}$$

The following proposition demonstrates that Problems (41) and (42) are in fact equivalent.

Proposition 2 *Problems* (41) *and* (42) *are equivalent in the following sense: both problems have the same optimal value, and there is a one-to-one mapping between feasible and optimal*

solutions in both problems.

Proof To prove the assertion, it is sufficient to show that Problems (41) and (42) have the same feasible region. Notice, that since the constraints are linear in $\boldsymbol{\xi}'$, the semi-infinite constraints in Problems (41) can be equivalently written as

$$\begin{aligned}(HR_\xi - BYR_G) &\in (\text{cone}(\Xi')^*)^m,\\ (YR_G) &\in (\text{cone}(\Xi')^*)^q,\\ (\boldsymbol{e}\boldsymbol{e}_1^\top - YR_G) &\in (\text{cone}(\Xi')^*)^q,\end{aligned}$$

where $\text{cone}(\Xi')^*$ is the dual cone of $\text{cone}(\Xi')$. Similarly, the semi-infinite constraints in Problems (42) can be equivalently written as

$$\begin{aligned}(HR_\xi - BYR_G) &\in (\text{cone}(\overline{\Xi}')^*)^m,\\ (YR_G) &\in (\text{cone}(\overline{\Xi}')^*)^q,\\ (\boldsymbol{e}\boldsymbol{e}_1^\top - YR_G) &\in (\text{cone}(\overline{\Xi}')^*)^q.\end{aligned}$$

From [47, Corollary 6.21], we have that $\text{cone}(\Xi')^* = \text{cone}(\overline{\Xi}')^*$, and therefore, we can conclude that the feasible region of Problems (41) and (42) is equivalent. ∎

Problem (42) is linear to both the decision variables Y and the uncertain vector $\boldsymbol{\xi}'$. Despite this nice bilinear structure, in the following we demonstrate that Problems (38) and (42) are generically intractable for decision rules of type (33).

Theorem 1 *Problems* (38) *and* (42) *defined through* $L(\cdot)$ *in* (37) *and* $G(\cdot)$ *in* (33b)*, are* NP*-hard even when* $Y \in \mathbb{Z}^{q\times g}$ *is relaxed to* $Y \in \mathbb{R}^{q\times g}$*.*

Proof See Appendix A. ∎

Theorem 1 provides a rather disappointing result on the complexity of Problems (38) and (42). Therefore, unless P = NP, there is no algorithm that solves generic problems of type (38) and (42) in polynomial time. This difficulty stems from the generic structure of $G(\cdot)$ in (33b), combined with a generic polyhedral uncertainty set Ξ, resulting in highly

non-convex sets $\overline{\Xi}'$. Nevertheless, in the following we derive the exact polyhedral representation of $\text{conv}(\overline{\Xi}')$, allowing us to use linear duality arguments from [10, 8], to reformulate Problem (42) into a mixed-integer linear optimization problem. We remark that Theorem 1 also covers decision rule problems involving real-valued, piecewise constant decisions rules constructed using (33b), and demonstrates that these problems are also computationally intractable.

We construct the convex hull of $\overline{\Xi}'$ by taking convex combinations of its extreme points, denoted by $\text{ext}(\overline{\Xi}')$. These points are either the extreme points of Ξ', or the limit points $\overline{\Xi}'\backslash\Xi'$. We construct these points as follows: First, using the definition of $G(\cdot)$ we partition Ξ into P polyhedra, Ξ_p. Then, for all extreme points of Ξ_p, $\boldsymbol{\xi} \in \text{ext}(\Xi_p)$, we calculate the one-side limit point at $L(\boldsymbol{\xi})$. The set of all one-side limit points will coincides with the set of extreme points of $\overline{\Xi}'$, see Figure 3. The partitions of Ξ can be formulated as follows:

$$\Xi_p = \left\{ \begin{array}{c} \boldsymbol{\xi} \in \Xi : \boldsymbol{\alpha}_i^\top \boldsymbol{\xi} \geq \beta_i, \;\; i \in \mathcal{G}_p \subseteq \{2, \ldots, g\}, \\ \boldsymbol{\alpha}_i^\top \boldsymbol{\xi} \leq \beta_i, \;\; i \in \{2, \ldots, g\} \backslash \mathcal{G}_p \end{array} \right\}, \quad p = 1, \ldots, P. \tag{43}$$

Here, $\mathcal{G}_p$ is constructed such that the number of partitions P is maximized, while ensuring that all Ξ_p are (a) non-empty, (b) any partition pair Ξ_i and Ξ_j, $i \neq j$, can overlap only on one their facets, and (c), $\Xi = \bigcup_{p=1}^{P} \Xi_p$, see Figure 2. Depending on the choices of $\boldsymbol{\alpha}_i$ and

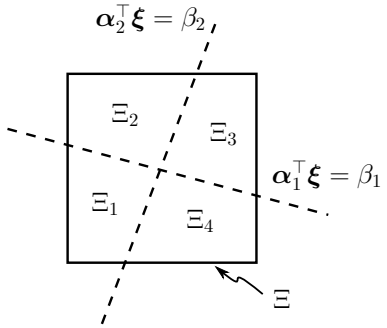

Figure 2: *The uncertainty set Ξ is partitioned into Ξ_p, $p = 1, \ldots, 4$ polyhedra using $\alpha_1^\top \xi = \beta_1$ and $\alpha_2^\top \xi = \beta_2$, such that $\Xi = \Xi_1 \cup \ldots \cup \Xi_4$.*

β_i, the number of partitions can be up to $P = 2^g$. We define V and $V(p)$ such that

$$V := \bigcup_{p=1}^{P} V(p), \qquad V(p) := \text{ext}(\Xi_p), \;\; p = 1, \ldots, P. \tag{44}$$

Due to the discontinuity of $G(\cdot)$, the set of points $\{\boldsymbol{\xi}' \in \mathbb{R}^{k'} : \boldsymbol{\xi}' = L(\boldsymbol{\xi}), \boldsymbol{\xi} \in V\}$ does not contain the points $\overline{\Xi}'\backslash\Xi'$. To construct these points, we now define the one-sided limit points $\widehat{L}_p(\boldsymbol{\xi}) = (\boldsymbol{\xi}^\top, \widehat{G}_p(\boldsymbol{\xi})^\top)^\top$ for all $\boldsymbol{\xi} \in V(p)$ and each partition $p = 1, \ldots, P$. Here, $\widehat{G}_p(\boldsymbol{\xi}) : \mathbb{R}^k \to \mathbb{R}^g$ is given by.

$$\widehat{G}_p(\boldsymbol{\xi}) := \lim_{u \in \Xi_p,\, u \to \xi} G(\boldsymbol{u}), \quad \forall \boldsymbol{\xi} \in V(p),\ p = 1, \ldots, P. \tag{45}$$

From definition (33b), for each partition p, $G(\boldsymbol{\xi})$ is constant for all $\boldsymbol{\xi}$ in the interior of Ξ_p, which is denoted by $\text{int}(\Xi_p)$. Therefore, for each $\tilde{\boldsymbol{\xi}} \in V(p)$, the one-side limit $\widehat{G}_p(\tilde{\boldsymbol{\xi}})$ is equal to $G(\boldsymbol{\xi})$ for all $\boldsymbol{\xi} \in \text{int}(\Xi_p)$. Furthermore, since each partition Ξ_p spans $\mathbb{R}^k$, then there exists at least $k+1$ vertices in Ξ_p, and therefore, at least $k+1$ one-sided limit points $\widehat{G}_p(\boldsymbol{\xi})$. From the definition (45), we have that $\widehat{L}_p(\boldsymbol{\xi})$ are the extreme points of $\overline{\Xi}'$ for all $\boldsymbol{\xi} \in V(p)$ and $p = 1, \ldots, P$, see Figure 3.

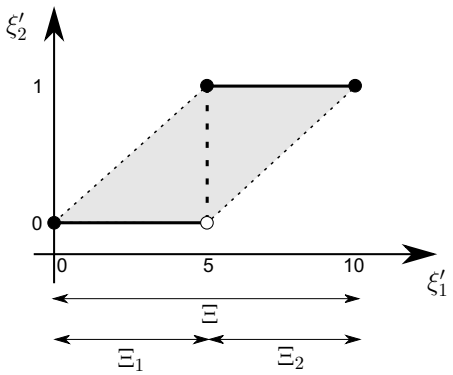

Figure 3: *Convex hull representation of $\overline{\Xi}'$ induced by lifting $\xi' = (\xi_1', \xi_2') = L(\xi) = (\xi, G(\xi))^\top$, where $G(\xi) = \mathbf{1}(\xi \geq 5)$ and $\xi \in [0, 10]$. Here, $V = \{\text{ext}(\Xi_1), \text{ext}(\Xi_2)\} = \{0, 5, 10\}$. The convex hull is constructed by taking convex combinations of the points $L(0), L(5), L(10)$ (black dots), and point $(5, \widehat{G}_1(5))^\top$ (white dot), where $\widehat{G}_1(5)$ is the one-side limit point at $\xi = 5$ using partition Ξ_1.*

The following proposition gives the polyhedral representation of the convex hull of $\overline{\Xi}'$.

Proposition 3 *Let $L(\cdot)$ be given by* (37) *and $G(\cdot)$ being defined in* (33b)*. Then, the tight,*

closed, outer approximation for the convex hull of $\overline{\overline{\Xi}}'$ *is given by the following polyhedron:*

$$
\begin{aligned}
conv(\overline{\overline{\Xi}}') = \Big\{ \boldsymbol{\xi}' = (\boldsymbol{\xi}_1'^{\top}, \boldsymbol{\xi}_2'^{\top})^{\top} \in \mathbb{R}^{k'} : \exists \zeta_p(\boldsymbol{v}) \in \mathbb{R}_+,\ \forall \boldsymbol{v} \in V(p),\ p = 1, \dots, P,\ \textit{such that} \\
\sum_{p=1}^{P} \sum_{v \in V(p)} \zeta_p(\boldsymbol{v}) = 1, \\
\boldsymbol{\xi}_1' = \sum_{p=1}^{P} \sum_{v \in V(p)} \zeta_p(\boldsymbol{v}) \boldsymbol{v}, \\
\boldsymbol{\xi}_2' = \sum_{p=1}^{P} \sum_{v \in V(p)} \zeta_p(\boldsymbol{v}) \widehat{G}_p(\boldsymbol{v}) \Big\}.
\end{aligned}
\tag{46}
$$

Proof The proof is split in two parts: First, we prove that $L(\Xi) \subseteq \text{conv}\,(\overline{\overline{\Xi}}')$ holds, and then we show that $\text{conv}\,(\text{cl}(L(\Xi))) \supseteq \text{conv}\,(\overline{\overline{\Xi}}')$ is true as well, concluding that $\text{conv}\,(\text{cl}(L(\Xi))) = \text{conv}\,(\overline{\overline{\Xi}}')$.

To prove assertion $L(\Xi) \subseteq \text{conv}\,(\overline{\overline{\Xi}}')$, pick any $\boldsymbol{\xi} \in \Xi$. By construction $\boldsymbol{\xi}$ belongs to some partition of Ξ, say Ξ_p, for which $L(\boldsymbol{\xi})$ is an element of Ξ'. There are two possibilities: (a) $\boldsymbol{\xi}$ lies on the boundary of the partition and hence on one of its facets, or (b) $\boldsymbol{\xi}$ lies in the interior of Ξ_p. If $\boldsymbol{\xi}$ lies on a facet, and since Ξ_p spans $\mathbb{R}^k$, then Carathéodory's Theorem implies that there are k extreme points of the facet, $\boldsymbol{v}_1, \dots, \boldsymbol{v}_k \in V(p)$, such that for some $\delta(\boldsymbol{v}) \in \mathbb{R}_+^k$ with $\sum_{i=1}^k \delta(\boldsymbol{v}_i) = 1$, we have $\boldsymbol{\xi} = \sum_{i=1}^k \delta(\boldsymbol{v}_i)\boldsymbol{v}_i$. Since $G(\cdot)$ is constant on each facet, then the k one-side limit points $\widehat{G}_p(\boldsymbol{v}_i)$, $\boldsymbol{v}_1, \dots, \boldsymbol{v}_k \in V(p)$, attain the same value as $G(\boldsymbol{\xi})$. Therefore, we have

$$
G\left(\sum_{i=1}^{k} \delta(\boldsymbol{v}_i)\boldsymbol{v}_i \right) = \sum_{i=1}^{k} \delta(\boldsymbol{v}_i) \widehat{G}_p(\boldsymbol{v}_i).
$$

Hence, setting $\zeta_p(\boldsymbol{v}_i) = \delta(\boldsymbol{v}_i)$, $i = 1, \dots, k$, in the definition of the convex hull with the rest of the $\zeta(\boldsymbol{v})$ equal to zero proves part (a) of the assertion. If $\boldsymbol{\xi}$ lies in the interior of Ξ_p, then again by Carathéodory's Theorem there are $k+1$ extreme points of Ξ_p such that for some $\delta(\boldsymbol{v}) \in \mathbb{R}_+^{k+1}$ and $\boldsymbol{v}_1, \dots, \boldsymbol{v}_{k+1} \in V(p)$ with $\sum_{i=1}^{k+1} \delta(\boldsymbol{v}_i) = 1$, we have $\boldsymbol{\xi} = \sum_{i=1}^{k+1} \delta(\boldsymbol{v}_i)\boldsymbol{v}_i$. By construction, there also exists $k+1$ one-sided limit in the collection $\{\widehat{G}_p(\boldsymbol{v}) : \boldsymbol{v} \in V(p)\}$

that attain the same value as $G(\boldsymbol{\xi})$ for all $\boldsymbol{\xi} \in \text{int}(\Xi_p)$. Thus, we have that

$$G\left(\sum_{i=1}^{k+1} \delta(\boldsymbol{v}_i)\boldsymbol{v}_i\right) = \sum_{i=1}^{k+1} \delta(\boldsymbol{v}_i)\widehat{G}_p(\boldsymbol{v}_i).$$

Setting $\zeta_p(\boldsymbol{v}_i) = \delta(\boldsymbol{v}_i)$, $i = 1, \ldots, k+1$, with the rest of the $\zeta(\boldsymbol{v})$ equal to zero proves part (b) of the assertion.

To prove the second part of assertion, $\text{conv}\,(\text{cl}(L(\Xi))) \supseteq \text{conv}\,(\overline{\Xi}')$, fix $\boldsymbol{\xi}' \in \overline{\Xi}'$. By construction, there exists $\zeta_p(v) \in \mathbb{R}_+$, $p = 1, \ldots, P$, $\boldsymbol{v} \in V$, such that

$$\sum_{p=1}^{P} \sum_{v \in V(p)} \zeta_p(\boldsymbol{v}) = 1, \quad \boldsymbol{\xi}'_1 = \sum_{p=1}^{P} \sum_{v \in V(p)} \zeta_p(\boldsymbol{v})\boldsymbol{v}, \quad \boldsymbol{\xi}'_2 = \sum_{p=1}^{P} \sum_{v \in V(p)} \zeta_p(\boldsymbol{v})\widehat{G}_p(\boldsymbol{v}).$$

This implies that

$$\begin{pmatrix} \boldsymbol{\xi}'_1 \\ \boldsymbol{\xi}'_2 \end{pmatrix} = \sum_{p=1}^{P} \sum_{v \in V(p)} \zeta_p(\boldsymbol{v}) \begin{pmatrix} \boldsymbol{v} \\ \widehat{G}_p(\boldsymbol{v}) \end{pmatrix},$$

that is, $\boldsymbol{\xi}'$ is a convex combination of either corner points or limit points of $L(\Xi)$. Therefore, $\text{conv}\,(\overline{\Xi}')$ is equal to $\text{conv}\,(\text{cl}(L(\Xi)))$ almost surely. This concludes the proof. ∎

The convex hull (46) is a closed and bounded polyhedral set described by a finite set of linear inequalities. Therefore, one can rewrite (46) as the following polyhedron

$$\text{conv}(\overline{\Xi}') = \left\{\boldsymbol{\xi}' \in \mathbb{R}^{k'} : \exists \boldsymbol{\zeta}' \in \mathbb{R}^{v'} \text{ such that } W'\boldsymbol{\xi}' + U'\boldsymbol{\zeta}' \geq \boldsymbol{h}'\right\}, \tag{47}$$

where $W' \in \mathbb{R}^{l' \times k'}$, $U' \in \mathbb{R}^{l' \times v'}$ and $\boldsymbol{h}' \in \mathbb{R}^{l'}$ are completely determined through Propositions 3.

The following proposition captures the essence of robust optimization and provides the tools for reformulating the infinite number of constraints in Problem (42), see [10, 8]. The proof is repeated here to keep the chapter self-contained.

Proposition 4 *For any $Z \in \mathbb{R}^{m \times k'}$ and $\text{conv}(\overline{\Xi}')$ given by (47), the following statements are equivalent.*

(i) $Z\boldsymbol{\xi}' \geq 0$ for all $\boldsymbol{\xi}' \in \text{conv}(\overline{\Xi}')$,

(ii) $\exists\, \Lambda \in \mathbb{R}_+^{m\times l'}$ *with* $\Lambda W' = Z,\ \Lambda U' = 0,\ \Lambda \boldsymbol{h}' \geq 0.$

Proof We denote by $Z_\mu^\top$ the μth row of the matrix Z. Then, statement (i) is equivalent to

$$\left.\begin{array}{lll} & Z\boldsymbol{\xi}' \geq 0 \text{ for all } \boldsymbol{\xi}' \in \operatorname{conv}(\Xi'), & \\ \iff & 0 \leq \min\limits_{\xi'} \left\{ Z_\mu^\top \xi : \exists \boldsymbol{\zeta}' \in \mathbb{R}^{v'},\ W'\boldsymbol{\xi}' + U'\boldsymbol{\zeta}' \geq \boldsymbol{h}' \right\}, & \forall\, \mu = 1,\ldots,m \\ \iff & 0 \leq \max\limits_{\Lambda_\mu \in \mathbb{R}^{l'}} \left\{ \boldsymbol{h}'^\top \Lambda_\mu : W'^\top \Lambda_\mu = Z_\mu,\ U'^\top \Lambda_\mu = 0,\ \Lambda_\mu \geq 0 \right\}, & \forall\, \mu = 1,\ldots,m \\ \iff & \exists\, \Lambda_\mu \in \mathbb{R}^{l'} \text{ with } W'^\top \Lambda_\mu = Z_\mu,\ U'^\top \Lambda_\mu = 0,\ \boldsymbol{h}'^\top \Lambda_\mu \geq 0,\ \Lambda_\mu \geq 0, & \forall\, \mu = 1,\ldots,m \end{array}\right\} \quad (48)$$

The equivalence in the third line follows from linear duality. Interpreting $\Lambda_\mu^\top$ as the μth row of a new matrix $\Lambda \in \mathbb{R}^{m\times l}$ shows that the last line in (48) is equivalent to assertion (ii). Thus, the claim follows. ∎

Using Proposition 4 together with the polyhedron (47), we can now reformulate Problem (42) into the following mixed-integer linear optimization problem.

$$\begin{array}{ll} \text{minimize} & \operatorname{Tr}\left(M' R_\xi^\top D^\top Y R_G \right) \\ \text{subject to} & Y \in \mathbb{Z}^{q\times g},\ \Lambda \in \mathbb{R}_+^{m\times l'},\ \Gamma \in \mathbb{R}_+^{q\times l'},\ \Theta \in \mathbb{R}_+^{q\times l'} \\ & BYR_G + \Lambda W' = HR_\xi,\ \Lambda U' = 0,\ \Lambda \boldsymbol{h}' \geq 0, \\ & YR_G = \Gamma W',\ \Gamma U' = 0,\ \Gamma \boldsymbol{h}' \geq 0, \\ & \boldsymbol{e}\boldsymbol{e}_1^\top - YR_G = \Theta W',\ \Theta U' = 0,\ \Theta \boldsymbol{h}' \geq 0. \end{array} \quad (49)$$

Here, $\Lambda \in \mathbb{R}_+^{m\times l'}$, $\Gamma \in \mathbb{R}_+^{q\times l'}$ and $\Theta \in \mathbb{R}_+^{q\times l'}$ are the auxiliary variables associated with constraints $BYR_G\boldsymbol{\xi}' \leq HR_\xi\boldsymbol{\xi}'$, $0 \leq YR_G\boldsymbol{\xi}'$ and $YR_G\boldsymbol{\xi}' \leq \boldsymbol{e}$, respectively. We emphasize that Problem (49) is the exact reformulation of Problem (42), since (46) is a tight outer approximation of the convex hull of $\overline{\Xi}'$. Therefore, *the solution of Problem* (49) *achieves the best binary decision rule associated with* $G(\cdot)$ *in* (33b). Notice that the size of the Problem (49) grows quadratically with respect to the number of binary decision q, and l' the number of constraints of $\operatorname{conv}(\overline{\Xi}')$. However, l' can be very large as it depends on the cardinality of V, which is constructed using the extreme points of the $P = 2^g$ partitions Ξ_p. Therefore, the size of Problem (49) can grow exponentially with respect to the description of Ξ and the complexity of the binary decision rule, g, defined in (33).

In the following, we present instances of Ξ and $G(\cdot)$, for which the proposed solution method will result to mixed-integer optimization problems whose size grows only polynomially with respect to the input parameters.

4.2.3 Scalable Binary Recision Rule Structures

In this section, we derive a scalable mixed-integer linear optimization reformulation of Problem (34) by considering simplified instances of the uncertainty set Ξ and binary decision rules $\boldsymbol{y}(\cdot)$. We will show that for the structure of Ξ and $\boldsymbol{y}(\cdot)$ considered in this section, the size of the resulting mixed-integer linear optimization problem grows polynomially in the size of the original Problem (30) as well as the description of the binary decision rules. We consider two cases: (i) uncertainty sets that can be described as hyperrectangles and (ii) general polyhedral uncertainty sets.

We now consider case (i), and define the uncertainty set of Problem (30) to have the following hyperrectangular structure:

$$\Xi = \left\{ \boldsymbol{\xi} \in \mathbb{R}^k \ : \ \boldsymbol{l} \leq \boldsymbol{\xi} \leq \boldsymbol{u}, \, \xi_1 = 1 \right\}, \tag{50}$$

for given $\boldsymbol{l}, \boldsymbol{u} \in \mathbb{R}^k$. We restrict the feasible region of the binary functions $\boldsymbol{y}(\cdot) \in \mathcal{B}_{k,q}$ to admit the piecewise constant structure (33a), but now $G : \mathbb{R}^k \to \{0,1\}^g$ is written in the form

$$G(\cdot) = (G_1(\cdot), G_{2,1}(\cdot), \ldots, G_{2,r}(\cdot), \ldots, G_{k,r}(\cdot))^\top,$$

with $g = 1 + (k-1)r$. Here, $G_1 : \mathbb{R}^k \to \{0,1\}$ and $G_{i,j} : \mathbb{R}^k \to \{0,1\}$, for $j = 1, \ldots, r, \ i = 2, \ldots, k$, are piecewise constant functions given by

$$G_1(\boldsymbol{\xi}) := 1, \qquad G_{i,j}(\boldsymbol{\xi}) := \mathbf{1}\left(\xi_i \geq \beta_{i,j}\right), \quad j = 1, \ldots, r, \ i = 2, \ldots, k, \tag{51}$$

for fixed $\beta_{i,j} \in \mathbb{R}$, $j = 1, \ldots, r$, $i = 2, \ldots, k$. By construction, we assume that $l_i < \beta_{i,1} < \ldots, \beta_{i,r} < u_i$, for all $i = 2, \ldots, k$. Structure (51) gives rise to binary decision rules that are discontinuous along the axis of the uncertainty set, and have r discontinuities in each direction. Notice that (51) is a special case of (33b), where vector $\boldsymbol{\alpha}$ are replaced by $\boldsymbol{e}_i$.

One can easily modify (51) such that the number of discontinuities r is different for each direction ξ_i. We will refer to $\beta_{i,1}, \ldots, \beta_{i,r}$ as the breakpoints in direction ξ_i.

Problem (34) still retains the same semi-infinite structure using (50) and (51). In the following, we use the same arguments as in Section 4.2.2, to (a) redefine Problem (34) into a higher dimensional space and (b) construct the convex hull of the lifted uncertainty set and use it together with Proposition 4 to reformulate the infinite number of constraints of the lifted problem.

We define the non-linear lifting operator $L : \mathbb{R}^k \to \mathbb{R}^{k'}$ to be $L(\cdot) = (L_1(\cdot)^\top, \ldots, L_k(\cdot)^\top)^\top$ such that

$$\begin{aligned} &L_1 : \mathbb{R}^k \to \mathbb{R}^2, \quad && L_1(\boldsymbol{\xi}) = (\xi_1, G_1(\boldsymbol{\xi}))^\top, \\ &L_i : \mathbb{R}^k \to \mathbb{R}^{r+1}, \quad && L_i(\boldsymbol{\xi}) = (\xi_i, G_i(\boldsymbol{\xi})^\top)^\top, \;\; i = 2, \ldots, k, \end{aligned} \tag{52}$$

with $k' = 2 + (k-1)(r+1)$. Here, with slight abuse of notation, $G_i(\cdot) = (G_{i,1}(\cdot), \ldots, G_{i,r})^\top$ for $i = 2, \ldots, k$. Notice that $L(\cdot)$ is separable in each component $L_i(\cdot)$, since $G_i(\cdot)$ only involves ξ_i. We also introduce matrices $R_\xi = (R_{\xi,1}, \ldots, R_{\xi,k}) \in \mathbb{R}^{k \times k'}$ with $R_{\xi,1} \in \mathbb{R}^{k \times 2}$, $R_{\xi,i} \in \mathbb{R}^{k \times (r+1)}$, $i = 2, \ldots, k$, such that

$$R_\xi L(\boldsymbol{\xi}) = \boldsymbol{\xi}, \qquad R_{\xi,i} L_i(\boldsymbol{\xi}) = \boldsymbol{e}_i \xi_i, \;\; i = 1, \ldots, k, \tag{53a}$$

and $R_G = (R_{G,1}, \ldots, R_{G,k}) \in \mathbb{R}^{g \times k'}$ with $R_{G,1} \in \mathbb{R}^{g \times 2}$, $R_{G,i} \in \mathbb{R}^{g \times (r+1)}$, $i = 2, \ldots, k$, such that

$$R_G L(\boldsymbol{\xi}) = G(\boldsymbol{\xi}), \qquad R_{G,i} L_i(\boldsymbol{\xi}) = (0, \ldots, G_i(\boldsymbol{\xi})^\top, \ldots, 0)^\top, \;\; i = 1, \ldots, k. \tag{53b}$$

As in Section 4.2.2, using the definition of $L(\cdot)$ and R_ξ, R_G in (52) and (53), respectively, we can rewrite Problem (34) into Problem (38).

We now define the uncertain vector $\boldsymbol{\xi}' = (\boldsymbol{\xi}_1'^\top, \ldots, \boldsymbol{\xi}_k'^\top)^\top \in \mathbb{R}^{k'}$ such that $\boldsymbol{\xi}' = L(\boldsymbol{\xi})$ and $\boldsymbol{\xi}_i' = L_i(\boldsymbol{\xi})$ for $i = 1, \ldots, k$. $\boldsymbol{\xi}'$ has probability measure $\mathbb{P}_{\xi'}$ defined using (39), and corresponding probability measure support is give by

$$\Xi' = L(\Xi) = \left\{ \boldsymbol{\xi}' \in \mathbb{R}^{k'} : \; \boldsymbol{l} \leq R_\xi \boldsymbol{\xi}' \leq \boldsymbol{u}, \;\; L(R_G \boldsymbol{\xi}') = \boldsymbol{\xi}' \right\}. \tag{54}$$

Notice that since both Ξ and $L(\cdot)$ are separable in each component ξ_i, Ξ' can be written in the following equivalent form:

$$\Xi' = L(\Xi) = \big\{(\boldsymbol{\xi}_1'^{\top}, \ldots, \boldsymbol{\xi}_k'^{\top})^{\top} \in \mathbb{R}^{k'} \,:\, \boldsymbol{\xi}_i \in \Xi_i',\, i = 1, \ldots, k\big\},$$

where $\Xi_i' = L_i(\Xi)$. Therefore, we can express $\operatorname{conv}(\operatorname{cl}(\Xi'))$ as

$$\begin{aligned} \operatorname{conv}(\operatorname{cl}(\Xi')) &= \big\{(\boldsymbol{\xi}_1'^{\top}, \ldots, \boldsymbol{\xi}_k'^{\top})^{\top} \in \mathbb{R}^{k'} \,:\, \boldsymbol{\xi}_i' \in \operatorname{conv}(\operatorname{cl}(\Xi_i')),\, i = 1, \ldots, k\big\}, \\ &= \big\{(\boldsymbol{\xi}_1'^{\top}, \ldots, \boldsymbol{\xi}_k'^{\top})^{\top} \in \mathbb{R}^{k'} \,:\, \boldsymbol{\xi}_i' \in \operatorname{conv}(\overline{\Xi}_i'),\, i = 1, \ldots, k\big\}. \end{aligned} \tag{55}$$

It is thus sufficient to derive a closed-form representation for the marginal convex hulls $\operatorname{conv}(\overline{\Xi}_i')$.

We now construct the polyhedral representation of $\operatorname{conv}(\overline{\Xi}_i')$. As before, $\operatorname{conv}(\overline{\Xi}_i')$ will be constructed by taking convex combinations of its extreme points. Set $\operatorname{conv}(\overline{\Xi}_1')$ has the trivial representation $\operatorname{conv}(\overline{\Xi}_1') = \{\boldsymbol{\xi}_1' \in \mathbb{R}^2 : \boldsymbol{\xi}_1' = \boldsymbol{e}\}$. We define the sets $V_i = \{l_i, \beta_{i,1}, \ldots, \beta_{i,r}, u_i\}$, $i = 2, \ldots, k$, and introduce the following partitions for each dimension of Ξ.

$$\left.\begin{aligned} &\Xi_{i,1} := \{\xi_i \in \mathbb{R} \,:\, l_i \le \xi_i \le \beta_{i,1}\}, \\ &\Xi_{i,p} := \{\xi_i \in \mathbb{R} \,:\, \beta_{i,p-1} \le \xi_i \le \beta_{i,p}\}, \quad p = 2, \ldots, r, \\ &\Xi_{i,r+1} := \{\xi_i \in \mathbb{R} \,:\, \beta_{i,r} \le \xi_i \le u_i\}, \end{aligned}\right\} i = 2, \ldots, k. \tag{56}$$

Therefore, $V_i(1) = \{l_i, \beta_{i,1}\}$, $V_i(p) = \{\beta_{i,p-1}, \beta_{i,p}\}$, $p = 2, \ldots, r$, $V_i(r+1) = \{\beta_{i,r}, u_i\}$. It is easy to see that points $L(\boldsymbol{e}_i\xi_i)$, $\xi_i \in V_i$, are extreme points of $\operatorname{conv}(\overline{\Xi}_i')$, see Figure 4. Notice that $G_i(\boldsymbol{e}_i\xi_i)$ is constant for all $\xi_i \in \operatorname{int}(\Xi_{i,p})$, $p = 1, \ldots, r+1$, but can attain different values on the boundaries of the partitions. To this end, for each partition and $\xi_i \in V_i(p)$, we define the one-sided limit points $\widehat{G}_{i,p}(\boldsymbol{e}_i\xi_i) \in \mathbb{R}^{r+1}$ such that

$$\widehat{G}_{i,p}(\boldsymbol{e}_i\xi_i) = \lim_{u \in \Xi_{i,p},\, u \to \xi_i} G_i(\boldsymbol{e}_i u), \quad \forall \xi_i \in V_i(p),\ p = 1, \ldots, r+1, \tag{57}$$

for all $i = 2, \ldots, k$. $G_i(\boldsymbol{e}_i\xi_i)$ is constant for all $\xi_i \in \operatorname{int}(\Xi_{i,p})$ and each partition p. Therefore, for each $\tilde{\xi}_i \in V_i(p)$, the one-side limit $\widehat{G}_{i,p}(\boldsymbol{e}_i\tilde{\xi})$ is equal to $G(\boldsymbol{e}_i\xi_i)$ for all $\xi \in \operatorname{int}(\Xi_{i,p})$.

The following proposition gives the polyhedral representation of the each $\operatorname{conv}(\overline{\Xi}_i')$, $i =$

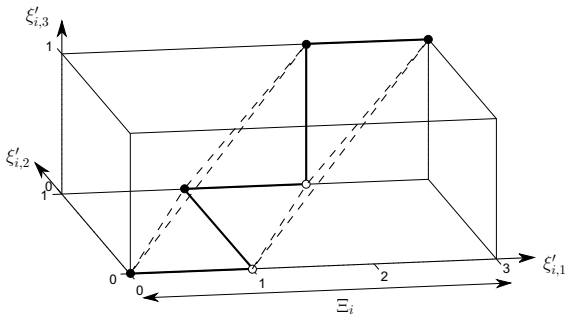

Figure 4: *Convex hull representation of $\overline{\Xi}'_i$ induced by lifting $\xi'_i = L_i(\xi) = (\xi_i, G_i(\xi)^\top)^\top$, where $G_i(\xi) = (\mathbf{1}(\xi_i \geq 1), \mathbf{1}(\xi_i \geq 2))^\top$ and $\xi_i \in [0,3]$. Here, $V_i = \{0,1,2,3\}$. The convex hull is constructed by taking convex combinations of the points $L(0), L(1), L(2), L(3)$ (black dots), and points $(1, \widehat{G}_{1,1}(1))^\top$, $(2, \widehat{G}_{1,2}(2))^\top$ (white dot), where $\widehat{G}_{1,1}(1)$ and $\widehat{G}_{1,2}(2)$ are the one-side limit point at points $\xi_i = 1$ and $\xi_i = 2$, respectively.*

$2, \ldots, k$. A visual representation of $\text{conv}(\overline{\Xi}'_i)$ is depicted in Figure 4.

Proposition 5 *For each $i = 2 \ldots, k$, let $L_i(\cdot)$ be given by* (52) *and $G_i(\cdot)$ being defined in* (51)*. Then, the tight, closed, outer approximation for the convex hull of $\overline{\Xi}'_i$ is given by the following polyhedron:*

$$
\begin{aligned}
conv(\overline{\Xi}'_i) = \Big\{ \boldsymbol{\xi}'_i = (\xi'^\top_{i,1}, \boldsymbol{\xi}'^\top_{i,2})^\top \in \mathbb{R}^{r+1} : \exists \zeta_{i,p}(v) \in \mathbb{R}_+,\ \forall v \in V_i(p),\ p = 1, \ldots, r+1,\ \textit{such that} \\
\sum_{p=1}^{r+1} \sum_{v \in V_i(p)} \zeta_{i,p}(v) = 1, \\
\xi'_{i,1} = \sum_{p=1}^{r+1} \sum_{v \in V_i(p)} \zeta_{i,p}(v) v, \\
\boldsymbol{\xi}'_{i,2} = \sum_{p=1}^{r+1} \sum_{v \in V_i(p)} \zeta_{i,p}(v) \widehat{G}_p(\boldsymbol{e}_i v) \Big\}.
\end{aligned}
\tag{58}
$$

Proof Proposition 5 can be proven using similar arguments as in Proposition 3. ∎

Set $\text{conv}(\overline{\Xi}'_i)$ has $2r+2$ extreme points. Using the extreme point representation (58), $\text{conv}(\overline{\Xi}'_i)$ can be represented through $4r+4$ constraints. Therefore, $\text{conv}(\overline{\Xi}') = \times_{i=1}^k \text{conv}(\overline{\Xi}'_i)$ can be represented using a total of $2 + k(4r+4)$ constraints, i.e., its size grows quadratically in the dimension of Ξ, k, and the complexity of the binary decision rule r. One can now use the polyhedral description (58) together with Proposition 4, to reformulate Problem (42) into a mixed-integer linear optimization problem which can be expressed in the form (49).

The size of the mixed-integer linear optimization problem will be polynomial in q, m, k and r. We emphasize that since the convex hull representation (58) is exact, *the solution of Problem* (49) *achieves the best binary decision rule associated with* $G(\cdot)$ *defined in* (51).

We now consider case (ii), and we assume that Ξ is a generic set of the type (90) and $G(\cdot)$ is given by (51). From Section 4.2, we know that the number of constraints needed to describe the polyhedral representation of $\mathrm{conv}(\overline{\Xi}')$ grows exponentially with the description of Ξ and complexity of the decision rule. In the following, we presented a systematic way to construct a tractable outer approximation for $\mathrm{conv}(\overline{\Xi}')$. Using this outer approximation to reformulate the lifted problem (38), will not yield the best binary decision rule structure associated with function $G(\cdot)$ but rather a conservative approximation. Nevertheless, the size of the resulting mixed-integer linear optimization problem will only grow polynomially with respect to the constrains of Ξ and complexity of the binary decision rule.

To this end, let $\{\boldsymbol{\xi} \in \mathbb{R}^k : \boldsymbol{l} \leq \boldsymbol{\xi} \leq \boldsymbol{u}\}$ be the smallest hyperrectangle containing Ξ. We have

$$\begin{aligned} \Xi &= \left\{\boldsymbol{\xi} \in \mathbb{R}^k : \exists \boldsymbol{\zeta} \in \mathbb{R}^v \text{ such that } W\boldsymbol{\xi} + U\boldsymbol{\zeta} \geq \boldsymbol{h},\ \xi_1 = 1\right\}, \\ &= \left\{\boldsymbol{\xi} \in \mathbb{R}^k : \exists \boldsymbol{\zeta} \in \mathbb{R}^v \text{ such that } W\boldsymbol{\xi} + U\boldsymbol{\zeta} \geq \boldsymbol{h},\ \xi_1 = 1,\ \boldsymbol{l} \leq \boldsymbol{\xi} \leq \boldsymbol{u}\right\}, \end{aligned} \tag{59}$$

which implies that the lifted uncertainty set $\Xi' = L(\Xi)$ can be expressed as $\Xi' = \Xi'_1 \cap \Xi'_2$, where

$$\begin{aligned} \Xi'_1 &:= \left\{\boldsymbol{\xi}' \in \mathbb{R}^{k'} : \exists \boldsymbol{\zeta} \in \mathbb{R}^v \text{ such that } WR_\xi\boldsymbol{\xi}' + U\boldsymbol{\zeta} \geq \boldsymbol{h},\ \xi'_1 = 1\right\} \\ \Xi'_2 &:= \left\{\boldsymbol{\xi}' \in \mathbb{R}^{k'} : \boldsymbol{l} \leq R_\xi\boldsymbol{\xi}' \leq \boldsymbol{u},\ L(R_G\boldsymbol{\xi}') = \boldsymbol{\xi}'\right\}. \end{aligned}$$

Notice that Ξ'_2 has exactly the same structure as (54). We thus conclude that

$$\widehat{\Xi}' := \{\Xi'_1 \cap \mathrm{conv}(\overline{\Xi}'_2)\} \quad \supseteq \quad \mathrm{conv}(\overline{\Xi}'), \tag{60}$$

and therefore, $\widehat{\Xi}'$ can be used as an outer approximation of $\mathrm{conv}(\overline{\Xi}')$. Since $\mathrm{conv}(\overline{\Xi}'_2)$ can be written as the polyhedron induced by Proposition 5, set $\widehat{\Xi}'$ has a total of $(l+1)+(2+k(4r+4))$ constraints. As before, one can now use the polyhedral description of $\widehat{\Xi}'$ together with Proposition 4, to reformulate Problem (42) into a mixed-integer linear optimization problem which can be expressed in the form (49). The size of the mixed-integer linear

optimization problem will be polynomial in q, m, k, l and r. Since $\widehat{\Xi}' \supseteq \text{conv}(\overline{\Xi}')$, the *solution of Problem* (49) *might not achieve the best binary decision rule induced by* (51)*, but rather a conservative approximation.* Nevertheless, using this systematic decomposition of the uncertainty set, we can efficiently apply binary decision rules to problems where the uncertainty set has arbitrary convex polyhedral structure.

One can use $G(\cdot)$ given by (51) together with this systematic decomposition of Ξ to achieve the same binary decision rule structures as those offered by $G(\cdot)$ defined in (33b). We will illustrate this through the following example. Lets assume that the problem in hand requires a generic Ξ of type (90), and we want to parameterize the binary decision rule to be a linear function of $G(\boldsymbol{\xi}) = \mathbf{1}\left(\boldsymbol{\alpha}^\top \boldsymbol{\xi} \geq \beta\right)$ for some $\boldsymbol{\alpha} \in \mathbb{R}^k$ and $\beta \in \mathbb{R}$. We can introduce an additional random variable $\tilde{\xi}$ in Ξ such that $\tilde{\xi} = \boldsymbol{\alpha}^\top \boldsymbol{\xi}$. $\tilde{\xi}$ is completely determined by the random variables already presented in Ξ. The modified support can now be written as follows,

$$\begin{aligned}
\Xi &= \left\{(\boldsymbol{\xi}, \tilde{\xi}) \in \mathbb{R}^{k+1} : \exists \boldsymbol{\zeta} \in \mathbb{R}^v,\ W\boldsymbol{\xi} + U\boldsymbol{\zeta} \geq \boldsymbol{h},\ \xi_1 = 1,\ \tilde{\xi} = \boldsymbol{\alpha}^\top \boldsymbol{\xi}\right\}, \\
&= \left\{(\boldsymbol{\xi}, \tilde{\xi}) \in \mathbb{R}^{k+1} : \exists \boldsymbol{\zeta} \in \mathbb{R}^v,\ W\boldsymbol{\xi} + U\boldsymbol{\zeta} \geq \boldsymbol{h},\ \xi_1 = 1,\ \tilde{\xi} = \boldsymbol{\alpha}^\top \boldsymbol{\xi},\ \boldsymbol{l} \leq \boldsymbol{\xi} \leq \boldsymbol{u},\ \tilde{l} \leq \tilde{\xi} \leq \tilde{u}\right\},
\end{aligned} \tag{61}$$

where $\{(\boldsymbol{\xi}, \tilde{\xi}) \in \mathbb{R}^{k+1} : \boldsymbol{l} \leq \boldsymbol{\xi} \leq \boldsymbol{u},\ \tilde{l} \leq \tilde{\xi} \leq \tilde{u}\}$ is the smallest hyperrectangle containing Ξ. We can now use

$$G(\boldsymbol{\xi}, \tilde{\xi}) = \mathbf{1}\left(\tilde{\xi} \geq \beta\right),$$

which is an instance of (51), to achieve the same structure as $G(\boldsymbol{\xi}) = \mathbf{1}\left(\boldsymbol{\alpha}^\top \boldsymbol{\xi} \geq \beta\right)$. Defining appropriate $L(\cdot)$ and R_ξ, R_G, we can use the following decomposition of the lifted uncertainty set $\Xi' = \Xi'_1 \cap \Xi'_2$ such that

$$\begin{aligned}
\Xi'_1 &:= \left\{(\boldsymbol{\xi}', \tilde{\xi}') \in \mathbb{R}^{k'+1} : \exists \boldsymbol{\zeta} \in \mathbb{R}^v \text{ such that } WR_\xi \boldsymbol{\xi}' + U\boldsymbol{\zeta} \geq \boldsymbol{h},\ \xi'_1 = 1,\ \tilde{\xi}' = \boldsymbol{\alpha}^\top R_\xi \boldsymbol{\xi}'\right\}, \\
\Xi'_2 &:= \left\{(\boldsymbol{\xi}', \tilde{\xi}') \in \mathbb{R}^{k'+1} : \boldsymbol{l} \leq R_\xi \boldsymbol{\xi}' \leq \boldsymbol{u},\ \tilde{l} \leq \tilde{\xi}' \leq \tilde{u},\ L(R_G(\boldsymbol{\xi}', \tilde{\xi}')) = (\boldsymbol{\xi}', \tilde{\xi}')\right\}.
\end{aligned}$$

By defining, $\widehat{\Xi}' := \{\Xi'_1 \cap \text{conv}(\Xi'_2)\} \supseteq \text{conv}(\overline{\Xi}')$ we can reformulate reformulate Problem (42) into a mixed-integer linear optimization problem with polynomial number of decision variable and constraints with respect to the the input data. Once more, we emphasize that this systematic decomposition might not achieve the best binary decision rule induced by

$G(\cdot)$ but rather a conservative approximation. Nevertheless, we can now achieve the same flexibility for the binary decision rules as those offered by (33b) without the exponential growth in the size of the resulting problem.

4.3 Binary Decision Rules for Random-Recourse Problems

In this section, we present our approach for one-stage adaptive optimization problems with random recourse. The solution method of this class of problems is an adaptation of the solution method presented in Section 4.2.2. Given function $B : \mathbb{R}^k \to \mathbb{R}^{m\times q}$ and matrices $D \in \mathbb{R}^{q\times k}$, $H \in \mathbb{R}^{m\times k}$ and a probability measure $\mathbb{P}_\xi$ supported on set Ξ for the uncertain vector $\boldsymbol{\xi} \in \mathbb{R}^k$, we are interested in choosing binary functions $\boldsymbol{y}(\cdot) \in \mathcal{B}_{k,q}$ in order to solve:

$$\left.\begin{array}{ll} \text{minimize} & \mathbb{E}_\xi\Big((D\boldsymbol{\xi})^\top \boldsymbol{y}(\boldsymbol{\xi})\Big) \\ \text{subject to} & \boldsymbol{y}(\cdot) \in \mathcal{B}_{k,q}, \\ & B(\boldsymbol{\xi})\boldsymbol{y}(\boldsymbol{\xi}) \leq H\boldsymbol{\xi}, \end{array}\right\} \quad \forall \boldsymbol{\xi} \in \Xi, \tag{62}$$

where Ξ is a generic polyhedron of type (90). We assume that the recourse matric $B(\boldsymbol{\xi})$ depends linearly on the uncertain parameters. Therefore, the μth row of $B(\boldsymbol{\xi})$ is representable as $\boldsymbol{\xi}^\top B_\mu$ for matrices $B_\mu \in \mathbb{R}^{k\times q}$ with $\mu = 1, \ldots, m$. Problem (62) can therefore be written as the following problem.

$$\left.\begin{array}{ll} \text{minimize} & \mathbb{E}_\xi\Big((D\boldsymbol{\xi})^\top \boldsymbol{y}(\boldsymbol{\xi})\Big) \\ \text{subject to} & \boldsymbol{y}(\cdot) \in \mathcal{B}_{k,q}, \\ & \boldsymbol{\xi}^\top B_\mu \boldsymbol{y}(\boldsymbol{\xi}) \leq H_\mu^\top \boldsymbol{\xi}, \;\; \mu = 1, \ldots, m \end{array}\right\} \quad \forall \boldsymbol{\xi} \in \Xi, \tag{63}$$

where $H_\mu^\top$ denotes the μth row of matrix H. Problem (63) involves a continuum of decision variables and inequality constraints. Therefore, in order to make the problem amenable to numerical solutions, we restrict $\boldsymbol{y}(\cdot)$ to admit structure (33). Applying the binary decision rules (33) to Problem (63) yields the following semi-infinite problem, which involves a finite

number of decision variables $Y \in \mathbb{Z}^{q\times g}$, and an infinite number of constraints:

$$\begin{array}{ll} \text{minimize} & \mathbb{E}_\xi\Big(\boldsymbol{\xi}^\top D^\top Y G(\boldsymbol{\xi})\Big) \\ \text{subject to} & Y \in \mathbb{Z}^{q\times g}, \\ & \boldsymbol{\xi}^\top B_\mu Y G(\boldsymbol{\xi}) \le H_\mu^\top \boldsymbol{\xi}, \;\; \mu = 1,\dots,m, \\ & 0 \le Y G(\boldsymbol{\xi}) \le \boldsymbol{e}, \end{array} \Bigg\} \quad \forall \boldsymbol{\xi} \in \Xi. \tag{64}$$

Notice that all decision variables appear linearly in Problem (34). Nevertheless, the objective and constraints are non-linear functions of $\boldsymbol{\xi}$.

We now define the non-linear lifting operator $L : \mathbb{R}^k \to \mathbb{R}^{k'}$ such that

$$L : \mathbb{R}^k \to \mathbb{R}^{k'}, \qquad L(\boldsymbol{\xi}) = \begin{pmatrix} \boldsymbol{\xi} \\ G(\boldsymbol{\xi}) \\ G(\boldsymbol{\xi})\xi_1 \\ \vdots \\ G(\boldsymbol{\xi})\xi_k \end{pmatrix}, \tag{65}$$

$k' = k+g+gk$. Note that including both components $G(\boldsymbol{\xi})$ and $G(\boldsymbol{\xi})\xi_1$ in $L(\boldsymbol{\xi})$ is redundant as by construction $\xi_1 = 1$. Nevertheless, in the following we adopt formulation (65) for easy of exposition. We also define matrices $R_\xi \in \mathbb{R}^{k\times k'}$, and $R_G \in \mathbb{R}^{q\times k'}$ such that

$$R_\xi L(\boldsymbol{\xi}) = \boldsymbol{\xi}, \qquad R_G L(\boldsymbol{\xi}) = G(\boldsymbol{\xi}). \tag{66}$$

In addition, we define $F : \mathbb{R}^{k\times g} \to \mathbb{R}^{k'}$ that expresses the quadratic polynomials $\boldsymbol{\xi}^\top B_\mu Y G(\boldsymbol{\xi})$, as linear functions of $L(\boldsymbol{\xi})$ through the following constraints:

$$\mathbf{y}'_\mu = F(B_\mu Y), \quad \mathbf{y}'^\top_\mu L(\boldsymbol{\xi}) = \boldsymbol{\xi}^\top B_\mu Y G(\boldsymbol{\xi}), \;\; \mathbf{y}'_\mu \in \mathbb{R}^{k'}, \quad \mu = 1,\dots,m, \quad \forall \boldsymbol{\xi} \in \Xi. \tag{67}$$

Constraints $\mathbf{y}'_\mu = F(B_\mu Y)$, $\mu = 1,\dots,m$, are linear, since $F(\cdot)$ effectively reorders the entries of matrix $B_\mu Y$ into the vector $\mathbf{y}'_\mu$. Using (65), (66) and (67), Problem (64) can now

be rewritten into the following optimization problem.

$$\begin{aligned} &\text{minimize} \quad \mathrm{Tr}\Big(M R_\xi^\top D^\top Y R_G\Big) \\ &\text{subject to} \quad Y \in \mathbb{Z}^{q\times g}, \\ &\qquad \mathbf{y}_\mu'^\top L(\boldsymbol{\xi}) \le H_\mu^\top R_\xi L(\boldsymbol{\xi}), \qquad \mu = 1,\dots,m, \\ &\qquad \mathbf{y}_\mu' = F(B_\mu Y),\ \mathbf{y}_\mu' \in \mathbb{R}^{k'}, \quad \mu = 1,\dots,m, \\ &\qquad 0 \le Y R_G L(\boldsymbol{\xi}) \le \boldsymbol{e}, \end{aligned} \Bigg\} \quad \forall \boldsymbol{\xi} \in \Xi. \tag{68}$$

Problem (68) has similar structure as Problem (38), i.e., both Y and $L(\boldsymbol{\xi})$ appear linearly in the constraints. Therefore, in the following we redefine Problem (68) in a higher dimensional space and apply Proposition 4 to reformulate the problem into a mixed-integer optimization problem.

We define the uncertain vector $\boldsymbol{\xi}' = (\boldsymbol{\xi}_{1,1}'^\top, \boldsymbol{\xi}_{1,2}'^\top, \boldsymbol{\xi}_{2,1}'^\top, \dots, \boldsymbol{\xi}_{2,k}'^\top)^\top \in \mathbb{R}^{k'}$ such that $\boldsymbol{\xi}' = L(\boldsymbol{\xi})$ and $\boldsymbol{\xi}_{1,1}' = \boldsymbol{\xi}$, $\boldsymbol{\xi}_{1,2}' = G(\boldsymbol{\xi})$, and $\boldsymbol{\xi}_{2,i}' = G(\boldsymbol{\xi})\xi_i$ for $i = 1,\dots,k$. $\boldsymbol{\xi}'$ has probability measure $\mathbb{P}_{\xi'}$ defined as in (39) and support $\Xi' = L(\Xi)$. Problem (68) can now be written as the equivalent semi-infinite problem defined on the lifted space $\boldsymbol{\xi}'$.

$$\begin{aligned} &\text{minimize} \quad \mathrm{Tr}\Big(M' R_\xi^\top D^\top Y R_G\Big) \\ &\text{subject to} \quad Y \in \mathbb{Z}^{q\times g}, \\ &\qquad \mathbf{y}_\mu'^\top \boldsymbol{\xi}' \le H_\mu^\top R_\xi\, \boldsymbol{\xi}', \qquad \mu = 1,\dots,m, \\ &\qquad \mathbf{y}_\mu' = F(B_\mu Y),\ \mathbf{y}_\mu' \in \mathbb{R}^{k'}, \quad \mu = 1,\dots,m, \\ &\qquad 0 \le Y R_G \boldsymbol{\xi}' \le \boldsymbol{e}, \end{aligned} \Bigg\} \quad \forall \boldsymbol{\xi}' \in \Xi', \tag{69}$$

We will construct the convex hull of $\overline{\Xi}'$ in the same way as in Section 4.2.2. Notice that $\mathrm{conv}(\overline{\Xi}')$ will have the same number of extreme points as the convex hull defined through the lifting $L(\boldsymbol{\xi}) = (\boldsymbol{\xi}^\top, G(\boldsymbol{\xi})^\top)^\top$, see Figure 5. By defining the partitions Ξ_p of Ξ as in (43), and $\widehat{G}_p(\boldsymbol{\xi})$ as in (45), it is easy to see that $\mathrm{conv}(\overline{\Xi}')$ can be described as a convex

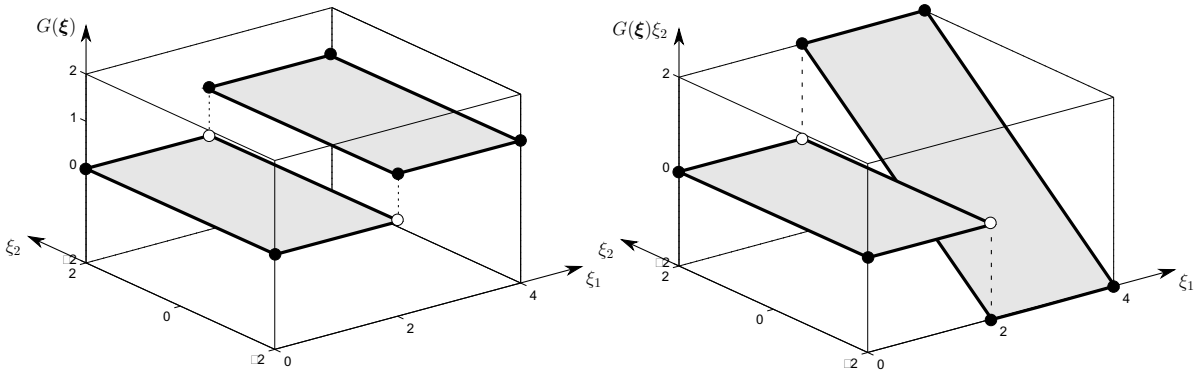

Figure 5: *Visualization of* $L(\xi) = (\xi_1, \xi_2, G(\xi))$ *(left) and* $\widetilde{L}(\xi) = (\xi_1, \xi_2, G(\xi)\xi_2)$ *(right) where* $G(\xi) = \mathbf{1}(\xi_1 \geq 2)$*, for* $\xi_1 \in [0, 4]$ *and* $\xi_2 \in [-2, 2]$*. Here,* $V = \{(0,-2), (0,2), (2,-2), (2,2), (4,-2), (4,2)\}$*. The convex hull of* $L(\xi)$ *is constructed by taken convex combinations of* $L(\xi)$ *for all* $\xi \in V$ *and* $(2, -2, \widehat{G}(2,-2)))^\top, (2, 2, \widehat{G}((2,2)))^\top$*, and the convex hull of* $\widetilde{L}(\xi)$ *is constructed by taken convex combinations of* $\widetilde{L}(\xi)$ *for all* $\xi \in V$ *and* $(2, -2, \widehat{G}(2,-2)(-2))^\top$*,* $(2, 2, \widehat{G}(2,2)2)^\top$*.*

combination of the following points.

$$\begin{pmatrix} \boldsymbol{\xi} \\ \widehat{G}_p(\boldsymbol{\xi}) \\ \widehat{G}_p(\boldsymbol{\xi})\xi_1 \\ \vdots \\ \widehat{G}_p(\boldsymbol{\xi})\xi_k \end{pmatrix}, \qquad \boldsymbol{\xi} \in V(p),\ p = 1, \ldots, P.$$

The following proposition gives the polyhedral representation of the convex hull of $\overline{\Xi}'$.

Proposition 6 *Let* $L(\cdot)$ *being defined in* (65) *and* $G(\cdot)$ *being defined in* (33b)*. Then, the tight, closed, outer approximation for the convex hull of* $\overline{\Xi}'$ *is given by the following*

polyhedron:

$$
\begin{aligned}
conv(\overline{\Xi}') = \Big\{ \boldsymbol{\xi}' = (\boldsymbol{\xi}_{1,1}'^{\top}, \boldsymbol{\xi}_{1,2}'^{\top}, \boldsymbol{\xi}_{2,1}'^{\top}, \ldots, \boldsymbol{\xi}_{2,k}'^{\top})^{\top} \in \mathbb{R}^{k'} : \exists \zeta_p(\boldsymbol{v}) \in \mathbb{R}_+, \, \forall \boldsymbol{v} \in V(p), \, p = 1, \ldots, P, \\
\sum_{p=1}^{P} \sum_{v \in V(p)} \zeta_p(\boldsymbol{v}) = 1, \\
\boldsymbol{\xi}_{1,1}' = \sum_{p=1}^{P} \sum_{v \in V(p)} \zeta_p(\boldsymbol{v}) \boldsymbol{v}, \\
\boldsymbol{\xi}_{1,2}' = \sum_{p=1}^{P} \sum_{v \in V(p)} \zeta_p(\boldsymbol{v}) \widehat{G}_p(\boldsymbol{v}), \\
\boldsymbol{\xi}_{2,i}' = \sum_{p=1}^{P} \sum_{v \in V(p)} \zeta_p(\boldsymbol{v}) \widehat{G}_p(\boldsymbol{v}) v_i, \; i = 1, \ldots, k \Big\}.
\end{aligned}
\tag{70}
$$

Proof Proposition 6 can be proven using similar arguments as in Proposition 3. ∎

The description of polyhedron (70) inherits the same exponential complexity as (46). That is, the number of constraints in (70) depends on the cardinality of V, which in the worst-case, is constructed using the extreme points of $P = 2^g$ partitions of Ξ. Nevertheless, the description (70) provides a tight outer approximation for conv $(\overline{\Xi}')$.

Using Proposition 4 together with the polyhedron (70), we can now reformulate Problem (69) into the following mixed-integer linear optimization problem.

$$
\begin{aligned}
\text{minimize} \quad & \operatorname{Tr}\Big(M' R_{\xi}^{\top} D^{\top} Y R_G \Big) \\
\text{subject to} \quad & Y \in \mathbb{Z}^{q \times g}, \; \Gamma \in \mathbb{R}_+^{q \times l'}, \; \Theta \in \mathbb{R}_+^{q \times l'}, \\
& \left.
\begin{aligned}
& \mathbf{y}_{\mu}' \in \mathbb{R}^{k'}, \; \boldsymbol{\lambda}_{\mu} \in \mathbb{R}_+^{m \times l'}, \\
& \mathbf{y}_{\mu}' + \boldsymbol{\lambda}_{\mu} W' = R_{\xi}^{\top} H_{\mu}, \; \boldsymbol{\lambda}_{\mu} U' = 0, \; \boldsymbol{\lambda}_{\mu} \boldsymbol{h}' \geq 0 \\
& \mathbf{y}_{\mu}' = F(B_{\mu} Y), \; \mathbf{y}_{\mu}' \in \mathbb{R}^{k'},
\end{aligned}
\right\} \quad \mu = 1, \ldots, m, \\
& Y R_G = \Gamma W', \; \Gamma U' = 0, \; \Gamma \boldsymbol{h}' \geq 0, \\
& \boldsymbol{e} \boldsymbol{e}_1^{\top} - Y R_G = \Theta W', \; \Theta U' = 0, \; \Theta \boldsymbol{h}' \geq 0.
\end{aligned}
\tag{71}
$$

Here, the auxiliary variables $\boldsymbol{\lambda}_{\mu} \in \mathbb{R}_+^{m \times l'}$ are associated with constraints $\mathbf{y}_{\mu}'^{\top} \boldsymbol{\xi}' \leq H_{\mu}^{\top} R_{\xi} \, \boldsymbol{\xi}'$, for $\mu = 1, \ldots, m$, and $\Gamma \in \mathbb{R}_+^{q \times l'}$ and $\Theta \in \mathbb{R}_+^{q \times l'}$ with constraints $0 \leq Y R_G \boldsymbol{\xi}'$ and $Y R_G \boldsymbol{\xi}' \leq \boldsymbol{e}$, respectively. Problem (71) is the exact reformulation of Problem (69), since (70) is a tight

outer approximation of the convex hull of $\overline{\Xi}'$, and thus *the solution of Problem* (71) *achieves the best binary decision rule associated with* $G(\cdot)$ *in* (33b). Nevertheless, the size of Problem (71) is affected by the exponential growth of the constraints in conv $(\overline{\Xi}')$. One can mitigate this exponential growth by considering simplified structures of Ξ and $G(\cdot)$, following similar guidelines as those those discussed in Section 4.2.3.

4.4 Binary Decision Rules for Multistage Problems

In this section, we extend the methodology presented in Section 4.2 to cover multistage adaptive optimization problems with fixed-recourse. The mathematical formulations presented can easily be adapted to random-recourse problems by following the guidelines of Section 4.3.

The dynamic decision process considered can be described as follows: A decision maker first observes an uncertain parameter $\boldsymbol{\xi}_1 \in \mathbb{R}^{k_1}$ and then takes a binary decision $\boldsymbol{y}_1(\boldsymbol{\xi}_1) \in \{0,1\}^{q_1}$. Subsequently, a second uncertain parameter $\boldsymbol{\xi}_2 \in \mathbb{R}^{k_2}$ is revealed, in response to which the decision maker takes a second decision $\boldsymbol{y}_2(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2) \in \{0,1\}^{q_2}$. This sequence of alternating observations and decisions extends over T time stages. To enforce the non-anticipative structure of the decisions, a decision taken at stage t can only depend on the observed parameters up to and including stage t, i.e., $\boldsymbol{y}_t(\boldsymbol{\xi}^t)$ where $\boldsymbol{\xi}^t = (\boldsymbol{\xi}_1^\top, \ldots, \boldsymbol{\xi}_t^\top)^\top \in \mathbb{R}^{k^t}$, with $k^t = \sum_{s=1}^{t} k_s$. For consistency with the previous sections, and with slight abuse of notation, we assume that $k_1 = 1$ and $\boldsymbol{\xi}_1 = 1$. As before, setting $\boldsymbol{\xi}_1 = 1$ is a non-restrictive assumption which allows to represent affine functions of the non-degenerate outcomes $(\boldsymbol{\xi}_2^\top, \ldots, \boldsymbol{\xi}_t^\top)^\top$ in a compact manner as linear functions of $(\boldsymbol{\xi}_1^\top, \ldots, \boldsymbol{\xi}_t^\top)^\top$. We denote by $\boldsymbol{\xi} = (\boldsymbol{\xi}_1^\top, \ldots, \boldsymbol{\xi}_T^\top)^\top \in \mathbb{R}^k$ the vector of all uncertain parameters, where $k = k^T$. Finally, we denote by $\mathcal{B}_{k^t, q_t}$ the space of binary functions from $\mathbb{R}^{k^t}$ to $\{0,1\}^{q_t}$.

Given matrices $B_{ts} \in \mathbb{R}^{m_t \times q_s}$, $D_t \in \mathbb{R}^{q_t \times k^t}$ and $H_t \in \mathbb{R}^{m_t \times k^t}$, and a probability measure $\mathbb{P}_\xi$ supported on set Ξ given by (90), for the uncertain vector $\boldsymbol{\xi} \in \mathbb{R}^k$, we are interested in

choosing binary functions $\boldsymbol{y}_t(\cdot) \in \mathcal{B}_{k^t,q_t}$ in order to solve:

$$\left.\begin{array}{ll} \text{minimize} & \mathbb{E}_\xi \left(\sum_{t=1}^{T} (D_t \boldsymbol{\xi}^t)^\top \boldsymbol{y}_t(\boldsymbol{\xi}^t) \right) \\ \text{subject to} & \boldsymbol{y}_t(\cdot) \in \mathcal{B}_{k^t,q_t}, \\ & \sum_{s=1}^{t} B_{ts} \boldsymbol{y}_s(\boldsymbol{\xi}^s) \leq H_t \boldsymbol{\xi}^t, \end{array}\right\} \quad t = 1, \ldots, T, \ \ \forall \boldsymbol{\xi} \in \Xi. \tag{72}$$

Problem (72) involves a continuum of decision variables and inequality constraints. Therefore, in order to make the problem amenable to numerical solutions, we restrict the feasible region of the binary functions $\boldsymbol{y}_t(\cdot) \in \mathcal{B}_{k^t,q_t}$ to admit the following piecewise constant structure:

$$\left.\begin{array}{l} \boldsymbol{y}_t(\boldsymbol{\xi}) = Y_t G^t(\boldsymbol{\xi}),\ Y \in \mathbb{Z}^{q_t \times g^t}, \\ 0 \leq Y_t G^t(\boldsymbol{\xi}) \leq \boldsymbol{e}, \end{array}\right\} \quad t = 1, \ldots, T, \quad \forall \boldsymbol{\xi} \in \Xi, \tag{73a}$$

where $G^t : \mathbb{R}^k \to \{0,1\}^{g^t}$ can be expressed by $G^t(\cdot) = (G_1(\cdot)^\top, \ldots, G_t(\cdot)^\top)$, and $G_t : \mathbb{R}^{k^t} \to \{0,1\}^{g_t}$ are the piecewise constant functions:

$$G_1(\boldsymbol{\xi}_1) := 1, \quad G_{t,i}(\boldsymbol{\xi}) := \mathbf{1}\left(\boldsymbol{\alpha}_{t,i}^\top \boldsymbol{\xi}^t \geq \beta_{t,i} \right), \quad i = 1, \ldots, g_t,\ t = 2, \ldots, T, \tag{73b}$$

for given $\boldsymbol{\alpha}_{t,i} \in \mathbb{R}^{k^t}$ and $\beta_{t,i} \in \mathbb{R}$, $i = 1, \ldots, g$, $t = 2, \ldots, T$. Here, we assume that the sets $\{\boldsymbol{\xi} \in \Xi \ : \ \boldsymbol{\alpha}_{t,i}^\top \boldsymbol{\xi}^t \geq \beta_{t,i}\}$ and $\{\boldsymbol{\xi} \in \Xi \ : \ \boldsymbol{\alpha}_{t,i}^\top \boldsymbol{\xi}^t \leq \beta_{t,i}\}$ are non-empty for all $i = 1, \ldots, g_t$, $t = 2, \ldots, T$, and $\boldsymbol{\alpha}_{t,i}^\top \boldsymbol{\xi}^t - \beta_{t,i} \neq \boldsymbol{\alpha}_{t,j}^\top \boldsymbol{\xi}^t - \beta_{t,j}$ for all $\boldsymbol{\xi} \in \Xi$ where $i, j \in \{1, \ldots, g_t\}$, $i \neq j$ and $t = 2, \ldots, T$. The dimension of $G^t(\cdot)$ is g^t, with $g^t = \sum_{s=1}^{t} g_s$. The non-anticipativity of $\boldsymbol{y}_t(\cdot)$ is ensured by restricting $G^t(\cdot)$ to depend only on random parameters up to and including stage t. Applying decision rules (73) to Problem (72) yields the following semi-infinite problem.

$$\left.\begin{array}{ll} \text{minimize} & \mathbb{E}_\xi \left(\sum_{t=1}^{T} (D_t \boldsymbol{\xi}^t)^\top Y_t G^t(\boldsymbol{\xi}) \right) \\ \text{subject to} & Y_t \in \mathbb{Z}^{q_t \times g^t}, \\ & \sum_{s=1}^{t} B_{ts} Y_s G^s(\boldsymbol{\xi}) \leq H_t \boldsymbol{\xi}^t, \\ & 0 \leq Y_t G^t(\boldsymbol{\xi}) \leq \boldsymbol{e} \end{array}\right\} \quad t = 1, \ldots, T, \ \ \forall \boldsymbol{\xi} \in \Xi. \tag{74}$$

We can now use the lifting techniques to first express Problem (74) in a higher dimensional space, compute the convex hull associated with the non-convex uncertainty set of the lifted problem, and finally reformulate the semi-infinite structure using Proposition 4. We define lifting $L^t : \mathbb{R}^k \to \mathbb{R}^{k't}$, such that $L^t(\cdot) = (L_1(\cdot)^\top, \ldots, L_t(\cdot)^\top)^\top$, where

$$L_t : \mathbb{R}^{k_t} \to \mathbb{R}^{k'_t}, \qquad L_t(\boldsymbol{\xi}) = (\boldsymbol{\xi}_t^\top, G_t(\boldsymbol{\xi})^\top)^\top,\ t = 1, \ldots, T, \tag{75a}$$

Here, $k'_t = k_t + g_t$. By convention $L(\cdot) = (L_1(\cdot)^\top, \ldots, L_T(\cdot)^\top)^\top$ and $k'^t = \sum_{s=1}^t k'_s$ with $k' = k'^T$. In addition, we define matrices, $R_{\xi,t} \in \mathbb{R}^{k^t \times k'}$ and $R_{G,t} \in \mathbb{R}^{g^t \times k'}$ such that

$$R_{\xi,t} L(\boldsymbol{\xi}) = \boldsymbol{\xi}^t, \qquad R_{G,t} L(\boldsymbol{\xi}) = G(\boldsymbol{\xi})^t, \quad t = 1, \ldots, T. \tag{75b}$$

Using (75), Problem (74) can now be rewritten in the following form:

$$\begin{array}{ll} \text{minimize} & \mathbb{E}_\xi \left(\displaystyle\sum_{t=1}^{T} (D_t R_{\xi,t} L(\boldsymbol{\xi}))^\top Y_t R_{G,t} L(\boldsymbol{\xi}) \right) \\ \text{subject to} & Y_t \in \mathbb{Z}^{q_t \times g^t}, \\ & \displaystyle\sum_{s=1}^{t} B_{ts} Y_s R_{G,s} L(\boldsymbol{\xi}) \leq H_t R_{\xi,t} L(\boldsymbol{\xi}), \\ & 0 \leq Y_t R_{G,t} L(\boldsymbol{\xi}) \leq \boldsymbol{e} \end{array} \Bigg\} \quad t = 1, \ldots, T, \ \ \forall \boldsymbol{\xi} \in \Xi, \tag{76}$$

Problem (76) has similar structure as Problem (38), i.e., both Y_t and $L(\boldsymbol{\xi})$ appear linearly in the constraints. Therefore, in the following we redefine Problem (76) in a higher dimensional space and apply Proposition 4 to reformulate the problem into a mixed-integer optimization problem.

We now define the uncertain vector $\boldsymbol{\xi}' = (\boldsymbol{\xi}_1'^\top, \ldots, \boldsymbol{\xi}_T'^\top)^\top \in \mathbb{R}^{k'}$ such that $\boldsymbol{\xi}' = L(\boldsymbol{\xi})$ and $\boldsymbol{\xi}'_t = L_t(\boldsymbol{\xi})$ for $t = 1, \ldots, T$. $\boldsymbol{\xi}'$ has probability measure $\mathbb{P}_{\xi'}$ defined using (39), and probability measure support $\Xi' = L(\Xi)$. Using the definition of $\boldsymbol{\xi}'$, the lifted semi-infinite problem is

given as follows:

$$\left.\begin{array}{ll} \text{minimize} & \mathbb{E}_{\xi'}\left(\sum_{t=1}^{T}(D_t R_{\xi,t}\boldsymbol{\xi}')^\top Y_t R_{G,t}\boldsymbol{\xi}'\right) \\ \text{subject to} & Y_t \in \mathbb{Z}^{q_t\times g^t}, \\ & \sum_{s=1}^{t} B_{ts} Y_s R_{G,s}\boldsymbol{\xi}' \le H_t R_{\xi,t}\boldsymbol{\xi}', \\ & 0 \le Y_t R_{G,t}\boldsymbol{\xi}' \le \boldsymbol{e} \end{array}\right\} \quad t=1,\ldots,T,\ \ \forall \boldsymbol{\xi}' \in \Xi'. \tag{77}$$

The definition of $L(\cdot)$ in (75a) share almost identical structure as $L(\cdot)$ in (37). Therefore, the convex hull of $\overline{\Xi}'$ can be expressed as a slight variant of the polyhedron given in (46), and will constitute a tight outer approximation of $\text{conv}(\overline{\Xi}')$. Using Proposition 4 together with the polyhedral representation of $\text{conv}(\overline{\Xi}')$, we can now reformulate Problem (69) into the following mixed-integer linear optimization problem.

$$\left.\begin{array}{ll} \text{minimize} & \sum_{t=1}^{T} \text{Tr}\Big(M' R_{\xi,t}^\top D_t^\top Y_t R_{G,t}\Big) \\ \text{subject to} & Y_t \in \mathbb{Z}^{q_t\times g^t},\ \Lambda \in \mathbb{R}_+^{m_t\times l'},\ \Gamma \in \mathbb{R}_+^{q^t\times l'},\ \Theta \in \mathbb{R}_+^{q^t\times l'} \\ & \sum_{s=1}^{t} B_{ts} Y R_{G,s} + \Lambda_t W' = H_t R_{\xi,t},\ \Lambda_t U' = 0,\ \Lambda_t \boldsymbol{h}' \ge 0, \\ & Y_t R_{G,t} = \Gamma_t W',\ \Gamma_t U' = 0,\ \Gamma_t \boldsymbol{h}' \ge 0, \\ & \boldsymbol{e}\boldsymbol{e}_1^\top - Y_t R_{G,t} = \Theta_t W',\ \Theta_t U' = 0,\ \Theta_t \boldsymbol{h}' \ge 0. \end{array}\right\} \quad t=1,\ldots,T, \tag{78}$$

where $M' \in \mathbb{R}^{k'\times k'}$, $M' = \mathbb{E}_{\xi'}(\boldsymbol{\xi}'\boldsymbol{\xi}'^\top)$. Once more, we emphasize that Problem (78) is the exact reformulation of Problem (77). Therefore, *the solution of Problem* (78) *achieves the best binary decision rule associated with* $G(\cdot)$ *in* (33b) at the cost of the exponential growth of its size with respect to the description of Ξ and the complexity of the binary decision rules (73). One can mitigate this exponential growth by considering simplified structures of Ξ and $G(\cdot)$, following similar guidelines as those those discussed in Section 4.2.3.

4.5 Technical Proofs

In this section, we will provide the proof for Theorem 1. To do this, we need the following auxiliary results given in Lemmas 1, 2 and 3. Lemma 1 gives the complexity of the integer feasibility problem, which will be used in Lemmas 2 and 3 to prove that checking feasibility of a semi-infinite constraint involving indicator functions is NP-hard.

Lemma 1 (Integer feasibility problem) *The following decision problem is* NP*-hard:*

$$\begin{aligned} &\textsc{Instance.} \quad W \in \mathbb{R}^{l\times k},\, \boldsymbol{h} \in \mathbb{R}^l \text{ and } N \in \mathbb{Z}_+ \text{ with } N < k. \\ &\textsc{Question.} \quad \text{Is there } \boldsymbol{\xi} \in \{0,1\}^k \text{ such that } W\boldsymbol{\xi} \geq \boldsymbol{h},\, \textstyle\sum_{i=1}^k \xi_i = N? \end{aligned} \tag{79}$$

Proof See [43]. ∎

Lemma 2 (Equivalent integer feasibility problem) *The following decision problem is* NP*-hard:*

$$\begin{aligned} &\textsc{Instance.} \quad W \in \mathbb{R}^{l\times k},\, \boldsymbol{h} \in \mathbb{R}^l \text{ and } N \in \mathbb{Z}_+ \text{ with } N < k. \\ &\textsc{Question.} \quad \text{Is there } \boldsymbol{\xi} \in [0,2]^k \text{ such that } W\boldsymbol{\xi} \geq \boldsymbol{h},\, \textstyle\sum_{i=1}^k \xi_i = N,\, \sum_{i=1}^k \mathbf{1}(\xi_i \geq 1) = N? \end{aligned} \tag{80}$$

Proof In the following, we will show that if there exists $\boldsymbol{\xi}$ that satisfies the assertion, then $\boldsymbol{\xi}$ must be in $\boldsymbol{\xi} \in \{0,1\}^k$. If that is the case, then decision problem (80) reduces to the decision problem (79), which from Lemma 1 we know to be NP-hard. We will prove this by contradiction. Assume, with loss of generality, that there exists $\boldsymbol{\xi}$ with $\xi_1 > 1$ that satisfy $W\boldsymbol{\xi} \geq \boldsymbol{h}$, $\sum_{i=1}^k \xi_i = N$, $\sum_{i=1}^k \mathbf{1}(\xi_i \geq 1) = N$. Since $\xi_1 > 1$, then $\sum_{i=2}^k \xi_i < N-1$ and as a result $\sum_{i=2}^k \mathbf{1}(\xi_i \geq 1) < N-1$, which contradicts the assumption that $\sum_{i=1}^k \mathbf{1}(\xi_i \geq 1) = N$. Now, assume that there exists $\boldsymbol{\xi}$ with $0 < \xi_1 < 1$ that satisfy $W\boldsymbol{\xi} \geq \boldsymbol{h}$, $\sum_{i=1}^k \xi_i = N$, $\sum_{i=1}^k \mathbf{1}(\xi_i \geq 1) = N$. Since $0 < \xi_1 < 1$, then $\sum_{i=2}^k \mathbf{1}(\xi_i \geq 1) < N-1$, which again contradicts the assumption that $\sum_{i=1}^k \mathbf{1}(\xi_i \geq 1) = N$. Therefore, we conclude that $\boldsymbol{\xi} \in \{0,1\}^k$, and thus the decision problem (80) is NP-hard. ∎

The following lemma provides the key ingredient for proving Theorem 1.

Lemma 3 *The following decision problem is* NP*-hard:*

$$\begin{aligned} &\textsc{Instance}. \quad \textit{A convex polytope } \Xi \subset \mathbb{R}^k \textit{ and } \tau \in \mathbb{R}. \\ &\textsc{Question}. \quad \textit{Do all } \boldsymbol{\xi} \in \Xi \textit{ satisfy } \textstyle\sum_{i=1}^k \mathbf{1}(\xi_i \geq 1) \leq \tau? \end{aligned} \tag{81}$$

Proof We will show that an instance of decision problem (81) can be reduced to an equivalent integer feasibility problem (80). Consider the instance $\tau = N - 1$, with $N < k$, and Ξ be given by

$$\Xi := \left\{ \boldsymbol{\xi} \in [0,2]^k \,:\, W\boldsymbol{\xi} \geq \boldsymbol{h},\ \sum_{i=1}^k \xi_i = N \right\}. \tag{82}$$

Therefore, the decision problem (81) evaluates to true if there is no $\boldsymbol{\xi}$ such that

$$\boldsymbol{\xi} \in [0,2]^k \text{ such that } W\boldsymbol{\xi} \geq, \sum_{i=1}^k \xi_i = N, \sum_{i=1}^k \mathbf{1}(\xi_i \geq 1) = N. \tag{83}$$

Checking the latter assertion is equivalent to checking if the decision problem (80) holds. Notice that by construction, there is no $\boldsymbol{\xi}$ such that $\sum_{i=1}^k \mathbf{1}(\xi_i \geq 1) > N$. Using Lemma 2, we can thus deduce that (81) is an NP-hard problem. ∎

We know have all the ingredients to prove Theorem 1.

Proof of Theorem 1 Let Ξ be a convex polytope given by

$$\Xi = \left\{ \boldsymbol{\xi} \in [0,2]^k \,:\, W\boldsymbol{\xi} \geq \boldsymbol{h}, \sum_{i=1}^k \xi_i = N \right\},$$

and denote by $\mathbb{P}_\xi$ the uniform distribution on Ξ. We define the following instance of (30).

$$\left.\begin{array}{ll} \text{minimize} & 0 \\ \text{subject to} & \boldsymbol{y}(\cdot) \in \mathcal{B}_{k,k}, \\ & \xi_i - 1 \leq y_i(\boldsymbol{\xi}) \leq \xi_i \ \ \forall i = 1, \ldots, k, \\ & \displaystyle\sum_{i=1}^k y_i\left(\boldsymbol{\xi}\right) \leq N - 1, \end{array}\right\} \quad \forall \boldsymbol{\xi} \in \Xi. \tag{84}$$

The optimal solution of Problem (84) is given by $y_i^*\left(\boldsymbol{\xi}\right) := \mathbf{1}(\xi_i \geq 1)$, $i = 1, \ldots, k$. We thus

have

$$\sum_{i=1}^{k} \mathbf{1}(\xi_i \geq 1) \leq N-1 \quad \forall \boldsymbol{\xi} \in \Xi \quad \Longleftrightarrow \quad \text{Problem (84) is feasible.}$$

Hence, Lemma 3 implies that checking the feasibility of Problem (84) is NP-hard. We now set $\boldsymbol{\alpha}_i = \boldsymbol{e}_i$ and $\beta_i = 1$, for $i = 2, \ldots, k$, in the description of $G(\cdot)$ in (33b). By construction, there exists $Y \in \{0,1\}^{k \times g}$ such that $\boldsymbol{y}^*(\boldsymbol{\xi}) = YG(\boldsymbol{\xi})$, that achieves the same objective value in all Problems (38), (42) and (84). The above arguments allow us to conclude that

$$\sum_{i=1}^{k} \mathbf{1}(\xi_i \geq 1) \leq N-1 \quad \forall \boldsymbol{\xi} \in \Xi \quad \Longleftrightarrow \quad \text{Problem (38) is feasible} \underset{\text{Propositions 1 \& 2}}{\Longleftrightarrow} \text{Problem (42) is feasible.}$$

Lemma 3 implies that Problems (38) and (42) are NP-hard, even when $Y \in \mathbb{Z}^{k \times g}$ is relaxed to $Y \in \mathbb{R}^{k \times g}$. ∎

4.6 Conclusions

In this chapter, we present linearly parameterised binary decision rule structures that can be used in conjunction with real-valued decision rules appearing in the literature, for solving multistage adaptive mixed-integer optimization problems. We provide a systematic way to reformulate the binary decision rule problem into a finite dimensional mixed-integer linear optimization problem, and we identify instances where the size of this problem grows polynomially with respect to the input data. The theory presented covers both fixed-recourse and random-recourse problems. Our numerical results demonstrate the effectiveness of the proposed binary decision rules and show that they are (*i*) highly scalable and (*ii*) provide significant improvements in solution quality.

5 Alternating Directing Methofd of Multipliers (ADMM)

5.1 Introduction

Distributed optimization algorithms sparked great interest in the research community [12, 58, 57], with the algorithms being applied a diverse range of applications such as image processing [25, 1], signal processing [41, 49], wireless communications [50, 64] and many more. These algorithms overcame the inherit limitation of classical optimization algorithms that require to be solved in a centralized manner. This is done by exploiting any special structure the problem might have, such as a network type structure. In this setting, the distributed optimization algorithms decompose the problem into a finite number of subproblems that can be solved independently and efficiently, with global feasibility and optimality being achieved through the solution of a sequence of such optimization problems.

The alternating direction method of multipliers (ADMM) has gained great momentum in recent years as it is well suited for solving distributed convex optimization, and in particular large-scale problems arising in statistics, machine learning, and related areas [21]. In our work, we propose to use ADMM together with policy design resulting from the decision rule approximations and dynamic programming. Using ADMM provides two distinct advantages: (i) One can directly model multiple entities, each having their own individual characteristics such as costs and constraints, trying to optimize their own problems but together satisfying global coupling constraints that need to be respected by everyone; (ii) Since the ADMM algorithm allows to solve optimization problems in parallel, greater computational efficiency is achieved. Preliminary results on large multistage stochastic optimization problems involving $\mathcal{O}(100)$ entities has shown to produce high quality solutions in $\mathcal{O}(5)$ minutes.

5.2 ADMM for Static Convex Problems

In this section, we overview the basic concepts behind the ADMM algorithm, addressing static convex optimization problems having the following structure. Given convex functions $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$ and convex sets $\mathcal{X}_i \subseteq \mathbb{R}^{n_i}$ for $i = 1, \ldots, N$, and convex set $\mathcal{C} \subseteq \mathbb{R}^{n_1+\ldots+n_N}$, we

are interested to choose vector $\boldsymbol{x} = (\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top)^\top$ with $\boldsymbol{x}_i \in \mathbb{R}^{n_i}$, $i = 1, \ldots, N$, in order to solve:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) \\ \text{subject to} \quad & \boldsymbol{x}_i \in \mathcal{X}_i, \;\; i = 1, \ldots, N, \\ & \boldsymbol{x} \in \mathcal{C}. \end{aligned} \tag{85}$$

Problem (85) can be thought as an optimization problem involving N agents, where $\boldsymbol{x}_i$ are the decisions of agent i. The objective function is decoupled with respect to the N agents, but their decisions are coupled through constraint $\boldsymbol{x} \in \mathcal{C}$. Notice that in the absence of constraint $\boldsymbol{x} \in \mathcal{C}$, Problem (85) can be completely decoupled among the N. Problem (85) can be equivalently written as the following optimization problem

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) + g(\boldsymbol{z}) \\ \text{subject to} \quad & \boldsymbol{x}_i \in \mathcal{X}_i, \;\; i = 1, \ldots, N, \\ & \boldsymbol{x} - \boldsymbol{z} = 0, \end{aligned} \tag{86}$$

where $\boldsymbol{z} = (\boldsymbol{z}_1^\top, \ldots, \boldsymbol{z}_N^\top)^\top$ and $g : \mathbb{R}^{n_1+\ldots+n_N}$ is the indicator function given by

$$g(\boldsymbol{z}) = \begin{cases} 0 & \text{if } \boldsymbol{z} \in \mathcal{C}, \\ \infty & \text{if } \boldsymbol{z} \notin \mathcal{C}. \end{cases}$$

Notice that in the absence of constraint $\boldsymbol{x} - \boldsymbol{z} = 0$, Problem (86) is again decoupled in the decision variables $\boldsymbol{x}_i$, $i = 1, \ldots, N$ and $\boldsymbol{z}$.

We now introduce the *augmented Lagrangian* of Problem (86) given by

$$L_\rho(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}) = \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) + g(\boldsymbol{z}) + \boldsymbol{y}^\top(\boldsymbol{x} - \boldsymbol{z}) + \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{z}\|_2^2.$$

Here, $\boldsymbol{y} \in \mathbb{R}^{n_1+\ldots+n_N}$ are the dual variables associated with constraint $\boldsymbol{x} - \boldsymbol{z} = 0$, and $\rho > 0$ is called the penalty parameter. In contrast with the Lagrangian of Problem (86), the augmented Lagrangian additionally contains regularization term $\frac{\rho}{2}\|\boldsymbol{x}-\boldsymbol{z}\|_2^2$. This additional term brings robustness to the dual ascent methods without the need to impose strong assumptions like string convexity or finiteness for $f_i(\cdot)$. Also, notice that if one sets $\rho = 0$,

then $L_\rho(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y})$ corresponds exactly to the Lagrangian of Problem (86). By introducing the scaled dual variable $\boldsymbol{u} = (\boldsymbol{u}_1^\top, \ldots, \boldsymbol{u}_N^\top)^\top$ with $\boldsymbol{u}_i \in \mathbb{R}^{n_i}$, $i = 1, \ldots, N$, the augmented Lagrangian (5.2) can be equivalently written as

$$L_\rho(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{u}) = \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) + g(\boldsymbol{z}) + \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{z} + \boldsymbol{u}\|_2^2.$$

Therefore, the ADMM algorithm can be described as follows.

$$\begin{aligned}
\boldsymbol{x}_i^{k+1} &:= \underset{x_i \in \mathcal{X}_i}{\arg\min} \left(f_i(\boldsymbol{x}_i) + (\rho/2)\|\boldsymbol{x}_i - \boldsymbol{z}_i^k + \boldsymbol{u}_i^k\|_2^2 \right), \quad i = 1, \ldots, N, \\
\boldsymbol{z}^{k+1} &:= \Pi_{\mathcal{C}}(\boldsymbol{x}^{k+1} + \boldsymbol{u}^k) \\
\boldsymbol{u}_i^{k+1} &:= \boldsymbol{u}_i^k + \boldsymbol{x}_i^{k+1} - \boldsymbol{z}_i^{k+1}, \quad i = 1, \ldots, N.
\end{aligned} \tag{87}$$

where $\Pi_{\mathcal{C}}(\cdot)$ is the Euclidean projection onto set $\mathcal{C}$. This projection can be carried out by solving the following optimization problem.

$$\begin{aligned}
\Pi_{\mathcal{C}}(\boldsymbol{x}^{k+1} + \boldsymbol{u}^k) = \arg\text{minimize} \quad & \|\boldsymbol{z} - (\boldsymbol{x}^{k+1} + \boldsymbol{u}^k)\|_2^2 \\
\text{subject to} \quad & \boldsymbol{z} \in \mathbb{R}^{n_1 + \ldots + n_N}, \\
& \boldsymbol{z} \in \mathcal{C}.
\end{aligned} \tag{88}$$

Notice that updates of $\boldsymbol{x}_i^{k+1}$ for each $i = 1, \ldots, N$, can be done in parallel as the problems are decoupled. A similar statement holds for the updates of $\boldsymbol{u}_i^{k+1}$.

The stopping criteria for Algorithm (93) can be determined by monitoring the primal and dual residuals given

$$\boldsymbol{r}^k = \boldsymbol{x}^k - \boldsymbol{z}^k, \qquad \boldsymbol{s}^k = -\rho(\boldsymbol{z}^k - \boldsymbol{z}^{k-1}),$$

respectively. The Algorithm terminates when the magnitude of the residuals $\boldsymbol{r}^k$ and $\boldsymbol{s}^k$ falls below the following thresholds

$$\|\boldsymbol{r}^k\|_2 \leq \epsilon^{\text{primal}}, \qquad \|\boldsymbol{s}^k\|_2 \leq \epsilon^{\text{dual}}.$$

Constants ϵ^{primal} and ϵ^{dual} can be typically inferred from the structure of sets $\mathcal{X}_i$, $i = 1, \ldots, N$, and $\mathcal{C}$.

5.3 ADMM for Adaptive Optimization Problems

In this section, we present our approach for one-stage adaptive optimization problems with fixed recourse, involving only real-valued decisions. Given matrices $A_i \in \mathbb{R}^{m_i \times n_i}$, $B_i \in \mathbb{R}^{m_i \times k}$, $C_i \in \mathbb{R}^{n_i \times k}$, $F_i \in \mathbb{R}^{m_c \times n_i}$, $G \in \mathbb{R}^{m_c \times k}$ for $i = 1, \ldots, N$, and a probability measure $\mathbb{P}_\xi$ supported on set Ξ for the uncertain vector $\boldsymbol{\xi} \in \mathbb{R}^k$, we are interested in choosing real-valued functions $\boldsymbol{x}_i(\cdot) \in \mathcal{R}_{k,n_i}$, $i = 1, \ldots, N$ in order to solve:

$$\begin{array}{lll} \text{minimize} & \mathbb{E}_\xi\Big(\sum_{i=1}^{N}(C_i\boldsymbol{\xi})^\top \boldsymbol{x}_i(\boldsymbol{\xi})\Big) & \\ \text{subject to} & \boldsymbol{x}_i(\cdot) \in \mathcal{R}_{n_i}, & i = 1, \ldots, N, \\ & A_i\boldsymbol{x}_i(\boldsymbol{\xi}) \geq B_i\boldsymbol{\xi}, & i = 1, \ldots, N, \\ & \sum_{i=1}^{N} F_i\boldsymbol{x}_i(\boldsymbol{\xi}) \geq G\boldsymbol{\xi}, & \end{array} \Bigg\} \quad \forall \boldsymbol{\xi} \in \Xi. \tag{89}$$

We assume that the uncertainty set Ξ is a non-empty, convex and compact polyhedron

$$\Xi = \Big\{\boldsymbol{\xi} \in \mathbb{R}^k : \exists \boldsymbol{\zeta} \in \mathbb{R}^v \text{ such that } W\boldsymbol{\xi} + U\boldsymbol{\zeta} \geq \boldsymbol{h},\ \xi_1 = 1\Big\}, \tag{90}$$

where $W \in \mathbb{R}^{l \times k}, U \in \mathbb{R}^{l \times v}$ and $\boldsymbol{h} \in \mathbb{R}^l$. Moreover, we assume that the linear hull of Ξ spans $\mathbb{R}^k$. The parameter ξ_1 is set equal to 1 without loss of generality as it allows us to represent affine functions of the non-degenerate outcomes $(\xi_2, \ldots, \xi_k)$ in a compact manner as linear functions of $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_k)$.

As with Problem (85), Problem (89) can be thought of an optimization problem involving N individual agents, where the decisions of agent i are denoted by $\boldsymbol{x}_i(\cdot) \in \mathcal{R}_{k,n_i}$. Notice that if one disregards the last constraint in Problem (89), the problem is decouples for each player i. In the following, we will take advantage of this structure allowing to decompose the problem and use the ADMM algorithm (87).

By construction, the decisions of all agents in Problem (89) are functions of $\boldsymbol{\xi}$, i.e., all agents can observe and adapt their decisions based on all elements of the uncertain vector $\boldsymbol{\xi}$. This is done without loss of generality. Indeed, one can alternative require that the decisions of agent i depend only on a portion of $\boldsymbol{\xi}$, say $\boldsymbol{\xi}_i \in \mathbb{R}^{k_i}$, with $\boldsymbol{\xi} = (\boldsymbol{\xi}_1^\top, \ldots, \boldsymbol{\xi}_N^\top)^\top$

and $\boldsymbol{x}_i(\cdot) \in \mathcal{R}_{k_i,n_i}$. Using this formulation, one can model the part of the system each agent can observe and consequently adapt to.

Problem (89) involves a continuum of decision variables and inequality constraints. Therefore, in order to make the problem amenable to numerical solutions, there is a need for suitable functional approximations for $\boldsymbol{x}_i(\cdot)$, $i = 1, \ldots, N$.

5.3.1 ADMM for Linear Decision Rules

We now restrict the feasible region of the real-valued functions $\boldsymbol{x}_i(\cdot)$, $i = 1, \ldots, N$ to admit the following linear structure:

$$\boldsymbol{x}_i(\boldsymbol{\xi}) = X_i\boldsymbol{\xi}, \quad X_i \in \mathbb{R}^{n_i \times k}, \quad i = 1, \ldots, N.$$

Applying the linear decision rule approximation to Problem (89) yields the following semi-infinite program, involving only finite number of decisions $X_i \in \mathbb{R}^{n_i \times k}, i = 1, \ldots, N$, and infinite number of constraints.

$$\left.\begin{array}{lll} \text{minimize} & \mathbb{E}_\xi\Big(\sum_{i=1}^{N}(C_i\boldsymbol{\xi})^\top X_i\boldsymbol{\xi}\Big) & \\ \text{subject to} & X_i \in \mathbb{R}^{n_i\times k}, & i = 1, \ldots, N \\ & A_iX_i\boldsymbol{\xi} \geq B_i\boldsymbol{\xi}, & i = 1, \ldots, N \\ & \sum_{i=1}^{N} F_iX_i\boldsymbol{\xi} \geq G\boldsymbol{\xi}, & \end{array}\right\} \quad \forall \boldsymbol{\xi} \in \Xi. \tag{91}$$

Using similar arguments as Proposition (4), we can equivalently write Problem (91) as the following finite dimensional linear program.

$$\begin{array}{lll} \text{minimize} & \sum_{i=1}^{N} \mathrm{Tr}(MC_i^\top X_i) & \\ \text{subject to} & X_i \in \mathbb{R}^{n_i\times k}, & i = 1, \ldots, N \\ & (A_iX_i - B_i) \in (\mathrm{cone}(\Xi)^*)^{m_i}, & i = 1, \ldots, N \\ & \big(\sum_{i=1}^{N} F_iX_i - G\big) \in (\mathrm{cone}(\Xi)^*)^{m_c}, & \end{array} \tag{92}$$

where $\text{cone}(\Xi)^*$ is the dual cone of $\text{cone}(\Xi)$.

Problem (92) has the same structure as Problem (85), with the decision variables X_i for each agent i being coupled in the last constraints. Therefore, the ADMM algorithm (87) can be used to solve Problem (92). The algorithm can be written as follows.

$$\begin{aligned} X_i^{k+1} &:= \underset{X_i \in \mathbb{R}^{n_i \times k}}{\arg\min} \Big\{ \text{Tr}(MC_i^\top X_i) + (\rho/2)\|X_i - Z_i^k + U_i^k\|_2^2 \ : \\ &\qquad\qquad (A_i X_i - B_i) \in \text{cone}(\Xi^*)^{m_i} \Big\}, \ \ i = 1, \ldots, N, \\ \boldsymbol{Z}^{k+1} &:= \Pi_{\mathcal{C}}(\boldsymbol{X}^{k+1} + \boldsymbol{U}^k) \\ U_i^{k+1} &:= U_i^k + X_i^{k+1} - Z_i^{k+1}, \ \ i = 1, \ldots, N. \end{aligned} \tag{93}$$

Here, we slight abuse of notation, we done by $\boldsymbol{X} = (X_1, \ldots, X_N)$, $\boldsymbol{Z} = (Z_1, \ldots, Z_N)$ and $\boldsymbol{U} = (U_1, \ldots, U_N)$, and $\mathcal{C} = \{X_i, \in \mathbb{R}^{n_i \times k}, i = 1, \ldots, N \ : \ \big(\sum_{i=1}^N F_i X_i - G\big) \in (\text{cone}(\Xi)^*)^{m_c}\}$. The projection $\Pi_{\mathcal{C}}$ can be carried out by solving the following optimization problem.

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{Z} - (\boldsymbol{X}^{k+1} + \boldsymbol{U}^k)\|_F^2 \\ \text{subject to} \quad & \boldsymbol{Z} = (Z_1, \ldots, Z_N), \ Z_i \in \mathbb{R}^{n_i \times k}, \ \ i = 1, \ldots, N \\ & \sum_{i=1}^N F_i Z_i - G \in \text{cone}(\Xi^*)^{m_i}, \end{aligned}$$

where

$$\|\boldsymbol{Z} - (\boldsymbol{X} + \boldsymbol{U})\|_F = \left(\sum_{i=1}^N \sum_{j=1}^{n_i} \sum_{p=1}^k \big(Z_{jp} - (X_{jp} + U_{jp})\big)^2 \right)^{\frac{1}{2}}.$$

As before, the updates of X_i^{k+1} and U_i^{k+1} for each $i = 1, \ldots, N$, can be done in parallel as the problems are decoupled.

The stopping criteria for Algorithm (87) can be determined by monitoring the primal and dual residuals given

$$\boldsymbol{R}^k = \boldsymbol{X}^k - \boldsymbol{Z}^k, \qquad \boldsymbol{S}^k = -\rho(\boldsymbol{Z}^k - \boldsymbol{Z}^{k-1}),$$

respectively, with the algorithm terminating magnitude of the residuals falls below the

following thresholds

$$\|\boldsymbol{R}^k\|_F \leq \epsilon^{\text{primal}}, \qquad \|\boldsymbol{S}^k\|_F \leq \epsilon^{\text{dual}}.$$

for given ϵ^{primal} and ϵ^{dual}.

5.4 Conclusion

ADMM is a powerful tool for solving optimization problems in a distributed fashion. In the realm of Local4Global, the ADMM algorithm can be interpreted as a finite set of individual agents, each of which try to optimize their own objective, while finding the tradeoff between optimality and feasibility for the global problem. Linear decision rules in conjunction with the ADMM provide an efficient platform to address dynamic problems, typically encountered in control problems. All the theoretical developments present in this section can be extended to the non-linear decision rules presented in Section 4, making the proposed algorithm an invaluable tool for solving computationally demanding, dynamic optimization problems.

6 Summary

In this deliverable, we have addressed the following: (a) We introduced the precise mathematical formulation of the class of optimization problems that appear in the two case studies of L4G (building control and traffic control). This includes the characterization of the general class of the optimization problems and a detailed description of the problem statement. Emphasis is put on the distinction between Local versus Global control, and the structural difference this induces to the underline optimization problems; (b) We discussed how concepts from approximate dynamic programming, can be used in a model-free control algorithms. This work is in line with the Cognitive-based Adaptive Optimization (CAO), and (PCAO) algorithms proposed by the consortium partners in the Centre for Research and Technology [3, 38, 46]; (c) We presented a novel work for multistage decision making under uncertainty involving both real-valued and discrete decisions. To this end, we extended the framework of the functional approximation known as *decision rules* to cover integer recourse decisions; (d) Finally, we discussed concepts from distributed optimization algorithms with special emphasis on the Alternating Direction Method of Multipliers and we present the basic concepts. We demonstrate how this can be used in conjunction with the decision rule approximation to solve distributed optimization problems under uncertainty. All the theoretical tools presented in this deliverable are essential tools for addressing the two case studies in Local4Global.

References

[1] M. V. Afonso, J. M. Bioucas-Dias, and M. A.T. Figueiredo. An augmented lagrangian approach to the constrained optimization formulation of imaging inverse problems. *IEEE Transactions on Image Processing*, 20(3):681–695, 2011.

[2] B.D.O. Anderson and J.B. Moore. *Optimal Control: Linear Quadratic Methods*. Prentice Hall, 1990.

[3] Simone Baldi, Iakovos Michailidis, Hossein Jula, Elias B Kosmatopoulos, and Petros A Ioannou. A "plug-n-play" computationally efficient approach for control design of large-scale nonlinear systems using co-simulation. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 436–441, 2013.

[4] D. Bampou and D. Kuhn. Scenario-free stochastic programming with polynomial decision rules. *50th IEEE Conference on Decision and Control and European Control Conference, Orlando, FL, USA*, pages 7806–7812, December 2011.

[5] Richard Ernest Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[6] Richard Ernest Bellman. *Eye of the Hurricane: an autobiography*. World Scientific Singapore, 1984.

[7] A. Ben-Tal and D. den Hertog. *Immunizing conic quadratic optimization problems against implementation errors*, volume 2011-060. Tilburg University, 2011.

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

[9] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376, 2004.

[10] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton University Press, 2009.

[11] A. Ben-Tal, B. Golany, and S. Shtern. Robust multi-echelon multi-period inventory control. *European Journal of Operational Research*, 199(3):922–935, 2009.

[12] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc., 1989.

[13] Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[14] D. Bertsimas and C. Caramanis. Adaptability via sampling. In *46th IEEE Conference on Decision and Control, New Orleans, LA, USA*, pages 4717–4722, December 2007.

[15] D. Bertsimas and C. Caramanis. Finite adaptability in multistage linear optimization. *IEEE Transactions on Automatic Control*, 55(12):2751–2766, 2010.

[16] D. Bertsimas and A. Georghiou. Design of near optimal decision rules in multi-stage adaptive mixed-integer optimization. *Submitted for publication, available via Optimization-Online*, 2013.

[17] D. Bertsimas, D.A. Iancu, and P.A. Parrilo. Optimality of affine policies in multi-stage robust optimization. *Mathematics of Operations Research*, 35(2):363–394, 2010.

[18] D. Bertsimas, D.A. Iancu, and P.A. Parrilo. A hierarchy of near-optimal policies for multistage adaptive optimization. *IEEE Transactions on Automatic Control*, 56(12): 2809–2824, December 2011.

[19] Brett Bethke and Jonathan P How. Approximate dynamic programming using bellman residual elimination and gaussian process regression. In *American Control Conference, 2009. ACC'09.*, pages 745–750, 2009.

[20] Brett Bethke and Jonathan P. How. Approximate dynamic programming using model-free bellman residual elimination. In *American Control Conference*, pages 4146 – 4151, June 2010.

[21] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[22] G. C. Calafiore. Multi-period portfolio optimization with linear control policies. *Automatica*, 44(10):2463–2473, 2008.

[23] G. C. Calafiore. An affine control method for optimal dynamic asset allocation with transaction costs. *SIAM Journal on Control and Optimization*, 48(4):2254–2274, 2009.

[24] C. Caramanis. *Adaptable optimization: Theory and algorithms*. PhD thesis, Massachusetts Institute of Technology, 2006.

[25] Y. A. Censor and S. A. Zenios. *Parallel Optimization: Theory, Algorithms and Applications*. Oxford University Press, 1997.

[26] X. Chen and Y. Zhang. Uncertain linear programs: Extended affinely adjustable robust counterparts. *Operations Research*, 57(6):1469–1482, 2009. doi: 10.1287/opre.1080.0605. URL `http://or.journal.informs.org/cgi/content/abstract/opre.1080.0605v1`.

[27] X. Chen, M. Sim, P. Sun, and J. Zhang. A linear decision-based approximation approach to stochastic programming. *Operations Research*, 56(2):344–357, 2008.

[28] D. P. De Farias and B. Van Roy. On the existence of fixed points for approximate value iteration and temporal-difference learning. *Journal of Optimization Theory and Applications*, 105(3):589–608, 2000.

[29] Daniela Pucci de Farias and Benjamin Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6):850–865, November-December 2003.

[30] M. Dyer and L. Stougie. Computational complexity of stochastic programming problems. *Mathematical Programming*, 106(3):423–432, 2006.

[31] Stanley J. Garstka and Roger J.-B. Wets. On decision rules in stochastic programming. *Mathematical Programming*, 7(1):117–143, 1974.

[32] A. Georghiou, W. Wiesemann, and D. Kuhn. Generalized decision rule approximations for stochastic programming via liftings. *Mathematical Programming, to appear*, 2014.

[33] J. Goh and M. Sim. Distributionally robust optimization and its tractable approximations. *Operations Research*, 58(4):902–917, 2010. doi: 10.1287/opre.1090.

0795. URL http://or.journal.informs.org/content/early/2010/04/23/opre.1090.0795.abstract.

[34] P. J Goulart, E. C Kerrigan, and J. M Maciejowski. Optimization over state feedback policies for robust control with constraints. *Automatica*, 42(4):523–533, 2006.

[35] C. E. Gounaris, W. Wiesemann, and C. A. Floudas. The robust capacitated vehicle routing problem under demand uncertainty. *Operations Research*, 61(3):677–693, 2013.

[36] G. A. Hanasusanto, D. Kuhn, and W. Wiesemann. Two-stage robust integer programming. *Submitted for publication, available via Optimization-Online*, 2014.

[37] Arezou Keshavarz and Stephen Boyd. Quadratic approximate dynamic programming for input-affine systems. *International Journal of Robust and Nonlinear Control*, 24 (3):432–449, July 2012.

[38] E. B. Kosmatopoulos and J. L. Piovesan. Clf-based control design for unknown multiinput nonlinear systems with good transient performance. *IEEE Transactions on Automatic Control*, 55(11):2635–2640, 2010.

[39] D. Kuhn, W. Wiesemann, and A. Georghiou. Primal and dual linear decision rules in stochastic and robust optimization. *Mathematical Programming*, 130(1):177–209, 2011. ISSN 0025–5610.

[40] Alan S. Manne. Linear programming and sequential decisions. *INFORMS - Mathematics of Operations Research*, 1960.

[41] G. Mateos, J. A. Bazerque, and G. B. Giannakis. Distributed sparse linear regression. *IEEE Transactions on Signal Processing*, 58(10):5262–5276, 2010.

[42] Iakovos Michailidis, Simone Baldi, Elias B. Kosmatopoulos, and Petros A Ioannou. Adaptive optimal control for large-scale non-linear systems.

[43] C. H. Papadimitriou. On the complexity of integer programming. *Journal of the Association for Computing Machinery*, 28(4):765–768, 1981.

[44] Bart P. Van Parys, Paul J. Goulart, and Marfred Morari. Infinite-horizon perforamnce bounds for constrained strochastic systems. In *51st IEEE Conference on Decision and Control*, pages 2171 – 2176, December 2012.

[45] Warren B. Powell. What you should know about approximate dynamic programming. *Naval Research Logistics (NRL)*, 56(3):239–249, February 2009.

[46] A. Renzaglia, L. Doitsidis, A. Martinelli, and E. Kosmatopoulos. Multi-Robot 3D Coverage of Unknown Areas. *International Journal of Robotics Research*, 31(6):738–752, 2012.

[47] R.T. Rockafellar and R.J.B. Wets. *Variational Analysis*. Springer, 1997.

[48] Benjamin Van Roy. Neuro-dynamic programming: Overview and recent trends. In *Handbook of Markov decision processes*, pages 431–459. Springer, 2002.

[49] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

[50] I. D. Schizas, A. Ribeiro, and G. B. Giannakis. Consensus in Ad Hoc WSNs with noisy links: Part I: Distributed estimation of deterministic signals. *IEEE Transactions on Signal Processing*, 56(1):350–364, 2008.

[51] Paul J. Schweitzer and Abraham Seidmann. Generalized polynomial approximations in markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110:568–582, 1985.

[52] C.-T. See and M. Sim. Robust approximation to multiperiod inventory management. *Operations Research*, 58(3):583–594, 2010.

[53] A. Shapiro and A. Nemirovski. On complexity of stochastic programming problems. *Applied Optimization*, 99(1):111–146, 2005.

[54] J. Skaf and S. P. Boyd. Design of affine controllers via convex optimization. *IEEE Transactions on Automatic Control*, 55(11):2476–2487, 2010.

[55] Tyler H. Summers, Konstantin Kunz, Nikoloas Kartiotoglou, Maryam Kamgarpour, Sean Summers, and John Lygeros. Approximate dynamic programming via sum of squares programming. 2013.

[56] Gerald Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 8: 257–277, 1992.

[57] J. N. Tsitsiklis. Problems in decentralized decision making and computation. Technical report, DTIC Document, 1984.

[58] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Rransactions on Automatic Control*, 31(9):803–812, 1986.

[59] Fei-Yue Wang, Huaguang Zhang, and Derong Liu. Adaptive dynamic programming: an introduction. *Computational Intelligence Magazine, IEEE*, 4(2):39–47, 2009.

[60] Yang Wang and Stephen Boyd. Performance bounds and suboptimal policies for linear stochastic control via lmis. *International Journal of Robust and Nonlinear Control*, 21 (14):1710–1728, 2011.

[61] Yang Wang, Brendan O'Donoghue, and Stephen Boyd. Approximate dynamic programming via iterated bellman inequalities. *International Journal of Robust and Nonlinear Control*, 2014.

[62] Christopher Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, May 1989.

[63] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4): 279–292, 1992.

[64] H. Zhu, A. Cano, and G. B. Giannakis. Distributed consensus-based demodulation: algorithms and error analysis. *Wireless Communications, IEEE Transactions on*, 9(6): 2044–2054, 2010.