Quantified Integer Programming with Polyhedral and Decision-Dependent Uncertainty

DISSERTATION
zur Erlangung des Grades eines Doktors
der Naturwissenschaften

vorgelegt von
M.Sc. Michael Hartisch

eingereicht bei der Naturwissenschaftlich-Technischen Fakultät
der Universität Siegen
Siegen 2020

Betreuer und erster Gutachter
Prof. Dr. rer. nat. Ulf Lorenz
Universität Siegen

Zweiter Gutachter
Prof. Dr. rer. nat. Ingo Althöfer
Friedrich-Schiller-Universität Jena

Dritter Gutachter
Prof. Dr. rer. nat. Marc Goerigk
Universität Siegen

Tag der mündlichen Prüfung
07. August 2020

Danksagung

An dieser Stelle möchte ich mich bei allen Personen bedanken, die mich während der Anfertigung dieser Arbeit unterstützt und motiviert haben.

Besonders danken möchte ich Herrn Prof. Dr. Ulf Lorenz, der diese Arbeit betreut hat und mir fortwährend mit fachlichem Rat zur Seite stand. Er hat mir großes Vertrauen entgegengebracht und durch zahlreiche konstruktive Diskussionen einen erheblichen Beitrag zum Gelingen dieser Arbeit geleistet.

Ich möchte mich bei Herrn Prof. Dr. Ingo Althöfer bedanken, der mich während meiner mathematischen Ausbildung lehrte immer aufgeschlossen zu sein und auch unkonventionelle Wege einzuschlagen. Desweiteren möchte ich mich bei Herrn Prof. Dr. Marc Goerigk bedanken, der durch wertvolle Ratschläge zur inhaltlichen Ausgestaltung dieser Arbeit beigetragen hat. Beiden möchte ich für die Begutachtung dieser Arbeit danken.

Danken möchte ich außerdem meinen Kollegen für viele wertvolle Anregungen und Diskussionen und ganz besonders für das freundschaftliche Arbeitsklima.

Weiterhin möchte ich mich für die finanzielle Unterstützung im Rahmen des DFG-Projekt “Fortgeschrittene Algorithmen und Heuristiken zur Lösung quantifizierter gemischt-ganzzahliger linearer Programme” bedanken.

Mein Dank gilt weiterhin meinen Eltern und meiner Familie, die mich in jeglicher Hinsicht unermüdlich unterstützt haben. Besonders danken möchte ich schließlich meiner Frau Josephine, die mir in anstrengenden Zeiten stets den Rücken freigehalten hat.

Zusammenfassung

Das in der Mitte des letzten Jahrhunderts etablierte Forschungsgebiet der Optimierung unter Unsicherheit hat in den letzten beiden Jahrzehnten viel Aufmerksamkeit erhalten. Der Umgang mit unsicheren Daten in Optimierungsproblemen bleibt aber trotz (oder gerade wegen) der Zeiten großer Datenmengen und Cloud-Lösungen eine große Herausforderung. Die bekanntesten Ansätze um mit solchen Optimierungsproblemen umzugehen sind *stochastische Programmierung* und *robuste Optimierung*. Um ein realistischeres Abbild des zugrunde liegenden Problems zu ermöglichen, können dabei auch mehrstufige Modelle zum Einsatz kommen. Während es einige echt-mehrstufige stochastische Ansätze gibt, gehen die meisten mehrstufigen Erweiterungen der robusten Optimierung nicht über ein zweistufiges Modell hinaus. Noch weniger Aufmerksamkeit wird der Berücksichtigung von entscheidungsabhängiger Unsicherheit geschenkt. Um diese Lücke zu schließen, wurden in dieser Arbeit mehrstufige Optimierungsprobleme auch mit entscheidungsabhängiger Unsicherheit mittels quantifizierter Programme untersucht.

Mit quantifizierten Programmen können mehrstufige Optimierungsprobleme unter Unsicherheit formuliert werden. Dabei handelt es sich um lineare Programme, deren Variablen mit jeweils einem Existenz- oder einem Allquantor versehen sind und eine feste Reihenfolge aufweisen. Die erzielten Fortschritte für *quantifizierte lineare Programme* gaben Anlass zur näheren Untersuchung von *quantifizierten ganzzahligen Programmen* (QIP) in dieser Arbeit. Während sich ein Großteil der Forschung in diesem Bereich der komplexitätstheoretischen Untersuchung spezieller QIP-Erfüllbarkeitsprobleme widmet, war das vorrangige Ziel dieser Arbeit neue Lösungstechniken und Unsicherheitskonzepte für das QIP-Optimierungsproblem zu erforschen.

QIPs können durch eine Spielbaumsuche gelöst werden, welche durch Techniken aus den Bereichen des SAT-, QBF- und MIP-Lösens erweitert werden können. Weitere Lösungstechniken, die in eine solche Spielbaumsuche mit einfließen, wurden in der vorliegenden Arbeit entwickelt und fundiert. Insbesondere wurde die *Strategic Copy-Pruning* Heuristik etabliert. Diese erlaubt es die Existenz einer Strategie in linearer Zeit implizit sicherzustellen, ohne diese explizit durchlaufen zu müssen. Außerdem konnte gezeigt werden, dass die Umsetzung der erlangten Ergebnisse den Suchprozess erheblich beschleunigen kann.

Um die Ausdrucksfähigkeit von QIPs zu erhöhen, wurden darüber hinaus QIPs mit polyedrischer Unsicherheitsmenge eingeführt. Infolge dieser Erweiterung konnten verschiedene mehrstufige kombinatorische Optimierungsprobleme deutlich schneller gelöst werden. Durch eine zusätzliche Erweiterung wurde außerdem ein allgemeiner Rahmen für mehrstufige Optimierungsprobleme unter entscheidungsabhängiger Unsicherheit geschaffen. In dieser Arbeit wurden dafür Lösungstechniken theoretisch fundiert, Implementierungsdetails bereitgestellt und explizite Polynomzeitreduktionen hergeleitet.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer, nicht angegebener Hilfsmittel angefertigt habe. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt.

Es wurden keine Dienste eines Promotionsvermittlungsinstituts oder einer ähnlichen Organisation in Anspruch genommen.

Siegen, 3. Mai 2020

Abstract

The research field of optimization under uncertainty, even though already established in the middle of the last century, gained much attention in the last two decades. The necessity to deal with uncertain data remains a major challenge despite (or because of) the times of big data and cloud solutions. The most prominent modeling paradigms dealing with such optimization tasks are *stochastic programming* and *robust optimization*. Sometimes, in order to obtain an even more realistic description of the underlying problem, multistage models can be used. While there are some real multistage stochastic approaches, most multistage extensions to robust optimization hardly ever consider more than two stages. Even less attention is paid to the consideration of decision-dependent uncertainty. To overcome these shortcomings, we explored multistage optimization under decision-dependent uncertainty via quantified programming.

Quantified programs, which are linear programs with ordered variables that are either existentially or universally quantified, provide a convenient framework for multistage optimization under uncertainty. While considerable research has been conducted with regard to *quantified linear programming* (QLP) this thesis focused on *quantified integer programming* (QIP). Whereas most research in this area is concerned with complexity results regarding the QIP satisfiability problem, we concentrate on solution and modeling techniques for the QIP optimization problem, which we tested and implemented in our open-source solver.

One way to solve a QIP is to apply a game tree search, enhanced with non-chronological backjumping. We developed and theoretically substantiated further solution techniques for QIPs within a game tree search framework and established the *strategic copy-pruning* mechanism, which allows to *implicitly* deduce the existence of a strategy in linear time (by static examination of the QIP-matrix) without explicitly traversing the strategy itself. We also showed that the implementation of our findings can massively speed up the search process.

Furthermore, in order to enhance the expressive power of QIPs, we introduced QIPs with a polyhedral uncertainty set. We showed that by exploiting this extension, we were able to significantly speed up our solver on various multistage combinatorial optimization problems. Additionally, we established QIPs with interdependent domains and thereby provide a general framework for multistage optimization problems under decision-dependent uncertainty. We theoretically substantiated solution techniques, provided implementation details and derived polynomial-time reduction functions mapping both extensions to the basic QIP.

Contents

Glossaries

Symbols

$\mathcal{A}$	set of universal variable blocks
$A_{k,\star}$	k-th row of matrix A
$A_{k,(i)}$	part of the k-th row of matrix A corresponding to variable block i
$A_{\star,k}$	k-th column of matrix A
$A_\exists$	columns of matrix A corresponding to existential variables
$A_\forall$	columns of matrix A corresponding to universal variables
$A^\exists$	coefficient matrix of the existential constraint system
$A^\forall$	coefficient matrix of the universal constraint system
$b^\exists$	right-hand side vector of the existential constraint system
$b^\forall$	right-hand side vector of the universal constraint system
β	number of variables blocks
B_i	set of variables of block i
$c_\exists$	vector containing the values of c corresponding to existential variables
$c_\forall$	vector containing the values of c corresponding to universal variables
$\mathcal{D}$	polyhedral uncertainty set; legal domain for universal variables in a QIP^{PU}
$\mathcal{E}$	set of existential variable blocks
E	set of edges of a game tree
e	vector of edge labels of a game tree
$\mathcal{F}(x_1, \ldots, x_k)$	set of legal variable assignments dependent on previous assignments
$\mathcal{F}^{(i)}$	set of legal variable block assignments of block i
$\mathcal{F}(v)$	set of legal successors at node v, dependent on the path from the root to v
f^{ID}	polynomial-time reduction function mapping QIP^{ID} to QIP
f^{PU}	polynomial-time reduction function mapping QIP^{PU} to QIP
$\mathcal{I}$	set of integer variables; equal to set of variables for QIP, QIP^{PU} and QIP^{ID}
$\mathcal{I}_\exists$	set of existentially quantified (integer) variables
$\mathcal{I}_\forall$	set of universally quantified (integer) variables
$\mathcal{L}$	domain of the variable vector
$\mathcal{L}_\forall$	universal variable domain
$\mathcal{L}_\exists$	existential variable domain
$\text{level}(v)$	number of edges in the path from the unique root r to v; $\text{level}(r) = 0$
$\mathcal{L}^{(i)}$	domain of variable block i
$\mathcal{L}_k$	domain of variable k

$\mathcal{L}(v)$	set of successors (or children) at node v
$m_\exists$	number of existential constraints
$m_\forall$	number of universal constraints
$\mu(i,j)$	function mapping the j-th variable of block i to its original index
$\mu_\forall(k)$	function mapping the k-th universal variable to its original index
$\mu_\exists(k)$	function mapping the k-th existential variable to its original index
$\mathbb{N}_0$	set of non-negative integers
$\mathbb{N}$	set of natural numbers
n	number of variables
$n_\exists$	number of existential variables
$n_\forall$	number of universal variables
parent(v)	parent node v within a game tree
v^+	positive part of a vector $v \in \mathbb{Q}^n$, i.e. $v_i^+ = \max(v_i, 0)$
$\mathbb{Q}$	set of rational numbers
Q_k	quantifier of variable k
$Q^{(i)}$	quantifier of the i-th variable block
r	root node of a game tree
$\leq_p$	$A \leq_p B$ if problem A is polynomial-time reducible to problem B
$\mathcal{T}(S)$	set of terminal nodes in S; nodes without outgoing edges in strategy S
$\mathcal{U}$	set of unavoidable scenarios
V	set of nodes of a game tree
$V_\exists$	set of existential nodes, representing an existential decision stage
$V_\forall$	set of universal nodes, representing a universal decision stage
V_L	set of leaves; nodes without successors in a game tree
x_k	k-th entry of vector x
$x^{(i)}$	entries of variable vector x that belong to variable block i
$x_\exists$	entries of x that are existentially quantified
$x_\forall$	entries of x that are universally quantified
x_v	(partial) variable assignment corresponding to path from the root to node v
$\mathbb{Z}$	set of integers

Acronyms

ARO	adjustable robust optimization
DEP	deterministic equivalent program
GB	gigabyte
KB	kilobyte
LP	linear program
MB	megabyte
MIP	mixed integer program
OR	operations research
PV	principal variation
QBF	quantified boolean formula problem
QCSP	quantified constraint satisfaction problem
QIP	quantified integer program
QIP^{ID}	quantified integer program with interdependent domains
QIP^{PU}	quantified integer program with polyhedral uncertainty set
QII	quantified integer implication
QLI	quantified linear implication
QLP	quantified linear program
QMIP	quantified mixed integer program
RO	robust optimization
SAT	boolean satisfiability problem
SCP	strategic copy-pruning

List of Figures

List of Tables

1. Introduction

1.1. Motivation

Planning under uncertainty is our everyday life. We try to plan ahead, cope with (un)expected incidents and even try to optimize small parts of our daily schedule, e.g. by minimizing the time it takes to leave the house in the morning in order to maximize the time we can sleep. We hedge against uncertain events by adding time buffers, packing an additional bottle of water or keeping the umbrella in the car—just in case. But it goes even further than just preparing for unexpected events, as we have the ability to actively alter the set of expected events: When doing the weekly groceries shopping on Wednesday evenings we can expect deserted corridors but also empty vegetable counters. Doing the same on Friday afternoon the opposite can be expected. Planning (usually) does not happen once a day but in episodes as we adapt to our surroundings as new input data emerges, e.g. we decide to skip a quick stop at the clothing store when the bus is 30 minutes late, in order to attend other appointments in time. Thus, our daily decisions are part of a multistage optimization task under decision-dependent uncertainty, which we (heuristically) try to master. Similar processes occur in game playing and—with wider implications—in operations research.

From the very beginning of the rise of mathematical optimization in the 1940s the need of understanding and dealing with uncertain data was apparent and rapid progress was made early on [Dan55, Bea55, Bel57]. In addition to understanding that input data is not deterministic and static, multistage optimization under uncertainty further considers the recurring alternation between the occurrence of uncertain events and planning decisions. Pioneering work was done in the research area of stochastic programming [BL11, Sha08], where uncertain data is assumed to obey a distribution and the average case is optimized. In the area of robust optimization [BTGN09, BBC11, GMT14], where an uncertainty set is considered and the worst case is optimized, the multistage approach only gained more interest with the beginning of the new millennium. Decision-dependent uncertainty has received more attention in the last ten years, but is certainly not the focus of optimization under uncertainty. This may be due to the fact that most researchers do not see the additional complexity as being proportionate to the improvement, or generally question the necessity of such an advanced optimization framework. However, as outlined above, most decision processes are multistage processes under—potentially decision-dependent—uncertainty, and computational complexity results should not prevent us from advancing our understanding and handling of such problems.

The consideration of uncertainty in optimization problems, in general, often results in an increased computational complexity: many problems in the classes `P` or `NP` become `PSPACE`-

complete if the input data is not deterministic [Pap85]. Thus, a trade-off between a complex, but (potentially) more realistic and a simplified, but tractable problem description is made. In application-oriented areas where good results must be found quickly, trading complexity for computability is reasonable and appropriate. Nevertheless, a quickly obtained solution becomes worthless if it does not reflect reality. Therefore, we stress the importance of researching solution techniques for theoretically hard problems, despite their potentially deterrent complexity:

> *"Perhaps only because we were so naive were we willing to try to solve set partitioning problems with several thousand variables, and I may add with some success. Had we known about computational complexity and* NP*-completeness, we might not have tried."*
>
> (George L. Nemhauser, 1994 [Nem94, p. 6])

One should always keep in mind that a) hardness can also be viewed as synonym for "very compact problem description", b) computational complexity only considers worst-case performance and gives little indication of the computability of relevant and real-world instances and finally c) despite the NP-completeness of mixed integer programming, nowadays large-scale MIPs of practical-relevance with up to millions of variables and constraints can be solved, which was unthinkable 50 years ago. Thus, technological progress and intensive research is able to move computational boundaries, which is why we are not deterred from tackling PSPACE-complete problems algorithmically.

We are interested in PSPACE-complete quantified integer programming (QIP), which is integer programming with specifically ordered quantified variables. A variable is either existentially ($\exists$) or universally ($\forall$) quantified and we frequently use the interpretation of a game between the so-called existential and universal player. The term QIP was coined in [Sub04] only referring to the satisfiability problem and later was extended to an optimization problem by adding an objective function in [EL+11a]. Thus, QIPs are multistage optimization problems under uncertainty where the objective function is optimized with respect to the worst possible case. Jan Wolf's work on quantified linear programs provides the basis for this research [Wol15] and advancing and enhancing the open-source[1] QIP solver Yasol [EH+17] was the incentive of this thesis.

In this thesis, QIP-specific solution techniques are presented and theoretically consolidated, and fundamental theoretical research regarding two extensions of QIPs that allow a) a polyhedral uncertainty set and b) a decision-dependent uncertainty set are presented. Explicit polynomial-time reduction functions for both extensions do not only provide insight into their close connection to the basic QIP but also illustrate that they remain PSPACE-complete. We theoretically elaborate an alpha-beta based solution approach for these extensions and implement it for an application-oriented subclass within the existing framework of the open-source solver Yasol. This (still PSPACE-complete) subclass demands that potential realizations of uncertain events can be "easily determined" and that the planner is not able to "obliterate" uncertainty, which can be a reasonable restriction when dealing with environmental influences and uncertain

[1] http://www.q-mip.org (accessed May 3, 2020)

input data. In this thesis, several experimental results are provided that show potentials and (current) limits of quantified integer programming and its extensions.

1.2. Structure of the Thesis

This introduction is followed by seven chapters, which are structured as follows:

- **Chapter 2: Multistage Optimization under Uncertainty.** The concept of quantified mixed integer programming is introduced and the connections and differences to other areas of research are presented. The chapter concludes with illustrating examples how real-world problems can be modeled using quantified integer programs.
- **Chapter 3: Algorithmic Properties of QIPs.** New insights into algorithmic properties of QIPs are presented and known properties are pointed out for use in later chapters. The novel strategic copy-pruning mechanism is introduced, which allows to implicitly deduce the existence of a strategy in linear time (by static examination of the QIP-matrix) without explicitly traversing the strategy itself. Furthermore, the deterministic equivalent program and known relaxations are reviewed and new relaxations that incorporate enhanced information regarding potential realizations of uncertain variables are presented.
- **Chapter 4: Quantified Integer Programming with Polyhedral Uncertainty Set.** The possibility to restrict the uncertainty set of a quantified program to some polytope, instead of the hypercube created by variable bounds, is introduced. A polynomial-time reduction function, allowing the conversion to a standard QIP, is presented and several examples illustrate how this modeling option can be utilized.
- **Chapter 5: Quantified Integer Programming with Interdependent Domains.** This chapter covers a further extension allowing the interaction of planning decisions with the uncertainty set, resulting in an interdependence of the variable domains. Substantiating theoretical results and a polynomial-time reduction function back to the standard QIP are presented. Furthermore, possible relaxations and general solution techniques are discussed, which are also applicable for the QIP with polyhedral uncertainty set. Additionally, it is shown to what extent QIP pruning techniques are applicable.
- **Chapter 6: Simply Restricted QIP$^{\text{ID}}$.** As the possible influence of planning decisions on the uncertainty set is not too excessive in real-world examples, a restricted version of the interdependence presented in the previous chapter is considered. The benefits of the assumptions made for the solution process are discussed and examples where they apply are outlined.
- **Chapter 7: Implementation Details and Experimental Results.** Details on how our open-source solver was enhanced in order to deal with the previously introduced extensions are presented. In a computational study the performance of models utilizing these extensions with the equivalent standard QIP and the robust counterpart are evaluated.

- **Chapter 8: Conclusion and Outlook.** A summary of the findings and an outlook on future research in the field of quantified programming conclude the thesis.

1.3. Own Contribution to Knowledge

The contribution of this thesis can be summarized as follows:

- We develop two extensions of quantified integer programming: We introduce the quantified integer program with polyhedral uncertainty set QIP^{PU}, which allows the modeler to restrict universal variables to some polytope instead of the hypercube created by the variable bounds. Furthermore, the quantified integer program with interdependent domains QIP^{ID} is presented in which existential variable assignments in early decision stages restrain universal variable assignments later on. We further provide explicit reduction functions mapping each extension back to the standard QIP.
- We present the novel strategic copy-pruning mechanism (SCP) for QIP, which allows to implicitly deduce the existence of a winning strategy in linear time (by static examination of the QIP-matrix) without explicitly traversing the strategy itself. We provide proof that SCP can also be applied during the solution process for QIP^{ID}.
- We substantiate relaxations for QIP, QIP^{PU} and QIP^{ID} and present solution techniques for QIP^{ID}. We further introduce the *simply restricted* QIP^{ID} in which the possible interdependencies are limited. We motivate its relevance and demonstrate how this limitation can be integrated in the existing alpha-beta framework of our solver.
- We provide several examples for QIP as well as QIP^{PU} and QIP^{ID} and compare different modeling techniques and solution approaches in a detailed computational study. Therefore, we enable the open-source solver Yasol to deal with QIP^{PU} and simply restricted QIP^{ID}. We demonstrate that SCP can massively boost the solution process for some problem types, while in cases where it is not applicable, it has no significant negative effects.

This thesis is based on the following publications, in which the work was done in close cooperation with the respective co-authors:

- M. Hartisch, T. Ederer, U. Lorenz and J. Wolf. Quantified integer programs with polyhedral uncertainty set. In *Computers and Games - 9th International Conference, CG 2016*, pages 156–166, Springer International Publishing, Cham, 2016.
- T. Ederer, M. Hartisch, U. Lorenz, T. Opfer and J. Wolf. Yasol: An open source solver for quantified mixed integer programs. In *15th International Conference on Advances in Computer Games, ACG 2017*, pages 224–233. Springer International Publishing, Cham, 2017.
- M. Hartisch, A. Herbst, U. Lorenz, and J. B. Weber. Towards resilient process networks - designing booster stations via quantified programming. In *Applied Mechanics and Materials*, volume 885, pages 199–210. Trans Tech Publ, 2018.

- M. Hartisch and U. Lorenz. Mastering uncertainty: Towards robust multistage optimization with decision dependent uncertainty. In *Pacific Rim International Conference on Artificial Intelligence*, pages 446–458. Springer International Publishing, Cham, 2019.
- M. Hartisch and U. Lorenz. Robust multistage optimization with decision-dependent uncertainty. To appear in: *Operations Research Proceedings 2019*.
- M. Hartisch and U. Lorenz: Game tree search in a robust multistage optimization framework: Exploiting pruning mechanisms. *arXiv preprint arXiv:1811.12146;* To appear in: *16th International Conference on Advances in Computer Games, ACG 2019*.

2. Multistage Optimization under Uncertainty

2.1. Quantified (Mixed) Integer Programming

2.1.1. Problem Statement and Notation

Quantified integer programming (QIP) is an extension of integer programming (IP) where some variables are existentially and others are universally quantified. The term was introduced by Subramani in [Sub04] who also coined the term *quantified linear programming* (QLP) [Sub03]. The semantics of universally quantified variables is that some linear constraint system must be fulfilled for all possible realizations of such variables. Hence, a solution of a quantifed program is a strategy [PdB01] for assigning existentially quantified variables such that the underlying constraint system is satisifed. By adding a minimax objective function the aim is to find the best strategy [EL+11a, Wol15]. QIPs are known to be PSPACE-complete [Wol15] and can be interpreted as two-person zero-sum games between an existential and a universal player on the one hand, or multistage optimization problems under uncertainty on the other hand.

We adapt the notation used in [HL19b] to formally introduce quantified programming. Let $n \in \mathbb{N}$ be the number of variables and $x = (x_1, \dots, x_n)^\top \in \mathbb{Q}^n$ a vector[2] of (ordered) variables. Let $\mathcal{I} \subseteq \{1, \dots, n\}$ denote the (ordered) index set of integer variables. For each variable x_i its domain $\mathcal{L}_i$ with $l_i, u_i \in \mathbb{Q}$, $l_i \leq u_i$, $1 \leq i \leq n$, is given by $\mathcal{L}_i = \{y \in \mathbb{Q} \mid l_i \leq y \leq u_i \wedge i \in \mathcal{I} \Rightarrow y \in \mathbb{Z}\}$. For each integer variable $i \in \mathcal{I}$ we further demand the integrality of its bounds, i.e. $l_i, u_i \in \mathbb{Z}$. Consequently, $\mathcal{L}_i \neq \emptyset$ for each variable $i \in \{1, \dots, n\}$. The domain of the variable vector is described by $\mathcal{L} = \{y \in \mathbb{Q}^n \mid \forall i \in \{1, \dots, n\} : y_i \in \mathcal{L}_i\}$. Let $Q \in \{\exists, \forall\}^n$ denote the vector of (ordered) quantifiers. We call each maximal consecutive subsequence in Q consisting of identical quantifiers a *quantifier block*. The quantifier corresponding to the i-th quantifier block is given by $Q^{(i)} \in \{\exists, \forall\}$ and the corresponding i-th *variable block* is given by the (ordered) index set $B_i \subseteq \{1, \dots, n\}$. Let $\beta \in \mathbb{N}$, $\beta \leq n$, denote the number of variable blocks and thus $\beta - 1$ is the number of quantifier changes. Note that $B_1 \cup B_2 \cup \ldots \cup B_\beta = \{1, \dots, n\}$ with $B_i \cap B_{i'} = \emptyset$ for $i \neq i'$. Let $\mu(i, j) = \sum_{k=1}^{i-1} |B_k| + j$, which maps the j-th variable of block i to its original index. The variable vector of variable block B_i is referred to as $x^{(i)}$ and its range is given by $\mathcal{L}^{(i)} = \{y \in \mathbb{Q}^{|B_i|} \mid y_j \in \mathcal{L}_{\mu(i,j)}\}$.

Definition 2.1.1 (Quantified Mixed Integer Linear Program (QMIP))**.**
Let $A^\exists \in \mathbb{Q}^{m_\exists \times n}$ and $b^\exists \in \mathbb{Q}^{m_\exists}$ for $m_\exists \in \mathbb{N}$. Let $\mathcal{L}$ and Q be given as described with $Q^{(1)} = Q^{(\beta)} = \exists$. Let $c \in \mathbb{Q}^n$ be the vector of objective coefficients, for which $c^{(i)}$ denotes the vector of coefficients belonging to variable block B_i. The term $Q \circ x \in \mathcal{L}$ with the component wise binding

[2] We suppress transposes when they are clear from the context to avoid excessive notation.

operator $\circ$ *denotes the* quantification sequence $Q^{(1)}x^{(1)} \in \mathcal{L}^{(1)} \ \dots \ Q^{(\beta)}x^{(\beta)} \in \mathcal{L}^{(\beta)}$, *such that every quantifier* $Q^{(i)}$ *binds the variables* $x^{(i)}$ *of block* i *ranging in their domain* $\mathcal{L}^{(i)}$. *We call* $(A^{\exists}, b^{\exists}, c, \mathcal{L}, Q)$ *with*

$$z = \min_{x^{(1)} \in \mathcal{L}^{(1)}} \left(c^{(1)}x^{(1)} + \max_{x^{(2)} \in \mathcal{L}^{(2)}} \left(c^{(2)}x^{(2)} + \min_{x^{(3)} \in \mathcal{L}^{(3)}} \left(c^{(3)}x^{(3)} + \dots \min_{x^{(\beta)} \in \mathcal{L}^{(\beta)}} c^{(\beta)}x^{(\beta)} \right) \right) \right)$$

$$\text{s.t. } Q \circ x \in \mathcal{L} : \ A^{\exists}x \leq b^{\exists} \tag{2.1}$$

a quantified mixed integer linear program *(QMIP) with objective function.*

We call $A^{\exists}x \leq b^{\exists}$ the *(existential) constraint system* and $\mathcal{E} = \{k \in \{1, \dots, \beta\} \mid Q^{(k)} = \exists\}$ the set of existential variable blocks and $\mathcal{A} = \{k \in \{1, \dots, \beta\} \mid Q^{(k)} = \forall\}$ the set of universal variable blocks. Further, we call variable x_i an existential (universal) variable if the corresponding quantifier Q_i is $\exists$ ($\forall$).

By adding dummy variable blocks one can obtain the structure as shown above, i.e. existential first and last variable block, without altering the complexity and result of the problem instance [Wol15]. In this case $\mathcal{E} = \{1, 3, \dots, \beta\}$ and $\mathcal{A} = \{2, 4, \dots, \beta - 1\}$. There are two reasons for demanding $Q^{(1)} = Q^{(\beta)} = \exists$: With existential first and final variable block the min/max function starts and ends with a minimization term making the problem statement independent of the instance, immediately allowing $\beta = 1$, in which case the instance is a *mixed integer program* (MIP). Furthermore, if the final variable block consists of universal variables they can be eliminated by setting them in a worst-case manner for each constraint separately (see [Wol15]).

Existential variables can be interpreted as decision variables under control of the (minimizing) decision-maker or planner. Universal variables on the other hand cannot be controlled by the planner, who must be prepared for any realization within the universal variable domain. Hence, universal variables represent uncertain events or actions of a (maximizing) opponent. The question is whether there is a strategy to assign existential decision variables in such a way that the fulfillment of the constraint system can be guaranteed for each realization of universal variables. If such a strategy exists, we search for the best one, i.e. the strategy that minimizes the objective value regarding c in the worst case. The quantifier alternation specifies the timing of actions by the decision-maker and the opponent. Therefore, QMIP is a multistage optimization problem under uncertainty and in particular a *robust multistage optimization problem*, as the goal is to hedge against the worst-case scenario.

For easier notation we explicitly distinguish the existential and universal components of a QMIP. Let $n_{\exists} = |\cup_{k \in \mathcal{E}} B_k|$ be the number of existential variables and $n_{\forall} = n - n_{\exists}$ the number of universal variables.

Definition 2.1.2 (Index Mapping Function μ_q)**.**
Let $q \in \{\exists, \forall\}$ *and let* $i_q : \{1, \dots, n_q\} \to \{1, \dots, \beta\}$ *map the* k*-th existential (universal) variable to its corresponding variable block, i.e.* $i_q(k) = \max\{i \in \{1, \dots, \beta\} \mid \sum_{\ell < i,\ Q^{(\ell)} = q} |B_\ell| < k\}$. *Let* $j_q : \{1, \dots, n_q\} \to \{1, \dots, n\}$, $j_q(k) = k - \sum_{\ell < i_q(k),\ Q(\ell) = q} |B_\ell|$, *i.e. the* k*-th existential*

(universal) variable is the $j_\exists(k)$-th ($j_\forall(k)$-th) variable of its corresponding variable block $B_{i_\exists(k)}$ ($B_{i_\forall(k)}$). Then

$$\mu_q : \{1, \dots, n_q\} \to \{1, \dots, n\}, \mu_q(k) = \mu(i_q(k), j_q(k))$$

maps the k-th variable with quantifier q to its original index.

We can now define the existential and universal variable domains.

Definition 2.1.3 (Existential and Universal Variable Domain $\mathcal{L}_\exists$)**.**
We call

$$\mathcal{L}_\exists = \left\{ y \in \mathbb{Q}^{n_\exists} \mid \forall k \in \{1, \dots n_\exists\} : \ y_k \in \mathcal{L}_{\mu_\exists(k)} \right\}$$

the existential variable domain *and*

$$\mathcal{L}_\forall = \left\{ y \in \mathbb{Q}^{n_\forall} \mid \forall k \in \{1, \dots n_\forall\} : \ y_k \in \mathcal{L}_{\mu_\forall(k)} \right\}$$

the universal variable domain*.*

We further split up the coefficient matrix $A^\exists$ as well as the objective vector c into their components: Let $A^\exists_\exists$ ($A^\exists_\forall$) denote the matrix containing only the columns of $A^\exists$ corresponding to existential (universal) variables. Similarly, we define $c_\exists$ ($c_\forall$) as the vector of the objective coefficients containing the entries of c corresponding to existential (universal) variables. Furthermore, $x_\exists$ and $x_\forall$ denote the entries of the variable vector x that are quantified existentially or universally, respectively. As already introduced, the vector of objective coefficients belonging to block i is denoted by $c^{(i)}$ and the entries of row k of the coefficient matrix belonging to block i are denoted by $A^\exists_{k,(i)}$.

Example 2.1.4 (QMIP Notation)**.**
Consider a QMIP with $n = 4$ binary variables, i.e. $\mathcal{I} = \{1, 2, 3, 4\}$ and $\mathcal{L} = \{0, 1\}^4$. The objective $c^\top x$, the quantification sequence $Q \circ x \in \mathcal{L}$ and the constraint system $A^\exists x \leq b^\exists$ with $m_\exists = 3$ constraints are given as follows:

$$\min \{-3x_1 + \max \{4x_2 - x_3\}\}$$

$$\text{s.t. } \exists\, x_1 \in \{0,1\}\ \forall\, x_2 \in \{0,1\}\ \forall\, x_3 \in \{0,1\}\ \exists\, x_4 \in \{0,1\} :$$

$$\begin{pmatrix} 1 & -1 & 1 & -1 \\ -1 & 0 & -1 & -1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \leq \begin{pmatrix} 1 \\ -2 \\ 2 \end{pmatrix}$$

In future examples we often omit the min/max alternation in the objective and only specify the optimization orientation for the existential variables, which in this example is minimization. There are $\beta = 3$ variable blocks given by $B_1 = \{1\}$, $B_2 = \{2, 3\}$ and $B_3 = \{4\}$ with $\mathcal{E} = \{1, 3\}$ and $\mathcal{A} = \{2\}$. The number of existential and universal variables is two each, i.e. $n_\exists = n_\forall = 2$. Further, the existential domain is the same as the universal domain: $\mathcal{L}_\exists = \mathcal{L}_\forall = \{0, 1\}^2$. The

optimal value is 1*, which can be achieved with the optimal first-stage solution* $x_1 = 1$*, which is demonstrated in Example 2.1.18 on page 14.*

In [Wol15] QLPs, i.e. QMIPs with $\mathcal{I} = \emptyset$, were thoroughly examined. In this thesis, we focus on the case where no continuous variables are present.

Definition 2.1.5 (Quantified Integer Linear Program (QIP))**.**
A QMIP $(A^\exists, b^\exists, c, \mathcal{L}, Q)$ *is called a* quantified integer (linear) program *(QIP) if* $\mathcal{I} = \{1, \dots, n\}$ *and thus* $\mathcal{L} \subseteq \mathbb{Z}^n$*.*

Definition 2.1.6 (Set of Variables)**.**
For a QIP we call $\mathcal{I}$ *the (ordered)* set of variables*. The (ordered)* set of existentially quantified variables *is given by*

$$\mathcal{I}_\exists = \{k \in \mathcal{I} \mid Q_k = \exists\}$$

and the (ordered) set of universally quantified variables *is given by*

$$\mathcal{I}_\forall = \{k \in \mathcal{I} \mid Q_k = \forall\}\,.$$

A QIP with $c = 0$ is closely related to the *quantified boolean formula problem* (QBF) and extends it by allowing general integer variables and arbitrary linear constraints, rather than binary variables and clauses. The *quantified constraint satisfaction problem* (QCSP) is a generalization of QBF and QIP by allowing arbitrary constraints. Both QCSP and QBF, however, are mainly investigated as satisfiability problems, while we consider an optimization problem.

Remark 2.1.7. *In other literature (e.g. [Sub03, Sub04]) the terms QLP and QIP often refer to the satisfiability problem instead of the optimization problem. Obviously, with* $c = 0$ *the satisfiability problem is a subproblem of the presented optimization problem and therefore we use the terms QLP, QIP and QMIP for the more general optimization problems.*

The general QMIP with $\beta = 2$ ($\beta = 3$) is closely related to (two-stage) robust discrete optimization with interval uncertainty (see e.g. [KY97, BTN98, KZ16]). Note that "two-stage" robust optimization problems are similar to QMIPs with quantifier alternation $\exists\forall\exists$, i.e. a QIP with three variable blocks. We, however, use the term *stage* as synonym for variable block, rather than solely referring to existential variable blocks as it is commonly used in robust optimization. Therefore, we call a QMIP with quantifier alternation $\exists\forall\exists$ a three-stage problem.

In the course of this thesis, we often restrict ourselves to the cases that integer variables are further restricted to be only 0 or 1, resulting in a *binary QIP*.

Definition 2.1.8 (Binary QIP)**.**
A QIP $(A^\exists, b^\exists, c, \mathcal{L}, Q)$ *is called a* binary QIP *if* $\mathcal{L} = \{0, 1\}^n$*.*

A QIP with binary universal variables and integer existential variables can easily be converted to a binary QIP by expressing each non-binary variable x_k via $\lfloor \log_2(u_k - l_k) \rfloor + 1$ binary auxiliary variables and adding auxiliary bound constraints. Note that such a transformation of integer

universal variables is in general invalid, as the additional bound constraint would alter the problem itself: the bound constraint consists of only binary universal variables and provides an easy way for the universal player to violate the constraint system. This can be avoided by either a) adding the arising bound constraint to a universal constraint system (see Chapter 4) or b) clever modeling. The main reason for the frequent limitation to binary variables is that our solver indeed binarizes general integer variables and is (or rather was) only able to cope with binary universal variables. Further information regarding our solver can be found in Subsection 2.1.3 and Section 7.1.

2.1.2. Quantified Integer Programming as Two-Person Zero-Sum Game

A QIP instance can be interpreted as a two-person zero-sum game between an *existential player* setting the existentially quantified variables and a *universal player* setting the universally quantified variables with payoff z. The variables are set in consecutive order according to the variable sequence $x_1, \dots, x_n$. We say that a player makes the move $x^{(i)} = y$, if she fixes the variable vector $x^{(i)}$ of block i to $y \in \mathcal{L}^{(i)}$. At each such move, the corresponding player knows the settings of $x^{(1)}, \dots, x^{(i-1)}$ before taking her decision $x^{(i)}$.

Remark 2.1.9. *In the course of this thesis, we frequently use the terms universal and existential player. From now on the existential player is referred to as "he" and the universal player as "she". If we do not refer to a specific player we also use "she".*

Each fixed vector $x \in \mathcal{L}$, that is, when the existential player has fixed the existential variables and the universal player has fixed the universal variables, is called *a play*. If x satisfies the linear constraint system $A^\exists x \leq b^\exists$, the existential player pays $z = c^\top x$ to the universal player. If x does not satisfy $A^\exists x \leq b^\exists$, we say *the existential player loses* and the payoff is $+\infty$. This is a small deviation from conventional zero-sum games, but using[3] $\infty + (-\infty) = 0$ also fits for zero-sum game. Therefore, it is the existential player's primary goal to ensure the fulfillment of the constraint system, while the universal player tries to violate some constraints. If the existential player is able to ensure that all constraints are fulfilled he tries to minimize $c^\top x$, whereas the universal player tries to maximize her payoff. A game tree can be used to represent the chronological order of all possible moves, given by the quantification sequence $Q \circ x \in \mathcal{L}$.

Definition 2.1.10 (Game Tree)**.**
Consider a QIP $(A^\exists, b^\exists, c, \mathcal{L}, Q)$*. Its* game tree $G = (V, E, e)$ *is an edge-labeled finite arborescence (a directed, rooted tree, e.g. [KV18]) with a set of nodes* $V = V_\exists \cup V_\forall \cup V_L$*, unique root node* $r \in V_\exists$*, a set of edges* E *and a vector of edge labels* $e \in \mathbb{Q}^{|E|}$*. The disjoint sets* $V_\exists$*,* $V_\forall$ *and* V_L *contain* existential decision nodes*,* universal decision nodes *and* leaf nodes*, respectively. The* level *of a node* $v \in V$ *is the number of edges in the path from* r *to* v*. Inner (non-leaf) nodes with the same level are either all existential decision nodes or all universal decision nodes. The* j*-th variable is represented by inner nodes with level* $j - 1$*. Therefore, outgoing edges from a node* $v \in V$ *in level* $j - 1$ *represent moves from* $\mathcal{L}_j$ *of the corresponding player, the edge labels encode*

[3] Since this is only a matter of interpretation the consequences of this are not discussed further.

the corresponding variable assignments. The set V_L contains all nodes without outgoing edges, which represent completely filled variable vectors. We define $\mathcal{L}(v) = \{v' \in V \mid (v, v') \in E\}$ as the set of successors of an inner node $v \in V_\exists \cup V_\forall$ at level $j-1$, which is implicitly given by the corresponding variable domain $\mathcal{L}_j$.

Remark 2.1.11. *Throughout this thesis, we assume a minimizing existential player and a maximizing universal player. Therefore, $V_\exists$ is the set of minimizing (MIN) nodes and $V_\forall$ the set of maximizing (MAX) nodes.*

A path from the root to a leaf represents a play of the QIP and the sequence of edge labels encodes its moves and hence the assignment of the corresponding variables.

Definition 2.1.12 (Variable Assignment x_v Corresponding to Node v)**.**
Consider a QIP $(A^\exists, b^\exists, c, \mathcal{L}, Q)$, its game tree $G = (V, E, e)$ and any node $r \neq v \in V$. Then, $x_v \in \mathbb{Q}^{level(v)}$ denotes the variable assignment corresponding to v *defined by the edge labels of the path from the root to v.*

For any play $\tilde{x} \in \mathcal{L}$ we call the corresponding universal variable assignments $\tilde{x}_\forall \in \mathcal{L}_\forall$ a *scenario*. Note that for a general QMIP infinitely many edges would be required for nodes representing continuous variables, which is why we restricted ourselves to QIP in this subsection. The most relevant term in order to describe solutions are so-called strategies.

Definition 2.1.13 (Existential Strategy)**.**
A strategy *(for the assignment of existential variables) $S = (V', E', e')$ is a subtree of a game tree $G = (V, E, e)$. V' contains the unique root node $r \in V_\exists$, each node $v_\exists \in V' \cap V_\exists$ has exactly one child in S, and each node $v_\forall \in V' \cap V_\forall$ has as many children in S as in G, i.e. as many as there are values in the corresponding variable domain.*

An existential strategy defines how to react to each possible move by the universal player. Similarly, a *universal strategy* can be defined, but from now on the term *strategy* always refers to the existential strategy, while a universal strategy is called as such. A strategy is called a *winning strategy* if all paths from the root to a leaf represent a vector x such that $A^\exists x \leq b^\exists$ [LMW10].

Definition 2.1.14 (Winning Strategy)**.**
Consider a QIP $(A^\exists, b^\exists, c, \mathcal{L}, Q)$ and its game tree $G = (V, E, e)$. A strategy $S = (V', E', e')$ is called winning strategy *if $A^\exists x_v \leq b^\exists$ for any leaf $v \in V' \cap V_L$.*

A QIP is called *feasible* if (2.1) is true, i.e. if a winning strategy for the assignment of existential variables exists. Therefore, a winning strategy is also referred to as *solution* of a feasible QIP. If there is more than one solution, the objective function aims for a certain (the "best") one.

Definition 2.1.15 (Minimax Value and Value of a Strategy)**.**
Let $S = (V', E', e')$ be a subtree of the game tree $G = (V, E, e)$ of a QIP, with either $S = G$ or

S is a strategy. For any node $v \in V'$ the minimax value with respect to *S is recursively defined by*

$$minimax^S(v) = \begin{cases} c^\top x_v & \text{, if } v \in V_L \text{ and } A^\exists x_v \leq b^\exists \\ +\infty & \text{, if } v \in V_L \text{ and } A^\exists x_v \not\leq b^\exists \\ \min\{minimax^S(v') \mid (v,v') \in E'\} & \text{, if } v \in V_\exists \\ \max\{minimax^S(v') \mid (v,v') \in E'\} & \text{, if } v \in V_\forall\,. \end{cases}$$

The minimax value of the root $r \in V'$ with respect to S defines the value *of S denoted by $minimax(S) = minimax^S(r)$. For $S = G$ and any node $v \in V$ we call $minimax(v) = minimax^G(v)$ the* minimax value *of v, which is the outcome if the remaining variables are assigned optimally starting from this node, i.e. the outcome of optimal play by both players.*

Note that $minimax^S(v) = minimax(v)$ applies for any leaf $v \in V_L$ and any strategy S. However, this does not generally apply to inner nodes. We recall Stockman's theorem [PdB01, Sto79], which connects the value of a strategy to the maximum value at its leaves.

Theorem 2.1.16 (Stockman's Theorem [Sto79])**.**
The value of a strategy $S = (V', E', e')$ is equal to the maximum value at its leaves, i.e.

$$minimax(S) = \max_{v \in V' \cap V_L} minimax(v)\,.$$

As a leaf v not fulfilling $A^\exists x_v \leq b^\exists$ is represented by the value $+\infty$, a strategy S is a winning strategy if and only if $minimax(S) \neq +\infty$. Furthermore, for any node $v \in V$ with $minimax(v) = +\infty$ there exists no winning substrategy. The *optimal value* of a feasible QIP is the value of the *optimal solution*, i.e. the winning strategy with the smallest value.

Definition 2.1.17 (Optimal Winning Strategy of a QIP)**.**
Consider a feasible QIP $P = (A^\exists, b^\exists, c, \mathcal{L}, Q)$ and its game tree $G = (V, E, e)$. A winning strategy $S = (V', E', e')$ is called optimal winning strategy *if $minimax(S) \leq minimax(\tilde{S})$ for all other winning strategies $\tilde{S}$. Then $minimax(S) = minimax(G)$ applies and we call $minimax(G)$ the* optimal value *of P. We call the variable assignment x_v corresponding to a leaf $v \in V' \cap V_L$ in S with $minimax(S) = c^\top x_v$* principal variation *(PV) [CM83], which is a sequence of variable assignments being chosen during optimal play in G.*

The PV also contains the *first-stage solution*—the optimal assignment of $x^{(1)}$—which is independent of the realization of universal variables. In the "real world", this decision should be implemented "here and now". As the entire optimal winning strategy might be too space-consuming or simply not available, the adapted QIP with $\beta - 2$ variable blocks can be solved after the realization of the universal variables $x^{(2)}$ in order to obtain the subsequent optimal decision. Note that as long as the universal variables are assigned according to the global PV no adapted QIP has to be solved.

Example 2.1.18 (Continuation of Example 2.1.4)**.**
The game tree of the binary QIP presented in Example 2.1.4 is depicted in Figure 2.1. Edges

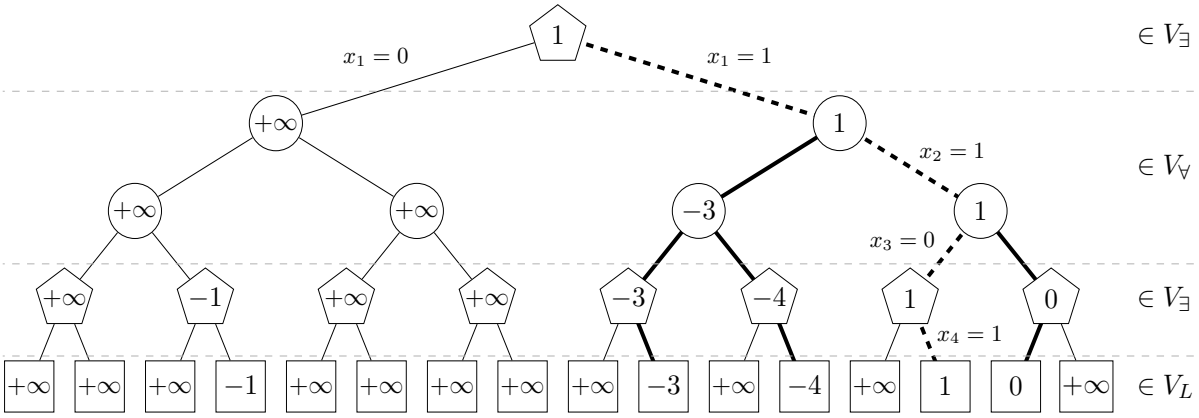

Figure 2.1.: Game tree with rectangular leaves, circular universal (MAX) nodes, and pentagonal existential (MIN) nodes. Values in the nodes are the corresponding minimax values. Dashed gray lines indicate the end of a variable block.

to the left stand for setting the corresponding variable to 0 *and edges to the right for assigning the value* 1*. The edges connecting the optimal winning strategy are drawn thicker, and the edges corresponding to the PV, given by* $x_1 = 1$*,* $x_2 = 1$*,* $x_3 = 0$ *and* $x_4 = 1$*, are additionally dashed. The optimal value of this instance is* 1*, which corresponds to the leaf in the optimal solution with the highest value. At (inner) nodes marked with* ∞ *no winning strategy exists in the underlying subtree and in particular the assignment* $x_1 = 0$ *ultimately results in a loss for the existential player, if the universal player plays optimally.*

2.1.3. The Open-Source QMIP Solver Yasol

The open-source solver *Yasol*[4] is a search based solver for QMIPs [EH+17]. While the solver is novel in the sense that there are no other general QMIP solvers that we know of, most basic ingredients are not new at all. The heart of the search algorithm is an arithmetic linear constraint database together with an alpha-beta algorithm, which has been successfully used in gaming programs, e.g. chess programs for many years [KM75, DL04]. In order to realize fast backjumps—as typically performed in SAT- and QBF-solvers (e.g. [GNT03, CVB01])—the alpha-beta algorithm was extended as roughly described in [EH+17]. Yasol deals with constraint learning on the so-called primal side as known from SAT- and QBF-solving (e.g. [MSLM09, GN+02]), as well as with constraint learning on the dual side known from MIP (e.g. [CC+98]). Several other techniques from various research fields are implemented, e.g. the killer heuristic [AN77], restart strategies [Bie08] and strong branching [AKM05]. Yasol is currently able to solve multistage quantified mixed integer programs with the following properties:

[4]Sources and further information regarding the solver can be found on http://www.q-mip.org (accessed May 3, 2020).

- The basic structure must be a QMIP as in Definition 2.1.1, i.e. linear constraints and objective function, existentially or universally quantified variables, all variables are bounded from below and above and finally $Q_1 = Q_n = \exists$.
- Integer variables are allowed in all existential and universal variable blocks.
- Continuous variables are allowed only in the last closing stage, i.e. we assume $\cup_{i=1}^{\beta-1} B_i \subseteq \mathcal{I}$.

For further information regarding the input format we refer to Section 7.1. Yasol makes intensive use of a linear program solver like the LP-solver of CPLEX. These tools are black-box used, while not exploiting the integer solving abilities of these foreign solvers.

As one of the results of this thesis, an extension of the solver is developed, which allows to enter an explicit universal constraint system the universal player must satisfy. This implicitly allows the input of general integer universal variables, which now can be binarized as the arising bound constraint can be added to the universal constraint system.

2.2. Related Work

This section provides an overview of related work. A very thorough overview in the context of quantified programs can be found in [Wol15] and therefore mainly relevant literature published after 2014 is presented in this section. Further, as a significant part of this thesis deals with the restriction of the uncertainty set—the domain of universally quantified variables—an emphasis is placed on the topic of decision-dependent uncertainty.

2.2.1. Quantification of Variables

Quantified Linear/Integer Programming and Quantified Linear/Integer Implication The course of quantified programming until 2014 is surveyed very thoroughly in [Wol15]. In further publications Wolf and his co-authors extended the framework of quantified programming by the aspect of optimization by introducing an objective function [EL+11a]. Additionally, they conducted a geometric analysis for QLPs [LMW10], presented several results regarding the computational complexity of QLPs [Wol15] and closely connected quantified programming to modeling games [EL+11b, LOW13]. Furthermore, algorithms for general QLPs (with objective function) were developed and implemented: an alpha–beta nested Bender's decomposition was proposed to solve the quantified linear optimization problem and tested in a computational study [LW15, Wol15]. After 2014 new complexity results were obtained for variants of the quantified linear satisfiability problem (cf. Remark 2.1.7) [WES17]. In [WSE16, NP20] the computational complexity of special quantified integer satisfiability programs were examined and in [CH17] it was shown that the quantified integer satisfiability program with β alternating variable blocks is complete for the β-th level of the polynomial hierarchy. In [BW19] and [AGV18] quantified programs were utilized to obtain complexity bounds.

The *quantified linear implication problem* (QLI) was introduced in [ER+12], which extends quantified programming to implications of linear systems:

Definition 2.2.1 (Quantifed Linear/Integer Implication)**.**

$$Q \circ x \in \mathbb{R}^n : \ (Bx \leq d \Rightarrow Ax \leq b)$$

where Q is the vector of quantifiers, x the vector of variables, A and B are matrices and b and d are column vectors of appropriate size, is called the quantified linear implication problem *(QLI). If the integrality of the variables is required it is called the* quantified integer implication problem *(QII).*

QLI can be viewed as adding a constraint system that the universal variables have to satisfy, as $Bx \not\leq d$ immediately results in a true implication. Complexity results were shown for special sublasses of QLI and in particular it was shown that the general QLI is PSPACE-hard [ER+12, ER+14]. For the general QII and a subclass only few computational complexity results were obtained in [WSE16].

In contrast to the results mentioned above regarding QLI and QII, the extensions of quantified programming presented in this thesis refer to the optimization problem, rather than the satisfiability problem, and we focus on algorithmic properties and solution techniques. Furthermore, we deal differently with the case that the constraint systems on both sides of the implication are violated (see Remarks 4.2.6 and 5.2.5).

Quantified Boolean Formula The *quantified boolean formula problem* (QBF) can be viewed as a boolean QIP with $c = 0$ and each constraint being a clause, i.e. $A^\exists \in \{-1, 0, 1\}^{m_\exists \times n}$ and $b_i^\exists = |\{j \in \mathcal{I} \mid \ A_{i,j}^\exists = 1\}| - 1$ for each $\leq$-constraint $i \in \{1, \ldots, m_\exists\}$. Beside being the prototypical PSPACE-complete problem [Sto76] QBF allows a very compact problem description and thus several areas of application arise [SB+19]. Within the last few years several new techniques for solving QBF were developed and enhanced, such as clause selection [JMS15], clause elimination [HJ+15], quantified blocked clause elimination [LB+15], counter example guide abstraction refinement [JK+16], dependency learning [PSS19a], long-distance Q-resolution [PSS19b] as well as several preprocessing techniques [WR+17, LE19]. Further, nested SAT solvers were utilized for solving QBF [BJT16] and even machine learning techniques were succesfully adapted [Jan18]. Several solvers evolved and emerged [RT15, LE17, Ten19] and a competitive evaluation of solvers was revived [PS19, LSvG16].

Further, the *dependency quantified boolean Formula problem* (DQBF) generalizes the QBF by allowing an explicit specification of variable dependencies [PRA01, BCJ14], which corresponds to the henkin quantifier [HK65]. They can be interpreted as a game between a single universal player and multiple non-cooperative existential players with incomplete information, whereat each existential player observes universal variables assignments on which their further own decisions depend. The so-called Skolem function describes the evaluation of an existential variable under the possible assignments to its dependencies [BC+16]. DQBF allow elegant encoding and are very expressive, resulting in a recent surge of novel research results [Rab17, WK+17, SJ+19].

Quantified Constraint Satisfaction The *quantified constraint satisfaction problem* (QCSP) is a generalization of the QIP with arbitrary, instead of only linear constraints [CK04, FO07]. A framework for quantified constraint optimization was presented in [BLV08]. After 2014 (see [Wol15] for overview of preceding years) only few publications on QCSP can be found. A survey on non-boolean QCSP was presented in [Mar17]. In [CS14] nested constraint programs were introduced, which include QCSP as special case. The detection and certification of the falsity of QCSP were discussed [Che14, MOQ15] and further pruning mechanisms [BS14] and branching rules were examined [GW+20]. QCSP was used to render the design space for complex systems [HAW14] and schedulability tests [Zha16]. Further, QCSP is examined on finite monoids [CM16] and on semicomplete digraphs [ĐMM17]. Regarding our focus on the restriction of universal variables the results obtained in [BC09] should be mentioned, as they studied QCSPs in which the domains for each variable can be arbitrarily restricted. Additionally, we want to mention *constraint games*, which provide a constraint satisfaction framework for more than two players [LB+13, Pal19].

2.2.2. Optimization under Uncertainty

Immediately after the advent of mathematical programming the topic of incomplete information on input data gave rise to the research area of optimization under uncertainty. Seminal works of Dantzig [Dan55], Beale [Bea55] and Bellman [Bel57] were followed by numerous approaches regarding the handling of uncertainty. The most prominent—but fundamentally different—approaches are *stochastic programming* and *robust optimization*. While uncertain input data is assumed to follow a probability distribution in the former approach, the latter approach deals with deterministic set-based uncertainties [BBC11]. Other solution approaches for (optimization) problems under uncertainty include dynamic programming [Bel57, Ber01], fuzzy programming [CF12], interval programming [Hla12], approximation techniques [CF+93, MSU99, MV06], sampling [GP+04] and monte carlo tree search methods [BP+12, SS+17]. For more information we also want to refer to recent surveys regarding optimization under uncertainty [KA19, NY19, BDN19] and the references therein. In the following paragraphs we briefly review robust optimization and stochastic programming with focus on multistage problems.

Robust Optimization Robust optimization problems are mathematical optimization problems with uncertain data, where a valid solution is sought for any (anticipated) realization of that data [BTN02, BS07, Zha07, GMT14]. We focus on robust *linear* optimization problems (e.g. [BTGN09, BBC11]), i.e. when the underlying optimization problem is a mixed integer linear optimization problem [Sch86, NW88]. A survey regarding nonlinear robust optimization was conducted in [LM+20]. In robust optimization one is interested in finding "good" solutions that are immune to all realizations of the uncertain data within the so-called *uncertainty set*, even for the worst-case realization [BBC11].

Definition 2.2.2 (Robust Counterpart)**.**
Consider a mixed integer linear program with n variables, m constraints and variable domain $\mathcal{X} \subseteq \mathbb{R}^n$:

$$\min_{x \in \mathcal{X}} \left\{ c^\top x \mid Ax \leq b \right\} \tag{2.2}$$

Let $U \subseteq \mathbb{R}^{m \times n} \times \mathbb{R}^m \times \mathbb{R}^n$ be the so-called uncertainty set. We call

$$\min_{x \in \mathcal{X},\, t \in \mathbb{R}} \left\{ t \mid \left(t \geq c^\top x \wedge Ax \leq b \right) \ \forall (A, b, c) \in U \right\}$$

the robust counterpart of (2.2).

Solving the robust counterpart ensures feasibility of the solution regarding U but results in a high *price of robustness* [BS04], i.e. the solution is often too conservative. Different concepts were developed to overcome this problem, e.g. a reliability-index [BTN00], minimax regret [KY97], flexible adjustment to the level of conservatism [BS04], the concepts of light robustness [FM09], soft robustness [BTBB10], adjustable robustness [BTG+04, YGdH19] and recoverable robustness [LL+09], lexicographic α-robustness [KLV12], decision-dependent uncertainty (see the paragraph starting on page 20) or explorable uncertainty (see the paragraph starting on page 21). In [CG16] the authors discuss which approach is suitable for the problem at hand. A compact overview of prevailing uncertainty sets and robustness concepts can be found in [GYdH15, GS16] where the latter focusses on the algorithm engineering methodology with regard to robust optimization. Furthermore, in [KZ16] a survey on robust discrete optimization is presented.

Two-stage models, e.g. adjustable robust optimization (ARO) and recoverable robust optimization, are often challenging to compute as even for rather simple cases the problem is NP-hard [BTG+04]. Nevertheless, within the last few years several results regarding multistage[5] models were obtained (e.g. [BTG+04, BBC11, BC10, BTGN09, DI15]) and a tutorial-like survey on robust multitstage decision-making was conducted in [DI15]. An interesting discussion on multistage optimization can be found in [BTGN09, pp. 408–410] where the authors acknowledge its "*extreme applied importance*" but point out the computational problems that arise and question the usefulness of most approximation techniques. Besides frequently used variants of lot sizing problems (e.g. [BTG+05, BTGS09, BG15, PdH16, dRBT+17, BSZ19]) robust multistage optimization has been used for the daily operation of power systems [LS+16] as well as planning and scheduling problems [LG16, NY17, MNL19]. Dynamic programming techniques can be use to solve such multistage problems under uncertainty [Sha11], but often suffer from the curse of dimensionality. Other solution methods include variations of Bender's decomposition [TTE09] and Fourier–Motzkin elimination [ZDHS18]. Additionally, iterative splitting of the uncertainty set is used to solve robust multistage problems in [PdH16] and a partition-and-bound algorithm is presented in [BD16]. By considering specific robust counterparts a solution can be approximated and sometimes even guaranteed [BTG+04, BTGS09, CZ09]. Furthermore, several approximation schemes based on (affine) decision rules can be found in the literature (e.g. [BIP10, KWG11, ISS13, BG15, GKW19]). Also worth mentioning is the generalization

[5] We critically remark that the term "multistage" is frequently used as synonym for "two-stage".

of adaptive distributionally robust linear optimization to multistage problems [BSZ19], where uncertainty sets contain probability measures [GS10, WKS14].

Quantified programming can be viewed as a general framework for robust multistage linear optimization with interval uncertainty, as variables are solely bound within the hypercube given by the variable bounds. Also, QIP with polyhedral uncertainty set (see Chapter 4) can be interpreted as multistage discrete optimization with polyhedral uncertainty.

Stochastic Programming In contrast to robust optimization, a stochastic uncertainty model is used in stochastic programming, i.e. a probability distribution over the uncertain parameters is assumed. Instead of ensuring that a solution is applicable for any data realization within the considered uncertainty set, here the solution must be valid (and optimal) in a probabilistic sense regarding stochastic uncertainty [BGS11]. An overview of application areas can be found in [WZ05]. For a general and comprehensive review of stochastic optimization we refer to [BL11, Sha08, SDR09] and the references therein. We focus on *multistage* stochastic *(mixed) integer linear* optimization and for an overview regarding such problems we recommend reading [Sch03, Sen05, EG+12, ZAS19]. If only discrete probabilities are considered, a multistage stochastic problem is also referred to as a *game against nature* [Pap85]. In this case a *scenario tree* can be used to describe the multistage uncertainty [HR09, SDR09]. This concept is very similar to the use of game trees for quantified programs (see Subsection 2.1.2) as also pointed out in [Wol15].

Similiar to theoretical results for robust multistage optimization the computational complexity explodes in the number of stages [SN05, DS06]. The *equivalent deterministic program* (DEP) [Wet74] can be used to reformulate the stochastic program into a traditional MIP but often cannot be solved using standard solvers, as their size grows exponentially with the input size. Hence, decomposition schemes [Bir85, Rus97, SZ14] gained in importance for large-scale instances. There are essentially two decomposition-based strategies: decomposition by time stages, e.g. the Bender's or L-shaped decomposition [Ben62, VSW69, QS17], and decomposition by scenario, e.g. progressive hedging [RW91, GH+16] and dual decomposition [CS99]. Recently a multistage scenario decomposition approach for mixed binary programs was published [EGU16]. Furthermore, the so-called branch-and-fix coordination method gained in importance, which is used to generate independent scenario clusters and coordinates the selection of branching nodes and variables (e.g. [AE+17]). Other recent advances for stochastic mixed integer programs consider decision rules (e.g. [BL18, DBL20]), sampling-based techniques (e.g. [PWB19]) and stochastic dual dynamic programming (e.g. [ZAS19]).

Quantified integer programming can be interpreted as worst-case multistage stochastic mixed integer linear programming with discrete probabilities for uncertain right-hand side parameters. In particular, in QIPs the worst case—and not the expected value—is optimized with regard to the set of scenarios (the uncertainty set). However, in Subsection 2.3.2 we present a QMIP model in which the uncertainty set contains a set of scenarios that appear with specific relative frequencies (i.e. probabilities) with the optimization goal to minimize the overall (expected) costs.

2.2.3. Optimization under Decision-Dependent Uncertainty

Uncertainty can be classified as *endogenous* and *exogenous*, whereby endogenous uncertainties can be affected by planning decisions and exogenous uncertainties cannot [LG18a]. In optimization under uncertainty in general it is frequently assumed that the occurring uncertainty is embedded in a predetermined uncertainty set or that it obeys a fixed random distribution, i.e. it is assumed to be exogenous. We use the term *decision-dependent* uncertainty for problems in which realizations of uncertain parameters can be manipulated by decisions made be the planner. Others stick with the term endogenous uncertainty (e.g. [GG14]) or use other terms like *variable uncertainty* (e.g. [Pos14]) or *adjustable uncertainty set* [ZK+17]. For both stochastic as well as robust optimization under uncertainty several results were obtained regarding decision-dependent uncertainty.

Robust Optimization with Decision-Dependent Uncertainty Decision-dependent uncertainty in (adjustable) robust optimization ((A)RO) results in an uncertainty set $U(x)$, which depends on the decision variables x. One of the first applications of RO with decision-dependent uncertainty was a software partitioning problem [SW+12] with uncertain execution order and unknown frequency of segment calls, where code segments must be assigned to different nodes. Poss introduced decision-dependent budgeted uncertainty and applied them to robust knapsack problems [Pos13] and RO with cost uncertainty [Pos14], and further provided results for RO with knapsack uncertainty [Pos18]. In [LRS+19] decision-dependent polyhedral uncertainty sets for the multitasking scheduling problem were constructed. In [LG18a] a generic polyhedral form with decision-dependence in both left- and right-hand sides is presented and the robust counterpart is derived. A more general framework for RO with decision-dependent uncertainty is presented in [NS18] and illustrated using the example of disaster control, where for a shortest path problem with uncertain edge lengths the edges to be reinforced must be determined. Combinatorial problems with uncertain costs were studied in [Con19].

In [AP19a] mixed integer ARO with decision-dependent uncertainty is interpreted as tri-level problem, allowing the adoption of algorithms for tri- and multilevel problems (e.g. [AP19b]). For process scheduling with uncertain processing-time of the jobs, an ARO model is used in [LG16], whereat the range of the uncertain processing-time parameters depend on the scheduling time of the job itself. The selection of specific scheduling times therefore actively determine the range in which the uncertain processing-times are expected. A general framework for resilient supply chain optimization is proposed in [ZY19] using an ARO model with decision-dependent uncertainty. Solution techniques for robust problems under decision-dependent uncertainty include (repeatedly) solving deterministic reformulations [Pos18, NS18], column and constraint generation algorithms [MGK19], decomposition techniques [Con19] and approximation algorithms [Pos18]. Further application-driven areas of robust optimization where decision-dependent uncertainty was deployed include the distribution-free dynamic pricing problem [BV17], scheduling problems [LG16, LG18b, VGM16], design of resilient networks [MGK19], radiation therapy

[NR17], adaptive water resources planning [EPH20] and robust kidney exchange [MBD19]. Also worth mentioning is the use of decision-dependent sets for robust optimal control [ZK+17].

Stochastic Optimization with Decision-Dependent Uncertainty One of the first results on decision-dependent uncertainty in stochastic processes can be found in [Pfl90] where a Markov chain with control-dependent transition is examined. Later, algorithmic procedures for solving a specific class of stochastic problems, where the distribution of random parameters depends on decision variables, was presented in [JWW98]. A framework in which decisions affect the disclosure time of uncertain parameters is introduced in [GG06]. Furthermore, decision-dependent ambiguity sets for distributionally robust stochastic programming are examined (e.g. [RW17, NRL18]) and computational strategies for non-convex, nonlinear stochastic programs with decision-dependent uncertainty are presented in [TGG13]. The classification of the different types of decision-dependent uncertainty in stochastic programming is surveyed in [HBT18].

Solution techniques for general multistage stochastic programs under decision-dependent uncertainty are decomposition algorithms [GG14, CC18], algorithms for the generation and exploitation of bounds [GG05, CC18, ZC19], and other heuristics [GG11, VKR11, HM16, Con17, EG+20]. There also are several application-oriented results: Stochastic programming is used for planning oil and gas field infrastructure, where uncertainties depend on design and operation decisions [Jon98, TGG09]. In [GP+06] preventive maintenance actions are optimized that can change future failure distributions with the goal of minimizing planned maintenance costs and unplanned repair costs. To strengthen an urban highway system against earthquakes, a stochastic optimization problem is used in [PS+10], where investment decisions decrease the likelihood of link failure by altering the probability distribution. The optimization of project portfolios is examined in [SC+10]. In [WSP12] a stochastic dynamic programming formulation is used to model global climate policy under decision-dependent uncertainty and dynamic programming techniques are applied. A stochastic model for a vehicle routing problem is examined in [HKM16], where the uncertain customer demand is disclosed if the customer is visited. The optimal selection of thermal and wind power units is examined in [ZZ18]. For a general introduction and a further detailed literature review regarding stochastic programming under decision-dependent uncertainty we recommend [Apa17].

Explorable Uncertainty Instead of allowing to change the nature of an uncertainty set, *explorable uncertainty* investigates the exploration of the uncertainty set: by querying parts of the uncertainty set, the decision-maker can gain better understanding of what to expect; but at a price. Hence, the question is which queries are indeed beneficial in order to obtain a better or even optimal solution. Such problems arise when more precise input data can be obtained with additional effort. This research question was first raised for a selection problem in [Kah91]. Other research deals with finding the median [FM+00], the shortest path problem [FM+07], finding a minimal spanning tree [EH+08, MMS17], selection problems [GSS11, EHK16], scheduling problems [Erl18, LMS19], knapsack problems [GG+15], sorting [HdL19], the computation of function values [KT01] and computing the convex hull [BH+05]. Further, a survey of queryable

uncertainty was published [EH+15]. Most of these publications aim at minimizing the number of queries to guarantee optimal solutions. For a caching problem a trade-off between performance (few queries) and the precision of the solution is investigated in [OW00]. For scheduling problems the trade-off between the benefit of a query (a scheduled job) and the resulting increase in the objective value is examined in [Erl18, DE+18].

2.3. Examples

2.3.1. Robust Runway Scheduling

Deviations in aircraft arrival times have an enormous impact on air traffic planning. An initial plan quickly becomes useless if it cannot be implemented in the real world, due to disturbances. The question whether quantified programs could be used for robust runway scheduling problems as in [HH+16] came up in a conversation with colleagues [LL16]. Under my supervision, a preliminary model was developed [Gna16], which served as the basis for further refinements and implementations.

Assume there are b runways at an airport and all arriving airplanes have to be assigned to exactly one time slot and runway for the landing. Each airplane is expected to arrive within some time window and hence the assigned time slot must be contained in this time window. Finding an initial matching, even an optimal one considering some objective function, can be modeled and solved using mixed integer programming techniques [Hel11]. However, the time windows are uncertain due to adjusted airspeed or operational problems and an initial schedule might become invalid (e.g. Figure 2.2). Thus, one is interested in a robust initial plan that can

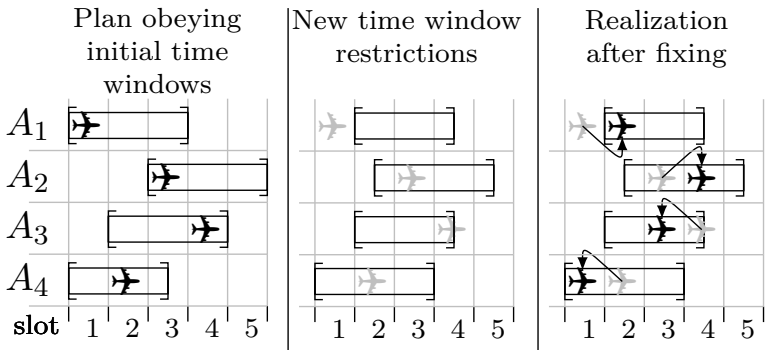

Figure 2.2.: Process of runway scheduling for $b = 1$: An initial plan is made (left) in which each airplane A_i is scheduled within its predicted time window. If the predicted time windows differ from the disclosed time windows (middle), the initial plan becomes invalid and a new scheduling must be found (right).

be adapted cheaply, e.g. the initial and adapted time slot of each airplane should not be too far apart [HH+16]. These uncertain events, however, do not uncover all at the same time: final time slots must be assigned to some airplanes while for other airplanes the actual time window is still unknown. We use a QIP to model and solve such a matching problem.

The problem is a b-matching: Each airplane must be assigned to one time slot and at most b airplanes can be assigned to one time slot. Further, the assigned time slot of an airplane must lie within the time window in which the airplane can land. Let S be the set of *time slots* and A the set of *airplanes*. For each airplane $i \in A$ the actual time window in which it must land is given by the first time slot $s_i \in \mathsf{S}_i \subseteq S$ and the length of the window $d_i \in D_i \subseteq \mathbb{N}$, i.e. the number of time slots within the time window (minus one). The realization of this time window $\{s_i, \dots, s_i + d_i\}$ is uncertain and therefore the variables s_i and d_i are universal variables. The potential starting time slots S and their lengths D are the domain of the universal variables and obviously must be chosen carefully. The existential player first has to specify an initial plan, using variables $x_{i,j}$ that indicate whether airplane $i \in A$ is assigned to time slot $j \in S$. Then the universal player first reveals the actual time windows for certain airplanes and the existential player then has to present the fixed plan (the final assignments) for those airplanes via indicator variables $y_{i,j}$. This can happen in a multistage manner, i.e. revealing and fixing occurs repeatedly until all airplanes are assigned to time slots within their actual time window.

The resulting costs are composed of the costs for the initial plan and the fixing costs. The costs for the initial plan only depend on the initial assignment of planes to time slots regarding predetermined costs. For the composition of the fixing costs several models can be applied, e.g. rescheduling an airplane results in a fixed fee, rescheduling an airplane results in costs depending on the newly selected time slot, or rescheduling an airplane results in costs depending on the initial and the newly selected time slot. There are several other options for the objective function, e.g. minimizing the workload of the air traffic controllers [KB+19], which aims at keeping the landing sequence of the airplanes unchanged. For a more general presentation, the cost of replacing airplane i depends on a cost function $c(x_{i,\star}, y_{i,\star})$ representing the (linear) relation between initial plan, fixed plan and fixing costs.

For simplicity the model below has only one universal variable block, i.e. all time windows are revealed simultaneously. The only aspect that needs to be changed in order to obtain a (real) multistage instance is the order of the variables and thus the quantification sequence, as briefly described below.

$$\min \sum_{i \in A} \sum_{j \in S} c_{i,j} x_{i,j} + \max \left\{ \min \left\{ \sum_{i \in A} c(x_{i,\star}, y_{i,\star}) \right\} \right\} \tag{2.3}$$

$$\text{s.t. } \exists\, x \in \{0,1\}^{|A| \times |S|} \quad \forall\, s \in \mathsf{S},\, d \in D \quad \exists\, y \in \{0,1\}^{|A| \times |S|}\text{:}$$

$$\sum_{j \in S} x_{i,j} = 1 \qquad \forall\, i \in A \tag{2.4}$$

$$\sum_{i \in A} x_{i,j} \leq b \qquad \forall\, j \in S \tag{2.5}$$

$$\sum_{j \in S} y_{i,j} = 1 \qquad \forall\, i \in A \tag{2.6}$$

$$\sum_{i \in A} y_{i,j} \leq b \qquad \forall\, j \in S \tag{2.7}$$

$$s_i \leq \sum_{j \in S} j \cdot y_{i,j} \leq s_i + d_i \qquad \forall\, i \in A \tag{2.8}$$

Constraints (2.4) and (2.5) as well as (2.6) and (2.7) ensure, that both the initial plan as well as the final fixed plan is a b-matching. Constraint (2.8) ensures that the finally assigned time slot for each airplane is contained in its time window given by the universal player. Note that we do not explicitly consider separation requirements between consecutive airplanes (see e.g. [Söl12]). But since we are using time slots, rather than actual landing times, this requirement can either be added by demanding a certain time slot distance, or the time slot length itself ensures that separation requirements are met. The Objective (2.3) aims at the minimization of the overall costs, consisting of costs for the initial plan and the costs for adapting the initial plan according to the disclosed time windows. For example, if there is a fixed fee for replanning an airplane, further existential variables $Z \in \{0,1\}^{|A|\times|S|}$ are added to the final existential stage and the following constraints would be added:

$$z_{i,j} \geq y_{i,j} - x_{i,j} \qquad \forall\, i \in A, j \in S \tag{2.9}$$

$$c(x_{i,\star}, y_{i,\star}) = \sum_{j \in S} r_i z_{i,j} \qquad \forall\, i \in A \tag{2.10}$$

In this case, variable $z_{i,j}$ indicates whether airplane i is ultimately scheduled in time slot j ($y_{i,j} = 1$) but was not initially scheduled there ($x_{i,j} = 0$), resulting in rescheduling cost r_i.

If the time windows are not revealed simultaneously, more than one universal variable block arises. In this case, the set of airplanes A as well as the universal domains S and D are split up according to the groups of airplanes with simultaneously disclosed time windows.

The variables for the time windows, s and d, and the planning variables y are then ordered in the quantification sequence according to these groups. Hence, such a multistage runway scheduling problem under uncertainty can have up to $|A|$ universal variable blocks if for each airplane the time window is revealed individually.

2.3.2. Resilient Booster Stations

On the following pages the goal is to generate cost-efficient, resilient booster stations out of predefined non-resilient ones, as proposed in [HH+18]. The requirements for the case of resilient booster stations are manifested in the *DVGW*[6] *code of practice "DIN 1988-500: Pressure boosting stations with RPM-regulated pumps"* [DIN11]. It states that booster stations must have at least one stand-by pump. If one pump breaks down, the system must be able to satisfy the peak flow and thus all demanded loads at any time. In order to avoid stagnation water, an automatic, cyclic interchange between all pumps including the stand-by pumps is necessary. Therefore, all pumps have to operate at least once in 24 hours. This additional requirement is strongly connected to the cost-efficiency goal.

The relevant costs in the considered case are the investment costs for the additional pumps as well as the operational costs of the overall system over a predefined lifespan. As the breakdown cases are expected to only take place in a small amount of time compared to the lifespan, due to short repair times, they do not significantly affect the operational costs of the system and are

[6] German Technical and Scientific Association for Gas and Water.

therefore neglected. However, the requirement for all pumps to operate once in 24 hours, i.e. in at least one of the daily repeating load scenarios, massively affects the operational costs. Given this circumstance, it is not a trivial task to determine by which stand-by pumps the system should be extended in order to obtain a cost-optimal system. Theoretically, a set of pumps or entire subsystems can be connected either in parallel or in series. However, according to today's practice only parallel connections are favorable from a technical point of view and we therefore assume a parallel arrangement (for further details we refer to [HH+18]).

The QMIP consists of $\beta = 5$ stages. The first existential block primarily represents the investment decision concerning the additional pumps. In the universal second variable block the load scenario is selected. The existential third variable block is used to determine the cost-optimal operating point of the available pumps for the given scenario. In the following universal variable block one of the initial pumps is chosen for breakdown. The final existential block is used to check whether the remaining pumps (without the broken one) are able to fulfill the selected load scenario.

As the handling of the breakdown- and standard-control is independent—and only depends on the first-stage investment decision—we could also have built a three-stage model: investment decision (first stage), selection of a load and a breakdown scenario (second stage), and finally computing the standard- and breakdown-control (third stage). However, using five stages has severe advantages. First, the chosen variable sequence indicates the processing order more accurately: for any scenario, we must provide a standard-control first and subsequently valid breakdown-controls must be ensured for the particular scenario. Second, the corresponding deterministic equivalent program contains significantly less variables, since the standard-control decision variables do not have to be duplicated for each breakdown scenario [Wol15]. A similar argument is valid for game tree search methods: if modeled as a three-stage QMIP the standard-control found for one breakdown scenario must be rediscovered for another breakdown scenario, even though it could simply stay the same. Table 2.1 displays the parameters and Table 2.2 the

Table 2.1.: Parameters of the QMIP for the design of resilient booster stations.

parameter	description
$I = \{1, \dots, n\}$	set of initial pumps
$A = \{n+1, \dots, n+m\}$	set of purchasable additional pumps
$P = A \cup I$	set of all available pumps
$S = \{1, \dots, \bar{S}\}$	set of scenarios
$C \in \mathbb{R}_+^A$	C_p: investment cost for pump p
$Q \in \mathbb{R}_+^S$	Q_i: demanded volume flow in scenario i
$H \in \mathbb{R}_+^S$	H_i: demanded pressure increase in scenario i
$R \in [0,1]^S$	R_i: relative frequency of scenario i
$C^{kWh} \in \mathbb{R}_+$	costs per kilowatt hour of electricity
$H^{\max} \in \mathbb{R}_+$	general upper bound for pressure increase
$Q^{\max} \in \mathbb{R}_+$	general upper bound for volume flow
$T \in \mathbb{R}_+$	projected operational lifespan of the system

used variables. For the sake of compact presentation, we do not explicitly state the quantification

sequence $\mathcal{Q} \circ x$. However, in Table 2.2 both the corresponding stage and thus the order of the variables, as well as the variable quantification is given.

Table 2.2.: Variables of the QMIP for the design of resilient booster stations.

variable	stage	description
$y \in \{0,1\}^A$	1(∃)	y_p : indicates buying decision of pump p
$u \in \{0,1\}^{P \times S}$	1(∃)	$u_{p,s}$: indicator whether pump p is used in scenario s
$c \in \mathbb{R}_+^S$	1(∃)	c_i: operational costs in scenario i
$s \in S$	2(∀)	s: variable for selection of the load scenario
$\sigma \in \{0,1\}^S$	3(∃)	σ_i: indicator whether scenario i was selected
$x \in \{0,1\}^P$	3(∃)	x_p: indicator whether pump p is used
$q \in \mathbb{R}_+^P$	3(∃)	q_p: volume flow through pump p
$h \in \mathbb{R}_+^P$	3(∃)	h_p: pressure increase by pump p
$\rho \in \mathbb{R}_+^P$	3(∃)	ρ_p: power consumption of pump p
$n \in [0,1]^P$	3(∃)	n_p: rotational speed of pump p
$b \in I$	4(∀)	b: variable for selection of the broken pump
$\beta \in \{0,1\}^I$	5(∃)	β_p: indicator whether initial pump p is broken
$x^B \in \{0,1\}^P$	5(∃)	x_p^B: indicator whether pump p is used in case of disturbance
$q^B \in \mathbb{R}_+^P$	5(∃)	q_p^B: volume flow through pump p in case of disturbance
$h^B \in \mathbb{R}_+^P$	5(∃)	h_p^B: pressure increase by pump p in case of disturbance

$$\min \sum_{i \in S} R_i c_i + \sum_{p \in A} C_p y_p \tag{2.11}$$

$$\text{s.t.} \quad x_p \le y_p,\ x_p^B \le y_p \quad \forall p \in A \tag{2.12}$$

$$u_{p,i} - x_p + s_i \le 1 \quad \forall p \in P,\ i \in S \tag{2.13}$$

$$\sum_{i \in S} u_{p,i} \ge 1 \quad \forall p \in I \tag{2.14}$$

$$\sum_{i \in S} u_{p,i} \ge y_p \quad \forall p \in A \tag{2.15}$$

$$\sum_{i \in S} \sigma_i \le 1 \tag{2.16}$$

$$\sum_{i \in S} i\sigma_i = s \tag{2.17}$$

$$\sum_{p \in I} \beta_p \le 1 \tag{2.18}$$

$$\sum_{p \in I} p\beta_p = b \tag{2.19}$$

$$x_p^B + \beta_p \le 1 \quad \forall p \in I \tag{2.20}$$

$$\rho_p = \rho_p(q_p, n_p) \quad \forall p \in P \tag{2.21}$$

$$h_p = h_p(q_p, n_p) \quad \forall p \in P \tag{2.22}$$

$$h_p^B = h_p^B(q_p^B) \quad \forall p \in P \tag{2.23}$$

$$h_p = x_p \sum_{i \in S} H_i \sigma_i \quad \forall p \in P \tag{2.24}$$

$$h_p^B = x_p^B \sum_{i \in S} H_i \sigma_i \quad \forall p \in P \tag{2.25}$$

$$\sum_{p \in P} q_p = \sum_{i \in S} Q_i \sigma_i \tag{2.26}$$

$$\sum_{p \in P} q_p^B = \sum_{i \in S} Q_i \sigma_i \tag{2.27}$$

$$q_p \le Q^{\max} x_p \quad \forall p \in P \tag{2.28}$$

$$h_p \le H^{\max} x_p \quad \forall p \in P \tag{2.29}$$

$$q_p^B \le Q^{\max} x_p^B \quad \forall p \in P \tag{2.30}$$

$$h_p^B \le H^{\max} x_p^B \quad \forall p \in P \tag{2.31}$$

$$M(1 - \sigma_i) + c_i \ge C^{kWh} T \textstyle\sum_{p \in P} \rho_p \quad \forall i \in S \tag{2.32}$$

The Objective Function (2.11) aims at minimizing the weighted operational costs in the scenarios as well as the costs resulting from buying additional pumps. Constraint (2.12) links the first and third variable block as well as the first and fifth variable block by demanding that only purchased pumps can be used. The feature that each pump must be used in at least one of the appearing load scenarios is guaranteed by Constraints (2.13), (2.14) and (2.15). Constraints

(2.16)–(2.19) link the universal variable decision of the selected scenario and the selected broken pump with the corresponding existential variables while Constraint (2.20) prohibits the use of a broken pump. The operating point of a used pump must lie on its characteristic curve, which describes the nonlinear relation between h_p, q_p, n_p and ρ_p. This coherence is outlined in (2.21) and (2.22) and must be linearized. As the power consumption of the booster station in the case of a breakdown is not subject of the optimization the nonlinear relation between h_p^B and q_p^B can be modeled more easily: (2.23) ensures that the selected operating point (h_p^B, q_p^B) lies somewhere within the characteristic map without specifying the speed of the pump. Hence, linearizing the boundaries of the map and checking their fulfillment suffices. Constraints (2.24) - (2.27) ensure that the demanded volume flow and pressure increase of the selected load scenario are fulfilled in both the standard-control and the breakdown-control. Note that resolving the nonlinearity in (2.24) and (2.25) is a trivial task by using a Big M formulation. Constraints (2.28)–(2.31) set bounds on the volume flow and the pressure increase of a used pump and deal with unused pumps in particular. In (2.32) the power consumption resulting from the selected standard-control is transformed into energy costs. Note that the cost variables c_i are first-stage variables. This is necessary in order to be able to compute the overall costs instead of only obtaining the costs in each scenario.

The universal integer variables s and b and the existential binary variables σ and β are very similar and closely linked through Constraints (2.16)–(2.19). One might suggest that the binary variables σ and β could just as well be universal variables and thus replacing s and b. However, exactly one load and one breakdown scenario each must be selected. This would lead to a restriction of these variables as it is done in Constraint (2.16) and (2.18). But restricting universal variables using linear constraints (instead of simple variable bounds) cannot be done straightforward using standard QIP (see Chapter 4). In Subsection 4.4.2 we present the corresponding model with constraints restricting such binary universal variables.

Our solver Yasol can only deal with continuous variables in the final (existential) stage (see Subsection 2.1.3). Thus, the above model cannot be solved via this solver. However, by building the deterministic equivalent program (DEP) (see Subsection 3.1.1) and utilizing a standard MIP solver this QMIP can be solved. Further, in order to be able to use Yasol directly for this problem one could do the following: The continuous cost variable c in the first stage could be converted into an integer variable, which only leads to a minor limitation. Furthermore, as the continuous operating variables in stage 3 are independent of the universal variables in stage 4 the entire QIP could be turned into a 3-stage instance, as discussed before. Consequently, all continuous variables would be in the third and final stage, making Yasol applicable.

3. Algorithmic Properties of QIPs

3.1. Solution Techniques and Pruning Mechanisms

There are two different ways known how to tackle a QIP: A deterministic equivalent program—also called robust counterpart—can be built, similar to the ones known from stochastic programming [Wet74] and robust optimization [BTN99], and solved using standard integer programming solvers. On the other hand, the more direct approach is to conduct a game tree search [Alt88, All94, FMM92, DL04, SH+16]. We are particularly interested in the latter, as our solver combines techniques known from game tree search, linear programming and (quantified) boolean formula within an alpha-beta framework [EH+17, KM75]. During such a game tree search we are interested in quickly evaluating or estimating the minimax value of nodes, in order to find the optimal (existential) strategy of the corresponding subtree. To speed up the search process, limiting the number of subtrees that need to be explored is extremely beneficial. Such pruning operations are applied in many search based algorithms, e.g. the alpha-beta algorithm [KM75], branch-and-bound [NW88] and the Davis–Putnam–Logemann–Loveland algorithm [DLL62, GNT03]. In the following subsections we first revisit the deterministic equivalent program for QIPs and then present three approaches that allow pruning in a QIP game tree search.

3.1.1. Deterministic Equivalent Program

Building a deterministic equivalent for problems under uncertainty is a powerful tool both in stochastic programming [Wet74, BL11] and in robust optimization [BTN99, LDF11, BBC11]. For quantified programs a deterministic equivalent can be built in some cases: If only continuous variables are present, i.e. a QLP, the corresponding exponentially grown DEP can be tackled by exploiting the resulting matrix structure by using decomposition techniques [Wol15]. For QMIPs with only continuous existential variables and only discrete universal variable the corresponding QLP-relaxation yields the optimal solution [Wol15] and hence relaxing the integrality of the universal variables and building the DEP of the QLP-relaxation is a suitable way to solve such problems. For QMIPs in general, however, it is not possible to utilize a deterministic equivalent: It no longer suffices to consider the lower and upper bounds of universal variables, as a convex combination of the corresponding winning strategies no longer yields a winning strategy in general.

Example 3.1.1. *Consider the following QMIP without objective function:*

$$\forall\, x_1 \in [0,1]\ \exists\, x_2 \in \{0,1\} :\ x_1 + x_2 = 1$$

Obviously for both $x_1 = 0$ and $x_1 = 1$ there is a fulfilling existential assignment $x_2 = 1 - x_1$. However, for any $x_1 \in]0, 1[$ the constraint cannot be fulfilled. Thus, the existence of a strategy cannot be deduced from the consideration of the bounds.

For pure QIPs it is possible to derive a deterministic equivalent. We first briefly revisit the DEP of a QIP. For further details on the DEP we refer to [Wol15], but note that we introduce a slightly different notation in order to reuse it in Section 5.6.1.

Definition 3.1.2 (Deterministic Equivalent Program of a QIP)**.**
Let $P = (A^\exists, b^\exists, c, \mathcal{L}, Q)$ be a QIP. For a given scenario $s \in \mathcal{L}_\forall$ and existential variable block $i \in \mathcal{E}$ we call $\Sigma(s, i)$ the set of scenarios similar to s up to block i given by

$$\Sigma(s,i) = \left\{ \sigma \in \mathcal{L}_\forall \mid \forall j \leq \sum_{k \in \mathcal{A}:\, k<i} |B_k| :\ s_j = \sigma_j \right\}.$$

For $s \in \mathcal{L}_\forall$ the vector $x_s \in \mathcal{L}$ represents a variable vector in which the universal variables are fixed according to scenario s. Then the deterministic equivalent program *of P is given as follows:*

$$\begin{aligned}
&\min\ k && && (3.1)\\
&\text{s.t. } c^\top x_s \leq k && \forall\, s \in \mathcal{L}_\forall && (3.2)\\
&\quad A^\exists x_s \leq b^\exists && \forall\, s \in \mathcal{L}_\forall && (3.3)\\
&\quad (x_s)_\forall = s && \forall\, s \in \mathcal{L}_\forall && (3.4)\\
&\quad (x_s)_\exists \in \mathcal{L}_\exists && \forall\, s \in \mathcal{L}_\forall && (3.5)\\
&\quad x_s^{(i)} = x_\sigma^{(i)} && \forall\, i \in \mathcal{E},\ s, \sigma \in \mathcal{L}_\forall,\ s \neq \sigma,\ \sigma \in \Sigma(s,i) && (3.6)
\end{aligned}$$

The presented deterministic equivalent is a *split-view formulation* [BH92] where the existential constraint system is replicated for each scenario. It must be ensured that scenarios with a common history must have the same set of decisions. This *nonanticipativity property* (also see [BH92] and [Wet74]) is ensured by Constraint (3.6): the existential variables $x_s^{(i)}$ and $x_\sigma^{(i)}$ of block i in the presence of scenarios s and σ, respectively, have to be invariant, if the scenario s and σ are similar up to block i.

Example 3.1.3. *Consider the following QIP with binary variables. We use the block notation, i.e. $x^{(i)}$ for variable block i instead of x_j for each variable j, in order to avoid rampant indexes, even though each block contains only a single variable.*

$$\begin{array}{llllll}
\min & x^{(1)} & +x^{(2)} & -x^{(3)} & -x^{(4)} & -x^{(5)} \\
\text{s.t.} & \exists\, x^{(1)} & \forall\, x^{(2)} & \exists\, x^{(3)} & \forall\, x^{(4)} & \exists\, x^{(5)} \quad \in \{0,1\}^5 \\
& -2x^{(1)} & +x^{(2)} & +x^{(3)} & & \leq 0 \\
& x^{(1)} & & +x^{(3)} & +x^{(4)} & +x^{(5)} \quad \leq 2
\end{array}$$

The set of possible scenarios is given by $\mathcal{L}_\forall = \{(0,0), (0,1), (1,0), (1,1)\}$. The corresponding deterministic equivalent with the fixed universal scenarios variable $(x_s)_\forall$ already incorporated into the right-hand side value looks as follows:

$$
\begin{array}{lrrrrrrrrrl}
\min & k & & & & & & & & & \\
\text{s.t.} & x^{(1)}_{(0,0)} & -x^{(3)}_{(0,0)} & & -x^{(5)}_{(0,0)} & & & & & -k & \leq 0 \\
 & x^{(1)}_{(0,1)} & -x^{(3)}_{(0,1)} & & & -x^{(5)}_{(0,1)} & & & & -k & \leq 1 \\
 & x^{(1)}_{(1,0)} & & -x^{(3)}_{(1,0)} & & & -x^{(5)}_{(1,0)} & & & -k & \leq -1 \\
 & x^{(1)}_{(1,1)} & & -x^{(3)}_{(1,1)} & & & & -x^{(5)}_{(1,1)} & & -k & \leq 0 \\
 & -2x^{(1)}_{(0,0)} & +x^{(3)}_{(0,0)} & & & & & & & & \leq 0 \\
 & x^{(1)}_{(0,0)} & +x^{(3)}_{(0,0)} & & +x^{(5)}_{(0,0)} & & & & & & \leq 2 \\
 & -2x^{(1)}_{(0,1)} & +x^{(3)}_{(0,1)} & & & & & & & & \leq 0 \\
 & x^{(1)}_{(0,1)} & +x^{(3)}_{(0,1)} & & & +x^{(5)}_{(0,1)} & & & & & \leq 1 \\
 & -2x^{(1)}_{(1,0)} & & +x^{(3)}_{(1,0)} & & & & & & & \leq -1 \\
 & x^{(1)}_{(1,0)} & & +x^{(3)}_{(1,0)} & & & +x^{(5)}_{(1,0)} & & & & \leq 2 \\
 & -2x^{(1)}_{(1,1)} & & +x^{(3)}_{(1,1)} & & & & & & & \leq -1 \\
 & x^{(1)}_{(1,1)} & & +x^{(3)}_{(1,1)} & & & & +x^{(5)}_{(1,1)} & & & \leq 1 \\
 & x^{(1)}_{(0,0)} = x^{(1)}_{(0,1)} = x^{(1)}_{(1,0)} = x^{(1)}_{(1,1)} & & & & & & & & & \\
 & x^{(3)}_{(0,0)} = x^{(3)}_{(0,1)} & & & & & & & & & \\
 & x^{(3)}_{(1,0)} = x^{(3)}_{(1,1)} & & & & & & & & & \\
 & x^{(i)}_{s} \in \{0,1\} \quad \forall i \in \mathcal{E},\ s \in \mathcal{L}_{\forall} & & & & & & & & &
\end{array}
$$

The invariant variables—linked through the Nonanticipativity Constraint (3.6)—already (visually) share the same column. Hence, if this link is realized explicitly by replacing them by surrogates, the DEP is equal to the compact-view formulation [RW91] (also see [Wol15]).

$$
\begin{array}{lrrrrrrrrrl}
\min & k & & & & & & & & & \\
\text{s.t.} & x^{(1)} & -x^{(3)}_{(0)} & & -x^{(5)}_{(0,0)} & & & & & -k & \leq 0 \\
 & x^{(1)} & -x^{(3)}_{(0)} & & & -x^{(5)}_{(0,1)} & & & & -k & \leq 1 \\
 & x^{(1)} & & -x^{(3)}_{(1)} & & & -x^{(5)}_{(1,0)} & & & -k & \leq -1 \\
 & x^{(1)} & & -x^{(3)}_{(1)} & & & & -x^{(5)}_{(1,1)} & & -k & \leq 0 \\
 & -2x^{(1)} & +x^{(3)}_{(0)} & & & & & & & & \leq 0 \\
 & x^{(1)} & +x^{(3)}_{(0)} & & +x^{(5)}_{(0,0)} & & & & & & \leq 2 \\
 & x^{(1)} & +x^{(3)}_{(0)} & & & +x^{(5)}_{(0,1)} & & & & & \leq 1 \\
 & -2x^{(1)} & & +x^{(3)}_{(1)} & & & & & & & \leq -1 \\
 & x^{(1)} & & +x^{(3)}_{(1)} & & & +x^{(5)}_{(1,0)} & & & & \leq 2 \\
 & x^{(1)} & & +x^{(3)}_{(1)} & & & & +x^{(5)}_{(1,1)} & & & \leq 1 \\
 & x^{(1)}, & x^{(3)}_{(0)}, & x^{(3)}_{(1)}, & x^{(5)}_{(0,0)}, & x^{(5)}_{(0,1)}, & x^{(5)}_{(1,0)}, & x^{(5)}_{(1,1)} & & & \in \{0,1\}
\end{array}
$$

3.1.2. Monotone Variables

For QIPs a rather simple argument exists such that certain variable assignments never need to be checked as they are worse than their counterparts. This concept of monotone variables is well known in the field of quantified boolean formulas [CS+02] and integer programming [NW88].

Definition 3.1.4 (Monotonicity in a QIP)**.**
A variable x_k is called positive (negative) monotone *if it occurs with only positive (negative) sign in the matrix and objective, i.e. if the entries of $A^\exists$ and c belonging to x_k are all non-negative (non-positive).*

First, we show that there is a relation between two leaves in the game tree corresponding to completely assigned variable vectors that only differ in the entry of a monontone (binary) variable. After that we show that for monotone (binary) variables only one assignment must be considered during a game tree search.

Lemma 3.1.5. *Given a binary QIP, its game tree $G = (V, E, e)$ and a positive monotone variable x_k, $k \in \mathcal{I}$. For any two leaves $v^{(0)}$ and $v^{(1)}$ of the game tree represented by the fixed variable vectors $\tilde{x}^{(0)} = (\tilde{x}_1, \dots, \tilde{x}_{k-1}, 0, \tilde{x}_{k+1}, \dots, \tilde{x}_n) \in \mathcal{L}$ and $\tilde{x}^{(1)} = (\tilde{x}_1, \dots, \tilde{x}_{k-1}, 1, \tilde{x}_{k+1}, \dots, \tilde{x}_n) \in \mathcal{L}$, respectively, it is $minimax(v^{(0)}) \leq minimax(v^{(1)})$.*

Proof. If $A^\exists \tilde{x}^{(0)} \not\leq b^\exists$ then $minimax(v^{(0)}) = +\infty$. Hence, a constraint $i \in \{1, \dots, m_\exists\}$ exists with $b_i^\exists < A_{i,\star}^\exists \tilde{x}^{(0)}$. Due to the monotonicity of variable k it is $A_{i,\star}^\exists \tilde{x}^{(0)} \leq A_{i,\star}^\exists \tilde{x}^{(1)}$ and hence also $minimax(v^{(1)}) = +\infty$. If, on the other hand, $A^\exists \tilde{x}^{(0)} \leq b^\exists$ it is $minimax(v^{(0)}) = c^\top \tilde{x}^{(0)} \leq c^\top \tilde{x}^{(1)} \leq minimax(v^{(1)})$. □

Theorem 3.1.6. *Given a binary QIP, its game tree $G = (V, E, e)$ and a positive monotone variable x_k, $k \in \mathcal{I}$. For any node $v \in V$ with $level(v) = k-1$ and its two successors $v^{(0)}$ and $v^{(1)}$ representing the assignments $x_k = 0$ and $x_k = 1$, respectively, $minimax(v^{(0)}) \leq minimax(v^{(1)})$ applies.*

Proof. Let there be an optimal winning strategy for the subtree below $v^{(1)}$. Due to Lemma 3.1.5 this strategy is also a winning strategy for the subtree below $v^{(0)}$ with all leaf values being less than or equal to the leaves of the strategy below $v^{(1)}$. Hence, $minimax(v^{(0)}) \leq minimax(v^{(1)})$. If no winning strategy below $v^{(1)}$ exists it is $minimax(v^{(0)}) \leq +\infty = minimax(v^{(1)})$. □

Obviously, similar conclusions can be drawn from negative monotone variables. Note that this monotonicity is a valid argument for both universal and existential variables: A positive monotone existential variable can be fixed to its lower bound in order to obtain an optimal solution. For a positive monotone universal variable its upper bound always yields the correct minimax value. In particular, it suffices to detect a winning existential substrategy for this universal variable assignment, because if there is one, the monotonicity ensures the existence of winning substrategy with smaller minimax value for the remaining universal variable assignments. We also refer to [HL19a] where we utilized robust runway scheduling instances as presented in Section 2.3.1 to show the impact of monotonicity in a small study.

3.1.3. Strategic Copy-Pruning

In contrast to such usage of prior knowledge we also can gather deep knowledge during the search process: found strategies in certain subtrees can be useful in order to assess the minimax

value of related subtrees rapidly. The idea is based upon the observation that typically in only a rather small part of the game tree a distinct and crafty strategy is required in order to ensure the fulfillment of the constraint system: in the right-hand side subtree of Figure 3.1 it suffices to find a fulfilling existential variable assignment for only one scenario (universal variable assignment) and reuse it in the other branches.

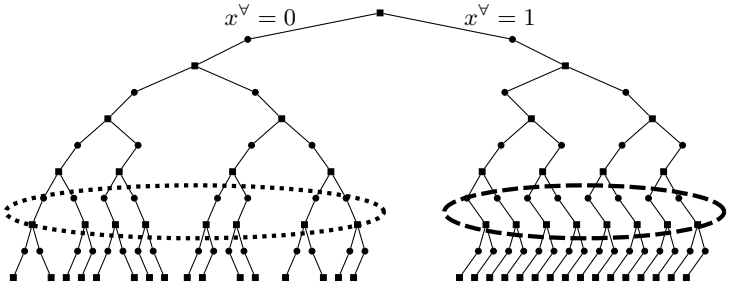

Figure 3.1.: The universal assignment $x^\forall = 1$ entails a simple winning strategy: regardless of future universal decisions existential variables can be set in a certain simple way, e.g. the existential decisions in the dashed ellipse are all the same. On the other hand, $x^\forall = 0$ calls for a more clever strategy: the existential decisions in the dotted ellipse differ depending on previous universal decisions.

An optimization task is often split up into two parts: finding the optimal solution itself and proving that no better solution can exist. For the latter, it turned out that applying backjumping techniques as utilized by QBF-solvers [Zha03, GNT03] and cutting planes as commonly used in integer programming [NW88] are also highly beneficial for QIPs in order to assess that no (better) strategy can exist in certain subtrees. For the first task, however, it seems that the exponential number of leaves belonging to a strategy must be traversed explicitly. This is certainly true in the worst case. However, as practical game trees are often structured irregularly, typically there are "difficult" parts of a game tree where a very deliberated substrategy must be found but also other parts where a less sophisticated substrategy suffices. We present a procedure, called *strategic copy-pruning* (SCP), that is capable of recognizing such subtrees making it possible to *implicitly* deduce the existence of a winning strategy therein. In contrast to similar ideas in QBF, as e.g. *counterexample guided abstraction refinement* [JK+16] and *solution directed backjumping* [GNT03] the fulfillment of linear constraints, rather than SAT clauses, must be ensured. Moreover, we consider an optimization process over a minimax objective rather than guaranteeing the mere satisfiability. For solving quantified constraint satisfaction problems a technique called *solution directed pruning* is used [GN+08], which has similarities to our proposed technique, but is also not applicable for an optimization problem. SCP draws its power not from memory-intensive learning, but from deep findings in the search tree. This perspective has led to remarkable achievements in the past [CHH99, vdHNL05, Sch13].

The effect of SCP is reinforced if the sequence of variable assignments predicted as optimal by minimax for both sides, called the principal variation [CM83], is traversed in an early stage of the tree search. Detecting and verifying this particular variable assignment is essential in order to obtain the objective value. Thus, having reasonable knowledge of which universal

variable assignments are particularly vicious can massively boost the search process. Several heuristics exist to analyze and find such promising moves in a game tree search environment [AN77, WvdW+04, Sch89, PS+96].

For game tree search there are already several algorithms trying to rapidly show the existence of winning strategies such as Kawano's simulation [Kaw96], sss* [Sto79], MTD(f) [PS+96] and (nega)scout [Rei83]. They, however, always have to traverse an exponential number of leaves. In our experiments, SCP often allows to conclude the existence of a winning strategy with a linear number of algebraic operations and in particular, in those cases it is not necessary to examine an exponential number of leaves resulting in a significant performance improvement both in time and number of solved instances (see page 153).

We first explain the basic idea of SCP using the example displayed in Figure 3.2. Consider

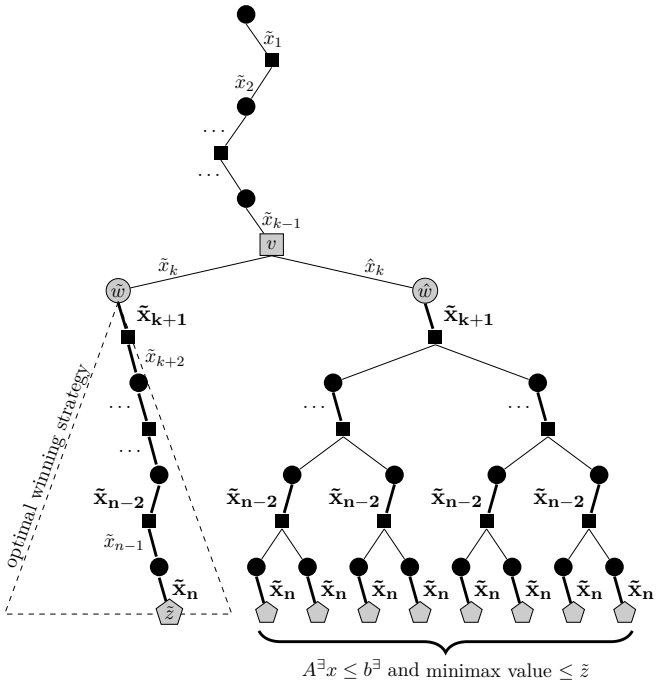

Figure 3.2.: Schematic display of SCP in a binary game tree.

a binary QIP and let the tree search have reached universal node $v \in V_\forall$ corresponding to the variable assignment $\tilde{x}_1, \ldots, \tilde{x}_{k-1}$. Keep in mind that this is a MAX node as the universal player is trying to maximize the objective. Assume the search first evaluates the left subtree below $\tilde{w} \in V_\exists$ corresponding to setting $x_k = \tilde{x}_k$ and assume the optimal solution is found for this subtree with principal variation $\tilde{x}_k, \ldots, \tilde{x}_n$. In particular, $minimax(\tilde{w}) = c^\top \tilde{x} = \tilde{z}$. The existential variable assignments of this principal variation are stored, as they seemed to have been the most effective in this subtree. Next up, in order to completely evaluate v, the existence of a winning strategy in the subtree below $\hat{w}$, corresponding to setting $x_k = \hat{x}_k = 1 - \tilde{x}_k$, must be ensured. Before searching the subtree explicitly we boldly attempt whether the strategy arising by adopting the existential assignments from the principal variation below $\tilde{w}$ is a winning strategy. If $A^\exists x \leq b^\exists$ for each leaf of this strategy, obviously a winning strategy for $\tilde{w}$ and thus for v is found. If further for each leaf the payoff is less than or equal to $\tilde{z}$ we even do not have to investigate this

subtree any further: a winning strategy has been found below $\hat{w}$ with value less than or equal to $\tilde{z}$. Hence, $minimax(\hat{w}) \leq \tilde{z}$, because there might exist even a better (existential) strategy. But since $minimax(v) = max(minimax(\tilde{w}), minimax(\hat{w}))$ no further search is required to validate $minimax(v) = \tilde{z}$.

The following Theorem 3.1.7 states the conditions necessary for a binary QIP such that this technique can be utilized. A similar result can be achieved for general QIPs, which we state in Theorem 3.1.8. But since our solver binarizes integer variables, only this following Theorem is practically used.

Theorem 3.1.7 (Strategic Copy-Pruning (SCP))**.**
Let $k \in \mathcal{I}_\forall$ and let $(\tilde{x}_1, \ldots, \tilde{x}_{k-1}) \in \{0,1\}^{k-1}$ be a fixed variable assignment of the variables $x_1, \ldots, x_{k-1}$. Let $v \in V_\forall$ be the corresponding node in the game tree. Let $\tilde{w} \in V$ and $\hat{w} \in V$ be the two children of v corresponding to the variable assignment $\tilde{x}_k$ and $\hat{x}_k = 1 - \tilde{x}_k$ of the universal variable x_k, respectively. Let there be an optimal winning strategy for the subtree below $\tilde{w}$ with minimax value $minimax(\tilde{w}) = \tilde{z}$ defined by the variable assignment $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_n) \in \{0,1\}^n$, i.e. $\tilde{z} = c^\top \tilde{x}$. If the minimax value of the copied strategy for the subtree below $\hat{w}$—obtained by adoption of future[7] existential variable assignments as in $\tilde{x}$—is not larger than $\tilde{z}$ and if this copied strategy constitutes a winning strategy, then $minimax(v) = \tilde{z}$. Formally: If both

$$c_k(\hat{x}_k - \tilde{x}_k) + \sum_{\substack{j \in \mathcal{I}_\forall: \\ j > k \wedge c_j \geq 0}} c_j(1 - \tilde{x}_j) - \sum_{\substack{j \in \mathcal{I}_\forall: \\ j > k \wedge c_j < 0}} c_j \tilde{x}_j \;\leq\; 0 \tag{3.7}$$

and

$$\sum_{\substack{j \in \mathcal{I}: \\ j \in \mathcal{I}_\exists \vee j < k}} A^\exists_{i,j} \tilde{x}_j + A^\exists_{i,k} \hat{x}_k + \sum_{\substack{j \in \mathcal{I}_\forall: \\ j > k \wedge A^\exists_{i,j} > 0}} A^\exists_{i,j} \leq b^\exists_i \qquad \forall i \in \{1, \ldots, m_\exists\} \tag{3.8}$$

then $minimax(v) = \tilde{z}$.

For clarification note that Condition (3.7) ensures that the change in the minimax value of the copied strategy, resulting from flipping x_k and using the worst-case assignment of the remaining future universal variables, is not positive, i.e. that its minimax value is still less than or equal to $\tilde{z}$. Condition (3.8) verifies that every constraint is satisfied in each leaf of the copied strategy by ensuring the fulfillment of each constraint in its specific worst-case scenario.

Proof. If (3.8) is satisfied there automatically exists a winning strategy for the subtree of v corresponding to $x_k = \hat{x}_k$ with root node $\hat{w}$, since for any future universal variable assignment the assignment of upcoming existential variables as in $\tilde{x}$ fulfills the constraint system. This is ensured, since for each constraint the worst-case setting of the future universal variables is

[7] *future* means variable blocks with index $\geq k$.

chosen. Hence, if (3.8) is fulfilled each leaf fulfills $A^{\exists}x \leq b^{\exists}$. Furthermore, the minimax value $\hat{z}$ of this strategy is less than or equal to $\tilde{z}$ due to Condition (3.7):

$$\hat{z} = \sum_{\substack{j\in\mathcal{I}:\\ j\in\mathcal{I}_\exists \vee j<k}} c_j\tilde{x}_j + c_k\hat{x}_k + \sum_{\substack{j\in\mathcal{I}_\forall:\\ j>k\wedge c_j\geq 0}} c_j \overset{(3.7)}{\leq} \sum_{\substack{j\in\mathcal{I}:\\ j\in\mathcal{I}_\exists \vee j<k}} c_j\tilde{x}_j + c_k\tilde{x}_k + \sum_{\substack{j\in\mathcal{I}_\forall:\\ j>k}} c_j\tilde{x}_j = \tilde{z}$$

Hence, if (3.7) and (3.8) are fulfilled there exists a winning strategy in the subtree below $\hat{w}$ with value less than or equal to $\tilde{z}$. Thus, the (still unknown) optimal solution for the subtree below $\hat{w}$ has a minimax value less than or equal to $\tilde{z}$, i.e. $minimax(\hat{w}) \leq \hat{z} \leq \tilde{z} = minimax(\tilde{w})$. Therefore, with Definition 2.1.15, $minimax(v) = minimax(\tilde{w}) = \tilde{z}$. □

Note that Condition (3.8) is trivially fulfilled for any constraint $i \in \{1,\dots,m_\exists\}$ with $A^{\exists}_{i,j} = 0$ for all $j \in \mathcal{I}_\forall, j \geq k$, i.e. constraints that are not influenced by future universal variables do not need to be examined. Hence, only a limited number of constraints need to be checked in case of a sparse matrix. Furthermore, Condition (3.7) is fulfilled if $c_j = 0$ for all $j \in \mathcal{I}_\forall$, $j \geq k$, i.e. if the future universal variables have no direct effect on the objective value. If even $c = 0$, i.e. it is a satisfiabilty problem rather than an optimization problem, Condition (3.7) holds trivially.

In Theorem 3.1.8 we present how SCP can come into effect for more general QIPs.

Theorem 3.1.8 (SCP for Integers)**.**
Let $k \in \mathcal{I}_\forall$ and let $(\tilde{x}_1,\dots,\tilde{x}_{k-1}) \in \mathcal{L}_1 \times\cdots\times \mathcal{L}_{k-1}$ be a fixed variable assignment of the variables $x_1,\dots,x_{k-1}$. Let $v \in V_\forall$ be the corresponding node in the game tree. Let $\tilde{w} \in V$ and $\hat{w} \in V$ be two children of v corresponding to the variable assignment $\tilde{x}_k \in \mathcal{L}_k$ and $\hat{x}_k \in \mathcal{L}_k$, $\hat{x}_k \neq \tilde{x}_k$, of the universal variable x_k, respectively. Let there be an optimal winning strategy for the subtree below $\tilde{w}$ with $minimax(\tilde{w}) = \tilde{z}$ defined by the variable assignment $\tilde{x} = (\tilde{x}_1,\dots,\tilde{x}_n) \in \mathcal{L}$, i.e. $\tilde{z} = c^\top\tilde{x}$. If both

$$c_k(\hat{x}_k - \tilde{x}_k) + \sum_{\substack{j\in\mathcal{I}_\forall:\\ j>k\wedge c_j\geq 0}} c_j(u_j - \tilde{x}_j) + \sum_{\substack{j\in\mathcal{I}_\forall:\\ j>k\wedge c_j<0}} c_j(l_j - \tilde{x}_j) \leq 0 \tag{3.9}$$

and

$$\sum_{\substack{j\in\mathcal{I}:\\ j\in\mathcal{I}_\exists \vee j<k}} A^{\exists}_{i,j}\tilde{x}_j + A^{\exists}_{i,k}\hat{x}_k + \sum_{\substack{j\in\mathcal{I}_\forall:\\ j>k\wedge A^{\exists}_{i,j}\geq 0}} A^{\exists}_{i,j}u_j + \sum_{\substack{j\in\mathcal{I}_\forall:\\ j>k\wedge A^{\exists}_{i,j}<0}} A^{\exists}_{i,j}l_j \leq b^{\exists}_i \qquad \forall i \in \{1,\dots,m_\exists\} \tag{3.10}$$

then $minimax(v) \geq minimax(\tilde{w}) \geq minimax(\hat{w})$ and hence the subtree of v corresponding to assigning $x_k = \hat{x}_k$ does not need to be investigated.

The proof of Theorem 3.1.8 is similar to the proof of Theorem 3.1.7. Note that in the situation described in Theorem 3.1.8 one is not able to determine the minimax value of v, but it can be ensured that a certain subtree (the one below $\hat{w}$) does not yield the principal variation and hence does not need to be searched explicitly, as the existence of a winning strategy is guaranteed.

SCP Implementation Details When implementing the theoretical result from Theorem 3.1.7 the most crucial part is not the checking of Conditions (3.7) and (3.8), but ensuring the optimality of $\tilde{z}$ for $minimax(\tilde{w})$ demanded in the theorem. Consider Figure 3.3 representing the final four variables of a binary QIP with strictly alternating quantifiers. We assume the search

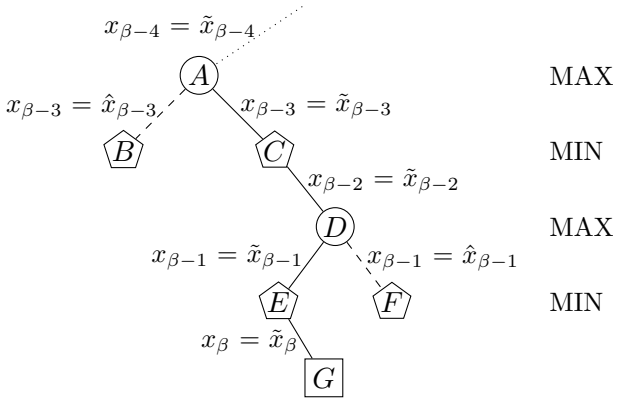

Figure 3.3.: Illustrative game tree: pentagonal existential decision nodes, circular universal decision nodes and rectangluar leaves. The dashed lines indicate that those underlying subtrees might be omitted if Theorem 3.1.7 applies.

has found the variable assignment $x = \tilde{x}$ (represented by node G) with $A^{\exists}\tilde{x} \leq b^{\exists}$. Furthermore, assume $x_\beta = \tilde{x}_\beta$ is the optimal assignment for the final variable block with regard to $x_1 = \tilde{x}_1, \ldots, x_{\beta-1} = \tilde{x}_{\beta-1}$, i.e. $minimax(E) = minimax(G)$. If the requirements of Theorem 3.1.7 for $k = \beta - 1$ are fulfilled it is $minimax(D) = minimax(E)$ and we do not have to calculate $minimax(F)$ explicitly as the existence of a winning strategy below F is ensured. If this attempt is successful the application of Theorem 3.1.7 at node A would be attractive. However, one must ensure, that $minimax(C) = minimax(D)$, i.e. that setting $x_{\beta-2} = \tilde{x}_{\beta-2}$ is indeed optimal in this stage. If this optimality cannot be guaranteed, but Conditions (3.7) and (3.8) are fulfilled at node A, we still can conclude the existence of a winning strategy for the subtree at B but we cannot yet specify $minimax(A)$. However, storing the information $minimax(B) \leq \hat{z}$ and $minimax(A) \leq \tilde{z}$ can be advantageous. In Algorithm 1 such a node A is marked as "potentially finished", as $minimax(C) = minimax(D)$ might not yet be ensured. If it turns out, that indeed setting $x_{\beta-2} = 1 - \tilde{x}_{\beta-2}$ is optimal at C such a marking is deleted.

As soon as a leaf v is found during the tree search with the corresponding variable assignment x_v being a potentially new PV for this subtree the mechanism described in Algorithm 1 is invoked: the two Conditions (3.7) and (3.8) of Theorem 3.1.7 are checked at each universal node starting from this leaf towards the root (Line 5). While both conditions are fulfilled the corresponding universal nodes are marked as potentially finished. If one of the conditions is not satisfied the remaining universal nodes above are marked as unfinished. If a level is closed during the tree search and the above universal node is marked as potentially finished this level also can be closed immediately as a strategy is guaranteed in the other branch with worse objective value (from the universal player's point of view). The unmarking of universal nodes (Line 9) is

Algorithm 1: Marking of potentially finished universal nodes.

Input: leaf node v

1: checkSCP=`TRUE`
2: **repeat**
3: v=parent(v)
4: **if** $v \in V_\forall$ **then**
5: **if** checkSCP **and** v fulfills Conditions (3.7) and (3.8) **then**
6: mark v as potentially finished
7: **else**
8: checkSCP=`FALSE`
9: mark v as unfinished
10: **end if**
11: **end if**
12: **until** v is root node

necessary since Theorem 3.1.7 demands x_v to be the actual PV and hence previous markings were made based on a false assumption.

Proposition 3.1.9. *Consider a QIP and let $v \in V_\forall$ be a node with level$(v) = k-1$, i.e. v represents the decision on universal variable x_k. In order to evaluate line 5 in Algorithm 1 for node v $\mathcal{O}(m_k \cdot n)$ operations are sufficient, with $m_k = |\{i \in \{1,\dots,m_\exists\} \mid A^\exists_{i,k} \neq 0\}|$ being the number of constraints in which variable k is present.*

Proof. The plain computation of the left-hand side of Condition (3.7) requires $\mathcal{O}(n)$ operations. However, the two sums in Condition (3.7) can be updated and stored for each loop and hence it suffices to carry out $\mathcal{O}(1)$ operations: check the increase in the objective if $x_k = \hat{x}_k$ by computing $c_k(\hat{x}_k - \tilde{x}_k)$ and adding the increase from choosing the worst-case setting of future universal variables computed in the previous loops. If the resulting increase is non-positive Condition (3.7) is fulfilled and we can update these sums by incorporating x_k accordingly. For Condition (3.8) to be fulfilled the fulfillment of each constraint in the worst-case setting of future universal variables must be ensured. In the setting of Algorithm 1 it suffices to check only constraints in which variable x_k is present, i.e. those constraints $i \in \{1,\dots,m_\exists\}$ with $A^\exists_{i,k} \neq 0$: Constraints with $A^\exists_{i,j} = 0$ for each $j \in \mathcal{I}_\forall$, $j \geq k$, are trivially fulfilled, since $A^\exists \tilde{x} \leq b^\exists$. Constraints with $A^\exists_{i,k} = 0$ and $A^\exists_{i,j} \neq 0$ for some future universal variable $j \in \mathcal{I}_\forall$, $j \geq k$, are fulfilled as they have been checked in a previous loop. Therefore, $\mathcal{O}(m_k \cdot n)$ operations are required in each loop. □

Obviously, in the worst case in each iteration $m_\exists \cdot n$ operations must be performed. In our experiments, however, where each of the universal variables occur in only a few rows and the matrix is sparse, the runtime of the heuristic is negligible. When exploiting Theorem 3.1.7 via Algorithm 1 it is especially advantageous if the search first investigates the PV as Condition (3.7) is more likely to be fulfilled. Hence, its applicability highly depends on the implemented diving and sorting heuristic.

Example 3.1.10. *Consider the following binary QIP (The min/max alternation in the objective is omitted):*

$$\begin{array}{llllll} \min & 2x_1 & +3x_2 & -2x_3 & -2x_4 & +x_5 \\ \text{s.t.} & \exists\, x_1 & \forall\, x_2 & \exists\, x_3 & \forall\, x_4 & \exists\, x_5 \end{array} \in \{0,1\}^5 :$$

$$\begin{pmatrix} 1 & -1 & 1 & 3 & -1 \\ 3 & 2 & 3 & 1 & -2 \end{pmatrix} x \leq \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

Starting at the root node of the corresponding game tree in Figure 3.4, we can immediately omit the subtree corresponding to $x_1 = 1$, as x_1 is positive monotone. Keep in mind that the result of Theorem 3.1.7 is particularly beneficial if the search process of a QIP solver first examines the principal variation, i.e. the variable assignment defining the actual minimax value. Assume the

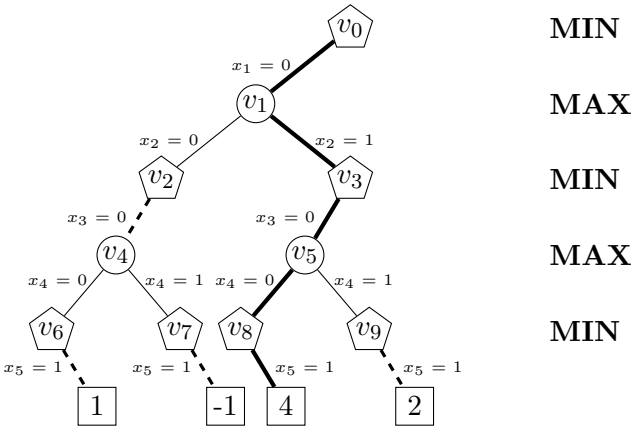

Figure 3.4.: Optimal winning strategy for the stated QIP. Circular nodes are existential decision nodes, rectangular nodes are universal decision nodes and pentagonal nodes are leaves. The values given in the leaves constitute the objective value corresponding to the variable assignment along the path from the root to this leaf. The dashed lines indicate that those existential decisions were simply copied from the path drawn thicker.

search process follows the path drawn thick in Figure 3.4 to node v_8, i.e. the path corresponding to the variable assignment $x_1 = 0$, $x_2 = 1$, $x_3 = 0$ and $x_4 = 0$. Setting $x_5 = 1$ is optimal in this case, as $x_5 = 0$ would violate the second constraint. Hence, the minimax value of v_8 is 4. On the way up in the search tree we then want to determine $minimax(v_5)$. As (3.7) and (3.8) are fulfilled for $k = 4$, $\tilde{z} = 4$ and $\tilde{x} = (0, 1, 0, 0, 1)$ we know that $minimax(v_5) = 4$. That means we have (easily) verified a winning strategy starting from v_9 with minimax value less than or equal to 4. In node v_3 setting $x_3 = 1$ is obviously to the detriment of the existential player, as the second constraint would become unfulfillable. Hence, $minimax(v_3) = minimax(v_5) = 4$. In node v_1 we once again try to apply Theorem 3.1.7 by copying the existential decisions of x_3 and x_5 in the thick path to the not yet investigated subtree associated with $x_2 = 0$. As (3.7) and (3.8) are fulfilled for $k = 2$, $\tilde{z} = 4$ and $\tilde{x} = (0, 1, 0, 0, 1)$ this attempt is successful and $minimax(v_1) = 4$. Note that by applying Theorem 3.1.7 the minimax value of the subtrees below v_2 and v_9 are not known exactly: in particular we only obtain $minimax(v_2) \leq \hat{z} = 1$, whereas a better strategy exists resulting in $minimax(v_2) = 0$ (Setting $x_5 = 0$ in node v_6).

Hence, by finding the principal variation first (thick path), exploiting monotonicity of x_1 at node v_0, Theorem 3.1.7 at node v_1 and v_5 and some further reasoning from linear programming at node v_3 and v_8 the minimax value at the root node v_0 was found to be 4 with optimal first-stage solution $x_1 = 0$.

In [HL19a] we showed that utilizing SCP in our solver resulted in a massive boost in both the number of solved instances and the runtime (about 4 times faster) on robust runway scheduling instances as presented in Section 2.3.1. In Section 7.3 we provide further evidence on the positive impact of SCP on several test sets and the results also indicate that the time for checking the Conditions (3.7) and (3.8) is indeed negligible.

3.1.4. Dominance Relations

Exploiting dominance relations is a common procedure in mixed integer programming [GK+15]. However, as shown below, one must be cautious when applying such techniques to QIPs. We first present the definition of a dominance relation in a QIP, which is similar to the one used in mixed integer programming. Then, we present an example how exploiting this property in a similar fashion as used for MIPs leads to wrong results for QIPs.

Definition 3.1.11 (Dominance Relation)**.**
Given a binary QIP and two variables x_j and x_k. We say x_j dominates *x_k ($x_j \succ x_k$), if*

a) $c_j \leq c_k$, and

b) $A^{\exists}_{i,j} \leq A^{\exists}_{i,k}$ for each constraint $i \in \{1, \dots, m_{\exists}\}$.

Example 3.1.12. *Consider the following QIP*

$$\min \; x_1 + x_2 + x_3$$

$$\text{s.t. } \exists x_1 \in \{0,1\} \, \forall x_2 \in \{0,1\} \, \exists x_3 \in \{0,1\} :$$

$$\begin{pmatrix} -3 & 2 & -2 \\ 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

x_1 dominates x_3, i.e. $x_1 \succ x_3$. Hence, one might expect (as it is valid for MIPs) that $x_1 = 0$ and $x_3 = 1$ cannot be part of an optimal winning strategy, i.e. no branch of the optimal solution contains the edges representing $x_1 = 0$ and $x_3 = 1$. However, in the above example the principal variation itself, given by $\tilde{x} = (0, 1, 1)$, contradicts this assumption: x_1 must be set to 0 as otherwise an immediate threat of a violation of the second constraint arises. Then the universal player chooses $x_2 = 1$ in order to maximize the objective by also forcing $x_3 = 1$ due to the first constraint. Hence, adding the constraint $x_3 \leq x_1$ to the constraint system, which is the usual procedure for binary variables in a MIP environment (or using this information implicitly in a conflict graph [GK+15]), would make this instance even infeasible. Now consider the DEP of the above QIP:

$$\min k$$
$$\text{s.t.} \quad \begin{pmatrix} -1 & 1 & 1 & 0 \\ -1 & 1 & 0 & 1 \\ 0 & -3 & -2 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -3 & 0 & -2 \\ 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} k \\ x_1 \\ x_3^{(0)} \\ x_3^{(1)} \end{pmatrix} \leq \begin{pmatrix} 0 \\ -1 \\ 0 \\ 0 \\ -2 \\ 2 \end{pmatrix}$$
$$x_1,\ x_3^{(0)},\ x_3^{(1)} \in \{0,1\}$$

For this MIP obviously $x_1 \not\succ x_3^{(0)}$ *and* $x_1 \not\succ x_3^{(1)}$. *This example demonstrates that different conclusions have to be drawn from dominance relations than we are used to for MIP.*

By using the insights obtained in the above example it becomes apparent that identifying dominances of existential variables in different variable blocks is not advantageous. However, the dominance relation of existential variables in the same variable block still is useful in order to easily avoid unnecessary computing time:

Proposition 3.1.13. *Consider a feasible binary QIP and* $i, j \in B_k$ *for some* $k \in \mathcal{E}$, *i.e.* i *and* j *are existential variables in the same block. Let* $x_i \succ x_j$. *Then there always exists an optimal winning strategy* $\mathcal{S} = (V', E', e')$ *with* $(x_v)_i \geq (x_v)_j$ *for all leaf nodes* $v \in V' \cap V_L$.

Proof. Let $\tilde{\mathcal{S}} = (\tilde{V}', \tilde{E}', \tilde{e}')$ be an optimal winning strategy with some leaf $\tilde{v} \in \tilde{V}' \cap V_L$ with $(x_{\tilde{v}})_i < (x_{\tilde{v}})_j$, i.e. $(x_{\tilde{v}})_i = 0$ and $(x_{\tilde{v}})_j = 1$. Let $t = \min\{i, j\}$ and $v_t \in \tilde{V}'$ be the node at level $t-1$ on the path from the root to $\tilde{v}$ in $\mathcal{S}$. For any leaf $\bar{v} \in \tilde{V}' \cap V_L$ in the subtree below v_t it is $(x_{\bar{v}})_i < (x_{\bar{v}})_j$ as only a single node for each level $i-1$ and $j-1$ is present in this substrategy (as both variables belong to the same variable block). Consider the strategy $\mathcal{S}^\star$ obtained by adapting the strategy $\tilde{\mathcal{S}}$ by flipping the two existential decisions below v_t corresponding to variable i and j. As $x_i \succ x_j$ for any leaf in $\mathcal{S}^\star$ below v_t the system $A^\exists x \leq b^\exists$ is also fulfilled with objective value not larger than at their counterpart in $\tilde{\mathcal{S}}$. Therefore, $\mathcal{S}^\star$ is a winning strategy with value not larger than the value of $\tilde{\mathcal{S}}$. Thus, either $\tilde{\mathcal{S}}$ is not optimal or $\mathcal{S}^\star$ is also optimal. □

If a dominance as described in Proposition 3.1.13 is identified, the information that the combination $x_i = 0$ and $x_j = 1$ never has to be investigated, can be exploited. Note that the reason why dominance relations of variables in different blocks cannot be exploited as in Proposition 3.1.13 is that it cannot be guaranteed (in general) that for the node v_t all leaves in its subtree have the demanded property (cf. Example 3.1.12).

A similar result can be obtained for dominance relations of universal variables in the same block. In preparation for this we first prove following lemma.

Lemma 3.1.14. *Let* $P = (A^\exists, b^\exists, c, \mathcal{L}, Q)$ *and* $\tilde{P} = (A^\exists, \tilde{b}^\exists, c, \mathcal{L}, Q)$ *be two QIPs with* $\tilde{b}^\exists \leq b^\exists$. *Then for the optimal values* z *of* P *and* $\tilde{z}$ *of* $\tilde{P}$ *it is* $z \leq \tilde{z}$.

Proof. Consider the game trees $G = (V, E, e)$ and $\tilde{G} = (\tilde{V}, \tilde{E}, \tilde{e})$ of P and $\tilde{P}$, respectively. For any leaf $v \in V_L \subset V$ there exists a leaf $\tilde{v} \in \tilde{V}_L \subset \tilde{V}$ with $x_v = x_{\tilde{v}}$ and $minimax(v) \leq minimax(\tilde{v})$

with Definition 2.1.15. Therefore, any strategy $\tilde{\mathcal{S}}$ in $\tilde{G}$ can be adopted to be a strategy $\mathcal{S}$ in G with $minimax(\mathcal{S}) \leq minimax(\tilde{\mathcal{S}})$. □

Proposition 3.1.15. *Consider a feasible binary QIP and its game tree $G = (V, E, e)$. Let $i, j \in B_k$ for some $k \in \mathcal{A}$, i.e. i and j are universal variables in the same block k. Let $x_i \succ x_j$. Then there exists an optimal winning strategy $\mathcal{S} = (V', E', e')$ for which the PV is defined by at least one path from the root to a leaf $v \in V' \cap V_L$ with $(x_v)_i \leq (x_v)_j$.*

Proof. Let $j = i + 1$ (Resorting within blocks does not change the nature of the problem). Consider any node $v \in V$ with $\text{level}(v) = i - 1$ representing the universal variable x_i. Let $w \in V$ be the node reached by setting $x_i = 0$ and $x_j = 1$ and $\tilde{w}$ be the node reached by setting $x_i = 1$ and $x_j = 0$ starting from v, with x_w and $x_{\tilde{w}}$ being the corresponding partial variable assignments. Let $c_w = \sum_{k=1}^{j} c_k (x_w)_k$ and $c_{\tilde{w}} = \sum_{k=1}^{j} c_k (x_{\tilde{w}})_k$. Due to $x_i \succ x_j$ it is $c_{\tilde{w}} \leq c_w$. Both w and $\tilde{w}$ represent QIPs $(A', b', c', \mathcal{L}', Q')$ and $(A', \tilde{b}', c', \mathcal{L}', Q')$ with offset c_w and $c_{\tilde{w}}$, respectively. Then $b' \leq \tilde{b}'$ applies due to $x_i \succ x_j$. Hence, with Lemma 3.1.14 it is $minimax(\tilde{w}) \leq minimax(w)$. □

Surprisingly, we are also able to exploit dominance relations between universal and existential variables: An existing dominance relation $x_i \succ x_j$ states that $x_i = 1$ and $x_j = 0$ is better for the existential player than $x_i = 0$ and $x_j = 1$ (if all other variables stay the same). Assume x_j is a universal variable in a variable block before the existential variable x_i. Assume after setting $x_j = 1$ the variable x_i is forced to be set $x_i = 0$ but an optimal winning strategy exists. Then setting $x_j = 0$ is to the detriment of the universal player, as the strategy from the branch $x_j = 1$ can be adapted by setting $x_i = 0$, which results in a winning strategy with better objective value (for the existential player).

Proposition 3.1.16. *Consider a binary QIP and let $j \in \mathcal{I}_\forall$. Let $\tilde{v} \in V_\forall$ be the node in the game tree representing a fixed variable sequence $\tilde{x}_1, \ldots, \tilde{x}_{j-1}$. Let $i \in \mathcal{I}_\exists$, $i > j$, with $x_i \succ x_j$. Let there be an optimal winning strategy $\mathcal{S} = (V', E', e')$ with $minimax(\mathcal{S}) = z$ for the subtree of $\tilde{v}$ corresponding to $x_j = 1$ and let $(x_v)_i = 0$ in every leaf $v \in V' \cap V_L$. Then $minimax(\tilde{v}) = z$.*

Proof. We construct a winning strategy $\tilde{\mathcal{S}} = (\tilde{V}', \tilde{E}', \tilde{e}')$ for the subtree of $\tilde{v}$ corresponding to $x_j = 0$ with $minimax(\tilde{\mathcal{S}}) \leq z$. For each leaf $w \in V' \cap V_L$ we construct a leaf node $\tilde{w} \in \tilde{V}'$ defined by the variable assignment $x_{\tilde{w}}$ given by

$$(x_{\tilde{w}})_k = \begin{cases} (x_w)_k & , \ k \in \mathcal{I} \setminus \{i, j\} \\ 1 - (x_w)_k & , \ k \in \{i, j\} \, . \end{cases}$$

Since $x_i \succ x_j$ for each leaf pair $(w,\tilde{w})$ it is

$$A^\exists x_{\tilde{w}} \leq A^\exists x_w \leq b^\exists \tag{3.11}$$

and

$$minimax(w) \geq minimax(\tilde{w}) \, . \tag{3.12}$$

The strategy $\tilde{S}$ consists of all such leaves $\tilde{w}$ and every path from its root to such a leaf. Due to Equation (3.11) $\tilde{\mathcal{S}}$ constitutes a winning strategy and due to Equation (3.12) this strategy has a minimax value not larger than z. Hence, the *optimal* winning strategy for the subtree corresponding to $x_j = 0$ has a minimax value not larger than z. Since $minimax(v) = max(minimax(\mathcal{S}), minimax(\tilde{\mathcal{S}}))$ it is $minimax(v) = z$. □

As it is often too memory-intensive to store the entire strategy in order to check the requirements of Proposition 3.1.16 it is hardly usable, in general. However, if the considered existential variable x_i is an element of the subsequent variable block of the universal variable x_j and if x_j is the last variable within its block (by rearrangement of the variables within this block) only one path must be considered: the PV of the optimal winning strategy of the subtree corresponding to $x_j = 1$.

3.2. Relaxations

In an alpha-beta search as well as in a branch-and-bound setting assessing the potential of a subtree is crucial for the search process. This can be done by relaxing some problem conditions in order to obtain a bound on the optimal value of a (sub)problem. In mixed integer linear programming, variants of the LP-relaxation of a problem are employed [Bal01]. In this section we briefly discuss relaxations applicable for QIPs. We first revisit relaxations discussed in [Wol15] and then present extended LP-relaxations with embedded scenarios.

A straightforward way to relax a QIP is to enlarge the variable domain by relaxing the integrality resulting in a QLP.

Definition 3.2.1 (QLP-Relaxation of a QIP)**.**
Let $P = (A^\exists, b^\exists, c, \mathcal{L}, Q)$ be a QIP (with integer bounds). By omitting the requirement that each variable must take integer values the relaxed variable domain is given by

$$\mathcal{L}_{relax} = \{y \in \mathbb{Q}^n \mid \forall i \in \mathcal{I} : l_i \leq y_i \leq u_i\} .$$

The QLP-relaxation *of P is given by $(A^\exists, b^\exists, c, \mathcal{L}_{relax}, Q)$.*

For the QLP-relaxation the following was shown in [Wol15]:

Proposition 3.2.2. *Given a QIP P and its QLP-relaxation R. Then the following holds:*

a) If R is infeasible, then also P is infeasible.

b) If R is feasible with optimal value z_R, then either P is infeasible or P is feasible with optimal value $z_P \geq z_R$, i.e. z_R constituted a lower bound.

In addition to relaxing constraints or the integrality of variables, the quantification sequence can be altered by changing the order of the variables or the quantification of variables. An LP-relaxation of a QIP can be built by additionally dropping universal quantification, i.e. each variable is considered to be an existential variable with continuous domain. If the universal quantification but not the integrity is relaxed, the IP-relaxation arises.

Definition 3.2.3 (LP- and IP-Relaxation of a QIP)**.**
Let $P = (A^\exists, b^\exists, c, \mathcal{L}, Q)$ be a QIP and let $\mathcal{L}_{relax}$ be given as in Definition 3.2.1.

a) The IP-relaxation *of P is given by $(A^\exists, b^\exists, c, \mathcal{L}, \exists^n)$.*

b) The LP-relaxation *of P is given by $(A^\exists, b^\exists, c, \mathcal{L}_{relax}, \exists^n)$.*

The relaxation of quantification can be viewed as taking control of the opponents variables, i.e. such a relaxation constitutes a single-player game. It was shown in [Wol15] that Proposition 3.2.2 is also valid if R is the LP-relaxation. Furthermore, Proposition 3.2.2 is valid if R is the IP-relaxation as the same gaming arguments as used in [Wol15] for the LP-relaxation apply.

A useful relaxation must be well balanced with regard to the quality of the resulting bound and its computing time. The IP-relaxation often provides a better bound than the LP-relaxation but with increased theoretical computational complexity. The same holds for the QLP-relaxation in comparison with the LP-relaxation. Despite the better runtime of the LP-relaxation compared to the QLP-relaxation, one major drawback of the LP-relaxation is that it totally neglects the uncertainty arising from universal variables: transferring the responsibility of universal variables to the existential player and solving the single-player game has nothing to do with the worst-case outcome in most cases. This, however, can easily be improved by arbitrarily fixing the universal variables. This can be interpreted as knowing the (deterministic) opponent moves beforehand and adapting one's own moves for this special game. As the original QIP optimizes the worst case, arbitrarily fixed universal variables also must be dealt with.

Definition 3.2.4 (LP-Relaxation with Fixed Scenario)**.**
Let $P = (A^\exists, b^\exists, c, \mathcal{L}, Q)$ be a QIP and let $\hat{x}_\forall \in \mathcal{L}_\forall$ be a fixed scenario. The LP

$$\min \left\{ c^\top x \mid x \in \mathcal{L}_{relax} \wedge x_\forall = \hat{x}_\forall \wedge A^\exists x \leq b^\exists \right\}$$

is called the LP-relaxation with fixed scenario $\hat{x}_\forall$.

Similarly, the IP-relaxation with fixed scenario can be defined, the interpretation of which is that the universal player discloses her (future) moves right at the beginning and the existential player can adjust his moves accordingly. The following $\forall\exists$-S-relaxation can be interpreted as repeatedly solving the IP-relaxation for each scenario in the scenario set S in order to improve the resulting bound.

Definition 3.2.5 ($\forall\exists$-S-Relaxation)**.**
Let $P = (A^\exists, b^\exists, c, \mathcal{L}, Q)$ be a QIP. Let $S \subseteq \mathcal{L}_\forall$. The QIP

$$\max_{x_\forall \in S} \left(c_\forall x_\forall + \min_{x_\exists \in \mathcal{L}_\exists} c_\exists x_\exists \right) \qquad \text{s.t. } \forall\, x_\forall \in S\ \exists\, x_\exists \in \mathcal{L}_\exists :\ A^\exists x \leq b^\exists$$

is called its $\forall\exists$-S-relaxation.

For $S = \mathcal{L}_\forall$ the $\forall\exists$-S-relaxation is simliar to the $\forall\exists$-relaxation as presented in [Wol15] with the slight difference that we deal with QIPs and they examine QLPs. Nevertheless, very similar results can be obtained for such a relaxation:

Proposition 3.2.6. *Let $P = (A^{\exists}, b^{\exists}, c, \mathcal{L}, Q)$ be a QIP. Let $S \subseteq \mathcal{L}_\forall$ and let R be the corresponding $\forall\exists$-S-relaxation. Then the following holds:*

a) If R is infeasible, then also P is infeasible.

b) If R is feasible with optimal value z_R, then either P is infeasible or P is feasible with optimal value $z_P \geq z_R$, i.e. z_R constituted a lower bound.

Proof.

a) If Q is infeasible there exists some scenario $\hat{x}_\forall \in S \subseteq \mathcal{L}_\forall$ such that

$$\nexists x_\exists \in \mathcal{L}_\exists : \; A^{\exists}_{\exists} x_\exists \leq b^{\exists} - A^{\exists}_{\forall} \hat{x}_\forall \,,$$

and since $\hat{x}_\forall \in \mathcal{L}_\forall$ there cannot exists a winning strategy for P. As a gaming argument we can interpret this the following way: If there is some move sequence of the opponent we cannot react to in a victorious way—even if we know the sequence beforehand—the game is lost for sure.

b) Let $\hat{x}_\forall \in S$ be the scenario defining $z_R = c^\top \hat{x}$, i.e. the worst-case scenario for R, and let $\hat{x}_\exists$ be the corresponding fixation of the existential variables. It is

$$\hat{x}_\exists = \operatorname*{arg\,min}_{x_\exists \in \mathcal{L}_\exists} \left\{ c_\exists^\top x_\exists \mid A^{\exists}_{\exists} x_\exists \leq b^{\exists} - A^{\exists}_{\forall} \hat{x}_\forall \right\} . \tag{3.13}$$

If P is feasible, scenario $\hat{x}_\forall$ must also be present in the corresponding winning strategy. Let $\tilde{x}$ be the corresponding game, i.e. $\tilde{x}_\forall = \hat{x}_\forall$. With Equation (3.13) obviously $z_R = c^\top \hat{x} \leq c^\top \tilde{x}$ and thus with Theorem 2.1.16 $z_R \leq z_P$. □

Proposition 3.2.6 also proves the usability of the IP- and LP-relaxation with fixed scenario, as for $|S| = 1$ the $\forall\exists$-S-relaxation is an IP with fixed universal variables, which can be relaxed—by dropping the variable integrality—yielding an LP-relaxation with fixed scenario.

A $\forall\exists$-S-relaxation can be solved by repeatedly solving the LP-relaxations for each scenario in S and selecting the worst outcome. Although now several scenarios can be considered, which to some extent takes the uncertainty arising from universal variables into account, some existential variables must be fixed for more than one scenario: the $\forall\exists$-S-relaxation allows fixations of the existential variables exactly matched for each single scenario (see Equation (3.13)). This is due to the altered order of the variables. Reintroducing the original order of the variables while taking the same subset of scenarios $S \in \mathcal{L}_\forall$ into account, can improve the resulting bound.

Definition 3.2.7 (S-Relaxation)**.**
Let $P = (A^{\exists}, b^{\exists}, c, \mathcal{L}, Q)$ be a QIP. Let $S \subseteq \mathcal{L}_\forall$ and let $\mathcal{L}_S = \{x \in \mathcal{L} \mid x_\forall \in S\}$. We call

$$z = \min_{x^{(1)} \in \mathcal{L}_S^{(1)}} \left(c^{(1)} x^{(1)} + \max_{x^{(2)} \in \mathcal{L}_S^{(2)}} \left(c^{(2)} x^{(2)} + \min_{x^{(3)} \in \mathcal{L}_S^{(3)}} \left(c^{(3)} x^{(3)} + \ldots \min_{x^{(\beta)} \in \mathcal{L}_S^{(\beta)}} c^{(\beta)} x^{(\beta)} \right) \right) \right)$$

$$\text{s.t. } Q \circ x \in \mathcal{L}_S : \; A^{\exists} x \leq b^{\exists} \tag{$\star$}$$

the S-relaxation of P.

Note that this S-relaxation is not a standard QIP, as the individual universal variable domains are not given by simple bounds. In particular, the local domain of a universal variable block $i \in \mathcal{A}$ depends on previous variable assignments $\tilde{x}^{(1)}, \dots, \tilde{x}^{(i-1)}$ and is given by

$$\mathcal{L}_S^{(i)}(\tilde{x}^{(1)}, \dots, \tilde{x}^{(i-1)}) = \{\hat{x}^{(i)} \in \mathcal{L}^{(i)} \mid \exists x = (\tilde{x}^{(1)}, \dots, \tilde{x}^{(i-1)}, \hat{x}^{(i)}, x^{(i+1)}, \dots, x^{(\beta)}) \in \mathcal{L}_S\}\,.$$

Nevertheless, an S-relaxation can be solved by building a deterministic equivalent program.

Example 3.2.8. *Consider the following binary QIP (The min/max alternation in the objective is omitted for clarity):*

$$\begin{array}{llllll} \min & -2x_1 & +x_2 & -x_3 & -x_4 & \\ \text{s.t.} & \exists\, x_1 & \forall\, x_2 & \exists\, x_3 & \forall\, x_4 & \in \{0,1\}^4 : \\ & x_1 & +x_2 & +x_3 & +x_4 & \leq 3 \\ & & -x_2 & -x_3 & +x_4 & \leq 0 \end{array}$$

The optimal first-stage solution is $\tilde{x}_1 = 1$, the PV is $(1,1,0,0)$ and hence the optimal value is -1. Let $S = (\{1,0\},\{1,1\})$ be a set of scenarios. The $\forall\exists$-S-relaxation can be solved by solving the two emerging IPs as shown in Table 3.1 when fixing the universal variables as in S and selecting the larger objective value of both instances. This $\forall\exists$-S-relaxation yields a lower bound

Table 3.1.: Solutions of the single IPs with fixed scenarios.

scenario	$x_2 = 1$, $x_4 = 0$	$x_2 = 1$, $x_4 = 1$
IP	$\begin{array}{llll} \min & -2x_1 & -x_3 & +1 \\ \text{s.t.} & x_1 & +x_3 & \leq 2 \\ & & -x_3 & \leq 1 \end{array}$	$\begin{array}{llll} \min & -2x_1 & -x_3 & +0 \\ \text{s.t.} & x_1 & +x_3 & \leq 1 \\ & & -x_3 & \leq 0 \end{array}$
solution	$x_1 = 1$, $x_3 = 1$	$x_1 = 1$, $x_3 = 0$
objective	-2	-2

of -2. The first-stage variable is set to 1 in both scenarios but note that even though in both scenarios $x_2 = 1$ the solutions of the IPs differ in x_3. This is due to the prior knowledge of the fixation of x_4. This obviously does not reflect the original QIP well, as x_3 should be set to exactly one value in the subtree corresponding to $x_1 = 1$ and $x_2 = 1$. Now consider the DEP of the S-relaxation in which $x_{3(\tilde{x}_2)}$ represents the assignment of x_3 after x_2 is set to $\tilde{x}_2$:

$$\begin{array}{llllll} \min & k & & & & \\ \text{s.t.} & -2x_1 & -x_{3(1)} & +1 & \leq k & \left.\begin{array}{l}\\\\\end{array}\right\} \textit{Scenario } (1,0) \\ & x_1 & +x_{3(1)} & & \leq 2 & \\ & & -x_{3(1)} & & \leq 1 & \\ & -2x_1 & -x_{3(1)} & +0 & \leq k & \left.\begin{array}{l}\\\\\end{array}\right\} \textit{Scenario } (1,1) \\ & x_1 & +x_{3(1)} & & \leq 1 & \\ & & -x_{3(1)} & & \leq 0 & \\ & x_1 & , \; x_{3(1)} & \in \{0,1\} & & \end{array}$$

In the S-relaxation it is ensured that variables following equal sub-scenarios are set to the same value. As x_2 is set to 1 in each considered scenario in S, x_3 must be set to the same value in both cases. The solution of the DEP is $x_1 = 1$, $x_{3(1)} = 0$ and $k = -1$. Thus, the S-relaxation yields the lower bound -1 for the original QIP. This is not only a better bound than the one obtained by the $\exists\forall$-S-relaxation but it is also a tight bound.

The DEP of the S-relaxation can easily be weakened by turning it into a feasibility problem rather than an optimization problem by dropping the k variable (see Example 3.2.8). This might result in a better runtime but no bound is computed if found feasible. Nevertheless, it might be helpful to quickly assess that no winning strategy can exist for a subtree. Further, dropping the integrality in this DEP results in a standard LP, which can be solved quickly using standard solvers.

Both for the $\exists\forall$-S-relaxation as well as the S-relaxation the selection of scenarios for S is crucial. If too many scenarios are chosen, solving the relaxations might consume too much time. In particular, for $S = \mathcal{L}_\forall$ the S-relaxation is identical to the underlying QIP. If S only consists of scenarios that are 'easy to handle' the resulting bound might not be very helpful. Adding the scenario corresponding to the PV, however, seems particularly advantageous. Therefore, information collected during the search [AN77, Sch83] is used in order to approximate the PV. For more information on implementation and experiments, we refer to Section 7.2.3.

4. Quantified Integer Programming with Polyhedral Uncertainty Set

4.1. Motivation

In optimization under uncertainty—in general—uncertain events and variables must be modeled and selected cautiously as their influence might become too powerful. In the worst-case optimization framework provided by QIPs the modeler must ensure that the modeled worst case is not undesirable or too conservative, i.e. the *price of robustness* must be appropriate [BS04]. For example consider the optimization task of finding an optimal machine scheduling. Expanding this problem by considering the possibility of failing machines yields a robust optimization problem. However, in order to obtain a reasonable instance, the modeler must cautiously specify the number (or set) of machines that may fail: allowing the failure of all machines at the same time would render any robust optimization pointless as in the worst case all machines are broken all the time. In classic robust optimization the modeler therefore has the option to specify the considered uncertainty set. Common are polyhedral uncertainty sets [BB09], ellipsoidal uncertainty sets [BTN99] or budgeted uncertainty sets [BS04]. Further, one can describe an uncertainty set by general norms [BPS04]. Another approach is to define a finite set of anticpated scenarios [MVZ95, RW91] e.g. given through historical data. For QIPs, the use of a finite set of scenarios is easy to apply and one approach has been demonstrated in [ELO14]. Thus, as for the mentioned machine scheduling, limiting the number of non-functional machines can be dealt with by introducing explicit scenarios, the universal player can pick from. This, however, either requires the existence of historical data in order to support the selected scenarios or the scenario set consists of more generic scenarios that can be described by (linear) constraints, e.g. if only scenarios with a maximum of two simultaneous machine failures are considered. Thus, having the ability to restrict the universal variables to be within their own polyhedral set rather than simply using a cubic uncertainty set—given by variable bounds—can be advantageous: it simplifies the modeling itself and in some generic cases bypasses the need of building or selecting scenarios explicitly. We therefore introduced a second linear constraint system $A^{\forall}x \leq b^{\forall}$ that restricts the universal variables to a polytope resulting in the QIP with polyhedral uncertainty set (also see [HE+16]).

4.2. Problem Statement QIP$^{\mathsf{PU}}$

Polyhedral uncertainty sets are easily applicable, straightforward usable and therefore frequently examined in robust optimization [BTN99, BB09]. Thus far, QIPs only allow a cubic uncertainty set, requiring the modeler to carefully implement the uncertainty. We extend the idea of quantified variables by restricting the universal variables to a polytope, described through a *universal constraint system*, given by $A^\forall x \leq b^\forall$ with $A^\forall \in \mathbb{Q}^{m_\forall \times n}$ and $b^\forall \in \mathbb{Q}^{m_\forall}$ for $m_\forall \in \mathbb{N}_0$. We restrict the universal variables in such way that their range only depends on other universal variables. In other words, we assume that existential variables have no influence on the domain of universal variables. We therefore demand

$$A^\forall_{i,k} = 0 \qquad \forall\, i \in \{1, \dots, m_\forall\},\, k \in \mathcal{I}_\exists\,, \tag{4.1}$$

i.e. each entry of $A^\forall$ belonging to an existential variable is zero. The case, where Restriction (4.1) is omitted is discussed in Chapter 5. We use the same notation as for standard QIPs (see Section 2.1).

Definition 4.2.1 (QIP with Polyhedral Uncertainty Set (QIP$^{\mathrm{PU}}$))**.**
Let $A^\exists \in \mathbb{Q}^{m_\exists \times n}$ and $b^\exists \in \mathbb{Q}^{m_\exists}$ for $m_\exists \in \mathbb{N}$. Let $\mathcal{L}$ and Q be given as in Section 2.1 with $\mathcal{I} = \{1, \dots, n\}$ and $Q^{(1)} = Q^{(\beta)} = \exists$. Let $c \in \mathbb{Q}^n$ be the vector of objective coefficients, for which $c^{(i)}$ denotes the vector of coefficients belonging to variable block B_i. Let $m_\forall \in \mathbb{N}$, $b^\forall \in \mathbb{Q}^{m_\forall}$ and $A^\forall \in \mathbb{Q}^{m_\forall \times n}$ complying with Restriction (4.1) and we define $\mathcal{D} = \{x \in \mathcal{L} \mid A^\forall x \leq b^\forall\} \neq \emptyset$. The term $Q \circ x \in \mathcal{D}$ with the component wise binding operator $\circ$ denotes the quantification sequence *$Q^{(1)}x^{(1)} \in \mathcal{D}^{(1)}\ Q^{(2)}x^{(2)} \in \mathcal{D}^{(2)}(x^{(1)})\ \dots\ Q^{(\beta)}x^{(\beta)} \in \mathcal{D}^{(\beta)}(x^{(1)}, \dots, x^{(\beta-1)})$ such that every quantifier $Q^{(i)}$ binds the variables $x^{(i)}$ of block i ranging in their domain $\mathcal{D}^{(i)}(x^{(1)}, \dots, x^{(i-1)})$, with*

$$\mathcal{D}^{(i)}(\tilde{x}^{(1)}, \dots, \tilde{x}^{(i-1)}) = \begin{cases} \mathcal{L}^{(i)} & , \text{ if } i \in \mathcal{E} \\ \{y \in \mathcal{L}^{(i)} \mid \exists x = (\tilde{x}^{(1)}, \dots, \tilde{x}^{(i-1)}, y, x^{(i+1)}, \dots, x^{(\beta)}) \in \mathcal{D}\} & , \text{ if } i \in \mathcal{A}\,. \end{cases}$$

We call

$$z \;=\; \min_{x^{(1)} \in \mathcal{D}^{(1)}} \left(c^{(1)}x^{(1)} + \max_{x^{(2)} \in \mathcal{D}^{(2)}} \left(c^{(2)}x^{(2)} + \min_{x^{(3)} \in \mathcal{D}^{(3)}} \left(c^{(3)}x^{(3)} + \dots \min_{x^{(\beta)} \in \mathcal{D}^{(\beta)}} c^{(\beta)}x^{(\beta)} \right) \right) \right)$$

$$s.t.\ Q \circ x \in \mathcal{D} : A^\exists x \leq b^\exists\,. \tag{4.2}$$

a QIP with polyhedral uncertainty set *(QIP^{PU}) given by the tuple $(A^\exists, A^\forall, b^\exists, b^\forall, c, \mathcal{L}, Q)$.*

In case of a standard QIP any variable x_i (existential or universal) must be integer and within its bounds $[l_i, u_i]$, i.e. $x_i \in \mathcal{L}_i$, in order to constitute a valid variable assignment. In a QIP$^{\mathrm{PU}}$, however, the assignment options for universal variables also depend on the assignment of previous (universal) variables: when assigning a value to the universal variable x_i there must exist a series of future assignments for $x_{i+1}, \dots, x_n$ such that the resulting vector x fulfills $A^\forall x \leq b^\forall$. In other words: such an assignment must not make it impossible to satisfy the system $A^\forall x \leq b^\forall$.

Definition 4.2.2 (Legal Variable Assignment for QIP^{PU})**.**
A legal assignment *of an existential variable x_i demands this variable to be integer and within its bounds $[l_i, u_i]$. The legal assignment of a universal variable additionally depends on the (legal) assignment of previous variables $x_1, \dots, x_{i-1}$, i.e. when assigning a value to the universal variable x_i there must exist a series of legal moves $x_{i+1}, \dots, x_n$ such that the resulting vector x fulfills $A^{\forall}x \leq b^{\forall}$.*

The legal domain $\mathcal{D}^{(i)}(\tilde{x}^{(1)}, \dots, \tilde{x}^{(i-1)})$ of a universal variable block $x^{(i)}$ can be determined by Fourier-Motzkin elimination (extended to integers) [Wil76] of the domain $\mathcal{D}$ and fixating the previous variable assignments $\tilde{x}_1, \dots, \tilde{x}_{i-1}$, or by explicitly computing the set $\mathcal{D}^{(i)}(\tilde{x}^{(1)}, \dots, \tilde{x}^{(i-1)})$ by checking the feasibility of $|\mathcal{L}^{(i)}|$ linear constraint systems.

In Definition 4.2.1 we demand $\mathcal{D} \neq \emptyset$, i.e. we demand the existence of at least one fixation of universal variables $x_{\forall} \in \mathcal{L}_{\forall}$ that fulfills the universal constraint system. This is equal to demanding a non-empty uncertainty set, which is a common condition in robust optimization. With $\mathcal{D} \neq \emptyset$ the universal player always has a strategy to fulfill $A^{\forall}x \leq b^{\forall}$. Therefore, one does not have to deal with plays (completely assigned vectors $\tilde{x} \in \mathcal{L}$) with $A^{\exists}\tilde{x} \not\leq b^{\exists}$ and $A^{\forall}\tilde{x} \not\leq b^{\forall}$ as such a play cannot be optimal. Furthermore, if $\mathcal{D} = \emptyset$ the first universally quantified variable in the quantification sequence has an empty variable domain. This results in a vacuous truth [Qui54, BF+05] and hence the evaluation of Condition (4.2) is independent of the subsequent logical statement.

Definition 4.2.3 (Vacuous Truth)**.**
Let $P(x)$ be some logical statement which depends on x. The statement

$$\forall x \in A: \ P(x)$$

is vacuously true *if $A = \emptyset$.*

Intuitively, the truth of such a statement might come as a surprise. However, as $\forall x \in A: \ P(x)$ is equivalent to $\forall x : (x \in A \Rightarrow P(x))$ the evaluation in case of $A = \emptyset$ is trivial.

Remark 4.2.4. *The statement $\exists x \in A: \ P(x)$ is* vacuously false *if $A = \emptyset$.*

In order to understand the impact in a QIP setting, a simple game argument can help to understand the meaning of such a vacuous statement: Let $\mathcal{L}_i = \emptyset$ for a universal variable x_i. Hence, at some place in the winning condition it is asked whether $\forall x_i \in \emptyset: \ P(x_i)$ which is vacuously true, resulting in a direct win for the existential player. Similarly, if an existential variable domain is empty a vacuously false statement would appear in the winning condition and hence, if this point is reached in the game the existential player loses. This can be restated to "If a player has no legal move she loses immediately".

Example 4.2.5. *Let there be three binary variables, $\mathcal{L} = \{0,1\}^3$ and $Q = (\exists, \forall, \exists)$. Let $c = 0$ and let the two constraint systems be given by a single universal constraint $x_2 \leq -1$ and a*

single existential constraint $x_1 + x_2 + x_3 \geq 4$. *Obviously* $\mathcal{D} = \emptyset$ *since* $\nexists x_2 \in \{0,1\} : \ x_2 \leq -1$. *Furthermore,* $\nexists x \in \mathcal{L} : \ A^{\exists}x \leq b^{\exists}$. *Thus, the winning condition states*

$$\exists\, x_1 \in \{0,1\}\ \forall\, x_2 \in \emptyset\ \exists\, x_3 \in \{0,1\} : \ x_1 + x_2 + x_3 \geq 4\,.$$

With Definition 4.2.3 the statement $\forall y \in \emptyset : F(y)$ *is true for any logical statement* F. *Therefore, the presented instance is feasible, resulting in a win for the existential player, even though he never had any chance fulfilling his constraint. This is due to the asymmetry arising when connecting the (local) universal variable domain to the possible fulfillment of the universal constraint system, whereas the existential variable domain remains independent of the fulfillment of the existential constraint system. This asymmetry can be prevented either by prohibiting* $\mathcal{D} = \emptyset$ *or by restricting the existential variable domain to variable assignments that do not make the existential system unfulfillable. This however results in a massive overhead and is not necessary for QIP*^{PU}*, as having the notion of an uncertainty set in mind, demanding* $\mathcal{D} = \emptyset$ *is the straightforward decision.*

Remark 4.2.6 (Connection between QIP$^{\text{PU}}$ and QII)**.**
The left-hand side system $Bx \leq d$ *of a QII (see Definition 2.2.1) can be interpreted as a constraint system for the universal player. If all entries corresponding to an existential variable are zero the QII is similar to the satisfiabilty QIP*^{PU}*: it is the primary goal of the universal player to fulfill* $Bx \leq d$ *and if she does not manage to do so the existential player wins.*

A QIP$^{\text{PU}}$ can also be represented by a game tree as given in Definition 2.1.10 on page 11. However, one has to adapt the term strategy as only legal universal variable assignments are of interest.

Definition 4.2.7 (Existential Strategy for QIP$^{\text{PU}}$)**.**
A strategy for a QIP$^{\text{PU}}$ *(for the assignment of existential variables)* $S = (V', E', e')$ *is a subtree of the game tree* $G = (V, E, e)$ *of a QIP*^{PU}*.* V' *contains the unique root node* $r \in V_{\exists}$*, each node* $v_{\exists} \in V' \cap V_{\exists}$ *has exactly one child in* S*, and each node* $v_{\forall} \in V' \cap V_{\forall}$ *has all the children that represent legal variable assignments.*

Similar to QIPs we now can define a winning strategy for a QIP$^{\text{PU}}$, which is a strategy for the QIP$^{\text{PU}}$ where all leaves represent a vector x such that $A^{\exists}x \leq b^{\exists}$. With Definition 4.2.7 and $\mathcal{D} \neq \emptyset$, also $A^{\forall}x \leq b^{\forall}$ is fulfilled at such a leaf. Hence, for any strategy as defined in Definition 4.2.7, we can reuse the minimax value presented in Definition 2.1.15. However, in order to enable a minimax search on the entire game tree, one slightly has to adapt the minimax value for QIP$^{\text{PU}}$:

Definition 4.2.8 (Minimax Value for QIP$^{\text{PU}}$)**.**
Let $S = (V', E', e')$ *be a subtree of the game tree* $G = (V, E, e)$ *of a QIP*^{PU}*, with either* $S = G$

or S is a strategy. For any node $v \in V'$ the minimax value under polyhedral uncertainty with respect to S *is recursively defined by*

$$minimax^S_{PU}(v) = \begin{cases} -\infty & , \text{ if } v \in V_L \text{ and } A^\forall x_v \not\leq b^\forall \\ c^\top x_v & , \text{ if } v \in V_L \text{ and } A^\exists x_v \leq b^\exists \text{ and } A^\forall x_v \leq b^\forall \\ +\infty & , \text{ if } v \in V_L \text{ and } A^\exists x_v \not\leq b^\exists \text{ and } A^\forall x_v \leq b^\forall \\ \min\{minimax^S_{PU}(v') \mid (v,v') \in E'\} & , \text{ if } v \in V_\exists \\ \max\{minimax^S_{PU}(v') \mid (v,v') \in E'\} & , \text{ if } v \in V_\forall \,. \end{cases}$$

The minimax value under polyhedral uncertainty with respect to S of the root $r \in V'$ defines the value *of S. For any node $v \in V$ $minimax_{PU}(v) = minimax^G_{PU}(v)$ is the outcome if the remaining variables are assigned optimally starting from v.*

For a leaf $v \in V_L$ with $A^\exists x_v \not\leq b^\exists$ and $A^\forall x_v \not\leq b^\forall$ we define $minimax_{PU}(v) = -\infty$, i.e. we mark this leaf as a loss for the universal player: as described above with $\mathcal{D} \neq \emptyset$ there always exists a universal strategy in the game tree in order to ensure $A^\forall x \leq b^\forall$ and reaching such a leaf can only be accomplished by an illegal universal variable assignment. Hence, regardless of the existential system, the universal player loses this game. In order to evaluate a winning strategy of a QIP$^{\mathrm{PU}}$, it suffices to have the basic minimax value presented in Definition 2.1.15 in mind, as the fulfillment of the universal constraint system is ensured for any strategy in Definition 4.2.7. Similar to QIP, a winning strategy is called optimal if the minimax value at its root is less than or equal to the minimax values of all other winning strategies and the vector $\tilde{x}$ representing the path which obeys the minimax rule is again referred to as the principal variation (PV) (see Definition 2.1.17).

4.3. Reduction QIP$^{\mathrm{PU}} \leq_p$ QIP

Obviously, any given QIP can be interpreted as a QIP$^{\mathrm{PU}}$ by adding a trivial constraint system yielding $\mathcal{D} = \mathcal{L}_\forall$. Thus, PSPACE-hardness of the QIP$^{\mathrm{PU}}$ is immediately given. The question arises, whether by adding the polyhedral uncertainty set—whereby universal variable assignments become dependent on earlier universal decisions in the game—a more complex problem was created (in the sense of complexity theory). We show that the altered uncertainty set has no effect on the complexity and QIP$^{\mathrm{PU}}$ remains PSPACE-complete. We provide a polynomial-time reduction function making it possible to transform any given QIP$^{\mathrm{PU}}$ into a QIP and hence proving the PSPACE-completeness of QIP$^{\mathrm{PU}}$. This is done by creating a collection of additional variables and constraints which have the impact that in the arising QIP the universal player may violate the encoded constraints (given by $A^\forall x \leq b^\forall$), at the price of basically abandoning the game: after such an illegal move the existential player can force a win regardless of what the universal player does afterwards, and the existential player receives a large payoff.

By explicitly providing a polynomial-time reduction function, we do not solely want to verify PSPACE-completeness but further grasp the tight entanglement between QIP and QIP$^{\mathrm{PU}}$. Ad-

ditionally it provides a (potentially ineffective) solution approach and further a deterministic equivalent program can be computed much more easily if a corresponding QIP is at hand. We also do not want to ignore that other arguments exist in order to classify QIP^{PU} with regard to its computational complexity. We briefly present two of them. First, the QIP^{PU}-game can directly be modeled by an alternating Turing machine [CKS81] working in polynomial time. Since $\texttt{AP} = \texttt{PSPACE}$ [AB09] the claim holds. Second, as a QIP^{PU} can be represented by a game tree we can solve it with the minimax algorithm or the alpha-beta algorithm. Both algorithms do not require the entire game tree in the memory but only the current path. Hence, despite the exponential size of the game tree, it can be solved using only polynomial space [KM75].

The reduction aims at transferring the additional condition $A^{\forall}x \leq b^{\forall}$ (from the QIP^{PU}) out of the domain of the variables into the system of constraints of the arising QIP. Note that we cannot simply add $A^{\forall}x \leq b^{\forall}$ as this would not restrict the universal player but tighten the conditions the existential player has to meet. In particular, this would give the universal player an easy way to win by selecting $x_{\forall} \in \mathcal{L}_{\forall}$ that does not fulfill $A^{\forall}x \leq b^{\forall}$. Instead, the fulfillment of the universal constraints is not enforced a priori. We introduce auxiliary constraints and variables that ensure that a violation of $A^{\forall}x \leq b^{\forall}$ is detected, with the effect that the existential player's constraints are relaxed and the payoff is altered to the detriment of the universal player, i.e. in addition to making all constraints trivially fulfilled, the universal player is penalized via the objective function. This can be rephrased in the terms of two-person games: "the existential player wins by default with huge payoff if the universal player cheats".

In order to do this each row of the universal constraint system is checked. Let us consider the k-th row, $k \in \{1, \ldots, m_{\forall}\}$, of the system $A^{\forall}x \leq b^{\forall}$, which is given by

$$\sum_{i \in \mathcal{I}} A^{\forall}_{k,i} \cdot x_i \leq b^{\forall}_k \,. \tag{4.3}$$

It is solely the universal player's task to meet this condition as the existential player cannot influence the left-hand side (due to Condition (4.1)). In case of a violation of row k it is

$$\sum_{i \in \mathcal{I}} A^{\forall}_{k,i} \cdot x_i > b^{\forall}_k \quad \Longleftrightarrow \quad \sum_{i \in \mathcal{I}} A^{\forall}_{k,i} \cdot x_i \geq b^{\forall}_k + \epsilon_k \tag{4.4}$$

for some $\epsilon_k > 0$. To select the parameter ϵ_k we need to find the smallest possible gap between the sum of integer multiples of the coefficients $A^{\forall}_{k,i}$ and $b^{\forall}_k$. It is sufficient to underestimate this smallest possible gap in order to ensure Equivalence (4.4). This can be achieved by using the reciprocal of the (lowest) common multiplier of the denominators—the lowest common denominator (LCD)—of the universal constraint's coefficients.

Lemma 4.3.1. *Let $a, b, \alpha, \beta \in \mathbb{Z}$, $\alpha, \beta > 0$. The smallest positive integer combination of the rational numbers $\frac{a}{\alpha}$ and $\frac{b}{\beta}$ can be underestimated by the reciprocal of the lowest common multiple (LCM) of α and β, i.e.*

$$\min\left\{\left|\frac{a}{\alpha}x + \frac{b}{\beta}y\right| : x, y \in \mathbb{Z} \text{ and } \left|\frac{a}{\alpha}x + \frac{b}{\beta}y\right| \neq 0\right\} \geq \frac{1}{LCM(\alpha, \beta)}$$

Proof. Since $\alpha\beta \geq LCM(\alpha, \beta)$ there exists $c \in \mathbb{Z}$ with $c = \frac{\alpha\beta}{LCM(\alpha,\beta)}$. In particular, c is a factor of both α and β. Therefore,

$$\frac{a}{\alpha}x + \frac{b}{\beta}y \quad = \quad \frac{a\beta x + b\alpha y}{\alpha\beta} \quad = \quad \frac{\frac{a\beta}{c}x + \frac{b\alpha}{c}y}{\frac{\alpha\beta}{c}} \quad = \quad \frac{\tilde{a}x + \tilde{b}y}{LCM(\alpha, \beta)}$$

with $\tilde{\alpha}, \tilde{\beta} \in \mathbb{Z}$. Hence, In order to obtain an underestimate of the smallest positive value of the above term it suffices to underestimate the smallest positive value of the numerator $\tilde{a}x + \tilde{b}y$. This numerator is an integer combination of integers and thus $\tilde{a}x + \tilde{b}y \geq 1$ applies for its smallest positive value. □

Note that in Lemma 4.3.1 it is $LCM(\alpha, \beta) = LCD(\frac{a}{\alpha}, \frac{b}{\beta})$. We henceforth use the LCD, as the denominators are not given explicitly. Let $R_k^{LCD} = (LCD(b_k^\forall, A_{k,1}^\forall, \ldots, A_{k,n}^\forall))^{-1}$ be the reciprocal of the LCD of $b_k^\forall$ and the coefficients $A_{k,\star}^\forall$. Then

$$\sum_{i \in \mathcal{I}} A_{k,i}^\forall \cdot x_i \geq b_k^\forall + R_k^{LCD} \tag{4.5}$$

is fulfilled if and only if the original Constraint (4.3) is not satisfied by the assignment of the universal variables. Note that $R_k^{LCD} = 1$ if all entries of row k are integer. In order to detect a violation of constraint k it thus suffices to check, whether Condition (4.5) is fulfilled. For this purpose, we introduce a new binary existential variable $y_k \in \{0, 1\}$, which is forced to be 0 if constraint k is fulfilled, and can be set to 1, if the constraint is violated, i.e.

$$y_k \begin{cases} = 0 & , \text{if } \sum_{i \in \mathcal{I}} A_{k,i}^\forall \cdot x_i \leq b_k^\forall \\ \in \{0, 1\} & , \text{if } \sum_{i \in \mathcal{I}} A_{k,i}^\forall \cdot x_i > b_k^\forall \,. \end{cases} \tag{4.6}$$

In the second case y_k is set to 1 in optimal play, as the existential player is interested in pointing out misconduct by the universal player. In order to establish Property (4.6) of y_k the constraint

$$\sum_{i \in \mathcal{I}} A_{k,i}^\forall \cdot x_i \geq L_k + (-L_k + b_k^\forall + R_k^{LCD}) \cdot y_k \tag{4.7}$$

is introduced, with

$$L_k = \min_{x \in \mathcal{L}} \left(A_{k,\star}^\forall x \right) = \sum_{\substack{i \in \mathcal{I} \\ A_{k,i}^\forall < 0}} A_{k,i}^\forall \cdot u_i + \sum_{\substack{i \in \mathcal{I} \\ A_{k,i}^\forall \geq 0}} A_{k,i}^\forall \cdot l_i \,, \tag{4.8}$$

which is the smallest possible value of the left-hand side of Constraint (4.3) with respect to $\mathcal{L}$. Let us take a closer look at (4.7). If $y_k = 0$ the constraint is always fulfilled, since $\sum_{i \in \mathcal{I}} A_{k,i}^\forall \cdot x_i \geq L_k$ is trivially fulfilled due to the definition of L_k. On the other hand, setting $y_k = 1$ in (4.7) results in (4.5). Hence, if and only if the original constraint is violated y_k also can take the value 1. However, if the original constraint is satisfied y_k has to be set to 0. Thus, we embedded the variable y_k in a new constraint such that Property (4.6) is fulfilled. As one must check the

fulfillment of each of the rows of the universal system $m_\forall$ many of such binary indicator variables are needed. Furthermore, the auxiliary existential variable $p \in \{0,1\}$ is introduced, which has the ability to indicate the non-compliance of any row of the universal system, i.e.

$$p \begin{cases} = 0 & \text{, if } A^\forall x \leq b^\forall \\ \in \{0,1\} & \text{, if } A^\forall x \not\leq b^\forall \,. \end{cases} \tag{4.9}$$

Using the introduced y_k variables this property can easily be embedded via the constraint

$$p \leq \sum_{k=1}^{m_\forall} y_k \,. \tag{4.10}$$

Hence, if $p = 1$ the system $A^\forall x \leq b^\forall$ is violated in at least one constraint and thus the existential player should win by default. Thus, in this case it must be ensured, that the constraint system of the existential player is fulfilled in any case: if the universal player did not abide by her rules the existential player should not be punished for a violation of his system. Therefore, the existential constraint system is modified as follows

$$A^\exists x - Mp \leq b^\exists \,. \tag{4.11}$$

This corresponds to the standard Big-M modeling technique often used in mixed integer linear programming (see e.g. [CRT90, ST08]), in the sense that if M is "large enough" constraint (4.11) is trivially fulfilled if $p = 1$. This is ensured if the parameter vector $M \in \mathbb{Q}^{m_\exists}$ fulfills

$$\begin{aligned} M_k &\geq \max_{x \in \mathcal{L}} A^\exists_{k,\star} x - b^\exists_k && (4.12) \\ &= \sum_{\substack{i \in \mathcal{I} \\ A^\exists_{k,i} < 0}} A^\exists_{k,i} \cdot l_i + \sum_{\substack{i \in \mathcal{I} \\ A^\exists_{k,i} \geq 0}} A^\exists_{k,i} \cdot u_i - b^\exists_k && (4.13) \end{aligned}$$

for each existential constraint $k \in \{1, \ldots, m_\exists\}$. Hence, by setting $p = 1$ (which is only possible if the universal player did not comply with her constraint system) the existential player does no longer need to fulfill his original constraint system. The global indicator p is now further used to punish the universal player by massively reducing the payoff. Since the universal player is trying to maximize the objective function we can penalize a violation of universal constraints by subtracting this new variable p with a sufficiently large coefficient $\tilde{M}$ in the innermost term of the objective function. This final variable block is an existential block and thus the existential player sets this variable to 1 if possible. For $\tilde{M}$ we choose

$$\tilde{M} = \sum_{\substack{i \in \mathcal{I} \\ c_i < 0}} c_i \cdot (l_i - u_i) + \sum_{\substack{i \in \mathcal{I} \\ c_i \geq 0}}^{n} c_i \cdot (u_i - l_i) + 1 \tag{4.14}$$

with

$$\max_{x \in \mathcal{L}} c^\top x - \tilde{M} < \min_{x \in \mathcal{L}} c^\top x \,. \tag{4.15}$$

Therefore, when subtracting this value the objective function definitely yields a better value for the existential player than he could have achieved without it. However, since $\mathcal{D} \neq \emptyset$, the universal player can always prevent this, by meeting her system of equations and thus forcing p to be zero. The presented variables and constraints are utilized in the following QIP.

Definition 4.3.2 (Reduced QIP^{PU})**.**
Let $P = (A^{\exists}, A^{\forall}, b^{\exists}, b^{\forall}, c, \mathcal{L}, Q)$ be a QIP^{PU}. Let $L \in \mathbb{Q}^{m_\forall}$ be a vector with entries according to Equation (4.8). Let $R^{LCD} \in \mathbb{Q}^{m_\forall}$ be a vector where each entry R^{LCD}_k, $k \in \{1, \ldots, m_\forall\}$, is positive and less than or equal to the reciprocal of the lowest common denominator of the entries of $A^{\forall}_{k,\star}$ and $b^{\forall}_k$. Let $\tilde{M}$ and M be given as in Conditions (4.14) and (4.13), respectively. Then the reduction function $f^{PU}(P)$ maps the QIP^{PU} P to the following QIP:

$$\min_{x^{(1)} \in \mathcal{L}^{(1)}} \left(c^{(1)}x^{(1)} + \max_{x^{(2)} \in \mathcal{L}^{(2)}} \left(c^{(2)}x^{(2)} + \min_{x^{(3)} \in \mathcal{L}^{(3)}} \left(\ldots + \min_{\substack{x^{(\beta)} \in \mathcal{L}^{(\beta)},\ y \in \{0,1\}^{m_\forall},\\ p \in \{0,1\}}} \left(c^{(\beta)}x^{(\beta)} - \tilde{M}p \right) \right) \right) \right)$$

$$\text{s.t. } Q \circ x \in \mathcal{L}\ \exists y \in \{0,1\}^{m_\forall}\ \exists p \in \{0,1\} :$$

$$A^{\exists}x - Mp \leq b^{\exists} \tag{4.16}$$

$$-A^{\forall}x - (L - b^{\forall} - R^{LCD})y \leq -L \tag{4.17}$$

$$p - \sum_{k=1}^{m_\forall} y_k \leq 0 \tag{4.18}$$

We show that f^{PU} indeed constitutes a polynomial-time reduction function.

Lemma 4.3.3. *Let $P = (A^{\exists}, A^{\forall}, b^{\exists}, b^{\forall}, c, \mathcal{L}, Q)$ be a QIP^{PU}. Then $f^{PU}(P)$ is a standard QIP.*

Proof. Let $f^{\text{PU}}(P) = (\bar{A}, \bar{b}, \bar{c}, \bar{\mathcal{L}}, \bar{Q})$. It is $\bar{\mathcal{L}} = \mathcal{L} \times \{0,1\}^{m_\forall + 1}$ and thus each variable is bound by integer lower and upper bound and must be integer, i.e. $\bar{\mathcal{L}}$ is a cubical integer lattice. Further, the first and final quantifier is existential, as $\bar{Q} = (Q, \exists^{m_\forall + 1})$. The objective vector $\bar{c} = (c, 0^{m_\forall}, -\tilde{M})$, as well as the constraint system $\bar{A}z \leq \bar{b}$, given through (4.16)–(4.18), have all rational entries. □

First, we show that $f^{\text{PU}}(P)$ can be computed in polynomial time with respect to the size of the input, which is given through n, $m_\exists$ and $m_\forall$.

Theorem 4.3.4. *Let $P = (A^{\exists}, A^{\forall}, b^{\exists}, b^{\forall}, c, \mathcal{L}, Q)$ be a QIP^{PU}. Then the QIP $f^{PU}(P)$ as given in Definition 4.3.2 can be computed in polynomial time.*

Proof. Let $f^{\text{PU}}(P) = (\bar{A}, \bar{b}, \bar{c}, \bar{\mathcal{L}}, \bar{Q})$. For $\bar{c} = (c, 0^{m_\forall}, -\tilde{M})$ the entry $\tilde{M}$ can be calculated by using the upper and lower bound given in $\mathcal{L}$ appropriately, depending on the sign of the corresponding entries in c and hence requires $\mathcal{O}(n)$ operations (see Condition (4.14)). The computation of L requires $\mathcal{O}(m_\forall \cdot n)$ operations (see Condition (4.8)). Similarly, M can be computed in $\mathcal{O}(m_\forall \cdot n)$ (see Condition (4.13)). As the entries of R^{LCD} only need to underestimate the lowest common denominator of the corresponding row's entries it suffices to multiply the

denominators of the non-zero entries and take its reciprocal. Hence, a valid assignment of R^{LCD} can be computed in $\mathcal{O}(m_\forall \cdot n)$. Therefore, each entry of $\bar{A}$, $\bar{b}$, and $\bar{c}$ can be computed in polynomial time. Further, there are $m_\forall + 1$ auxiliary binary variables y and p and hence the resulting QIP has $n + m_\forall + 1$ variables. The number of constraints is $m_\exists + m_\forall + 1$, and $\bar{A}$ has $\mathcal{O}((n + m_\forall) \cdot (m_\exists + m_\forall))$ entries. □

In order to constitute a reduction function we must show that the QIP$^{\text{PU}}$ P and the transformed QIP $f^{\text{PU}}(P)$ are closely connected. For further investigations the variable vector of the transformed problem $f^{\text{PU}}(P)$ is denoted by $z = (x, y, p) \in \mathcal{L} \times \{0,1\}^{m_\forall} \times \{0,1\}$. The following Lemma 4.3.5 shows that a solution for $f^{\text{PU}}(P)$ always contains a solution for P.

Lemma 4.3.5. *Any winning strategy of $f^{PU}(P)$ contains a winning strategy for P as a subgraph.*

Proof. Obviously the game tree $G = (V, E, e)$ of P is a subgraph of the game tree $\bar{G} = (\bar{V}, \bar{E}, \bar{e})$ of $f^{\text{PU}}(P)$: considering only the nodes until level n of $\bar{G}$ yields G. Assume $\bar{S} = (\bar{V}', \bar{E}', \bar{e}')$ is a winning strategy of $f^{\text{PU}}(P)$ in $\bar{G}$, i.e., in each leaf the system of Constraints (4.16)–(4.18) is fulfilled. Note that this arborescence has a depth of $n + m_\forall + 1$. We consider the arborescence $S = (V', E', e')$ in G with $V' \subseteq \bar{V}'$, $E' \subseteq \bar{E}'$ and $e'((v,w)) = \bar{e}'((v,w))$ for each $(v,w) \in E'$. V' contains no node of a level larger than n and E' contains no edges leading to such nodes. Further, edges describing illegal assignments (see Definition 4.2.2) in terms of the QIP$^{\text{PU}}$ are deleted as well as their entire underlying subtrees. This designed arborescence S describes a strategy for P, because

- the depth is n and thus for each variable a decision level exists.
- nodes of universal variables have only legal assignment options leading out.
- the remaining strategy properties are adopted from $\bar{S}$.

This strategy S is also a winning strategy for P, since each path from the root to a leaf represents a vector x such that $A^\exists x \leq b^\exists$: Let us consider such a path $x_1, \ldots, x_n$ in S and the unique[8] associated overlying path $z = (x_1, \ldots, x_n, y_1, \ldots, y_{m_\forall}, p)$ in $\bar{S}$. As $A^\forall x \leq b^\forall$ holds, since illegal universal assignments were deleted, and due to the fulfillment of the Constraints (4.16)–(4.18) we conclude $p = 0$ and $y_i = 0$ for all $i \in \{1, \ldots, m_\forall\}$ since $\bar{S}$ is a winning strategy. Thus, because of Constraint (4.16), also $A^\exists x \leq b^\exists$ holds and hence S is a winning strategy for the QIP$^{\text{PU}}$ P. □

Note that the first-stage solution of the transformed QIP is identical[9] to the first-stage solution of such a constructed QIP$^{\text{PU}}$. The following two theorems prove that f^{PU} indeed is a reduction. In particular, we show that P is feasible with optimal value z_{opt} if and only if $f^{\text{PU}}(P)$ is feasible with the same optimal value.

Theorem 4.3.6. *Let the QIP^{PU} $P = (A^\exists, A^\forall, b^\exists, b^\forall, c, \mathcal{L}, Q)$ have an optimal winning strategy with PV $\tilde{x}$ and optimal value $c^\top \tilde{x}$. Then the transformed QIP $f^{PU}(P)$ has an optimal winning*

[8] The path is unique, because all nodes with level $\geq n$ belong to existential variables and thus have only one successor in a strategy.

[9] except for auxiliary variable p in single-stage instances.

strategy with PV $\tilde{z} = (\tilde{x}, \tilde{y}, \tilde{p})$ *with* $\tilde{y}_i = 0$ *for* $i = 1, \ldots, m_\forall$ *and* $\tilde{p} = 0$ *with the same optimal value.*

Proof. Let $f^{\text{PU}}(P) = (\bar{A}, \bar{b}, \bar{c}, \bar{\mathcal{L}}, \bar{Q})$. As outlined in the proof of Lemma 4.3.5 the tree $G = (V, E, e)$ of P is a subgraph of the game tree $\bar{G} = (\bar{V}, \bar{E}, \bar{e})$ of $f^{\text{PU}}(P)$. In particular, $\bar{G}$ can be created by adding a perfect binary tree of depth $m_\forall + 1$ to each leaf G representing the assignment of the auxiliary binary variables y and p. First we show that the optimal winning strategy $S = (V', E', e')$ of P can easily be expanded into a winning strategy for $f^{\text{PU}}(P)$. Each leaf $v \in V'$ of S has the property $A^\exists x_v \leq b^\exists$ and $A^\forall x_v \leq b^\forall$. Hence, by choosing $y = 0$ and $p = 0$ in $f^{\text{PU}}(P)$ results in $z_v = (x_v, 0, \ldots, 0, 0)$ with $\bar{A} z_v \leq \bar{b}$. Further, a winning strategy in $\bar{G}$ must take all possible universal variable assignments from $\mathcal{L}$ into account, not only the legal ones (cf. Definition 2.1.13 and Definition 4.2.7). Consider any universal decision node of the optimal winning strategy for P with less successors than elements in the corresponding variable domain $\mathcal{L}_i$. Let $\hat{x}_i \in \mathcal{L}_i$ be a variable assignment not considered in the strategy of P, i.e. setting this universal variable to $\hat{x}_i$ at this node is not a legal variable assignment according to Definition 4.2.2. Starting from this node, the winning strategy of P can be extended into a winning strategy for $f^{\text{PU}}(P)$ as follows:

a) at existential nodes in V add an arbitrary successor.

b) at universal nodes in V add all successors.

c) at nodes in $\bar{V} \setminus V$ select one constraint of $A^\forall x \leq b^\forall$ that is not met (according to the assignments along the given path) and set the corresponding y_k variable to 1, the other variables of y to 0 and finally set $p = 1$.

Each leaf of such an additional substrategy fulfills $\bar{A} z \leq \bar{b}$ as with $p = 1$ and at least one $y_k = 1$ the Constraints (4.16) and (4.18) are obviously fulfilled. As the subtree arises from an illegal variable assignment, the assignments of the universal variables along the path cannot fulfill $A^\forall x \leq b^\forall$ and hence there exists at least one constraint that is not fulfilled. For such a constraint the corresponding y_k variable was set to 1 and thus the k-th Constraint of (4.17) is fulfilled. As the remaining y variables were set to 0 the entire constraint system is fulfilled. Using this procedure a strategy for the QIP $f^{\text{PU}}(P)$ arises. It also constitutes a winning strategy for $f^{\text{PU}}(P)$ since for each leaf v and its corresponding variable assignment $z_v = (x, y, p)$ it is $\bar{A} z_v \leq \bar{b}$ and further

$$\bar{c}^\top z \begin{cases} = c^\top x & \text{, if there is a leaf } v \text{ in } S \text{ with } x_v = x \\ < \min_{x \in \mathcal{L}} c^\top x & \text{, if } x \text{ is not represented in } S, \text{ since } A^\forall x \not\leq b^\forall . \end{cases}$$

Thus, with Theorem 2.1.16, the value of the constructed winning strategy is equal to $c^\top \tilde{x}$ and the PV is $\tilde{z} = (\tilde{x}, 0, \ldots, 0, 0)$.

We now must show, that this constructed winning strategy for $f^{\text{PU}}(P)$ indeed is the optimal winning strategy. Let $\hat{z} = (\hat{x}, \hat{y}, \hat{p})$ be the PV of the optimal winning strategy of the transformed problem and thus $c^\top \hat{x} - \tilde{M} \hat{p} \leq c^\top \tilde{x}$. If $\hat{x} \notin \mathcal{D}\ \hat{z}$ would also fulfill $\hat{p} = 1$, since at least one row of the system $A^\forall \hat{x} \leq b^\forall$ is violated. However, because of Condition (4.15) the resulting value of the

objective function is less than any other solution obeying $A^{\forall}\hat{x} \leq b^{\forall}$. This is a contradiction to the minimax optimality of $\hat{z}$ since the universal player can avoid this by assigning her variables such that $A^{\forall}\hat{x} \leq b^{\forall}$ holds. Therefore, $\hat{x} \in \mathcal{D}$ and $A^{\forall}\hat{x} \leq b^{\forall}$ and hence $\hat{y} = 0$ and $\hat{p} = 0$. With Lemma 4.3.5 and the optimality of $\tilde{x}$ for P it is $c^{\top}\hat{x} \geq c^{\top}\tilde{x}$ and therefore $c^{\top}\hat{x} = c^{\top}\tilde{x}$. □

Theorem 4.3.7. *If a* QIP^{PU} P *has no winning strategy, then* $f^{PU}(P)$ *is also infeasible.*

Proof. The claim follows immediately with Lemma 4.3.5. □

Corollary 4.3.8. *QIP*^{PU} *is in* PSPACE. *Since the QIP with cubical uncertainty set is a special case of the QIP*^{PU} *it is even* PSPACE*-complete.*

4.4. Examples

4.4.1. Weighted Dynamic Graph Reliability

We consider a simple graph game where one player has to traverse a given graph while the opponent is allowed to erase some edges. However, the opponent is not allowed to erase edges arbitrarily but must obey some rules. This problem is closely related to the dynamic graph reliability problem [Pap85] with the difference that edges have weights and an objective function should be minimized. Further, edges are erased depending on the point in time instead of the location of the player. We investigate the problem examplarily for the graph given in Figure 4.1, present its QIP^{PU} formulation and apply the reduction function presented in Section 4.3.

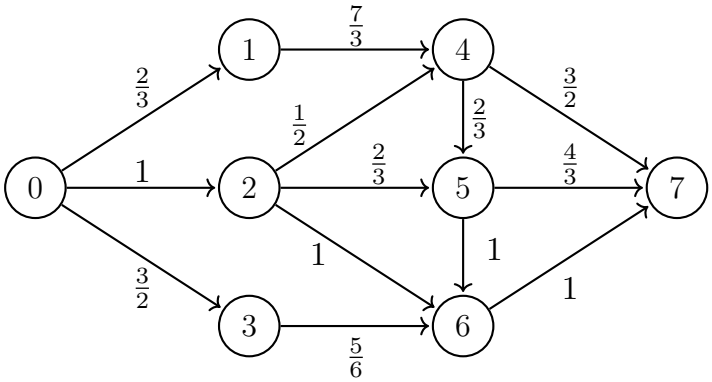

Figure 4.1.: Directed acyclic weighted graph with starting node 0 and target node 7.

The graph consists of eight nodes and the starting node is labeled with 0 and the target node with 7. The existential player, starting at node 0, wants to reach node 7 in the cheapest possible way (in terms of edge weights), whereas the universal player can delete certain edges in order to either prevent the existential player from reaching node 7, or to force him on an expensive path. Hence, the following questions arise: Is there a strategy for the existential player which allows him to reach the target node no matter how the opponent acts? And if there are multiple: which one is the winning strategy with the cheapest worst-case path to the target node?

Let $G = (V, E, c)$ describe the graph given in Figure 4.1 with V being the set of vertices, E the set of edges and $c : E \to \mathbb{Q}$ a function assigning weights to each edge. Let $x_{i,j} \in \{0, 1\}$ be variables indicating whether the existential player uses edge $(i, j) \in E$ or not. For each edge $(i, j) \in E$ with $i \neq 0$ let $d_{i,j} \in \{0, 1\}$ indicate whether the universal player deleted this edge or not. The existential player has the first move and can choose between node 1, 2 or 3. Subsequently, the universal player can delete certain edges. Both players take turns, whereat the existential player selects one edge per move and the universal player is allowed to deactivate upcoming edges. The entire turn order is given by the following quantifier string, omitting the binary variable domains for better readability:

$$\exists\, x_{0,1}, x_{0,2}, x_{0,3}\ \forall\, d_{1,4}, d_{2,4}, d_{2,5}, d_{2,6}, d_{3,6}\ \exists\, x_{1,4}, x_{2,4}, x_{2,5}, x_{2,6}, x_{3,6}$$
$$\forall\, d_{4,7}, d_{4,5}, d_{5,7}, d_{5,6}, d_{6,7}\ \exists\, x_{4,7}, x_{4,5}, x_{5,7}, x_{5,6}, x_{6,7}$$

The minimax objective function adds up the weights of the edges selected by the existential player. The variables active in each min/max block are given by the order and are abbreviated by their corresponding variable block $B^{(i)}$:

$$\min_{B^{(1)}} \left(\frac{2}{3}x_{0,1} + x_{0,2} + \frac{3}{2}x_{0,3} + \max_{B^{(2)}} \left(\min_{B^{(3)}} \left(\frac{7}{3}x_{1,4} + \frac{1}{2}x_{2,4} + \frac{2}{3}x_{2,5} + x_{2,6} + \frac{5}{6}x_{3,6} \right.\right.\right.$$
$$\left.\left.\left. + \max_{B^{(4)}} \left(\min_{B^{(5)}} \left(\frac{3}{2}x_{4,7} + \frac{2}{3}x_{4,5} + \frac{4}{3}x_{5,7} + x_{5,6} + x_{6,7} \right) \right) \right) \right) \right)$$

The existential player's system of equations $A^{\exists}x \leq b^{\exists}$ is given as follows:

$$\sum_{(0,j)\in E} x_{0,j} = 1 \tag{4.19}$$
$$\sum_{(i,7)\in E} x_{i,7} = 1 \tag{4.20}$$
$$\sum_{(i,k)\in E} x_{i,k} = \sum_{(k,j)\in E} x_{k,j} \qquad \forall\, k \in \{1, \ldots, 6\} \tag{4.21}$$
$$x_{i,j} + d_{i,j} \leq 1 \qquad \forall\, (i, j) \in E,\ i \neq 0 \tag{4.22}$$

Constraints (4.19)–(4.21) ensure the flow from node 0 to 7 and Constraint (4.22) forbids the use of edges that have been deleted by the universal player. The universal player on the other hand is restricted by her system $A^{\forall}x \leq b^{\forall}$ as follows:

$$\sum_{\substack{(i,j)\in E \\ i\neq 0}} d_{i,j} \leq 3\,, \qquad \sum_{\substack{(i,j)\in E \\ i\neq 0}} c(i, j) \cdot d_{i,j} \geq \frac{3}{2}\,, \qquad \sum_{\substack{(i,j)\in E \\ i\neq 0}} c(i, j) \cdot d_{i,j} \leq 2 \tag{4.23}$$

This system states that the universal player is allowed to delete at most 3 edges and the sum of the weights of the deleted edges must be between 1.5 and 2. Note that we did not convert either system into a "less or equal" system in order to make their actual use more clear. This, however, is necessary in order to use the transformation described in Section 4.3: each of the Equations

(4.19), (4.20) and (4.21) must be split up into two $\leq$-constraints and the $\geq$-Constraint (4.23) must also be flipped. This is an important step in order to obtain the correct values for M and L using Conditions (4.13) and (4.8), respectively. The reduced QIP is displayed below. Again, for convenience, the repeating variable domains $\{0,1\}$ are omitted in the quantifier string and in the objective we abbreviate the original variables active by their block $B^{(i)}$.

$$\min_{B^{(1)}}\left(\frac{2}{3}x_{0,1}+x_{0,2}+\frac{3}{2}x_{0,3}+\max_{B^{(2)}}\left(\min_{B^{(3)}}\left(\frac{7}{3}x_{1,4}+\frac{1}{2}x_{2,4}+\frac{2}{3}x_{2,5}+x_{2,6}+\frac{5}{6}x_{3,6}\right.\right.\right.$$
$$\left.\left.\left.+\max_{B^{(4)}}\left(\min_{B^{(5)},y,p}\left(\frac{3}{2}x_{4,7}+\frac{2}{3}x_{4,5}+\frac{4}{3}x_{5,7}+x_{5,6}+x_{6,7}-15p\right)\right)\right)\right)\right)$$
$$\text{s.t. } \exists\, x_{0,1},x_{0,2},x_{0,3}\ \forall\, d_{1,4},d_{2,4},d_{2,5},d_{2,6},d_{3,6}\ \exists\, x_{1,4},x_{2,4},x_{2,5},x_{2,6},x_{3,6}$$
$$\forall\, d_{4,7},d_{4,5},d_{5,7},d_{5,6},d_{6,7}\ \exists\, x_{4,7},x_{4,5},x_{5,7},x_{5,6},x_{6,7},y_1,y_2,y_3,p:$$

$$-\sum_{(0,j)\in E}x_{0,j}-p\leq -1\ ,\quad \sum_{(0,j)\in E}x_{0,j}-2p\leq 1 \tag{4.24}$$
$$-\sum_{(i,7)\in E}x_{i,7}-p\leq -1\ ,\quad \sum_{(i,7)\in E}x_{i,7}-2p\leq 1 \tag{4.25}$$
$$\sum_{(i,k)\in E}x_{i,k}-\sum_{(k,j)\in E}x_{k,j}-\deg^-(k)\cdot p\leq 0 \qquad \forall\, k\in\{1,\dots,6\} \tag{4.26}$$
$$\sum_{(k,j)\in E}x_{k,j}-\sum_{(i,k)\in E}x_{i,k}-\deg^+(k)\cdot p\leq 0 \qquad \forall\, k\in\{1,\dots,6\} \tag{4.27}$$
$$x_{i,j}+d_{i,j}-p\leq 1 \qquad \forall\,(i,j)\in E,\ i\neq 0 \tag{4.28}$$
$$4y_1-\sum_{\substack{(i,j)\in E\\ i\neq 0}}d_{i,j}\leq 0 \tag{4.29}$$
$$\sum_{\substack{(i,j)\in E\\ i\neq 0}}c(i,j)\cdot d_{i,j}+9.5y_2\leq \frac{65}{6} \tag{4.30}$$
$$-\sum_{\substack{(i,j)\in E\\ i\neq 0}}c(i,j)\cdot d_{i,j}+\frac{13}{6}y_3\leq 0 \tag{4.31}$$
$$p-\sum_{k=1}^{3}y_k\leq 0 \tag{4.32}$$

Obviously, the resulting QIP is less comprehensibly than simply stating the two constraint systems $A^{\exists}x\leq b^{\exists}$ and $A^{\forall}x\leq b^{\forall}$ separately. Constraints (4.24)–(4.28) describe the transformed existential system (cf. Constraint (4.16)), Constraints (4.29)–(4.31) are the embedded universal constraints (cf. Constraint (4.17)) and Constraint (4.32) is similar to Constraint (4.18). In Constraints (4.26) and (4.27) the coefficients of p are the number of incoming edges $\deg(k)^- = |\{(i,j)\in E \mid j=k\}|$ and the number of outgoing edges of node k $\deg(k)^+ = |\{(i,j)\in E \mid i=k\}|$, respectively. These values arise when computing the corresponding values of M and guarantee the trivial fulfillment for these rows if $p=1$. In Constraint (4.30) the coefficients result from

$L_2 = -\frac{65}{6}$, $R_2^{LCD} = \frac{1}{6}$ and $b_2^{\forall} = -\frac{3}{2}$. This standard QIP is easily solved by our solver. It turns out that there is a winning strategy for the existential player. The objective value of the PV is $\frac{11}{3}$ and the optimal first-stage decision is moving from the starting node 0 to node 2. The (perfect) universal player then deletes the edge between 2 and 4. The existential player is forced to move to node 5: directly moving to node 6 would result in a loss, as deleting edge $(6, 7)$ then perfectly fits into the universal player's budget. After that edge $(5, 7)$ is deleted and finally the target node is reached by taking the detour via node 6.

4.4.2. Resilient Booster Stations with QIP$^{\text{PU}}$

Consider the example given in Subsection 2.3.2 where a booster station is made more resilient by adding additional pumps. Among other things, the system must be able to cope with any load scenario, even if one of the initial pumps is malfunctioning. Both the selected load scenarios $s \in S$ as well as the selected broken pump $b \in I$ are modeled via integer universal variables. Those universal decisions are transformed into existential binary indicator variables σ and β, via Constraints (2.16)–(2.19), i.e.

$$\sum_{i \in S} \sigma_i \leq 1\,, \qquad \sum_{i \in S} i\sigma_i = s\,, \qquad \sum_{p \in I} \beta_p \leq 1 \qquad \text{and} \qquad \sum_{p \in I} p\beta_p = b\,.$$

Even just for modeling reasons it would be advantageous if this workaround was not necessary and instead σ and β could be directly used as the universal variables indicating the selected load scenario and the broken pump. This, however, is not straightforward possible with regular QIP, as without the condition $\sum_{p \in I} \beta_p \leq 1$ all pumps would fail simultaneously in the worst case, which does not correspond to the problem description. With the QIP$^{\text{PU}}$ framework, however, the universal constraint system can be utilized: by adding the two constraints $\sum_{p \in I} \beta_p \leq 1$ and $\sum_{i \in S} \sigma_i \leq 1$ to the system $A^{\forall} x \leq b^{\forall}$, the universal player has to fulfill them. Consequently, the model presented in Subsection 2.3.2 could be restated as QIP$^{\text{PU}}$ by

1. leaving out the integer universal variables s and b,
2. converting σ and β into universal variables of the second and fourth stage, respectively,
3. removing the Constraints (2.17) and (2.19),
4. marking the Constraints (2.16) and (2.18) as universal constraints.

Therefore, less constraints and fewer variables are required in order to grasp the problem as a QIP$^{\text{PU}}$. An additional advantage of the QIP$^{\text{PU}}$ is its better adaptability, e.g. if the requirements for a special booster station make it necessary to protect against the failure of two pumps simultaneously, the universal constraint system could simply be altered. This, in contrast, would require massive changes in the presented QIP as a scenario number would have to be transferred into a specific pump combination rather than a single pump.

4.4.3. Robust Runway Scheduling with Restricted Universal Options

In the robust runway scheduling problem presented in Subsection 2.3.1 the universal variables define the set of anticipated arrival time windows of the airplanes. If they are only restricted by their bounds, it is difficult to create meaningful scenarios: for example one might want to allow the time windows for a few airplanes to consist of only one time slot, but this should certainly not be the case for all airplanes. One conceivable demand for the time windows could be that they have a length of 2 on average (i.e. consist of three time slots). Therefore, the universal variables d_i, which specify this duration for each airplane $i \in A$, should not only obey their bounds, but also should fulfill the following condition:

$$\frac{1}{|A|} \sum_{i \in A} d_i \geq 2 \tag{4.33}$$

Constraint (4.33) can either be added to a universal constraint system $A^{\forall}x \leq b^{\forall}$ resulting in a QIP$^{\text{PU}}$. Or the rules regarding the universal variables can be enforced implicitly as shown in Definition 4.3.2: In a final existential block the fulfillment of such a constraint is checked and if a violation is detected the remaining constraint system is relaxed and the objective value is reduced drastically. This has the effect that a violation provoked by the assignment of universal variables results in a very good objective value (regarding the existential objective of minimization) and is thus unfavorable with respect to the universal maximization objective. To achieve this, we first rewrite Condition (4.33) to $-\sum_{i \in A} d_i \leq -2|A|$. Then the constraint is in the desired $\leq$-form and further $R^{LCD} = 1$ for this constraint, as each coefficient is integer. Let the domain of the time window lengths be given by $D = \{d \in \mathbb{N}_0^{|A|} \mid \forall i \in A : \ a_i \leq d_i \leq b_i\}$, with $a_i, b_i \in \mathbb{N}_0$ and $a_i \leq b_i$. Let $M_D = \sum_{i \in A} b_i$. Then the lowest value of the left-hand side is $-M_D$. Note that the anticipated lengths in D should not immediately contradict Restriction (4.33), i.e. it should be $M_D \geq 2|A|$. We can add constraint

$$\sum_{i \in A} d_i + (M_D - 2|A| + 1)p \leq M_D \tag{4.34}$$

to the existential constraint system and the checking variable $p \in \{0, 1\}$ to the final existential variable block. For $p = 0$ Constraint (4.34) is always fulfilled. On the other hand, p can be set to 1 only if $\frac{1}{|A|} \sum_{i \in A} d_i < 2$. Further, p is added to the objective function with coefficient $\tilde{M}$, which highly depends on the selected fixing costs (see Equation (4.14) and Objective (2.3)). Further, it is necessary to relax the existential constraint system, if the universal variables do not fulfill Constraint (4.33). But note that it is not necessary to relax each of the Constraints (2.4)–(2.8). It suffices to relax Constraint (2.7) for each time slot $j \in S$ in the following way: $\sum_{i \in A} y_{i,j} \leq b + |A|p$. This way, there always exists a final schedule y if p can be set to 1, because all airplanes can simply be planned in the same slot if necessary.

In Subsection 7.3.1 we present the results of computational experiments on robust runway scheduling instances and compare the performance of our solver on the QIP$^{\text{PU}}$ with its performance on the equivalent QIP.

4.4.4. Multistage Selection Problem

Consider a combinatorial problem of the form

$$\min \sum_{i=1}^{n} c_i x_i \qquad \text{s.t. } x \in \mathcal{X} \subseteq \{0,1\}^n .$$

In this section we examine the selection problem, i.e. $\mathcal{X} = \{x \in \{0,1\}^n : \sum_{i=1}^n x_i = p\}$, where p out of n items must be selected, such that the costs are minimized. We assume that uncertainty is only present in the objective, and a discrete list of $N \in \mathbb{N}$ potential cost vectors is given. A thorough overview of the two-stage robust selection problem can be found in [CG+18]. The two-stage robust problem is given by $\min_{x\in\mathcal{X}} \max_{c\in\mathcal{C}} cx$ with $\mathcal{C}$ being the set of anticipated scenarios. We build a robust counterpart, i.e. an equivalent MIP. Therefore, let $c_{i,k} \in \mathbb{N}_0$ be the cost for item $i \in \{1,\dots,n\}$ in scenario $k \in \{1,\dots,N\}$ and let x_i be the variable indicating the selection of item i. The robust counterpart is given as follows:

$$\min\ z \tag{4.35}$$

$$\text{s.t.} \sum_{i=1}^{n} c_{i,k} x_i \le z \qquad \forall\, k \in \{1,\dots,N\} \tag{4.36}$$

$$\sum_{i=1}^{n} x_i = p \tag{4.37}$$

$$x_i \in \{0,1\} \qquad \forall\, i \in \{1,\dots,n\} \tag{4.38}$$

This can be formulated as a QIP^{PU} by introducing universal variables q_k that indicate whether cost scenario k is selected. As only one scenario can occur, the universal constraint $\sum_{k=1}^N q_k = 1$ is used. Therefore, the universal variable domain $\mathcal{D}$ established in this chapter is given by $\mathcal{D} = \{q \in \{0,1\}^N \mid \sum_{k=1}^N q_k = 1\}$. A first straightforward attempt to model the objective function results in the nonlinear expression $\sum_{i=1}^n \sum_{k=1}^N q_k(c_{i,k}x_i)$. This nonlinearity is avoided by using the auxiliary variable z, which bundles the costs, and Constraint (4.41), which connects the selected scenario to the resulting costs using the Big-M method. The entire QIP^{PU} model for the robust selection problem is given as follows:

$$\min\ z \tag{4.39}$$

$$\text{s.t. } \exists\, x \in \{0,1\}^n \quad \forall\, q \in \mathcal{D} \quad \exists\, z \in \mathbb{N}_0\text{:}$$

$$\sum_{i=1}^{n} x_i = p \tag{4.40}$$

$$\sum_{i=1}^{n} c_{i,k} x_i \le z + M_k(1 - q_k) \qquad \forall\, k \in \{1,\dots,N\} \tag{4.41}$$

If M_k is selected appropriately for each potential scenario k (e.g. $M_k \ge \sum_{i=1}^n c_{i,k}$), all but one of the Constraints (4.41) are trivially fulfilled for a realization of $q \in \mathcal{D}$: if scenario k is selected by the universal player ($q_k = 1$) the corresponding costs $c_{\star,k}$ are decisive for the cost calculation.

The presented selection problem can be extended to a multistage decision problem: In the first (existential) decision stage a set of items can be selected for fixed costs c^0. Then, in a universal decision stage, a cost scenario is selected and in the subsequent existential decision stage further items can be selected. Those two stages can be repeated iteratively several times. If an item is selected, it cannot be selected again in a later stage and the goal remains to select p items such that the resulting costs are minimized. Let $S \in \mathbb{N}$ be the number of iterations and thus S is the number of universal decision stages. The universal domain for each universal stage $s \in \{1, \dots, S\}$ is given by $\mathcal{D}_s = \{q^s \in \{0,1\}^N \mid \sum_{k=1}^{N} q_k^s = 1\}$ and $q^s \in \mathcal{D}_s$ is the vector indicating the selected scenario. Let again N be the number of scenarios, i.e. at each iteration one of N scenarios is revealed. The cost of item i in scenario k of iteration s are given by $c_{i,k}^s$. This multistage selection problem under uncertainty can be modeled as a quantified program with a polyhedral uncertainty set (SELQ$^{\text{PU}}$) as follows:

Problem SELQ$^{\text{PU}}$

$$\min \sum_{i=1}^{n} c_i^0 x_i^0 + \sum_{s=1}^{S} z_s \tag{4.42}$$

$$\text{s.t. } \exists x^0 \in \{0,1\}^n \quad \forall q^1 \in \mathcal{D}_1 \quad \exists x^1 \in \{0,1\}^n \ \cdots \ \forall q^S \in \mathcal{D}_S \quad \exists x^S \in \{0,1\}^n \ \exists z \in \mathbb{N}_0^S\text{:}$$

$$\sum_{i=1}^{n} \sum_{s=0}^{S} x_i^s = p \tag{4.43}$$

$$\sum_{s=0}^{S} x_i^s \leq 1 \qquad \forall\, i \in \{1, \dots, n\} \tag{4.44}$$

$$\sum_{i=1}^{n} c_{i,k}^s x_i^s \leq z_s + M(1 - q_k^s) \qquad \forall\, k \in \{1, \dots, N\},\ s \in \{1, \dots, S\} \tag{4.45}$$

The Objective Function (4.42) consists of the expenses from the first stage with invariable costs and the expenses of subsequent iterations in which the cost for each item depend on the selected scenario. The first Constraint (4.43) demands that overall exactly p items must be selected. Constraint (4.44) prevents that an item is selected more than once. Constraint (4.45) enforces the link between the selected scenario, selected items and resulting costs in each iteration s. Note that for potential future models with cost vectors containing rational numbers the existential variables z also could be continuous. As our solver can only deal with (existential) continuous variables in the very last variable block, we put the z variables at the end of the quantification sequence. Here, however, the cost variables z_s also could be placed immediately after the corresponding selection in iteration s. Additionally, when explicitly stating the model one has to specify an upper bound on z_s, which easily can be computed by taking the cost vectors of the corresponding scenarios into account. Note that in generated SELQ$^{\text{PU}}$ instances we actually did use continuous variables for z as they have two immediate advantages: a) continuous variables are not binarized in our solver and b) the LP-relaxation immediately yields the optimal assignment for z_s after q^s and x^s are assigned.

In order to build an equivalent QIP one does not need to use the presented reduction function. Instead of using universal indicator variables $q^s \in \mathcal{D}_s$ the universal player can use a single integer variable $\ell_s \in \{1, \dots, N\}$ in order to select one of N scenarios in each iteration. This integer can then be transformed into existential indicator variables:

Problem SELQ

$$\min \sum_{i=1}^{n} c_i^0 x_i^0 + \sum_{s=1}^{S} z_s \tag{4.46}$$

$$\text{s.t. } \exists x^0 \in \{0,1\}^n \quad \forall \ell_1 \in \{1, \dots, N\} \quad \exists q^1 \in \{0,1\}^N \quad \exists x^1 \in \{0,1\}^n \ \cdots$$

$$\cdots \ \forall \ell_S \in \{1, \dots, N\} \quad \exists q^S \in \{0,1\}^N \quad \exists x^S \in \{0,1\}^n \ \exists z \in \mathbb{N}_0^S:$$

$$\sum_{i=1}^{n} \sum_{s=0}^{S} x_i^s = p \tag{4.47}$$

$$\sum_{s=0}^{S} x_i^s \leq 1 \qquad \forall\, i \in \{1, \dots, n\} \tag{4.48}$$

$$\sum_{i=1}^{n} c_{i,k}^s x_i^s \leq z_s + M(1 - q_k^s) \qquad \forall\, k \in \{1, \dots, N\},\ s \in \{1, \dots, S\} \tag{4.49}$$

$$\sum_{k=1}^{N} q_k^s = 1 \qquad \forall\, s \in \{1, \dots, S\} \tag{4.50}$$

$$\sum_{k=1}^{N} k \cdot q_k^s = \ell_s \qquad \forall\, s \in \{1, \dots, S\} \tag{4.51}$$

The variables q^s, which were universal variables in SELQ$^{\text{PU}}$, are now used as existential variables indicating the selected scenario. Constraints (4.50) and (4.51) ensure that the selected scenario number ℓ_s is transformed correctly into a corresponding assignment of q^s. Thus, the number of variables and constraints in SELQ increased compared to SELQ$^{\text{PU}}$.

We also want to provide a robust counterpart, i.e. an equivalent MIP, for which each possible scenario sequence must be listed explicitly. The set containing all possible sequences of scenarios is $R = \{1, \dots, N\}^S$. For one such sequence $r \in R$ and an iteration $s \in \{1, \dots, S\}$ the scenario in iteration s is denoted by $r[s]$. The entire sub-sequence up to iteration s is denoted by $sub(r, s) \in \{1, \dots, N\}^s$. For each item $i \in \{1, \dots, n\}$, iteration $s \in \{1, \dots, S\}$ and sequence $r \in R$, the variable $x_i^{sub(r,s)}$ indicates the decision of selecting item i in iteration s after the subsequence $sub(r, s)$ of r occurred.

Example 4.4.1. *For $N = 4$ and $S = 6$ a possible sequence of scenarios is $r = (1, 4, 2, 3, 1, 1)$. The sub-sequence until iteration $s = 4$ is $sub(r, 4) = (1, 4, 2, 3)$. The variable indicating whether item i is selected after 4 iterations and the occurrence of this particular sub-sequence is denoted $x_i^{sub(r,4)} = x_i^{(1,4,2,3)}$ and the scenario in iteration 4 for this sequence is $r[4] = 3$. The cost of item i in iteration $s = 4$ does not depend on the entire sequence, but only on the occured scenario and is given by $c_{i,r[s]}^s = c_{i,3}^4$. For the sequence of scenarios $\hat{r} = (1, 4, 2, 3, 2, 4)$ it is $sub(r, 4) = sub(\hat{r}, 4)$*

and therefore, the variables $x_i^{sub(r,4)}$ *and* $x_i^{sub(\hat{r},4)}$ *are the same. With the DEP of a QIP in mind, this ensures the nonanticipativity property (cf. Constraint (3.6)): even for different scenario sequences* r *and* $\hat{r}$ *the selection decisions must be the same, as long as the subsequences are identical.*

The robust counterpart of SELQ$^{\text{PU}}$ is called SELRC and is given as follows.

Problem SELRC

$$\min \sum_{i=1}^{n} c_i^0 x_i^0 + z \tag{4.52}$$

$$\text{s.t. } z \geq \sum_{i=1}^{n} \sum_{s=1}^{S} c_{i,r[s]}^{s} x_i^{sub(r,s)} \qquad \forall\, r \in R \tag{4.53}$$

$$\sum_{i=1}^{n} \left(x_i^0 + \sum_{s=1}^{S} x_i^{sub(r,s)} \right) = p \qquad \forall\, r \in R \tag{4.54}$$

$$x_i^0 + \sum_{s=1}^{S} x_i^{sub(r,s)} \leq 1 \qquad \forall\, i \in \{1, \ldots, n\},\ r \in R \tag{4.55}$$

$$x_i^0 \in \{0,1\} \qquad \forall\, i \in \{1, \ldots, n\} \tag{4.56}$$

$$x_i^{sub(r,s)} \in \{0,1\} \qquad \forall\, i \in \{1, \ldots, n\},\ r \in R,\ s \in \{1, \ldots, S\} \tag{4.57}$$

$$z \in \mathbb{R} \tag{4.58}$$

Constraint (4.53) now must ensure that the expenses from the worst-case scenario sequence appear in the objective function. Constraint (4.54) ensures for each scenario sequence that exactly p items are selected in the end, whereas Constraint (4.55) ensures that each item is selected at most once. We use this compact robust counterpart for a comparison between our solver and the general MIP solver CPLEX for which the results are presented in Subsection 7.3.2. We already refer to Appendix B.2 where statistics regarding sizes of selected instances of the three models SELQ$^{\text{PU}}$, SELQ and SELRC for various n, p, S and N are presented.

When it comes to such multistage problems under uncertainty, the question arises to what extent their solution is superior to applying heuristics and whether there are simple online decision strategies that come close to the optimal solution. Therefore, we present three online decision strategies in order to be able to grasp the relevance of the optimization model.

Strategy 1: Buy All Now The easiest strategy is to neglect any knowledge of future events and buy the p cheapest items right away. The resulting costs of this strategy in terms of the presented models are

$$\min \left\{ \sum_{i=1}^{n} c_i^0 x_i^0 \mid \sum_{i=1}^{n} x_i^0 = p \right\}.$$

Since knowledge of future iterations and scenarios is not taken into account this trivial strategy almost always leads to significantly worse results.

Strategy 2: Buy Now, If Never Cheaper in Worst Case In this decision strategy partial knowledge of future scenarios is incorporated: Items are sorted according to their lowest guaranteed costs incurred in the current or in future iterations. Starting with the cheapest, an item is bought if its best price is the current price. Let P be the set of already bought items. Let $s \in \{0, \dots, S\}$ be the current iteration and $k \in \{1, \dots, N\}$ the current scenario (if $s > 0$). In such a situation we propose to look at the $p - |P|$ cheapest items according to their best worst-case price and buy such an item now, if there is no future iteration in which this item is cheaper *in the worst case* as shown in Algorithm 2. By applying this strategy, obviously detrimental

Algorithm 2: Selection strategy 2: "Buy Now, If Never Cheaper in Worst Case".

Input: target p, current iteration s, current scenario k, costs c, set of bought items P

for each item $i \in \{1, \dots, n\}$ **do**
  $b_i = c^s_{i,k}$ // `best price of item` i `initialized to current price`
  $d_i =$"Buy Now" // `initialize decision for item` i
  if $i \in P$ **then**
    $b_i = \infty$
    $d_i =$"Already bought"
    continue loop with $i = i + 1$
  end if
  for each future iteration $\bar{s} > s$ **do**
    if $b_i \geq \max_{1 \leq \bar{k} \leq N} c^{\bar{s}}_{i,\bar{k}}$ **then**
      $b_i = \max_{1 \leq \bar{k} \leq N} c^{\bar{s}}_{i,\bar{k}}$
      $d_i =$"Buy Later"
    end if
  end for
end for
Sort b and d in ascending order according to the values in b
for $i = 1$ to $p - |P|$ **do** // `the cheapest items according to` b
  if $d_i =$"Buy Now" **then**
    $P = P \cup \{i\}$
  end if
end for

purchases are prevented, i.e. if it is guaranteed that the same item is available later for a cheaper price. However, other items are not considered.

Strategy 3: Don't Buy, If Others Will Be Cheaper Similar to Strategy 2 a buying decision depends on the worst-case costs in future scenarios, but incorporates prices of all remaining items. Let $\min(a, b)$ be the function returning the value of the b-smallest element of vector a. Let $a(R) \in \mathbb{Q}^{|R|}$ be the entries of vector $a \in \mathbb{Q}^n$ corresponding to indexes as in set $R \subseteq \{1, \dots, n\}$. Let again $s \in \{0, \dots, S\}$ be the current iteration, $k \in \{1, \dots, N\}$ the current scenario (if $s > 0$) and P the set of already bought items. In such a situation we propose to look at the $p - |P|$ cheapest items according to $c^s_{\star,k}$ and buy the cheapest item as long as no future iteration is found in which $p - |P|$ items are cheaper, even in the worst case. In Algorithm 3 this strategy is presented. This way it is prevented that by buying an item now an obviously cheaper selection

Algorithm 3: Selection strategy 3: "Don't Buy, If Others Will Be Cheaper".

Input: target p, current iteration s, current scenario k, costs c, set of bought items P

Remove the items in P from each cost vector
Resort the not yet selected items according to the current costs $c^s_{\star,k}$
for $i = 1$ to $p - |P|$ **do** // the $p - |P|$ cheapest items according to $c^s_{\star,k}$
  $R = \{1, \dots, n\} \setminus P$
  if $c^s_{i,k} < \min_{s<\bar{s}\leq S} \max_{1\leq\bar{k}\leq N} \min(c^s_{\star,k}(R), p - |P|)$ **then**
    $P = P \cup \{i\}$
  else
    return
  end if
end for

in the future is no longer possible. In particular, if only a single item remains to be bought it is checked whether there is a guaranteed cheaper item in a later iteration.

Example 4.4.2. *Let $n = 6$, $p = 3$, $S = 2$ and $N = 2$. The costs in the initial stage and each scenario is given in Table 4.1.*

Table 4.1.: Cost scenarios for an instance of the multistage selection problem.

i	1	2	3	4	5	6
c^0_i	84	14	76	61	31	45
$c^1_{i,1}$	40	24	29	41	90	71
$c^1_{i,2}$	45	30	15	18	44	44
$c^2_{i,1}$	13	25	12	11	75	50
$c^2_{i,2}$	80	10	29	32	64	30

a) *Strategy 1: "Buy All Now"*
Buying the three cheapest items in the first stage yields costs of 90.

b) *Strategy 2: "Buy Now, If Never Cheaper in Worst Case"*
In the first decision stage the vector b—containing the best worst-case costs of each item—is filled with the values $(45, 14, 29, 32, 31, 45)$. *The three smallest values are examined resulting in the decision of buying items* 2 *and* 5 *now. Item* 3 *is not bought in this iteration, as a better price is guaranteed later on. If in iteration* 1 *scenario* 1 *occurs, the vector b holds the values* $(40, \infty, 29, 32, \infty, 50)$. *Since only one item must be bought to reach $p = 3$ only the cheapest item is considered, but again item* 3 *is not bought, as later on the same price is ensured. If scenario* 2 *occurs in iteration* 1, *item* 3 *would be bought. The overall worst-case costs when this strategy is applied is* 74.

c) *Strategy 3: "Don't Buy, If Others Will Be Cheaper"*
In the first decision stage items 2, 5 *and* 6 *are considered. Item* 2 *costs* 14. *Note that if the first scenario of iteration* 2 *occurs, buying item* 2 *right away would be bad as there would*

be three cheaper items. However, this is the best cast scenario. Hence, the question is whether buying item 2 *now eliminates the option of buying three cheaper items in a single future iteration, even in the worst case. Therefore, the worst-case third cheapest cost of each iteration is calculated, which are* 40 *in iteration* 1 *and* 30 *in iteration* 2*. Since both values are larger than* 14 *item* 2 *is bought in the first stage. For item* 5 *this procedure is repeated, but now the second cheapest of the remaining items are considered. Those are* 40 *in iteration* 1 *(since item* 2 *is excluded) and* 30 *in iteration* 2*. Therefore, item* 5 *is not bought, as it is ensured, that in another future scenario two cheaper items exist. If in iteration* 1 *scenario* 1 *occurs, items* 3 *and* 1 *are considered and only item* 3 *is bought. If scenario* 2 *occurs in iteration* 1*, both items* 3 *and* 4 *are bought. The overall worst-case costs when this strategy is applied is* 73*.*

In the optimal strategy no item is bought in the first stage and the overall worst-case costs are 69*. The explicit strategies of the three heuristics, as well as the optimal solution can be found in Appendix A.2 on page 172.*

Even for such a small example the optimal value is less than the worst-case outcome after applying the presented heuristic strategies. A general comparison between optimal solutions of the (robust) multistage selection problem and the results of the presented heuristics is given in Subsection 7.3.2 on pages 159ff. and box plots regarding the relative deviation from the optimal value of the two more sophisticated heuristics 2 and 3 are presented in Appendix B.4.

4.4.5. Multistage Assignment Problem

Another frequently consulted combinatorial problem is the assignment problem: Given a complete bipartite graph $G = (V, E)$ with $V = A \cup B$, $n = |A| = |B|$. For each edge $(i, j) \in E$ it is $i \in A \Leftrightarrow j \in B$ and a cost value $c_{i,j} \in \mathbb{N}_0$ is associated with each edge. The assignment problem consists of determining a perfect matching of minimum total costs. Research on the min-max and min-max regret assignment problems can be found in [ABV05] and further complexity results are obtained in [DW+06]. We want to extend the already investigated robust approach to a multistage problem. In a first decision stage the existential player can select edges with known costs. Then, iteratively, new costs of the edges are presented (by the universal player) which then can be selected (by the existential player). Similar to the preceding subsection, the costs selected by the universal player come from a predefined scenario pool. Let N be the number of scenarios and S the number of iterations. We use the universal variable q_k^s to indicate whether cost scenario k is selected in iteration s. As only one scenario can occur at each iteration the universal constraint $\sum_{k=1}^{N} q_k^s = 1$ must be fulfilled and thus at each iteration $s \in \{1, \ldots, S\}$ the universal variables have to obey the domain $\mathcal{D}_s = \{q^s \in \{0,1\}^N \mid \sum_{k=1}^{N} q_k^s = 1\}$. The cost for edge $(i, j) \in E$ in scenario k in iteration s is given by $c_{i,j,k}^s \in \mathbb{N}_0$. The objective remains minimizing the costs for a perfect matching and once again auxiliary variables z_s are used to

bundle the costs incurred in iteration s and to avoid a nonlinear term in the objective function. The QIP$^{\text{PU}}$ model for this multistage assigment problem (ASSQ$^{\text{PU}}$) is given below.

Problem ASSQ$^{\text{PU}}$

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} c^0_{i,j}x^0_{i,j} + \sum_{s=1}^{S} z_s \tag{4.59}$$

$$\text{s.t. } \exists x^0 \in \{0,1\}^{n\times n} \quad \forall q^1 \in \mathcal{D}_1 \quad \exists x^1 \in \{0,1\}^{n\times n} \quad \cdots \quad \forall q^S \in \mathcal{D}_S \quad \exists x^S \in \{0,1\}^{n\times n}$$
$$\exists z \in \mathbb{N}_0^S :$$

$$\sum_{j=1}^{n}\sum_{s=0}^{S} x^s_{i,j} = 1 \qquad \forall\, i \in \{1,\dots,n\} \tag{4.60}$$

$$\sum_{i=1}^{n}\sum_{s=0}^{S} x^s_{i,j} = 1 \qquad \forall\, j \in \{1,\dots,n\} \tag{4.61}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c^s_{i,j,k}x^s_{i,j} \le z_s + M(1-q^s_k) \qquad \forall\, k \in \{1,\dots,N\},\ s \in \{1,\dots,S\} \tag{4.62}$$

The Objective Function (4.59) consists of the expenses from the first stage with fixed costs and each iteration with uncertain costs. Constraints (4.60) and (4.61) ensure that the found solution is indeed a perfect matching. Constraint (4.62) linearizes the dependence between selected scenario and incurred costs. Similar as we did it for the multistage selection problem, in order to build an equivalent QIP we represent the universal player's decision as an integer variable $\ell_s \in \{1,\dots,N\}$ and then convert it into existential indicator variables $q^s \in \{0,1\}^N$:

Problem ASSQ

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} c^0_{i,j}x^0_{i,j} + \sum_{s=1}^{S} z_s \tag{4.63}$$

$$\text{s.t. } \exists x^0 \in \{0,1\}^{n\times n} \quad \forall \ell_1 \in \{1,\dots,N\} \quad \exists q^1 \in \{0,1\}^N \quad \exists x^1 \in \{0,1\}^{n\times n} \quad \cdots$$
$$\cdots \quad \forall \ell_S \in \{1,\dots,N\} \quad \exists q^S \in \{0,1\}^N \quad \exists x^S \in \{0,1\}^{n\times n}\ \exists z \in \mathbb{N}_0^S:$$

$$\sum_{j=1}^{n}\sum_{s=0}^{S} x^s_{i,j} = 1 \qquad \forall\, i \in \{1,\dots,n\} \tag{4.64}$$

$$\sum_{i=1}^{n}\sum_{s=0}^{S} x^s_{i,j} = 1 \qquad \forall\, j \in \{1,\dots,n\} \tag{4.65}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c^s_{i,j,k}x^s_{i,j} \le z_s + M(1-q^s_k) \qquad \forall\, k \in \{1,\dots,N\},\ s \in \{1,\dots,S\} \tag{4.66}$$

$$\sum_{k=1}^{N} q^s_k = 1 \qquad \forall\, s \in \{1,\dots,S\} \tag{4.67}$$

$$\sum_{k=1}^{N} k \cdot q^s_k = \ell_s \qquad \forall\, s \in \{1,\dots,S\} \tag{4.68}$$

Regarding the positioning and the domain of the variables z_s, we refer to the discussion on page 66 in context of the multistage selection problem.

Again, we are interested in a robust counterpart that can be solved using standard MIP solvers. Similar to the notation used in the previous subsection, R denotes the set of all possible sequences of scenarios and $sub(r,s)$ denotes the sub-sequence of scenario sequence r up to iteration s (see Example 4.4.1). The variable $x_{i,j}^{sub(r,s)}$ indicates the decision of selecting edge (i,j) in iteration s after the subsequence $sub(r,s)$ of r occurred. The robust counterpart of ASSQ$^{\text{PU}}$ is referred to as ASSRC and is given below.

Problem ASSRC

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{i,j}^{0} x_{i,j}^{0} + z \tag{4.69}$$

$$\text{s.t. } z \geq \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{s=1}^{S} c_{i,j,r[s]}^{s} x_{i,j}^{sub(r,s)} \qquad \forall\, r \in R \tag{4.70}$$

$$\sum_{j=1}^{n}\sum_{s=0}^{S} x_{i,j}^{sub(r,s)} = 1 \qquad \forall\, i \in \{1,\dots,n\},\ r \in R \tag{4.71}$$

$$\sum_{j=1}^{n}\sum_{s=0}^{S} x_{i,j}^{sub(r,s)} = 1 \qquad \forall\, j \in \{1,\dots,n\},\ r \in R \tag{4.72}$$

$$x_{i.j}^{0} \in \{0,1\} \qquad \forall\, i,\ j \in \{1,\dots,n\} \tag{4.73}$$

$$x_{i,j}^{sub(r,s)} \in \{0,1\} \qquad \forall\, i,\ j \in \{1,\dots,n\},\ r \in R,\ s \in \{1,\dots,S\} \tag{4.74}$$

$$z \in \mathbb{R} \tag{4.75}$$

In Subsection 7.3.3 we compare the performance of the general MIP solver CPLEX on the ASSRC model with our solver on the ASSQ$^{\text{PU}}$ and ASSQ models.

5. Quantified Integer Programming with Interdependent Domains

5.1. Motivation

A QIP is inherently asymmetric, as even though the min-max semantic of the objective is symmetric, the universally quantified variables are only restricted to the domain $\mathcal{L}$, whereas the existential player—in addition to having to obey the variable bounds—also must ensure the fulfillment of the constraint system $A^{\exists}x \leq b^{\exists}$. In other words: only the existential player has to cope with a polytope influenced by the opponent's decisions whereby an interdependence between existential and universal decisions can only be represented in one direction through $A^{\exists}x \leq b^{\exists}$. The previously presented QIP$^{\mathrm{PU}}$ does not change this, as only the hypercube domain $\mathcal{L}$ is replaced by a polyhedral domain $\mathcal{D}$, where an existential influence on the universal domain is explicitly prohibited. Thus, in either setting it is difficult to model most two-person games as moves by either player almost always depend on previous own and opponent decisions. But also from the viewpoint of operations research this asymmetry does not always reflect reality. For example consider the task of finding an optimal machine scheduling. A robust optimization problem arises when the possibility of machine failures is also taken into account, in which case the modeler must cautiously specify the number (or set) of machines that may fail, e.g. by using QIP$^{\mathrm{PU}}$ or a budgeted uncertainty set in standard robust optimization. By also taking maintenance into account an advanced optimization approach is needed, which allows an influence of planning decisions in earlier stages on possible uncertain events later on: machine maintenance will prevent machine failure for a certain amount of time. Such interdependence of planning decisions and uncertain events is rarely dealt with in the literature and has only recently received more attention in the field of robust optimization (see page 20).

In the following section, a novel extension for QIP is introduced and investigated, which allows the modeling of decision-dependent uncertainty: similar to the QIP$^{\mathrm{PU}}$ presented in Chapter 4, a universal constraint system $A^{\forall}x \leq b^{\forall}$ is added to the problem statement, with the essential difference that existential variables can now occur in the universal constraint system, i.e. existential player's decisions can alter the polytope the universal player must comply with. Note that QIP$^{\mathrm{PU}}$ is a subproblem of this extension and therefore the results regarding its algorithmic properties are also valid for QIP$^{\mathrm{PU}}$.

5.2. Problem Statement QIP$^{\mathsf{ID}}$

We consider a second constraint system $A^{\forall}x \leq b^{\forall}$, $A^{\forall} \in \mathbb{Q}^{m_{\forall} \times n}$ and $b^{\forall} \in \mathbb{Q}^{m_{\forall}}$, $m_{\forall} \in \mathbb{N}$, the universal player must satisfy. In contrast to the QIP$^{\mathrm{PU}}$ presented in Chapter 4 we no longer restrict the universal constraint system to be only dependent on universal variables (cf. Condition (4.1)). Thus, in contrast to the QIP$^{\mathrm{PU}}$, the universal player does not necessarily have a strategy in order to fulfill her system. Therefore, situations where a completely assigned variable vector $x \in \mathcal{L}$ does not fulfill either system, i.e. $A^{\exists}x \not\leq b^{\exists}$ and $A^{\forall}x \not\leq b^{\forall}$, must be dealt with explicitly. We conduct the following preliminary considerations in order to adequately deal with such situations:

1. The superordinate goal of each player is to fulfill their own constraint system. In particular, one player should not be allowed to make her own system unfulfillable in order to violate the opponent's system.
2. The subordinate goal for both players remains trying to optimize the objective function: the existential player is trying to minimize and the universal player is trying to maximize the objective value.
3. If $A^{\forall}x \not\leq b^{\forall}$ and $A^{\exists}x \leq b^{\exists}$ for a play (a filled variable vector $x \in \mathcal{L}$) the payoff is $-\infty$, i.e. the universal player "loses".
4. If $A^{\forall}x \leq b^{\forall}$ and $A^{\exists}x \not\leq b^{\exists}$ for a play the payoff is $+\infty$, i.e. the existential player "loses".
5. If $A^{\forall}x \leq b^{\forall}$ and $A^{\exists}x \leq b^{\exists}$ for a play the payoff is $c^{\top}x$.
6. If $A^{\forall}x \not\leq b^{\forall}$ and $A^{\exists}x \not\leq b^{\exists}$ for a play the player whose system became unfulfillable first loses.

Hence, our intention is that an *illegal move* (making one's own system unfulfillable) immediately results in a loss for this player. However, this remains rather vague and in particular the term "unfulfillable" must be specified. The following characterizations come into mind:

a) A constraint system is unfulfillable, if for a partially filled variable vector no single assignment of the remaining variables exists such that the system is met.

b) A constraint system is unfulfillable, if for a partially filled variable vector no strategy for the assignment of the remaining variables exists such that the system is met.

Even though they capture core ideas of "unfulfillable", these two characterizations still remain imprecise. We pursue a) and elaborate a formal definition. But note that we also considered b) in order to define legal moves: it turned out that the resulting problems are essentially identical (see Remark 5.3.20).

For the following considerations assume that the first $k-1$ variables are already (legally) assigned and assume that $Q_k = \forall$, i.e. the next move—assigning x_k—is a move by the universal player. According to the above considerations and a), this next move by the universal player should be legal if afterwards there still is an assignment of the remaining variables such that the universal constraint system can be fulfilled. Obviously, such an assignment should adhere

to the global variable domain $\mathcal{L}$. But the question arises, whether such an assignment of future variables must itself consist of legal moves (by both players). However, the existence of a legal counter move for the existential player should not be of interest when determining the set of legal assignments of the universal variable x_k. In particular, a move by the universal player should not be made illegal by the fact that the existential player has no legal counter move. Thus, only the future universal variables should be required to correspond to legal moves, whereas the existential variables only must obey $\mathcal{L}$. Such an implicit definition, however, is also not necessary since any assignment of future variables fulfilling $A^{\forall}x \leq b^{\forall}$ must consequently consist of legal universal assignments. Hence, it is reasonable that the assignment of future variables must only obey $\mathcal{L}$ and we therefore define a legal variable assignments as follows.

Definition 5.2.1 ($\mathcal{F}$ - Legal Assignment of Variable x_k)**.**
The set of legal assignments $\mathcal{F}(\tilde{x}_1, \dots, \tilde{x}_{k-1}) \subseteq \mathcal{L}_k$ *of variable* x_k *depends on the assignment of previous variables* $\tilde{x}_1, \dots, \tilde{x}_{k-1}$ *and is given by*

$$\mathcal{F}(\tilde{x}_1, \dots, \tilde{x}_{k-1}) = \left\{ \hat{x}_i \in \mathcal{L}_k \mid \exists x = (\tilde{x}_1, \dots, \tilde{x}_{k-1}, \hat{x}_k, x_{k+1}, \dots, x_n) \in \mathcal{L} : \; A^{Q_k} x \leq b^{Q_k} \right\}.$$

Therefore, after assigning variable x_k *there still must exist an assignment of the remaining variables according to their bounds such that the system of the responsible player* $Q_k \in \{\exists, \forall\}$ *is fulfilled. The dependence on the previous variables* $\tilde{x}_1, \dots, \tilde{x}_{k-1}$ *is omitted when clear.*

With this definition the legal domain for a variable depends on previous decisions, and moves that eliminate any chance of fulfilling one's own constraint system are explicitly excluded. Similar to the range $\mathcal{L}^{(i)}$ of the i-th variable block we define the set of legal variable assignments of variable block i.

Definition 5.2.2 ($\mathcal{F}^{(i)}$ - Legal Assignment of Variable Block i)**.**
The set of legal assignments of variable block i *(dependent on the assignment of previous variable blocks)* $\mathcal{F}^{(i)}(\tilde{x}^{(1)}, \dots, \tilde{x}^{(i-1)})$ *is given by*

$$\mathcal{F}^{(i)} = \left\{ \hat{x}^{(i)} \in \mathcal{L}^{(i)} \mid \exists x = (\tilde{x}^{(1)}, \dots, \tilde{x}^{(i-1)}, \hat{x}^{(i)}, x^{(i+1)}, \dots, x^{(\beta)}) \in \mathcal{L} : \; A^{Q^{(i)}} x \leq b^{Q^{(i)}} \right\}$$

i.e. after assigning the variables of block i *there still must exist an assignment of* x *such that the system of* $Q^{(i)} \in \{\exists, \forall\}$ *is fulfilled. The dependence on the previous variable blocks* $\tilde{x}^{(1)}, \dots, \tilde{x}^{(i-1)}$ *is omitted when clear.*

In particular, $\mathcal{F}^{(i)} = \emptyset$ means that there is no move such that the own constraint system still can be satisfied, which is interpreted as a loss for the player in turn.

Definition 5.2.3 (Quantified Integer Program with Interdependent Domain)**.**
Let $A^{\exists} \in \mathbb{Q}^{m_{\exists} \times n}$ *and* $b^{\exists} \in \mathbb{Q}^{m_{\exists}}$ *for* $m_{\exists} \in \mathbb{N}$. *Let* $\mathcal{L}$ *and* Q *be given as in Section 2.1 with* $\mathcal{I} = \{1, \dots, n\}$. *We further demand* $\beta \geq 2$, *i.e. there are at least two variable blocks, with* $Q^{(1)} = \exists$ *and* $Q^{(\beta)} = \forall$. *Let* $c \in \mathbb{Q}^n$ *be the vector of objective coefficients, for which* $c^{(i)}$ *denotes*

the vector of coefficients belonging to variable block B_i. *Let* $b^\forall \in \mathbb{Q}^{m_\forall}$ *and* $A^\forall \in \mathbb{Q}^{m_\forall \times n}$ *for* $m_\forall \in \mathbb{N}$ *with*

$$\left\{x \in \mathcal{L} \mid A^\forall x \leq b^\forall\right\} \neq \emptyset\,. \tag{5.1}$$

We call

$$z = \min_{x^{(1)} \in \mathcal{F}^{(1)}} \left(c^{(1)}x^{(1)} + \max_{x^{(2)} \in \mathcal{F}^{(2)}} \left(c^{(2)}x^{(2)} + \min_{x^{(3)} \in \mathcal{F}^{(3)}} \left(c^{(3)}x^{(3)} + \dots \max_{x^{(\beta)} \in \mathcal{F}^{(\beta)}} c^{(\beta)}x^{(\beta)}\right)\right)\right) \tag{5.2}$$

$$\text{s.t. } \exists\, x^{(1)} \in \mathcal{F}^{(1)}\ \forall\, x^{(2)} \in \mathcal{F}^{(2)}\ \exists\, x^{(3)} \in \mathcal{F}^{(3)} \dots\ \forall\, x^{(\beta)} \in \mathcal{F}^{(\beta)} :\ A^\exists x \leq b^\exists \tag{5.3}$$

a QIP with interdependent domains *(QIP^{ID}) given by the tuple* $(A^\exists, A^\forall, b^\exists, b^\forall, c, \mathcal{L}, Q)$.

Note that if there is no legal variable assignment for an existential variable block, the objective value is $+\infty$, whereas $\mathcal{F}^{(i)} = \emptyset$ for $Q^{(i)} = \forall$ results in the objective value $-\infty$ (see Definition 4.2.3 and the subsequent discussion). Hence, the case of both systems being violated is bypassed: if both systems are not satisfied there must have been an illegal variable assignment at some earlier point.

Definition 5.2.3 contains three restrictive conditions:

a) $\left\{x \in \mathcal{L} \mid A^\forall x \leq b^\forall\right\} \neq \emptyset$

b) $Q^{(1)} = \exists$ and $Q^{(\beta)} = \forall$

c) $\beta \geq 2$

We demonstrate that these restrictions are necessary to avoid undesirable properties and at the same time are not too restrictive. As c) follows from b) we only consider b). The first demand is related to the restriction $\mathcal{D} \neq \emptyset$ for QIP$^{\text{PU}}$. However, in contrast to the QIP$^{\text{PU}}$, this restriction does not automatically result in the existence of a strategy for the universal player to fulfill $A^\forall x \leq b^\forall$, as now the existential player's decisions affect the universal constraint system. Further, since now the existential player's variable assignments also must be legal, the situation described in Example 4.2.5, where the existential player wins while violating his system, no longer poses a problem. However, another undesirable effect occurs if $\{x \in \mathcal{L} \mid A^\forall x \leq b^\forall\} = \emptyset$: the optimal value can change if a dummy variable—a variable with only zero-entries in $A^\exists$, $A^\forall$ and c—is added up front (see Example 5.2.4). Such behavior in itself is undesirable, but it becomes more drastic with respect to b): If adding dummy variables up front can change the result of an instance, the demanded property $Q^{(1)} = \exists$ is not always attainable without changing the problem itself. Hence, $\{x \in \mathcal{L} \mid A^\forall x \leq b^\forall\} \neq \emptyset$ is added to the definition and the following simple arguments even show that this demand is consistent:

- Assume the QIP$^{\text{ID}}$ describes not an entire game, but a certain game situation. With $Q^{(1)} = \exists$ it is the existential player's turn. Then $\{x \in \mathcal{L} \mid A^\forall x \leq b^\forall\} = \emptyset$ would imply that the previous universal move was illegal and hence, such a game situation cannot be reached via legal play.
- From the OR perspective the universal player often describes events that are by nature uncertain. Therefore, even when the existential player is able to restrict possible universal

actions, the planner cannot (and probably should not) "defeat" uncertainty by taking away all options. In particular, it makes little sense to allow uncertainty to be "defeated" from the outset.

- If we allow both constraint systems to have no solution from the start, then the outcome of such instance only depends on who the starting player is (see Example 5.2.4). This, however constitutes neither a meaningful game, nor a reasonable optimization task.

Example 5.2.4. *Let there be three binary variables, $\mathcal{L} = \{0,1\}^3$ and $Q = (\forall, \exists, \forall)$. Let $c = 0$ and let the two constraint systems be given as follows*

$$A^{\exists}x \leq b^{\exists} \qquad\qquad A^{\forall}x \leq b^{\forall}$$

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & -1 & -1 \\ -1 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \leq \begin{pmatrix} 1 \\ -2 \\ -2 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 1 \\ -1 & 0 & -1 \\ -1 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \leq \begin{pmatrix} 1 \\ -2 \\ -2 \end{pmatrix}$$

Both constraint systems have no solution. Therefore, $\mathcal{F}^{(1)} = \emptyset$ and the instance constitutes a loss for the starting universal player as she has no legal move in the first place. If an existential dummy variable is added upfront—in order to bring this instance into the demanded form with $Q^{(1)} = \exists$—suddenly the instance constitutes a loss for the existential player as no legal move for the dummy variable exists.

This minimal example illustrates that excluding the special case—with both constraint systems being a priori unfulfillable—is important for the well-defined character of QIP$^{\mathrm{ID}}$. Note that demanding $\{x \in \mathcal{L} \mid A^{\forall}x \leq b^{\forall}\} \neq \emptyset$ is also in line with robust optimization, where an empty uncertainty set is usally neglected (e.g. [BTN98]).

The restriction $Q^{(1)} = \exists$ in b), simplifies forthcoming definitions and proofs, without losing generality: by adding an existential dummy variable upfront in order to ensure $Q^{(1)} = \exists$ the result of the QIP$^{\mathrm{ID}}$ remains unchanged (if $\{x \in \mathcal{L} \mid A^{\forall}x \leq b^{\forall}\} \neq \emptyset$). In addition, consistency with the QIP is established, which also requires a first existential variable block.

On the other hand, the restriction $Q^{(\beta)} = \forall$ is more crucial and its necessity is of technical nature. After a play of a QIP$^{\mathrm{ID}}$ instance, the fulfillment of $A^{\exists}x \leq b^{\exists}$ for the fixed variable vector $x \in \mathcal{L}$ must automatically be checked (due to Condition (5.3)). The compliance with the universal constraint system, however is only checked during variable assignments by the universal player herself. In particular, if the existential player has the final move, a legal assignment of the corresponding existential variables could result in a violation of the universal constraint system. This, however, would remain undetected both in the Objective (5.2) as well as Condition (5.3), resulting in the objective value $c^{\top}x$ rather than $-\infty$. In this situation, adding a universal dummy variable at the end has an important effect: If the universal constraint system is fulfilled up to this point the corresponding legal variable domain $\mathcal{F}^{(\beta)} = \mathcal{L}^{(\beta)} \neq \emptyset$, i.e. the dummy variable can take any value. If, however, the final (legal!) existential variable assignment results in a violation of $A^{\forall}x \leq b^{\forall}$ it is $\mathcal{F}^{(\beta)} = \emptyset$, resulting in a loss for the universal player and the payoff $-\infty$. Hence, adding a universal variable at the end—such that, $Q^{(\beta)} = \forall$—ensures the detection of late game losses for the universal player.

Remark 5.2.5 (Connection between QIP^{ID} and QII (cf. Remark 4.2.6))**.**
The general QII (see Definition 2.2.1) is also asymmetric, but in the other direction as the original QIP: if the left-hand side system $Bx \leq d$ is not satisfied the statement is immediately true, regardless of the right-hand side system. Hence, the existential player in a QII aims at violating $Bx \leq d$ without having to obey "his own" right-hand side constraint system. Hence, a QIP^{ID} allows a more immediate description of an interdependence.

5.3. Use of Game Trees for QIP^{ID}

Similar to QIPs and QIP^{PU} we want to use game trees in order to describe and solve QIP^{ID}. Again, we can refer to Definition 2.1.10 for a valid definition of a game tree for any QIP^{ID} instance. This game tree, however, also contains illegal moves according to Definition 5.2.1. Therefore, both the term strategy and minimax value must be adjusted in order to describe and find solutions of this problem type, as a play of a QIP^{ID} might end before all variables are assigned, if a player has no legal moves left ($\mathcal{F}^{(i)} = \emptyset$). Furthermore, strategies for QIP^{ID} must not consider all possible moves from $\mathcal{L}^{(i)}$ but only all legal moves $\mathcal{F}^{(i)}$. For sake of simplicity we transfer the set of legal moves into the game tree terminology such that for each node the set of legal successors is formally defined.

Definition 5.3.1 ($\mathcal{F}(v)$ - Legal Successors of Node v)**.**
Let $G = (V, E, e)$ be the game tree of a QIP^{ID} and let $v \in V \setminus V_L$, i.e. level$(v) = k < n$. Let $\tilde{x}_v = (\tilde{x}_1, \ldots, \tilde{x}_k)$ be the partial variable assignment along the path from the root to v. The set of legal successors *of v is given by*

$$\mathcal{F}(v) = \{\hat{v} \in V \mid (v, \hat{v}) \in E \ \wedge \ e((v, \hat{v})) \in \mathcal{F}(\tilde{x}_1, \ldots, \tilde{x}_k)\} \ .$$

In a legal play of a QIP^{ID} only edges to legal successors can be used. In particular, the game tree contains subtrees that can never be reached during legal play. Therefore, a strategy must not contain such subtrees, which is why they are truncated.

Definition 5.3.2 (Truncated Existential Strategy)**.**
A truncated strategy *for the assignment of existential variables, is a subtree $T = (V', E', e')$ of a game tree $G = (V, E, e)$ of a QIP^{ID}. V' contains the unique root node $r \in V_\exists$. Each node $v_\exists \in V' \cap V_\exists$ has either a) exactly one child if and only if the corresponding legal domain is not empty, i.e. $\mathcal{F}(v_\exists) \neq \emptyset$, or b) no child if and only if $\mathcal{F}(v_\exists) = \emptyset$. Each node $v_\forall \in V' \cap V_\forall$ has all the children as in G for which at least one leaf in their corresponding sub-tree in G exists with $A^\forall x \leq b^\forall$, i.e. as many as there are in the legal domain $\mathcal{F}(v_\forall)$. Thus, each edge in T must represent a legal variable assignment according to Definition 5.2.1.*

A truncated universal strategy can be defined similarly:

Definition 5.3.3 (Truncated Universal Strategy)**.**
A truncated universal strategy*, is a subtree $T = (V', E', e')$ of a game tree $G = (V, E, e)$ of a*

QIP^{ID}. V' contains the unique root node $r \in V_\exists$. Each node $v_\forall \in V' \cap V_\forall$ has either a) exactly one child if and only if the corresponding legal domain is not empty, or b) no child if and only if $\mathcal{F}(v_\forall) = \emptyset$. Each node $v_\exists \in V' \cap V_\exists$ has all the children as in G for which at least one leaf in their corresponding sub-tree in G exists with $A^\exists x \leq b^\exists$, i.e. as many as in $\mathcal{F}(v_\exists)$.

The term truncated strategy refers to a truncated existential strategy, whereas a truncated universal strategy will be called as such. In truncated strategies some nodes have no successor even though they are not leaves in G. We call such nodes *terminal nodes*.

Definition 5.3.4 (Terminal Nodes)**.**
Let $S = (V, E, e)$ be an edge-labeled finite arborescence. We call

$$\mathcal{T}(S) = \{v \in V \mid \nexists u \in V : (v, u) \in E\}$$

the set of terminal nodes *in S, i.e. the set of nodes without any successor in S.*

Note that for any strategy S of a QIP (see Definition 2.1.13) it is $\mathcal{T}(S) \subseteq V_L$. This is also true for strategies of QIP$^{\mathrm{PU}}$ (see Definition 4.2.7). For a truncated strategy T of a QIP$^{\mathrm{ID}}$ the set of terminal nodes $\mathcal{T}(T)$ might also contain inner nodes from $V_\forall$ or $V_\exists$. Therefore, for a *winning truncated strategy* it must be ensured that no terminal node represents a loss for the existential player.

Definition 5.3.5 (Winning Truncated (Existential) Strategy)**.**
A truncated existential strategy $T = (V', E', e')$ is called a winning truncated existential strategy*, if for all terminal nodes $\hat{v} \in \mathcal{T}(T)$ it holds*

$$\left(\hat{v} \in V_L \wedge A^\exists x_{\hat{v}} \leq b^\exists\right) \vee \hat{v} \in V_\forall\,,$$

i.e. terminal nodes of T either represent a fully assigned vector $x \in \mathcal{L}$ with $A^\exists x \leq b^\exists$ or a partially filled vector $\tilde{x}_1, \ldots, \tilde{x}_{k-1}$ with $Q_k = \forall$ and $\mathcal{F}(\tilde{x}_1, \ldots, \tilde{x}_{k-1}) = \emptyset$.

Similar to QIP and QIP$^{\mathrm{PU}}$, we call a QIP$^{\mathrm{ID}}$ instance *feasible* if a winning truncated existential strategy exists, and sometimes call such a winning truncated existential strategy a *solution* of the QIP$^{\mathrm{ID}}$. Furthermore, we define winning truncated strategies for the universal player:

Definition 5.3.6 (Winning Truncated Universal Strategy)**.**
A truncated universal strategy $T = (V', E', e')$ is called a winning truncated universal strategy*, if for all terminal nodes $\hat{v} \in \mathcal{T}(T)$ it holds*

$$\left(\hat{v} \in V_L \wedge A^\forall x_{\hat{v}} \leq b^\forall\right) \vee \hat{v} \in V_\exists\,.$$

The existence of a winning truncated strategy ensures that there is always a legal move if it is one's turn and that one's constraint system is fulfilled after a play. Thus, the utmost goal of either player—to fulfill their constraint system and to perform only legal moves—can be achieved. Furthermore, if it is possible to ensure a loss for the opponent we call the corresponding strategy *destructive*.

Definition 5.3.7 (Destructive Strategy)**.**
Let $G = (V, E, e)$ be a game tree of a QIP^{ID}.

a) A winning truncated existential strategy $T = (V', E', e')$ is called destructive*, if for each terminal node $\hat{v} \in \mathcal{T}(T)$ it holds $(\hat{v} \in V_L \wedge A^{\forall} x_{\hat{v}} \not\leq b^{\forall}) \vee \hat{v} \in V_{\forall}$.*

b) A winning truncated universal strategy $T = (V', E', e')$ is called destructive*, if for each terminal node $\hat{v} \in \mathcal{T}(T)$ it holds $(\hat{v} \in V_L \wedge A^{\exists} x_{\hat{v}} \not\leq b^{\exists}) \vee \hat{v} \in V_{\exists}$.*

In particular, the existence of a winning truncated strategy for one player is closely connected to the existence of a destructive strategy for the opponent.

Corollary 5.3.8. *A winning truncated existential strategy exists if and only if no destructive universal strategy exists.*

Remark 5.3.9. *Similar to a QIP, a winning existential (universal) strategy can be defined for QIP^{ID}. Hence, a winning existential (universal) strategy $S = (V', E', e')$ is an existential (universal) strategy in which each leaf $v \in V_L \cap V'$ fulfills $A^{\exists} x_v \leq b^{\exists}$ ($A^{\forall} x_v \leq b^{\forall}$) but does not necessarily fulfill $A^{\forall} x_v \leq b^{\forall}$ ($A^{\exists} x_v \leq b^{\exists}$).*

Corollary 5.3.10. *The existence of a winning existential (universal) strategy for a QIP^{ID} implies the existence of a winning truncated existential (universal) strategy.*

Since the outcome "universal player loses" can occur for QIP$^{\mathrm{ID}}$ the value $-\infty$ must be added as possible outcome in order to be able to apply an adapted minimax search. Furthermore, if both constraint systems are violated for a fixed variable vector $x \in \mathcal{L}$, the corresponding leaf node itself does not hold the information who made the first illegal move, i.e. who loses. The symbolic value $\pm\infty$ is introduced for such nodes.

Definition 5.3.11 (Extended Minimax Value)**.**
Let $G = (V, E, e)$ be a game tree for a QIP^{ID} and $S = (V', E', e')$ some subtree of G, with either $S = G$ or S is a strategy. Let $w(v) : V_L \to \mathbb{Q} \cup \{+\infty, -\infty, \pm\infty\}$ be a weighting function with

$$w(v) = \begin{cases} c^{\top} x_v & , A^{\exists} x_v \leq b^{\exists} \text{ and } A^{\forall} x_v \leq b^{\forall} \\ +\infty & , A^{\exists} x_v \not\leq b^{\exists} \text{ and } A^{\forall} x_v \leq b^{\forall} \\ -\infty & , A^{\exists} x_v \leq b^{\exists} \text{ and } A^{\forall} x_v \not\leq b^{\forall} \\ \pm\infty & , A^{\exists} x_v \not\leq b^{\exists} \text{ and } A^{\forall} x_v \not\leq b^{\forall} \end{cases} .$$

For any node $v \in V$ the extended minimax value *with respect to S is defined recursively by*

$$minimax_e^S(v) = \begin{cases} w(v) & , \text{if } v \in V_L \\ \min\{minimax_e^S(v') \mid (v, v') \in E' \wedge minimax_e^S(v') \neq \pm\infty\} & , \text{if } v \in V_{\exists} \setminus V_{\pm\infty} \\ \max\{minimax_e^S(v') \mid (v, v') \in E' \wedge minimax_e^S(v') \neq \pm\infty\} & , \text{if } v \in V_{\forall} \setminus V_{\pm\infty} \\ \pm\infty & , \text{if } v \in V_{\pm\infty} . \end{cases}$$

with the set $V_{\pm\infty}$, given by

$$V_{\pm\infty} = \{v \in V \setminus V_L \mid \forall v' \in V : (v, v') \in E \Rightarrow minimax_e(v') = \pm\infty\}. \tag{5.4}$$

The extended minimax value of the root $r \in V'$ with respect to S defines the value *of S denoted by $minimax_e(S) = minimax_e^S(r)$. For $S = G$ we denote $minimax_e(v) = minimax_e^G(v)$.*

The nodes in set $V_{\pm\infty}$ represent partial variable assignments after which neither constraint system can be fulfilled. Thus, it contains nodes that cannot be reached via legal variable assignments, but note that $V_{\pm\infty}$ does not contain *all* nodes resulting from illegal moves.

Corollary 5.3.12. *Let $G = (V, E, e)$ be the game tree of a QIP^{ID}. Then $minimax_e(v) = \pm\infty$ for $v \in V$, if and only if every leaf $\hat{v} \in V_L$ in the subtree below v has the property $w(\hat{v}) = \pm\infty$.*

The restriction $\{x \in \mathcal{L} \mid A^{\forall}x \leq b^{\forall}\} \neq \emptyset$ for a QIP$^{\mathrm{ID}}$—discussed in the previous section—now reveals another positive characteristic: with the above Corollary 5.3.12 the extended minimax value of the root cannot be $\pm\infty$, as in at least one leaf of the game tree $A^{\forall}x \leq b^{\forall}$ is fulfilled.

Corollary 5.3.13. *For a QIP^{ID} as in Definition 5.2.3 and its corresponding game tree $G = (V, E, e)$, $minimax_e(r) \neq \pm\infty$ holds for the root node $r \in V$.*

Therefore, with $\{x \in \mathcal{L} \mid A^{\forall}x \leq b^{\forall}\} \neq \emptyset$, the root node can have any extended minimax value within $\mathbb{Q} \cup \{+\infty, -\infty\}$. This can be interpreted as follows:

- Root node has extended minimax value $z = c^{\top}x \in \mathbb{Q}$:
  Both the universal as well as the existential player have a winning truncated strategy to satisfy their system. The payoff if both players play optimally is z.
- Root node has extended minimax value $+\infty$:
  The existential player has no winning truncated strategy to satisfy his system: The universal player can enforce $A^{\exists}x \not\leq b^{\exists}$. We call this instance *infeasible*.
- Root node has extended minimax value $-\infty$:
  The universal player has no winning truncated strategy to satisfy her system: The existential player can enforce $A^{\forall}x \not\leq b^{\forall}$.

We now investigate the properties of the extended minimax value at the terminal nodes in order to eventually adapt Stockman's theorem for truncated strategies.

Lemma 5.3.14. *Let $G = (V, E, e)$ be the game tree of a QIP^{ID} and let $T = (V', E', e')$ be a truncated strategy in G. For every terminal node $v \in \mathcal{T}(T)$ it holds $minimax_e^T(v) \neq \pm\infty$.*

Proof. Let $v \in \mathcal{T}(T)$ with $minimax_e(v) = \pm\infty$. With Corollary 5.3.13 we know that v is not the root and therefore an edge $(u, v) \in E'$ must exist. Thus, $v \in \mathcal{F}(u) \neq \emptyset$. Hence, there exists some leaf $w \in V_L$ below v with $minimax_e(v) \neq \pm\infty$, which contradicts the assumption using Corollary 5.3.12. □

Lemma 5.3.15. *Given a truncated strategy $T = (V', E', e')$ in the game tree $G = (V, E, e)$ of a QIP^{ID}. For any terminal node $v \in \mathcal{T}(T)$ the following statements hold:*

a) $v \in V_L \Rightarrow minimax_e(v) = w(v)$.

b) $v \in V_\forall \Rightarrow minimax_e(v) = -\infty$.

c) $v \in V_\exists \Rightarrow minimax_e(v) = +\infty$.

Proof.

a) See Definition 5.3.11.

b) Let $k \in \mathcal{I}_\forall$ and $\tilde{x}_1, \ldots, \tilde{x}_k$ be the variable assignment along the path from the root to v. Since v has no child in T it is $\mathcal{F}(\tilde{x}_1, \ldots, \tilde{x}_k) = \emptyset$ with Definition 5.3.2. Therefore, for each leaf $\hat{v} \in V_L$ in the corresponding subtree below v in G it is $w(\hat{v}) \in \{-\infty, \pm\infty\}$, because the universal constraint system is violated in each leaf. Therefore, $minimax_e(v) \in \{-\infty, \pm\infty\}$. If $minimax_e(v) = \pm\infty$ it is $w(\hat{v}) = \pm\infty$ for each such leaf. Hence, $\tilde{x}_k \notin \mathcal{F}(\tilde{x}_1, \ldots, \tilde{x}_{k-1})$, which contradicts $v \in V'$. Thus, $minimax_e(v) = -\infty$.

c) Analogous to b) with $w(\hat{v}) \in \{+\infty, \pm\infty\}$. □

The above results are now utilized to adapt Stockman's theorem [PdB01] for QIP$^{\text{ID}}$, which links the extended minimax value of a truncated strategy to the largest value of its terminal nodes.

Theorem 5.3.16 (Extended Stockman's Theorem)**.**
The extended minimax value of a truncated strategy $S = (V', E', e')$ is equal to the largest extended minimax value at the terminal nodes of the truncated strategy, i.e.

$$minimax_e(S) = \max_{v_t \in \mathcal{T}(S)} minimax_e(v_t)\,. \tag{5.5}$$

Proof. For any truncated strategy $S = (V', E', e')$ and any terminal node $v \in \mathcal{T}(S)$ it is $w(v) \in \mathbb{Q} \cup \{+\infty, -\infty\}$ due to Lemmas 5.3.14 and 5.3.15. Thus, computing the extended minimax with respect to S is equivalent to computing the conventional minimax value with the small extension of also having leaves with minimax values $-\infty$ and $+\infty$. Therefore, similar to the results stated in [Sto79, PdB01], the (extended) minimax value of a truncated strategy is given by the maximum value at its leafs, i.e. its terminal nodes. □

Definition 5.3.17 (Optimal Solution of a QIP$^{\text{ID}}$)**.**
Consider a feasible QIP^{ID} P and its game tree G. A winning truncated strategy S is called optimal *if $minimax_e(S) = minimax_e^G(r)$. In particular, $minimax_e(S) \leq minimax_e(\hat{S})$ applies for all other winning truncated strategies $\hat{S}$ and we call $minimax_e(S)$ the* optimal value *of P.*

For QIP and QIP$^{\text{PU}}$ the PV represents the path within the optimal solution from the root to the leaf with largest minimax value, i.e. the move sequence chosen during optimal play. This has to be adapted for QIP$^{\text{ID}}$ as optimal play might end in an inner terminal node rather than an actual leaf.

Definition 5.3.18 (Principal Variation of a QIP$^{\text{ID}}$)**.**
Let $G = (V, E, e)$ be a game tree of a feasible QIP^{ID} P. Let $S = (V', E', e')$ be an optimal winning truncated existential strategy.

a) If $minimax_e(S) \in \mathbb{Q}$ *then the variable assignment* $x_v \in \mathcal{L}$ *corresponding to the path from the root* r *to a leaf node* $v \in V_L \cap V'$ *in* S *with* $c^\top x_v = minimax_e(v) = minimax_e(S)$*, is called* principal variation *of* P*.*

b) If $minimax_e(S) = -\infty$ *for the root node* r *then the (possibly) partial variable assignment corresponding to any path from* r *to a terminal node* $t \in \mathcal{T}(S)$ *with* $minimax_e(t) = -\infty$ *is called a* truncated principal variation*. A variable assignment* $\tilde{x}_v$ *corresponding to a leaf* $v \in V_L$ *below* t *with* $A^\exists \tilde{x}_v \leq b^\exists$ *and* $minimax_e(v) = -\infty$ *is called the* principal variation *of* P*.*

Case a) is analogous to the PV for QIP and QIP$^{\mathrm{PU}}$. In case b) the truncated PV contains the path from the root to the corresponding terminal node, while the PV further provides the path to an underlying leaf with $A^\exists \tilde{x}_v \leq b^\exists$, which confirms that the existential player indeed used legal moves in order to violate the universal player's system.

Example 5.3.19. *Let* $c = (-1,-1,-2)^\top$*,* $Q = (\exists, \forall, \forall)$*,* $\mathcal{L}_1 = \{1,2,3\}$ *and* $\mathcal{L}_2 = \mathcal{L}_3 = \{0,1\}$ *and the two constraint systems given as follows:*

$$A^\exists x \leq b^\exists: \quad \begin{array}{rcrcrcl} x_1 & + & x_2 & + & x_3 & \leq & 3 \\ 2x_1 & - & 3x_2 & & & \leq & 3 \end{array}$$

$$A^\forall x \leq b^\forall: \quad \begin{array}{rcrcrcl} -x_1 & + & 2x_2 & + & x_3 & \geq & 0 \end{array}$$

The game tree of this QIP^{ID} instance is give in Figure 5.1. The values at the leaves are

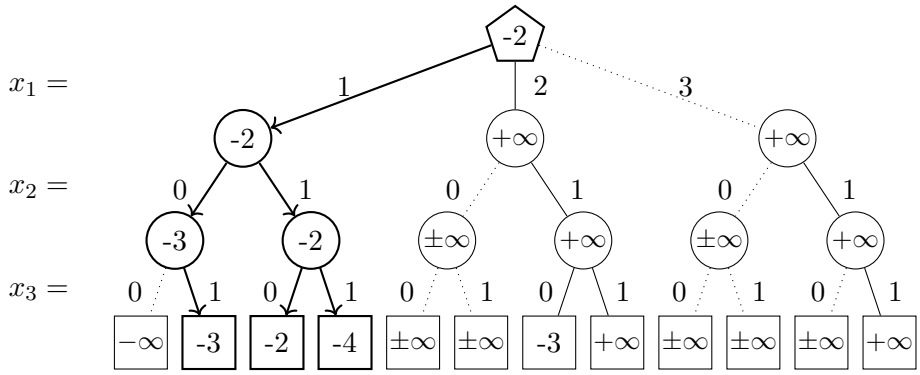

Figure 5.1.: Game tree for the given QIP$^{\mathrm{ID}}$ with rectangular leaves, circular universal nodes, and pentagonal existential root node. The values given in the nodes are the corresponding extended minimax values. Illegal moves are indicated as dotted lines. The optimal winning truncated strategy is indicated by thicker arrows.

assigned using the weighting function $w(v)$*. The value of inner nodes were received by applying the extended minimax value. Hence, node values show the value according to optimal play starting from this node, whereat* $\pm\infty$ *denotes an illegal game situation that cannot occur during legal play. In the first existential stage* $\mathcal{F}^{(1)} = \{1,2\}$ *and in particular,* $3 \notin \mathcal{F}^{(1)}$ *since the existential system cannot be fulfilled for* $x_3 = 3$ *(all leaves in the subtree beneath the decision* $x_3 = 3$ *have the property* $+\infty$ *or* $\pm\infty$*). Note that if* $x_1 = 2$ *the universal variable* x_2 *cannot be set to* 0*, since a*

violation of the universal system would become inevitable. In particular, $\mathcal{F}^{(2)}(2) = \{(1,0),(1,1)\}$. *Furthermore, even though the extended minimax value of the node resulting from setting* $x_1 = 2$ *is* $+\infty$ *this move itself is not illegal; it only results in a game situation where no winning strategy for the existential player exists. Thus, even though* $x_1 = 2$ *is a bad move, it is still legal. The optimal course of play (the principal variation) is* $x_1 = 1$, $x_2 = 1$, $x_3 = 0$ *with objective value* $c^\top x = -2$.

Remark 5.3.20. *At the very beginning of our research regarding QIP^{ID} we assumed that a move should only be legal, if a strategy exists—which itself consist of legal moves—such that the satisfiability of one's own constraint system is ensured. This has the obvious drawback that in this case a legal move can only be validated by solving a* PSPACE*-complete problem. We then realized that it suffices to use the set of legal moves as given in Definition 5.2.1 without changing the outcome. This solely changes the notion from being an illegal move to being a bad move, if no such strategy is available. Due to Condition (5.3) both players already aim for such a strategy and hence, rather than explicitly forbidding such moves we exploit that they eventually result in the correct extended minimax value* $+\infty$, $-\infty$ *or* $\pm\infty$. *Demanding this during the legality check solely anticipates that outcome at a high computational price.*

5.4. Computational Complexity of QIP$^{\mathsf{ID}}$

5.4.1. Complexity Results

The NP-hardness of determining the set of legal variable assignments $\mathcal{F}$ in each game position raises the question whether the presented extension results in a more difficult problem in the sense of complexity theory. However, a few simple arguments show that indeed QIP$^{\mathrm{ID}}$ remains in the same complexity class as the original QIP. Clearly a polynomial-time reduction function can be developed to show QIP $\leq_p$ QIP$^{\mathrm{ID}}$: by adding a trivial universal constraint system and a universal dummy variable at the end a QIP can be reduced to a QIP$^{\mathrm{ID}}$ The non-trivial reduction function proving QIP$^{\mathrm{ID}}$ $\leq_p$ QIP is presented in Subsection 5.4.2. Nevertheless, the existence of an extended minimax algorithm is already sufficient to show that QIP$^{\mathrm{ID}}$ is in PSPACE.

Theorem 5.4.1. *QIP^{ID} is* PSPACE*-complete.*

Proof. PSPACE-hardness is established via the outlined trivial reduction QIP $\leq_p$ QIP$^{\mathrm{ID}}$ and the PSPACE-completeness of QIP [Wol15]. To show that QIP$^{\mathrm{ID}}$ is in PSPACE, we can solve it with an extended minimax algorithm, that performs the evaluation of $minimax_e(v)$ for each node (cf. Algorithm 4 on page 107). When using a depth-first search one does not have to keep the entire game tree in memory, but only the current path. Therefore, only polynomial space is required in order to evaluate an exponentially sized game tree [All94, KM75]. □

However, PSPACE is a large class, and we are interested in the connection between QIP$^{\mathrm{ID}}$ and QIP: How can a QIP be utilized in order to describe the interdependence of legal moves and in particular the problem of distinguishing between legal and illegal variable assignments. In the

following subsection we present a reduction function in order to explicitly prove the non-trivial relationship QIP$^{\mathrm{ID}}$ $\leq_p$ QIP. The presented reduction gives us further structural insights into the nature of QIP$^{\mathrm{ID}}$ and helps us understand how a solution process in the Yasol framework is realizable and reasonable. Second it provides a checking routine for implementations: we want to solve QIP$^{\mathrm{ID}}$ via an extended game tree search, and the results can be checked by solving the DEP of the reduced QIP or directly solving the QIP via Yasol.

5.4.2. Reduction QIP$^{\mathrm{ID}}$ $\leq_p$ QIP

The presented problem QIP$^{\mathrm{ID}}$ has one major difficulty compared to the QIP: In order to know whether a variable assignment is legal one has to solve an NP-complete subproblem, i.e. one has to check the feasibility of several integer programs in order to create the set $\mathcal{F}^{(i)}\left(x^{(1)},\ldots,x^{(i-1)}\right)$, instead of simply ensuring compliance with the variable bounds. Hence, in order to transform a QIP$^{\mathrm{ID}}$ into a QIP, the resulting QIP has to ensure that setting one of the original variables illegally (i.e. outside of the specific $\mathcal{F}$-domain) cannot occur or would be detrimental to the player conducting such move. We take the latter path and therefore the QIP must be enabled to to detect illegal moves and penalize the respective player by allowing the opponent to get a much better payoff than usual. Hence, it is in each player's best interest to make moves that stay within the legal domain $\mathcal{F}$.

For a better understanding we first roughly sketch the functional principal of the reduction. The original variables $x^{(i)}$ of a variable block are also used in the arising QIP and in between some further auxiliary variables are added in order to ensure that illegal moves are disadvantageous compared to legal ones. For each existential variable block $x^{(i)}$ a verification vector $v^{(i)}$ is added to check, whether the existential system still can be satisfied. If for some assignment of $x^{(i)}$ the system $A^{\exists}x \leq b^{\exists}$ can no longer be fulfilled then no appropriate assignment of $v^{(i)}$ exists, resulting in a violation of the global constraint system of the QIP, which is to the detriment of the existential player. Hence, a legal existential variable assignment $x^{(i)} \in \mathcal{F}^{(i)}$ is preferred for which a corresponding assignment of $v^{(i)}$ is available. Note, however, that this must only come into effect, if in each earlier stage only legal moves were selected. A similar approach is used for universal variable blocks: If there is no assignment of the verification vector $v^{(i)}$ that fulfills the universal system, i.e. if $x^{(i)} \notin \mathcal{F}^{(i)}$, the objective value can be reduced massively and the constraint system of the QIP is fulfilled trivially. Auxiliary variables $y^{(i)}$ and t_i are used to detect and indicate such illegal assignments for universal variable block i. Hence, selecting the universal variable $x^{(i)}$ from $\mathcal{F}^{(i)}$ is preferable for the universal player.
The presented reduction function allows the coefficients of the resulting QIP to range in given intervals. There are the following reasons for the reduction function to be not unique:

1. By specifying bounds on certain parameters rather than fixing them to specific values this reduction leaves more leeway for application and simultaneously does not lose theoretical value.

2. There are technical reason as the proof for Theorem 5.4.4 is inductive and the structure of a subproblem and its reduction is not maintained if the parameters are fixed.

3. When proving that such a reduction can be computed in polynomial time we argue in Theorem 5.4.14 that "some" reduction can be computed in linear time.

Several observations from Section 4.3 can be reused in order to build the reduction, but an additional difficulty, in contrast to the QIP$^{\text{PU}}$ reduction, is that now in each stage the legality of a variable block assignment has to be ensured, and it no longer suffices to conduct a single check at the end of a play in order to determine whether the universal player complied with $A^{\forall}x \leq b^{\forall}$. This is necessary as the first illegal move must be detected in order to determine the losing player. Let the first $i-1$ variable blocks be assigned with $\bar{x}^{(1)}, \dots, \bar{x}^{(i-1)}$ and assume those assignments are all legal according to Definition 5.2.2. Let $i \in \mathcal{E}$. In this situation, an assignment $\hat{x}^{(i)} \in \mathcal{L}^{(i)}$ is legal if there exists some verification vector $v^{(i+1)} = (\hat{\hat{x}}^{(i+1)}, \dots, \hat{\hat{x}}^{(\beta)}) \in \mathcal{L}^{(i+1)} \times \dots \times \mathcal{L}^{(\beta)}$ such that for the resulting vector $s = (\bar{x}^{(1)}, \dots, \bar{x}^{(i-1)}, \hat{x}^{(i)}, \hat{\hat{x}}^{(i+1)}, \dots, \hat{\hat{x}}^{(\beta)})$ the existential system is satisfied, i.e. $A^{\exists}s \leq b^{\exists}$. If, on the other hand no verification vector can be found the assignment $\hat{x}^{(i)}$ is not legal. Hence, for each existential variable block $i \in \mathcal{E}$ the sub system $A^{\exists}s^{(i)} \leq b^{\exists}$ is added to the arising QIP with $s^{(i)} = (\bar{x}^{(1)}, \dots, \bar{x}^{(i-1)}, x^{(i)}, v^{(i)})$ being the vector consisting of the previous and current variable assignments as well as the verification vector. Therefore, in each existential block i the corresponding variable vector $x^{(i)}$ should be assigned legally as else the entire constraint system would become infeasible, resulting in a loss for the existential player. However, a slight adjustment must be made as the assumption that $\bar{x}^{(1)}, \dots, \bar{x}^{(i-1)}$ are legal assignments is not always true. If one of the previous variable assignments already was illegal the current i-th variable block does not have any effect on the outcome as the loser is already determined. In particular, a previous illegal move could render each assignment of $x^{(i)}$ illegal. Therefore, in such a case the mentioned sub system is relaxed via the indicator variable t_{i-1}, which signals whether some previous universal variable block was illegally assigned. In order to detect an illegal universal variables assignment we reuse findings of Subsection 4.3 and in particular Equivalence (4.4) together with Lemma 4.3.1 and Constraint (4.5). Note that for $i \in \mathcal{A}$, and $\hat{x}^{(i)} \in \mathcal{L}^{(i)}$ it is the universal player's obligation to provide an assignment for $v^{(i)}$ in order to prove the legality of her move. If she cannot (or does not) provide the validation that $\hat{x}^{(i)} \in \mathcal{F}^{(i)}$ at least one of the universal player's constraints is violated by $s = (\bar{x}^{(1)}, \dots, \bar{x}^{(i-1)}, \hat{x}^{(i)}, v^{(i)})$, which eventually allows the existential player to set the corresponding indicator $t_i = 1$.

In the following definition we present the reduction function mapping any QIP$^{\text{ID}}$ instance to a set of QIPs. We show in Theorem 5.4.14 that it is possible to compute one of those QIP instances in polynomial time.

Definition 5.4.2 (Reduction Function QIP$^{\text{ID}} \leq_p$ QIP).
The reduction function f^{ID} maps a given QIP^{ID} $P = (A^{\exists}, A^{\forall}, b^{\exists}, b^{\forall}, c, \mathcal{L}, Q)$ to a QIP with the following form:

$$\min_{x^{(1)} \in \mathcal{L}^{(1)}} \left(c^{(1)}x^{(1)} + \max_{x^{(2)} \in \mathcal{L}^{(2)}} \left(c^{(2)}x^{(2)} + \ldots + \max_{x^{(\beta)} \in \mathcal{L}^{(\beta)}} \left(c^{(\beta)}x^{(\beta)} - \min_{p \in \{0,1\}} \tilde{M}p \right) \right) \right)$$

$$
\begin{aligned}
\text{s.t. } & \exists\, x^{(1)} \in \mathcal{L}^{(1)} \quad \exists\, v^{(1)} \in \mathcal{L}^{(2)} \times \ldots \times \mathcal{L}^{(\beta)} \\
& \forall\, x^{(2)} \in \mathcal{L}^{(2)} \quad \forall\, v^{(2)} \in \mathcal{L}^{(3)} \times \ldots \times \mathcal{L}^{(\beta)} \quad \exists\, y^{(2)} \in \{0,1\}^{m_\forall} \quad \exists\, t_2 \in \{0,1\} \\
& \exists\, x^{(3)} \in \mathcal{L}^{(3)} \quad \exists\, v^{(3)} \in \mathcal{L}^{(4)} \times \ldots \times \mathcal{L}^{(\beta)} \\
& \forall\, x^{(4)} \in \mathcal{L}^{(4)} \quad \forall\, v^{(4)} \in \mathcal{L}^{(5)} \times \ldots \times \mathcal{L}^{(\beta)} \quad \exists\, y^{(4)} \in \{0,1\}^{m_\forall} \quad \exists\, t_4 \in \{0,1\} \\
& \ldots \\
& \forall\, x^{(\beta)} \in \mathcal{L}^{(\beta)} \quad \exists\, y^{(\beta)} \in \{0,1\}^{m_\forall} \quad \exists\, t_\beta \in \{0,1\} \quad \exists\, p \in \{0,1\} :
\end{aligned}
$$

$$
\begin{aligned}
& A^{\exists} s^{(i)} - M t_{i-1} \leq b^{\exists} && \forall\, i \in \mathcal{E} && (5.6)\\
& A^{\exists} x - M p \leq b^{\exists} && && (5.7)\\
& -A^{\forall} s^{(i)} - (L - b^{\forall} - R^{LCD}) y^{(i)} \leq -L && \forall\, i \in \mathcal{A} && (5.8)\\
& p - \sum_{i \in \mathcal{A}} t_i \leq 0 && && (5.9)\\
& t_i - t_{i-2} - \sum_{k=1}^{m_\forall} y_k^{(i)} \leq 0 && \forall\, i \in \mathcal{A} && (5.10)
\end{aligned}
$$

The symbol $s^{(i)}$ is the abbreviation for the vector $(x^{(1)}, \ldots, x^{(i)}, v^{(i)})$. M, $\tilde{M}$, L, R^{LCD} and t_0 are parameters that must fulfill the following criteria: $t_0 = 0$, $L \in \mathbb{Q}^{m_\forall}$ with

$$
L_k \leq \min_{x \in \mathcal{L}} A^{\forall}_{k,\star} x = \sum_{\substack{i \in \mathcal{I} \\ A^{\forall}_{k,i} < 0}} A^{\forall}_{k,i} \cdot u_i + \sum_{\substack{i \in \mathcal{I} \\ A^{\forall}_{k,i} \geq 0}}^{n} A^{\forall}_{k,i} \cdot l_i \qquad \forall\, k \in \{1, \ldots, m_\forall\}\,, \tag{5.11}
$$

being less than the smallest possible value of the left-hand side of row k of the universal constraint system resulting in $A^{\forall} x \geq L$ being valid for any $x \in \mathcal{L}$. $M \in \mathbb{Q}^{m_\exists}$ must fulfill

$$
M_k \geq \max_{x \in \mathcal{L}} A^{\exists}_{k,\star} x - b^{\exists}_k = \sum_{\substack{i \in \mathcal{I} \\ A^{\exists}_{k,i} < 0}} A^{\exists}_{k,i} \cdot l_i + \sum_{\substack{i \in \mathcal{I} \\ A^{\exists}_{k,i} \geq 0}} A^{\exists}_{k,i} \cdot u_i - b^{\exists}_k \qquad \forall k \in \{1, \ldots, m_\exists\}\,, \tag{5.12}
$$

i.e. M_k must be a bound on the largest possible violation of row k of the existential system, resulting in $A^{\exists} x \leq b^{\exists} + M$ being valid for all $x \in \mathcal{L}$. $\tilde{M} \in \mathbb{Q}$ is set such that

$$
\max_{x \in \mathcal{L}} c^{\top} x - \tilde{M} < \min_{x \in \mathcal{L}} c^{\top} x \tag{5.13}
$$

for example

$$
\tilde{M} \geq \sum_{\substack{i \in \mathcal{I} \\ c_i < 0}} c_i \cdot (l_i - u_i) + \sum_{\substack{i \in \mathcal{I} \\ c_i \geq 0}}^{n} c_i \cdot (u_i - l_i) + 1\,. \tag{5.14}
$$

$R^{LCD} \in \mathbb{Q}^{m_\forall}$ is a vector with positive entries less than or equal to the reciprocals of the lowest common denominators of the rows of $A^{\forall}$ and $b^{\forall}$, ensuring for any row $k \in \{1, \ldots, m_\forall\}$ and any

$x \in \mathcal{L}$ that $A^{\forall}_{k,\star} x \not\leq b^{\forall}_k \Leftrightarrow A^{\forall}_{k,\star} x \geq b^{\forall}_k + R^{LCD}_k$. In particular, $0 < R^{LCD}_k \leq 1$ suffices if all entries in row k are integer. We call $f^{ID}(P)$ the set of QIPs corresponding to the QIP^{ID} P.

Remark 5.4.3. *For the entries of L and R^{LCD} only upper bounds are given. Hence, for smaller values the operating principle is still intact. For M and $\tilde{M}$ lower bounds are given. Therefore, the result of the above mechanism is not unique, since the parameters L, M, $\tilde{M}$ and R^{LCD} only have to satisfy the given bounds.*

Note that the number of variable blocks of a resulting QIP in $f^{\mathrm{ID}}(P)$ is increased by one compared to the QIP$^{\mathrm{ID}}$. This final additional variable block is needed to check the fulfillment of the universal constraint system after the final variable block has been set. A result of the reduction of the QIP$^{\mathrm{ID}}$ given in Example 5.3.19 can be found in Appendix A.1.

5.4.3. Correctness of the Reduction QIP$^{\mathrm{ID}} \leq_p$ QIP

The close relationship between a QIP$^{\mathrm{ID}}$ and the corresponding QIPs resulting from the reduction function is given in the following theorem:

Theorem 5.4.4 (Correctness of f^{ID})**.**
The following equivalences hold:

1. *The QIP^{ID} instance P is feasible with optimal value $z \in \mathbb{Q}$*
   $\Longleftrightarrow \forall R \in f^{ID}(P)$: The QIP instance R is feasible with optimal value $\bar{z} = z$.
2. *The QIP^{ID} instance P is feasible with optimal value $z = -\infty$*
   $\Longleftrightarrow \forall R \in f^{ID}(P)$: The QIP instance R is feasible with optimal value $\bar{z} < \min\limits_{x \in \mathcal{L}} c^{\top} x$.
3. *The QIP^{ID} instance P is infeasible*
   $\Longleftrightarrow \forall R \in f^{ID}(P)$: The QIP instance R is infeasible.

In preparation for the proof of Theorem 5.4.4 some further considerations are required.

Definition 5.4.5 (Verification Set $\mathcal{V}$)**.**
Let $P = (A^{\exists}, A^{\forall}, b^{\exists}, b^{\forall}, c, \mathcal{L}, Q)$. For $1 \leq i < \beta$ and fixed $x^{(1)} \in \mathcal{L}^{(1)}, \ldots, x^{(i)} \in \mathcal{L}^{(i)}$ the verification set *$\mathcal{V}(x^{(1)}, \ldots, x^{(i)})$ is given by*

$$\mathcal{V}(x^{(1)}, \ldots, x^{(i)}) = \left\{ v^{(i)} \in \mathcal{L}^{(i+1)} \times \cdots \times \mathcal{L}^{(\beta)} \mid A^{Q^{(i)}}(x^{(1)}, \ldots, x^{(i)}, v^{(i)}) \leq b^{Q^{(i)}} \right\},$$

i.e. $\mathcal{V}(x^{(1)}, \ldots, x^{(i)})$ contains those vectors that verify that $x^{(i)} \in \mathcal{F}^{(i)}(x^{(1)}, \ldots, x^{(i-1)})$.

Corollary 5.4.6. *For $1 \leq i < \beta$ and given $x^{(1)} \in \mathcal{L}^{(1)}, \ldots, x^{(i-1)} \in \mathcal{L}^{(i-1)}$ it holds*

$$\mathcal{F}^{(i)}(x^{(1)}, \ldots, x^{(i-1)}) = \emptyset \Longleftrightarrow \forall \tilde{x}^{(i)} \in \mathcal{L}^{(i)} : \mathcal{V}(x^{(1)}, \ldots, x^{(i-1)}, \tilde{x}^{(i)}) = \emptyset$$

Definition 5.4.7 (Valid Variable Assignment until Stage k)**.**
Let $P = (A^{\exists}, A^{\forall}, b^{\exists}, b^{\forall}, c, \mathcal{L}, Q)$, $R \in f^{ID}(P)$ and $1 \leq k \leq \beta$. An assignment of the variables

$x^{(j)} \in \mathcal{L}^{(j)}$ and $v^{(j)} \in \mathcal{L}^{(j+1)} \times \cdots \times \mathcal{L}^{(\beta)}$ for all $j \leq k$ and $y^{(j)} \in \{0,1\}^{m_\forall}$ and $t_j \in \{0,1\}$ for all $j \in \mathcal{A}$, $j \leq k$, is called valid until stage k *for R, if the following constraint system is fulfilled*

$$\begin{aligned} & A^{\exists} s^{(i)} - M t_{i-1} \leq b^{\exists} && \forall\, i \in \mathcal{E},\ i \leq k \\ & -A^{\forall} s^{(i)} - (L - b^{\forall} - R^{LCD}) y^{(i)} \leq -L && \forall\, i \in \mathcal{A},\ i \leq k \\ & t_i - t_{i-2} - \sum_{k=1}^{m_\forall} y_k^{(i)} \leq 0 && \forall\, i \in \mathcal{A},\ i \leq k \end{aligned}$$

with $s^{(i)} = (x^{(1)}, \ldots, x^{(i)}, v^{(i)}) \in \mathcal{L}$. Such an assignment is denoted by $(x, v, y, t)_k$.

Lemma 5.4.8. *Let $P = (A^{\exists}, A^{\forall}, b^{\exists}, b^{\forall}, c, \mathcal{L}, Q)$, $R \in f^{ID}(P)$ and $1 \leq k \leq \beta$ with $k \in \mathcal{E}$. For a given valid variable assignment $(x, v, y, t)_{k-1}$ until stage $k-1$ with $t_{k-1} = 0$ and an arbitrary $x^{(k)} \in \mathcal{L}^{(k)}$ the constraint system of R is violated for $v^{(k)} \notin \mathcal{V}(x^{(1)}, \ldots, x^{(k)})$.*

Proof. Constraint (5.6) of R for $i = k$ states $A^{\exists} s^{(k)} \leq b^{\exists}$. For $s^{(k)} = (x^{(1)}, \ldots, x^{(k)}, v^{(k)})$ with $v^{(k)} \notin \mathcal{V}(x^{(1)}, \ldots, x^{(k)})$ it is $A^{\exists} s^{(k)} \not\leq b^{\exists}$ by definition of $\mathcal{V}(x^{(1)}, \ldots, x^{(k)})$. □

Corollary 5.4.9. *Let $P = (A^{\exists}, A^{\forall}, b^{\exists}, b^{\forall}, c, \mathcal{L}, Q)$, $R \in f^{ID}(P)$ and $1 \leq k \leq \beta$ with $k \in \mathcal{E}$. For a given valid variable assignment until stage $k-1$ with $t_{k-1} = 0$ and $x^{(k)} \in \mathcal{L}^{(k)} \setminus \mathcal{F}^{(k)}$ the system of R cannot be fulfilled.*

Lemma 5.4.10. *Let $P = (A^{\exists}, A^{\forall}, b^{\exists}, b^{\forall}, c, \mathcal{L}, Q)$, $k \in \{1 \ldots, m_\forall\}$ and L and R^{LCD} complying with the bounds as given in Definition 5.4.2. Let $\tilde{x} \in \mathcal{L}$. For*

$$-A^{\forall}_{k,\star} \tilde{x} - (L_k - b^{\forall}_k - R^{LCD}_k) y_k \leq -L_k \tag{5.15}$$

the following holds:

a) Condition (5.15) is valid for $\tilde{x}$ with $y_k = 0$.

b) Condition (5.15) is valid for $\tilde{x}$ with $y_k = 1 \Leftrightarrow A^{\forall}_{k,\star} \tilde{x} > b_k$.

Proof.

a) Since $L_k \leq \min_{x \in \mathcal{L}} A^{\forall}_{k,\star} x$ the statement is true.

b) "⇒" Let $-A^{\forall}_{k,\star} \tilde{x} - (L_k - b^{\forall}_k - R^{LCD}_k) \leq -L_k$ be true. Hence, $A^{\forall}_{k,\star} \tilde{x} \geq b^{\forall}_k + R^{LCD}_k$. Since $R^{LCD}_k > 0$ it is $A^{\forall}_{k,\star} \tilde{x} > b_k$.

"⇐" Let $A^{\forall}_{k,\star} \tilde{x} > b_k$. Hence, there is an $\epsilon > 0$ with $A^{\forall}_{k,\star} \tilde{x} \geq b_k + \epsilon$. With Lemma 4.3.1 and $\epsilon = R^{LCD}_k$, i.e. ϵ selected less than or equal to the reciprocal of the lowest common denominator of b_k and entries in $A^{\forall}_{k,\star}$, it is $A^{\forall}_{k,\star} \tilde{x} \geq b^{\forall}_k + R^{LCD}_k$. □

Lemma 5.4.11. *Let $P = (A^{\exists}, A^{\forall}, b^{\exists}, b^{\forall}, c, \mathcal{L}, Q)$, $R \in f^{ID}(P)$, and $1 \leq k < \beta$ with $k \in \mathcal{A}$. For a given valid variable assignment until stage $k-1$ with $t_{k-2} = 0$ and an arbitrary $x^{(k)} \in \mathcal{L}^{(k)}$ the following holds: For $v^{(k)} \in (\mathcal{L}^{(k+1)} \times \ldots \times \mathcal{L}^{(\beta)}) \setminus \mathcal{V}(x^{(1)}, \ldots, x^{(k)})$ a winning strategy exists to fulfill the constraint system of R with minimax value less than $\min_{x \in \mathcal{L}} c^{\top} x$.*

This lemma shows that if existential variables in R up to stage $k-1$ represent legal variable assignments with respect to the underlying QIP$^{\text{ID}}$ P and if the universal player fails to provide a correct legality validation $v^{(k)}$, then optimal play results in a very small payoff, which is to the detriment of the universal player.

Proof. Since the given variable assignment is valid until stage $k-1$ the Constraints (5.6), (5.8) and (5.10) of R are fulfilled for $i \leq k-1$ and in particular with $t_{k-2} = 0$ we know $A^{\exists}s^{(k-1)} \leq b^{\exists}$. Therefore, we only have to consider the remaining constraints of R. Since $v^{(k)} \notin \mathcal{V}(x^{(1)}, \dots, x^{(k)})$ and with $s^{(k)} = (x^{(1)}, \dots, x^{(k)}, v^{(k)})$ it is $A^{\forall}s^{(k)} \not\leq b^{\forall}$. Thus, a universal constraint $j \in \{1, \dots, m_{\forall}\}$ exists with $\sum_{i=1}^{n} A^{\forall}_{j,i}s^{(k)}_i > b^{\forall}_j$. Hence, $y^{(k)}_j = 1$ and $y^{(k)}_{j'} = 0$ for all $j' \neq j$ is a valid assignment such that

$$-A^{\forall}s^{(k)} - (L - b^{\forall} - R^{LCD})y^{(k)} \leq -L$$

is fulfilled as show in Lemma 5.4.10. Therefore, $t_k = 1$ is valid with Constraint (5.10) for $i = k$ and Constraint (5.6) for $i = k+1$ is fulfilled for any $x^{(k+1)}$ and $v^{(k+1)}$. For any $x^{(k+2)}$ and $v^{(k+2)}$ Constraint (5.8) for $i = k+2$ is fulfilled with $y^{(k+2)} = 0$ (Lemma 5.4.10). Since $t_k = 1$ we can set $t_{k+2} = 1$ according to (5.10). This argument can be used recursively until the final stage is reached such that Constraints (5.6), (5.7), (5.8), (5.10) are fulfilled. For $p = 1$ Constraint (5.9) is fulfilled resulting in $z = c^{\top}x - \tilde{M}p < \min_{x \in \mathcal{L}} c^{\top}x$ due to the selection of $\tilde{M}$ (see (5.13)). □

Corollary 5.4.12. *Let $P = (A^{\exists}, A^{\forall}, b^{\exists}, b^{\forall}, c, \mathcal{L}, Q)$, $R \in f^{ID}(P)$ and $1 \leq k \leq \beta$ with $k \in \mathcal{A}$. For a given valid variable assignment until stage $k-1$ with $t_{k-2} = 0$ and fixed $x^{(k)} \in \mathcal{L}^{(k)} \setminus \mathcal{F}^{(k)}$ the constraint system of R can be fulfilled and a winning strategy exists with minimax value less than $\min_{x \in \mathcal{L}} c^{\top}x$.*

Lemma 5.4.13. *Let $P = (A^{\exists}, A^{\forall}, b^{\exists}, b^{\forall}, c, \mathcal{L}, Q)$ be a QIP^{ID} with $\beta \geq 4$ and $R \in f^{ID}(P)$ a corresponding QIP according to Definition 5.4.2.*

a) For already fixed variables $\tilde{x}_1 \in \mathcal{F}^{(1)}$ and $\tilde{x}_2 \in \mathcal{F}^{(2)}(\tilde{x}^{(1)})$ the subproblem $\tilde{P}$ of P

$$\min_{x^{(3)} \in \mathcal{F}^{(3)}(\tilde{x}^{(1)}, \tilde{x}^{(2)})} \left(c^{(3)}x^{(3)} + \max_{x^{(4)} \in \mathcal{F}^{(4)}(\tilde{x}^{(1)}, \tilde{x}^{(2)}, x^{(3)})} \left(c^{(4)}x^{(4)} + \dots \max_{x^{(\beta)} \in \mathcal{F}^{(\beta)}} c^{(\beta)}x^{(\beta)} \right) \right)$$

$$\text{s.t. } \exists\, x^{(3)} \in \mathcal{F}^{(3)}(\tilde{x}^{(1)}, \tilde{x}^{(2)})\ \forall\, x^{(4)} \in \mathcal{F}^{(4)}(\tilde{x}^{(1)}, \tilde{x}^{(2)}, x^{(3)}) \dots\ \forall\, x^{(\beta)} \in \mathcal{F}^{(\beta)} : \tilde{A}^{\exists}x \leq \tilde{b}^{\exists}$$

is a QIP^{ID} according to Definition 5.2.3 with $\beta - 2$ variable blocks.

b) For the subproblem $\tilde{R}$ of R with already fixed variables $x^{(1)} = \tilde{x}^{(1)}$, $\tilde{v}^{(1)} \in \mathcal{V}(\tilde{x}^{(1)})$, $x^{(2)} = \tilde{x}^{(2)}$, $\tilde{v}^{(2)} \in \mathcal{V}(\tilde{x}^{(1)}, \tilde{x}^{(2)})$, $y^{(2)} = \bar{0}$ and $t_2 = 0$ it holds $\tilde{R} \in f^{ID}(\tilde{P})$ for $\tilde{P}$ as in a).

Proof. Let $\tilde{B} = |B_1 \cup B_2|$ and $\tilde{n} = n - |\tilde{B}|$.

a) For already fixed $\tilde{x}_1 \in \mathcal{F}^{(1)}$ and $\tilde{x}_2 \in \mathcal{F}^{(2)}(\tilde{x}^{(1)})$ the right hand side vectors $\tilde{b}^q$, for $q \in \{\exists, \forall\}$, of the two constraint systems are given by

$$\tilde{b}^q_j = b^q_j - \sum_{i=1}^{|B_1|} A^q_{j,\mu(1,i)} \tilde{x}^{(1)}_i - \sum_{i=1}^{|B_2|} A^q_{j,\mu(2,i)} \tilde{x}^{(2)}_i \qquad \forall\, j \in \{1, \ldots, m_q\}$$

and the left-hand side matrix $\tilde{A}^q \in \mathbb{Q}^{m_q \times \tilde{n}}$ is the submatrix of A^q without the columns belonging to B_1 and B_2. $\tilde{Q} \in \{\exists, \forall\}^{\tilde{n}}$ and $\tilde{c} \in \mathbb{Q}^{\tilde{n}}$ are the subvectors of Q and c, respectively, without the entries belonging to B_1 and B_2. Let $\tilde{\mathcal{L}} = \{x \in \mathbb{Z}^{\tilde{n}} \mid x_i \in \mathcal{L}_{|\tilde{B}|+i}\}$. Hence, $\tilde{P} = (\tilde{A}^\forall, \tilde{A}^\exists, \tilde{b}^\forall, \tilde{b}^\exists, \tilde{c}, \tilde{\mathcal{L}}, \tilde{Q})$ is a QIP$^{\mathrm{ID}}$ with $\beta - 2$ variable blocks. Furthermore, $\tilde{Q}^{(1)} = \exists$, $\tilde{Q}^{(\beta-2)} = \forall$ and $\{x \in \tilde{\mathcal{L}} \mid \tilde{A}^\forall x \leq \tilde{b}^\forall\} \neq \emptyset$ since $\tilde{x}^{(2)} \in \mathcal{F}^{(2)}(\tilde{x}^{(1)})$.

b) For fixed $x^{(1)} = \tilde{x}^{(1)}$, $\tilde{v}^{(1)} \in \mathcal{V}(\tilde{x}^{(1)})$, $x^{(2)} = \tilde{x}^{(2)}$, $\tilde{v}^{(2)} \in \mathcal{V}(\tilde{x}^{(1)}, \tilde{x}^{(2)})$, $y^{(2)} = 0$, $t_2 = 0$, Constraint (5.6) for $i = 1$ is satisfied, and also Constraints (5.8) and (5.10) are satisfied for $i = 2$. Omitting these constraints and inserting the fixed variables, using the notation as in a), yield:

$$\tilde{A}^\exists \hat{s}^{(i)} - M t_{i-1} \leq \tilde{b}^\exists \qquad \forall\, i \in \mathcal{E} \setminus \{1\} \tag{5.16}$$

$$\tilde{A}^\exists \hat{x} - Mp \leq \tilde{b}^\exists \tag{5.17}$$

$$-\tilde{A}^\forall \hat{s}^{(i)} - (\tilde{L} - \tilde{b}^\forall - R^{LCD}) y^{(i)} \leq -\tilde{L} \qquad \forall\, i \in \mathcal{A} \setminus \{2\} \tag{5.18}$$

$$p - \sum_{i \in \mathcal{A} \setminus \{2\}} t_i \leq 0 \tag{5.19}$$

$$t_i - t_{i-2} - \sum_{k=1}^{m_\forall} y^{(i)}_k \leq 0 \qquad \forall\, i \in \mathcal{A} \setminus \{2\}\,, \tag{5.20}$$

with $\hat{x} = (x^{(3)}, \ldots, x^{(\beta)})$, $\hat{s}^{(i)} = (x^{(3)}, \ldots, x^{(i)}, v^{(i)})$ and $\tilde{L} = L - b^\forall + \tilde{b}^\forall$. Constraints (5.19) and (5.20) obviously are the corresponding constraints of any instance from $f^{\mathrm{ID}}(\tilde{P})$. For Constraints (5.16) and (5.17) one must consider the vector M. For any $k \in \{1, \ldots, m_\exists\}$ it is

$$\begin{aligned}
M_k &\geq \max_{x \in \mathcal{L}} A^\exists_{k,\star} x - b^\exists_k \\
&= \sum_{\substack{i \in \mathcal{I} \\ A^\exists_{k,i} < 0}} A^\exists_{k,i} \cdot l_i + \sum_{\substack{i \in \mathcal{I} \\ A^\exists_{k,i} \geq 0}} A^\exists_{k,i} \cdot u_i - b^\exists_k \\
&\geq \sum_{\substack{i \in \mathcal{I},\, i \notin \tilde{B} \\ A_{k,i} < 0}} A^\exists_{k,i} \cdot l_i + \sum_{\substack{i \in \mathcal{I},\, i \notin \tilde{B} \\ A_{k,i} \geq 0}} A^\exists_{k,i} \cdot u_i + \sum_{i=1}^{|B_1|} A^\exists_{k,\mu(1,i)} \tilde{x}^{(1)}_i + \sum_{i=1}^{|B_2|} A^\exists_{k,\mu(2,i)} \tilde{x}^{(2)}_i - b^\exists_k \\
&= \sum_{\substack{1 \leq i \leq \tilde{n} \\ \tilde{A}^\exists_{k,i} < 0}} \tilde{A}^\exists_{k,i} \cdot l_{i+|\tilde{B}|} + \sum_{\substack{1 \leq i \leq \tilde{n} \\ \tilde{A}^\exists_{k,i} \geq 0}} \tilde{A}^\exists_{k,i} \cdot u_{i+|\tilde{B}|} - \tilde{b}^\exists_k \;=\; \max_{x \in \tilde{\mathcal{L}}} \tilde{A}^\exists_{k,\star} x - \tilde{b}^\exists_k
\end{aligned}$$

Hence, Constraints (5.16) and (5.17) are valid for $f^{\mathrm{ID}}(\tilde{P})$. Further, for $k \in \{1, \dots, m_\forall\}$ it is

$$\begin{aligned}
\tilde{L}_k &\leq \sum_{\substack{i \in \mathcal{I} \\ A^\forall_{k,i} < 0}} A^\forall_{k,i} \cdot u_i + \sum_{\substack{i \in \mathcal{I} \\ A^\forall_{k,i} \geq 0}}^{n} A^\forall_{k,i} \cdot l_i - b^\forall_k + \tilde{b}^\forall_k \\
&= \sum_{\substack{i \in \mathcal{I} \\ A^\forall_{k,i} < 0}} A^\forall_{k,i} \cdot u_i + \sum_{\substack{i \in \mathcal{I} \\ A^\forall_{k,i} \geq 0}} A^\forall_{k,i} \cdot l_i - \sum_{i=1}^{|B_1|} A^\forall_{k,\mu(1,i)} \tilde{x}^{(1)}_i - \sum_{i=1}^{|B_2|} A^\forall_{k,\mu(2,i)} \tilde{x}^{(2)}_i \\
&\leq \sum_{\substack{1 \leq i \leq \tilde{n} \\ \tilde{A}^\forall_{k,i} < 0}} \tilde{A}^\forall_{k,i} \cdot u_{i+|\tilde{B}|} + \sum_{\substack{1 \leq i \leq \tilde{n} \\ \tilde{A}^\forall_{k,i} \geq 0}} \tilde{A}^\forall_{k,i} \cdot l_{i+|\tilde{B}|}
\end{aligned}$$

and obviously R^{LCD} still fulfills the demanded property. Hence, the entire constraint system (5.16), (5.17), (5.18), (5.19) and (5.20) is valid for $f^{\mathrm{ID}}(\hat{P})$. Further, since

$$\begin{aligned}
&\max_{x \in \tilde{\mathcal{L}}} \tilde{c}^\top x - \tilde{M} + \sum_{i=1}^{|B_1|} c_{\mu(1,i)} \tilde{x}^{(1)}_i + \sum_{i=1}^{|B_2|} c_{\mu(2,i)} \tilde{x}^{(2)}_i \\
&< \min_{x \in \tilde{\mathcal{L}}} \tilde{c}^\top x + \sum_{i=1}^{|B_1|} c_{\mu(1,i)} \tilde{x}^{(1)}_i + \sum_{i=1}^{|B_2|} c_{\mu(2,i)} \tilde{x}^{(2)}_i
\end{aligned}$$

the value of $\tilde{M}$ fulfills the demanded property and therefore $\tilde{R} \in f^{\mathrm{ID}}(\tilde{P})$. □

We are now able to prove the "⇒" implications of all three equivalences of Theorem 5.4.4.

Proof. (Theorem 5.4.4, third equivalence, direction "⇒")
Let P be infeasible, i.e.

$$\forall x^{(1)} \in \mathcal{F}^{(1)} \; \exists x^{(2)} \in \mathcal{F}^{(2)} \; \forall x^{(3)} \in \mathcal{F}^{(3)} \dots \; : A^\exists x \not\leq b^\exists \,.$$

Consider any $R \in f^{\mathrm{ID}}(P)$. Proof by induction in the number of variable blocks β. Remember that by definition the number of variable blocks is even.

Basis: The statement is true for $\beta = 2$.
Case 1: $\mathcal{F}^{(1)} = \emptyset$
With Corollary 5.4.6 and 5.4.9 the constraint system of R cannot be fulfilled. Therefore, R is infeasible.
Case 2: $\mathcal{F}^{(1)} \neq \emptyset$
With Corollary 5.4.9 a fulfilling assignment of the first-stage variables must have the property $x^{(1)} \in \mathcal{F}^{(1)}$. For any $\hat{x}^{(1)} \in \mathcal{F}^{(1)}$ we must select $v^{(1)} \in \mathcal{V}(\hat{x}^{(1)})$ in order to be part of a winning strategy according to Corollary 5.4.8. Since P is infeasible we know $\exists x^{(2)} \in \mathcal{F}^{(2)}(\hat{x}^{(1)}) : A^\exists x \not\leq b^\exists$. Let $\hat{x}^{(2)} \in \mathcal{F}^{(2)}(\hat{x}^{(1)})$ be such an assignment. Therefore,

$$A^\forall \begin{pmatrix} \hat{x}^{(1)} \\ \hat{x}^{(2)} \end{pmatrix} \leq b^\forall$$

and $y^{(2)} = 0$ is enforced as shown in Lemma 5.4.10 and consequently $t_2 = 0$ and $p = 0$. Due to the selection of $\hat{x}^{(2)}$

$$A^{\exists} \begin{pmatrix} \hat{x}^{(1)} \\ \hat{x}^{(2)} \end{pmatrix} - Mp \not\leq b^{\exists}$$

and therefore the universal player can always enforce a violation of Constraint (5.6) and R is infeasible.

Inductive Step: Let the statement be true for any instance with $k \in \mathbb{N}$ stages, $k \geq 2$, k even. Let us consider an instance with $k + 2$ stages.
Case 1: $\mathcal{F}^{(1)} = \emptyset$
With the same argument as above R is infeasible.
Case 2: $\mathcal{F}^{(1)} \neq \emptyset$
In order to have a winning strategy the existential player must select $\hat{x}^{(1)} \in \mathcal{F}^{(1)}$ and $\hat{v}^{(1)} \in \mathcal{V}(\hat{x}^{(1)})$. Since P is infeasible

$$\exists\, x^{(2)} \in \mathcal{F}^{(2)}(\hat{x}^{(1)})\ \forall\, x^{(3)} \in \mathcal{F}^{(3)}\ \exists\, x^{(4)} \in \mathcal{F}^{(4)} \ldots : A^{\exists} x \not\leq b^{\exists}.$$

The universal player can select such $\hat{x}^{(2)} \in \mathcal{F}^{(2)}(\hat{x}^{(1)})$ and $\hat{v}^{(2)} \in \mathcal{V}^{(2}(\hat{x}^{(1)}, \hat{x}^{(2)})$ (following the same reasoning as above). Hence, in a winning strategy the existential variables $y^{(2)}$ and t_2 must be set to 0 (Lemma 5.4.10). As shown in Lemma 5.4.13 the subproblem $\hat{P}$ of P with $x^{(1)} = \hat{x}^{(1)}$ and $x^{(2)} = \hat{x}^{(2)}$ already fixed is a QIP$^{\mathrm{ID}}$ with k stages. Since t_2 must be set to 0 in a potential winning strategy for the subproblem $\hat{R}$ of R with $x^{(1)} = \hat{x}^{(1)}$, $v^{(1)} = \hat{v}^{(1)}$, $x^{(2)} = \hat{x}^{(2)}$,$v^{(2)} = \hat{v}^{(2)}$, $y^{(2)} = 0$ and $t_2 = 0$ fixed it holds $\hat{R} \in f^{\mathrm{ID}}(\hat{P})$ (also Lemma 5.4.13). Since P is infeasible and according to the selection of $x^{(1)} = \hat{x}^{(1)}$ and $x^{(2)} = \hat{x}^{(2)}$ we know:

$$\forall\, x^{(3)} \in \mathcal{F}^{(3)}(\hat{x}^{(1)}, \hat{x}^{(2)})\ \exists\, x^{(4)} \in \mathcal{F}^{(4)} \ldots\ : A^{\exists} x \not\leq b^{\exists}.$$

Hence, $\hat{P}$ is infeasible. Thus, by inductive hypothesis, $\hat{R}$ is also infeasible. Hence,

$$\forall\, x^{(1)}\ \forall\, v^{(1)}\ \exists\, x^{(2)}\ \exists\, v^{(2)}\ \forall\, y^{(2)}\ \forall\, t_2 : \text{no strategy exists to fulfill (5.6)–(5.10)}$$

and therefore R is infeasible. □

Proof. (Theorem 5.4.4, first equivalence, direction "⇒")
Let $P = (A^{\exists}, A^{\forall}, b^{\exists}, b^{\forall}, c, \mathcal{L}, Q)$ have an optimal winning truncated strategy $\mathcal{S}$ with optimal value $\tilde{z} = c^{\top}\tilde{x} \in \mathbb{Q}$ given by the PV $\tilde{x} = (\tilde{x}^{(1)}, \ldots, \tilde{x}^{(\beta)})$. We show that any $R \in f^{\mathrm{ID}}(P)$ has an optimal winning strategy with PV $\hat{o} = (\tilde{x}^{(1)}, v^{(1)}, \tilde{x}^{(2)}, v^{(2)}, y^{(2)}, t_2, \tilde{x}^{(3)}, \ldots, \tilde{x}^{(\beta)}, y^{(\beta)}, t_\beta, p)$, with $p = 0$, $v^{(i)} \in \mathcal{V}(\tilde{x}^{(1)}, \ldots, \tilde{x}^{(i)})$, $y^{(i)} = 0$ and $t_i = 0$ for all $i \in \{1, \ldots, \beta\}$ and optimal value $\bar{z} = \tilde{z}$. First we show that a winning strategy for $R \in f^{\mathrm{ID}}(P)$ exists with value $\tilde{z}$. Since $\tilde{z} \in \mathbb{Q}$ the PV $\tilde{x} = (\tilde{x}^{(1)}, \ldots, \tilde{x}^{(\beta)})$ is represented by a leaf $v \in V_L$ within the winning truncated strategy of P. In particular, $A^{\forall}\tilde{x} \leq b^{\forall}$ and $A^{\exists}\tilde{x} \leq b^{\exists}$. Therefore, $\mathcal{V}(\tilde{x}^{(1)}, \ldots, \tilde{x}^{(i)}) \neq \emptyset$ for all $i \in \{1, \ldots, \beta\}$. Thus, $\hat{o}$ as given above obviously satisfies the constraint system of any $R \in f^{\mathrm{ID}}(P)$ with objective

value $c^\top \tilde{x} - \tilde{M}p = \tilde{z}$. The remaining branches of the strategy are built inductively as follows depending on the node type:

Root node:
At the existential root node the children are selected as in the PV $\hat{o}$: $\tilde{x}^{(1)}$ and $v^{(1)}$ as described above. Hence, this path represents a valid variable assignment until stage 1.

Universal node representing $x^{(k)}$, $k \in \mathcal{A}$:
Let us consider a universal node emerging from a valid variable assignment until stage $k-1$ with $t_{k-2} = 0$. For children representing $x^{(k)} \notin \mathcal{F}^{(k)}$ we know with Corollary 5.4.12 that a winning strategy exists such that the constraint system of R is fulfilled resulting in an objective value less than $\min_{x \in \mathcal{L}} c^\top x$. For any $x^{(k)} \in \mathcal{F}^{(k)}$ we then must consider any selection of $v^{(k)}$. For $v^{(k)} \notin \mathcal{V}(x^{(1)}, \dots, x^{(k)})$ we know from Lemma 5.4.11 that a winning strategy exists with value less than $\min_{x \in \mathcal{L}} c^\top x$. For $v^{(k)} \in \mathcal{V}(x^{(1)}, \dots, x^{(k)})$ the subsequent existential decision in a winning strategy for R must be $y^{(k)} = 0$ and $t_k = 0$. If $k = \beta$ we also set $p = 0$. Therefore, only existential nodes resulting from paths representing valid variable assignments until stage k with $t_k = 0$, $x^{(k)} \in \mathcal{F}^{(k)}$ and $v^{(k)} \in \mathcal{V}(x^{(1)}, \dots, x^{(k)})$ need to be considered.

Existential node representing $x^{(k)}$, $k \in \mathcal{E}$:
Let us consider an existential node emerging from a valid variable assignment $(\bar{x}, \bar{v}, \bar{y}, \bar{t})_{k-1}$ until stage $k-1$ with $t_{k-1} = 0$, $x^{(k-1)} \in \mathcal{F}^{(k-1)}$ and $v^{(k)} \in \mathcal{V}(x^{(1)}, \dots, x^{(k)})$. Hence, $\bar{x}^{(i)} \in \mathcal{F}^{(i)}(\bar{x}^{(1)}, \dots, \bar{x}^{(i-1)})$ for all $i \in \{1, \dots, k-1\}$ and a node in the truncated strategy for P exists representing this variable assignment. As successor we select the path representing $x^{(k)}$ as in $\mathcal{S}$ and any $v^{(k)} \in \mathcal{V}(\bar{x}^{(1)}, \dots, \bar{x}^{(k-1)}, x^{(k)}) \neq \emptyset$. Hence, this path represents a valid variable assignment until stage k with $t_{k-2} = 0$.

Note that all leaves of a strategy built as described above represent variable assignments of R with objective values less than or equal to $\tilde{z}$, since they either represent variable assignments with $p = 1$ or a variable assignment with $p = 0$ and x as in some path of $\mathcal{S}$. Hence, the minimax value of the built strategy is $\tilde{z}$.

Let now $\hat{o} = (\hat{x}, \hat{v}, \hat{y}, \hat{t}, \hat{p})$ be the PV of the *optimal* winning strategy $\tilde{\mathcal{O}}$ of R and thus $c^\top \hat{x} - \tilde{M}\hat{p} \leq c^\top \tilde{x} = \tilde{z}$. If $A^\forall \hat{x} \not\leq b^\forall$ it holds $\hat{p} = 1$. However, since $\max_{x \in \mathcal{L}} c^\top x - \tilde{M} < \min_{x \in \mathcal{L}} c^\top x$ the resulting value of the objective function is less than any other path obeying $A^\forall x \leq b^\forall$. This is a contradiction to the minimax optimality of $\hat{o}$ since there exists a strategy for assigning the universal variables such that $A^\forall x \leq b^\forall$ is fulfilled: if there were no such strategy there would be a destructive winning strategy for the existential player and thus the optimal value of P would be $-\infty$. Thus, $A^\forall \hat{x} \leq b^\forall$. Hence, $\hat{p} = 0$, $\hat{y}^{(i)} = \bar{0}$ and $\hat{t}_i = 0$ for all $i \in \mathcal{A}$ and consequently $A^\exists \hat{x} \leq b^\exists$. Thus, $\hat{x}$ represents a legal path from the root to a leaf of the game tree of P. Further, all other leaves of $\tilde{\mathcal{O}}$ have values less than or equal to $\hat{z}$. Hence, for any leaf $(\bar{x}, \bar{v}, \bar{y}, \bar{t}, \bar{p})$ of $\tilde{\mathcal{O}}$ it is either

- $A^\forall \bar{x} \leq b^\forall$ and $A^\exists \bar{x} \leq b^\exists$

- or $\underset{i\in\{1,\dots,\beta\}}{\arg\min}\{\mathcal{F}^{(i)}(\bar{x}^{(1)},\dots,\bar{x}^{(i-1)})=\emptyset\}\in\mathcal{A}$ and $\bar{p}=1$.

Thus, $\tilde{\mathcal{O}}$ can be converted to a winning truncated strategy for P by adopting the branches representing legal moves. Therefore, $\bar{z}\geq\tilde{z}$ and consequently $\bar{z}=\tilde{z}$. □

Proof. (Theorem 5.4.4, second equivalence, direction "⇒")
Let there be a destructive winning truncated strategy S for P. We show that in any leaf of the optimal winning strategy for $R\in f^{\mathrm{ID}}(P)$ the objective value of the corresponding variable assignment is less than $\min_{x\in\mathcal{L}} c^\top x$. Therefore, we build a strategy using the same procedure as in the proof of the first equivalence. Any path from the root to a leaf in this built strategy represents a variable assignment resulting in an objective value less than $\min_{x\in\mathcal{L}} c^\top x$, as there always exists a stage where no verification vector $v^{(k)}\in\mathcal{V}$ can be found. Thus, a winning strategy exists with PV less than $\min_{x\in\mathcal{L}} c^\top x$. Therefore, the optimal winning strategy for R must also have a PV with objective value less than $\min_{x\in\mathcal{L}} c^\top x$. □

Proof. ("⇐" implications of Theorem 5.4.4)
The proof of the "⇐" implications shortly works as follows: Let $A\Leftrightarrow X$ be the abbreviation for Theorem 5.4.4 item 1, $B\Leftrightarrow Y$ for item 2 and $C\Leftrightarrow Z$ for item 3. Since the cases A, B, C and X, Y, Z, respectively, are disjoint, we additionally know[10] $A\veebar B\veebar C$ and $X\veebar Y\veebar Z$. Surprisingly, as a consequence the proof is complete: For example in order to show $X\Rightarrow A$ it suffices to show $\neg A\Rightarrow\neg X$. $\neg A$ implies $B\veebar C$ which itself implies $Y\veebar Z$ (as shown in the proofs before) and hence $\neg X$. □

In order to constitute a polynomial-time reduction function we also must show that one can find an element from $f^{\mathrm{ID}}(P)$ within polynomial time regarding the input size.

Theorem 5.4.14. *Let $P=(A^\exists,A^\forall,b^\exists,b^\forall,c,\mathcal{L},Q)$ be a given QIP^{ID}. An element of the set $f^{ID}(P)$ of the reduction function given in Definition 5.4.2 can be computed in polynomial time with respect to the input size.*

Proof. The size of the input $P=(A^\exists,A^\forall,b^\exists,b^\forall,c,\mathcal{L},Q)$ only depends on the number of variables n and the number of constraints of both systems $m_\exists$ and $m_\forall$. Obviously, values for the vectors L and M and the value for $\tilde{M}$ can be computed in polynomial time with respect to n, $m_\exists$ and $m_\forall$ by computing the bounds given in (5.11), (5.12) and (5.14), respectively. For the entries of R^{LCD} it is not necessary to find the *lowest* common denominator, i.e. it suffices to simply multiply the denominators of the non-zero entries of row k and take its reciprocal for R_k^{LDC}, which can be computed in polynomial time. If the denominators are not at hand it suffices to determine the largest number of decimal places d of the entries in row k and select $R_k^{LDC}=10^{-d}$. The number of variables in the resulting QIP is in $\mathcal{O}(n+n^2+n\cdot m_\forall)$: $\mathcal{O}(n)$ x and t variables, $\mathcal{O}(n^2)$ v variables, $\mathcal{O}(n\cdot m_\forall)$ y variables and one p variable. The number of constraints is $\mathcal{O}(n\cdot m_\exists+n\cdot m_\forall)$: $\mathcal{O}(n\cdot m_\exists)$ constraints in (5.6), $m_\exists$ constraints in (5.7), $\mathcal{O}(n\cdot m_\forall)$ in (5.8)

[10] $\veebar$ is used for the exclusive disjunction.

and 1 and $\mathcal{O}(n)$ constraints in (5.9) and (5.10), respectively. Hence, an element of $f^{\mathrm{ID}}(P)$ can be computed in polynomial time. □

We have shown the connection between the solution of a QIP$^{\mathrm{ID}}$ P with the solution of any element in $f^{\mathrm{ID}}(P)$ in Theorem 5.4.4 and with Theorem 5.4.14 we proved that the mapping of P to some element in $f^{\mathrm{ID}}(P)$ can be done in polynomial time. Together with the trivial claim QIP $\leq_p$ QIP$^{\mathrm{ID}}$ this shows that QIP$^{\mathrm{ID}}$ is indeed PSPACE-complete.

5.5. Relaxations

As already outlined in in Section 3.2, relaxations are a powerful tool to generate bounds on the optimal value of a (sub)problem and sometimes even to find valid solutions. Most relaxations presented in Section 3.2, however, are only applicable under very specific circumstances for QIP$^{\mathrm{ID}}$. For example, the LP-relaxation that only considers the existential constraint system, generally does not produce any usable information for a QIP$^{\mathrm{ID}}$ as its objective value does not necessarily constitute a bound on the optimal value and not even its infeasibility allows unequivocal conclusions. We propose several relaxations for QIP$^{\mathrm{ID}}$ and investigate what conclusions can be drawn from them. The relaxations are obtained by

- relaxing the integrality of variables,
- altering the quantification of variables,
- alternating the order of the variables,
- ignoring the interdependence,
- omitting one of the constraint systems, or
- shrinking the set of considered scenarios.

First, we investigate relaxations that arise by clustering equally quantified variables, ignoring the interdependence of variable domains and further omitting one of the constraint systems. Four relaxations arise, each containing only two variable blocks. We start with the two relaxations scrutinizing the satisfiability of $A^{\exists}x \leq b^{\exists}$ in Definitions 5.5.1 and 5.5.3 and then present in Definitions 5.5.6 and 5.5.7 relaxations that focus on the fulfillment of the universal constraint system.

Definition 5.5.1 ($(\exists\forall)^{\exists}$-Relaxation)**.**
For a QIP^{ID} $P = (A^{\exists}, A^{\forall}, b^{\exists}, b^{\forall}, c, \mathcal{L}, Q)$ the QIP

$$\min_{x_{\exists}\in\mathcal{L}_{\exists}} \left(c_{\exists}x_{\exists} + \max_{x_{\forall}\in\mathcal{L}_{\forall}} (c_{\forall}x_{\forall}) \right) \qquad \text{s.t. } \exists\, x_{\exists} \in \mathcal{L}_{\exists}\ \forall\, x_{\forall} \in \mathcal{L}_{\forall} :\ A^{\exists}x \leq b^{\exists}$$

is called its $(\exists\forall)^{\exists}$-relaxation.

For the $(\exists\forall)^{\exists}$-relaxation the variable sequence is altered such that the existential variables must be set first within their rigid bounds $\mathcal{L}_{\exists}$ (in particular not within some interdependent

domain) and subsequently the universal variables are assigned. Thus, a single fixation of the existential variables $x_\exists$ must exist, such that the existential constraint system holds for all assignments of the universal variables. The universal constraint system is completely neglected.

Proposition 5.5.2. *Given a QIP^{ID} P and its $(\exists\forall)^\exists$-relaxation R. If R is feasible with optimal first-stage solution $\tilde{x}_\exists$ and optimal value $\tilde{z}_R$ then P is also feasible with optimal value $\tilde{z}_P \leq \tilde{z}_R$.*

Proof. As R is feasible with first-stage solution $\tilde{x}_\exists$ the following holds:

$$\forall x_\forall \in \mathcal{L}_\forall : \; A^\exists_\forall x_\forall \leq b^\exists - A^\exists_\exists \tilde{x}_\exists. \tag{5.21}$$

Consider the truncated strategy $T = (V', E', e')$ arising from applying the variable assignment at existential nodes as in $\tilde{x}_\exists$. This truncated strategy constitutes a winning truncated strategy: For each existential node $v_\exists \in V' \cap V_\exists$ it is $\mathcal{F}(v_\exists) \neq \emptyset$ because of (5.21) and furthermore for each leaf $v_L \in V_L \cap V'$ it is $A^\exists x_{v_L} \leq b^\exists$ and thus $minimax_e(v_L) \in \{c^\top x_{v_L}, -\infty\}$. As x_{v_L} also corresponds to a leaf in the optimal winning strategy for R we know $minimax_e(v_L) \leq \tilde{z}_R$. Furthermore, for any other terminal node $\hat{v} \in \mathcal{T}(T) \setminus V_L$ it is $\hat{v} \in V_\forall$ with $minimax_e(\hat{v}) = -\infty$. Hence, for any terminal node $v \in \mathcal{T}(T)$ it is $minimax_e(v) \leq \tilde{z}_R$ and therefore $minimax_e(T) \leq \tilde{z}_R$ with Theorem 5.3.16. □

Note that this relaxation is closely linked to SCP (see Section 3.1.3 and 5.6.3), as a single existential variable assignment is searched such that the existential constraint system is fulfilled, regardless of the universal variable assignment. Unfortunately, the infeasibility of an $(\exists\forall)^\exists$-relaxation provides no useful information about the underlying QIP$^{\text{ID}}$ and even the decision problem of finding such an existential variable assignment remains NP-complete [Wol15].

The following $(\forall\exists)^\exists$-relaxation differs from the $(\exists\forall)^\exists$-relaxation in the order of the variables.

Definition 5.5.3 ($(\forall\exists)^\exists$-Relaxation)**.**
For a QIP^{ID} $P = (A^\exists, A^\forall, b^\exists, b^\forall, c, \mathcal{L}, Q)$ the QIP

$$\max_{x_\forall \in \mathcal{L}_\forall} \left(c_\forall x_\forall + \min_{x_\exists \in \mathcal{L}_\exists} (c_\exists x_\exists) \right) \qquad \text{s.t. } \forall\, x_\forall \in \mathcal{L}_\forall \; \exists\, x_\exists \in \mathcal{L}_\exists : \; A^\exists x \leq b^\exists$$

is called its $(\forall\exists)^\exists$-relaxation.

Again, the universal constraint system is neglected in the $(\forall\exists)^\exists$-relaxation and for any assignment of the universal variables it suffices to find an assignment of the existential variables such that $A^\exists x \leq b^\exists$. In constrast to the $\forall\exists$-relaxation of a QIP (see [Wol15]), one cannot conclude the infeasibility of a QIP$^{\text{ID}}$ instance, if its $(\forall\exists)^\exists$-relaxation is infeasible.

Example 5.5.4. *Let $n = 3$, $\mathcal{L} = \{0,1\}^3$, $Q = (\exists, \forall, \exists)$ and $c = 0$. The universal constraint system only contains $x_1 + x_2 \leq 1$. The two existential constraints are $x_1 + x_2 \leq 1$ and $x_1 + x_3 \geq 2$. The $(\forall\exists)^\exists$-relaxation of this QIP^{ID} is infeasible, as $\nexists x_1, x_3 \in \{0,1\} : x_1 \leq 0$ and $x_1 + x_3 \geq 2$. The QIP^{ID} itself, however, is feasible with first-stage solution $\tilde{x}_1 = 1$, since $\mathcal{F}^{(2)}(\tilde{x}_1) = \{0\} \neq \mathcal{L}^{(2)}$.*

Furthermore, the optimal value of a feasible $(\forall\exists)^\exists$-relaxation generally does not yield a lower bound on the optimal value of an underlying feasible QIP$^{\text{ID}}$.

Example 5.5.5. *Let $n = 4$, $\mathcal{L} = \{0,1\}^4$, $Q = (\exists, \forall, \exists, \forall)$ and $c = (0, 1, -1, 0)$. The universal constraint system only contains $x_2 + x_3 \leq 1$. The only existential constraint is $x_1 + x_2 + x_3 \geq 2$. By the way: x_4 is just an auxiliary universal variable in order to detect a violation caused by x_3. The $(\forall\exists)^{\exists}$-relaxation of this QIP^{ID} is feasible, with first-stage solution $\tilde{x}_\forall = (1, 0)$ and optimal value 0. The optimal value of the QIP^{ID}, however, is -1.*

So far the relaxations aimed at scrutinizing the fulfillment of the existential constraint system. Similar relaxations can be defined with focus on $A^\forall x \leq b^\forall$. For the sake of completeness we define the $(\forall\exists)^{\forall}$-relaxation, even though—similar to the $(\forall\exists)^{\exists}$-relaxation—it also does not yield any useful information for the underlying QIP$^{\text{ID}}$ in general.

Definition 5.5.6 ($(\forall\exists)^{\forall}$-Relaxation)**.**
For a QIP^{ID} $P = (A^\exists, A^\forall, b^\exists, b^\forall, c, \mathcal{L}, Q)$ the QIP

$$\min_{x_\exists \in \mathcal{L}_\exists} \left(c_\exists x_\exists + \max_{x_\forall \in \mathcal{L}_\forall} (c_\forall x_\forall) \right) \qquad \text{s.t. } \forall\, x_\exists \in \mathcal{L}_\exists\ \exists\, x_\forall \in \mathcal{L}_\forall :\ A^\forall x \leq b^\forall$$

is called its $(\forall\exists)^{\forall}$-relaxation.

The $(\forall\exists)^{\forall}$-relaxation has similar properties as the $(\forall\exists)^{\exists}$-relaxation: one cannot conclude the feasibility (infeasibility) of a QIP$^{\text{ID}}$ if its $(\forall\exists)^{\forall}$-relaxation is feasible (infeasible) and no general statements about upper bounds can be made. But by solving its counterpart—presented in the following definition—valuable information can be obtained.

Definition 5.5.7 ($(\exists\forall)^{\forall}$-Relaxation)**.**
For a QIP^{ID} $P = (A^\exists, A^\forall, b^\exists, b^\forall, c, \mathcal{L}, Q)$ the QIP

$$\max_{x_\forall \in \mathcal{L}_\forall} \left(c_\forall x_\forall + \min_{x_\exists \in \mathcal{L}_\exists} (c_\exists x_\exists) \right) \qquad \text{s.t. } \exists\, x_\forall\ \in \mathcal{L}_\forall\ \forall\, x_\exists \in \mathcal{L}_\exists :\ A^\forall x \leq b^\forall$$

is called its $(\exists\forall)^{\forall}$-relaxation.

Proposition 5.5.8. *Given a QIP^{ID} P and its $(\exists\forall)^{\forall}$-relaxation R. If R is feasible then either P is infeasible or for the optimal value $\tilde{z}_P$ of P $\tilde{z}_P \neq -\infty$ applies.*

Proof. Let $G = (V, E, e)$ be the game tree of P and let $\tilde{x}_\forall$ be the optimal first-stage solution of R. We show, that for any truncated strategy $T = (V', E', e')$ of P there exists a terminal node $w \in \mathcal{T}(T)$ with $minimax_e(w) \neq -\infty$. This is done by traversing T along the (unique) path arising from selecting edges according to the universal decisions as in $\tilde{x}_\forall$. We start at level $\ell = 0$ and the root node $w_0 \in V_\exists$. If $\mathcal{F}(w_0) = \emptyset$ it is $w_0 \in \mathcal{T}(T)$ and with Lemma 5.3.15 $minimax_e(w_0) = +\infty$. If $\mathcal{F}(w_0) \neq \emptyset$ there exists exactly one $w_1 \in V'$ with $(w_0, w_1) \in E'$. The remaining path is now constructed inductively. Let $0 < \ell \leq n$ be the current level and w_ℓ the current node.

- If $w_\ell \in V_\exists$ it is either $\mathcal{F}(w_\ell) = \emptyset$ and thus $w_\ell \in \mathcal{T}(T)$ resulting in $minimax_e(w_\ell) = +\infty$, or we select the unique successor node $w_{\ell+1} \in V'$.

- If $w_\ell \in V_\forall$ it is $\mathcal{F}(w_\ell) \neq \emptyset$ since previous universal variables were set according to $\tilde{x}_\forall$. We select $w_{\ell+1} \in V'$ according to the universal decision as in $\tilde{x}_\forall$.
- If $w_\ell \in V_L$ it is $(x_{w_\ell})_{\mu_\forall(k)} = (\tilde{x}_\forall)_k$ for each $k \in \{1, \dots, n_\forall\}$, i.e. each universal variable is set as in $\tilde{x}_\forall$ along the path from the root to w_ℓ by construction. Since $\tilde{x}_\forall$ is a first-stage solution of R it is $A^\forall x_{w_\ell} \leq b^\forall$ and hence $minimax_e(w_\ell) \neq -\infty$.

Therefore, for any truncated strategy $T = (V', E', e')$ of P there exists a terminal node $w \in \mathcal{T}(T)$ with $minimax_e(w) \neq -\infty$ and with Theorem 5.3.16 $minimax_e(T) \neq -\infty$. □

We now exploit the set of first-stage solutions of the $(\exists\forall)^\forall$-relaxation, because such universal variable assignments can also be used in order to obtain a bound on the optimal value of a feasible QIP$^{\text{ID}}$. We call such universal variable assignments *unavoidable scenarios*, since the existential player is not able to remove these assignments from the set of legal moves.

Definition 5.5.9 (Set of Unavoidable Scenarios)**.**
For a QIP^{ID} $P = (A^\exists, A^\forall, b^\exists, b^\forall, c, \mathcal{L}, Q)$ we call

$$\mathcal{U} = \left\{ x_\forall \in \mathcal{L}_\forall \mid \forall x_\exists \in \mathcal{L}_\exists : \ A^\forall x \leq b^\forall \right\}$$

the set of unavoidable scenarios.

Obviously, with Proposition 5.5.8, if $\mathcal{U} \neq \emptyset$ there exists a winning truncated universal strategy and the extended minimax value of the root note is not $-\infty$. Such unavoidable scenarios can now be used in an adapted $(\forall\exists)^\exists$-relaxation.

Definition 5.5.10 (U-Relaxation)**.**
Given a QIP^{ID} P and let $\emptyset \neq U \subseteq \mathcal{U}$. We call

$$\max_{x_\forall \in U} \left(c_\forall x_\forall + \min_{x_\exists \in \mathcal{L}_\exists} (c_\exists x_\exists) \right) \qquad \text{s.t. } \forall x_\forall \in U \ \exists\, x_\exists \in \mathcal{L}_\exists : \ A^\exists x \leq b^\exists$$

a U-relaxation *of* P.

A U-relaxation is related to the $(\forall\exists)^\exists$-relaxation with the important difference that useful information can be obtained from solving a U-relaxation.

Proposition 5.5.11. *Given a feasible QIP^{ID} P. Consider a U-relaxation R with $\emptyset \neq U \subseteq \mathcal{U}$ and optimal value $\tilde{z}_R$. Then $\tilde{z}_R$ is a lower bound on the optimal value $\tilde{z}_P$ of P, i.e. $\tilde{z}_R \leq \tilde{z}_P$.*

Proof. Since P is feasible R is also feasible and $\tilde{z}_R \neq -\infty$ since $U \neq \emptyset$. Let $\tilde{x}_\forall$ be the optimal first-stage solution of R. Further, as $\mathcal{U} \neq \emptyset$, obviously the $(\exists\forall)^\forall$-relaxation of P is feasible and $\tilde{z}_P \neq -\infty$ with Proposition 5.5.8. In the optimal winning truncated strategy $T = (V', E', e')$ for P there exists a path from the root to a terminal node $v \in \mathcal{T}(T) \cap V_L$ with the universal decisions as in $\tilde{x}_\forall$ (Construction as in Proof 5.5.8) and

$$\begin{aligned} minimax_e(v) \ = \ c^\top x_v \ &= \ c_\forall (x_v)_\forall + c_\exists (x_v)_\exists \\ &= \ c_\forall \tilde{x}_\forall + c_\exists (x_v)_\exists \ \geq \ c_\forall \tilde{x}_\forall + \min_{x_\exists \in \mathcal{L}_\exists :\ A^\exists x \leq b^\exists} c_\exists x_\exists \ = \ \tilde{z}_R \,. \end{aligned}$$

Therefore, with Theorem 5.3.16, $\tilde{z}_P \geq minimax_e(v) \geq \tilde{z}_R$. □

Proposition 5.5.12. *Given a QIP^{ID} P and a U-relaxation R with $\emptyset \neq U \subseteq \mathcal{U}$. If R is infeasible then also P is infeasible.*

Proof. If R is infeasible there is $\tilde{x}_\forall \in U$ such that $\forall x_\exists \in \mathcal{L}_\exists : \ A^\exists x \not\leq b^\exists$. Consider the game tree $G = (V, E, e)$ of P and any path $r, v_1, \ldots, v_n$ from the root to a leaf $v_n \in V_L$ with the universal variables along the path being assigned as in $\tilde{x}_\forall$. Then $\mathcal{F}(v_{i-1}) \neq \emptyset$ for each $i \in \mathcal{I}_\forall$ because $\tilde{x}_\forall \in U$. Furthermore, $A^\exists x_{v_n} \not\leq b^\exists$ and thus $minimax_e(v_n) = +\infty$. Since any truncated strategy must contain such a (partial) path there cannot be a winning truncated strategy in G. □

Remark 5.5.13. *For a QIP^{ID} $P = (A^\exists, A^\forall, b^\exists, b^\forall, c, \mathcal{L}, Q)$ with $A^\forall = 0$ and $b^\forall = 0$, i.e. a basic QIP, it is $\mathcal{U} = \mathcal{L}_\forall$. Thus, one obtains a valid U-relaxation by selecting any subset of the set of scenarios $\mathcal{L}_\forall$ for U, which is similar to the $\forall\exists$-S-relaxation (see Definition 3.2.5).*

If all entries in $A^\forall$ corresponding to existential variables are zero, the QIP^{ID} is a QIP^{PU}. In this case $\mathcal{U} = \mathcal{D}$. Hence, in case of a QIP^{PU}, relaxations that take unavoidable scenarios into account can be particularly useful, as the set of unavoidable scenarios is potentially more accessible.

We can also relax a QIP$^{\text{ID}}$ by entirely dropping the quantification of the variables, or rather by turning every quantifier in to an existential quantifier. This way we are no longer interested in strategies but in particular variable assignments from which we want to draw conclusions.

Definition 5.5.14 (IP-q-Relaxation)**.**
For a QIP^{ID} $P = (A^\exists, A^\forall, b^\exists, b^\forall, c, \mathcal{L}, Q)$ and $q \in \{\exists, \forall\}$ its IP-q-relaxation *is given by*

$$\begin{aligned} \min \ & \mathbb{1}_\exists(q) c^\top x - \mathbb{1}_\forall(q) c^\top x \\ \text{s.t.} \ & A^q x \leq b^q \\ & x \in \mathcal{L}\,, \end{aligned}$$

with indicator function $\mathbb{1}_{\tilde{q}}(q) = 1$, if $q = \tilde{q}$ and $\mathbb{1}_{\tilde{q}}(q) = 0$, otherwise.

In terms of games the IP-$\exists$-relaxation can be interpreted as a single player game, where the existential player both sets his own variables, as well as the variables of the universal player, trying to fulfill the existential constraint system and minimizing the objective value. Relaxing the integrality of the variables yields the LP-q-relaxation.

Definition 5.5.15 (LP-q-Relaxation)**.**
For a QIP^{ID} $P = (A^\exists, A^\forall, b^\exists, b^\forall, c, \mathcal{L}, Q)$ and $q \in \{\exists, \forall\}$ its LP-q-relaxation *is given by*

$$\begin{aligned} \min \ & \mathbb{1}_\exists(q) c^\top x - \mathbb{1}_\forall(q) c^\top x \\ \text{s.t.} \ & A^q x \leq b^q \\ & l \leq x \leq u\,. \end{aligned}$$

Unfortunately, within a game tree search, both the IP-q- and LP-q-relaxation do not yield a lower bound or allow extensive conclusions about the feasibility of the QIP$^{\text{ID}}$ in general. With Condition (5.1) an infeasible IP- or LP-$\exists$-relaxation yields the infeasibility of the QIP$^{\text{ID}}$ instance, i.e. applying this relaxation at the root during a tree search is reasonable. However, as Condition (5.1) is only valid at the root, such a relaxation can no longer be interpreted unambiguously within the search tree: if at an arbitrary inner game tree node $v \in V$ the corresponding IP- or LP-$\exists$-relaxation is infeasible one can only conclude $minimax_e(v) \in \{\pm\infty, +\infty\}$. In order to rule out $\pm\infty$ the satisfiability of the universal constraint system must be shown, i.e. the feasibility of the IP-$\forall$-relaxation at this node must be checked.

In case of a feasible LP-$\exists$-relaxation one would hope for a lower bound on the optimal value. Unfortunately, this is not the case either as $z_{LP} \in \mathbb{Q}$, whereas the underlying QIP$^{\text{ID}}$ might still constitute a loss for the universal player, i.e. the optimal value could be $-\infty$. This is also true at the root node. Nevertheless, there are some special cases where an IP-q- or LP-q-relaxation can yield more valuable results.

Proposition 5.5.16. *Given a QIP^{ID} P and its game tree $G = (V, E, e)$. Consider an existential node $v \in V_\exists$. Let the corresponding IP-$\exists$-relaxation (with previous variables fixed according to the edge labels along the path from the root to v) be infeasible. If the corresponding IP-$\forall$-relaxation is feasible then $minimax_e(v) = +\infty$.*

Proof. As the IP-$\exists$-relaxation is infeasible it is $\mathcal{F}(v) = \emptyset$ and no legal move for the existential player exists. Hence, $minimax_e(v) \in \{\pm\infty, +\infty\}$. With the feasibility of the IP-$\forall$-relaxation there exists some leaf node $w \in V_L$ below v with $A^\forall x_w \leq x^\forall$. With Corollary 5.3.12 $minimax_e(v) \neq \pm\infty$. □

Proposition 5.5.17. *Given a QIP^{ID} P and its IP-$\exists$-relaxation R. Let $\mathcal{U} \neq \emptyset$ and let R be feasible with optimal value $\tilde{z}_R$. Then either P is infeasible or $\tilde{z}_R$ is a lower bound on the optimal value $\tilde{z}_P$ of P, i.e. $\tilde{z}_R \leq \tilde{z}_P$.*

Proof. With $\mathcal{U} \neq \emptyset$ and Proposition 5.5.8 it is $\tilde{z}_P \neq -\infty$. Let $\tilde{x}_\forall \in \mathcal{U}$. For any leaf $v \in V_L \subseteq V$ in the game tree $G = (V, E, e)$ of P with $(x_v)_\forall = \tilde{x}_\forall$ it is $minimax_e(v) \neq -\infty$. A winning truncated strategy for P must contain a path $r, v_1 \ldots, v_\ell$ from the root r to a terminal node $v_\ell \in \mathcal{T}(G)$ with universal decisions as in $\tilde{x}_\forall$. Since $\tilde{x}_\forall \in \mathcal{U}$ it is $\mathcal{F}(v_{i-1}) \neq \emptyset$ for each $i \in \mathcal{I}_\forall$, $i \leq \ell$. Hence, either $v_\ell \in V_\exists$ and thus $minimax_e(v_\ell) = +\infty > \tilde{z}_R$ or $v_\ell \in V_L$. If, in the latter case, $A^\exists x_{v_\ell} \not\leq b^\exists$ then $minimax_e(v_\ell) = +\infty > \tilde{z}_R$ and if $A^\exists x_{v_\ell} \leq b^\exists$ then $minimax_e(v_\ell) = c^\top x_{v_\ell} \geq \min_{v \in V_L:\ A^\exists x_v \leq b^\exists} c^\top x_v = \tilde{z}_R$. With Theorem 5.3.16 $\tilde{z}_P \geq \tilde{z}_R$. □

Remark 5.5.18. *If $\mathcal{U} \neq \emptyset$ the IP-$\exists$-relaxation can be strengthened by assigning the universal variables according to an unavoidable scenarios within the relaxation explicitly. This is similar to the LP-relaxation with fixed scenarios for QIPs (Definition 3.2.4). Hence, rather than letting the LP decide on the assignment of universal variables in a best-case manner the universal variables are fixed to some (hopefully worst-case) scenario, for which a solution is sought.*

If $\mathcal{U} = \emptyset$ incorporating information of both constraint systems into the relaxation can be beneficial in order receive useful bounds on the optimal value. The following IP-($\exists\wedge\forall$)-relaxation is probably the most straightforward relaxation that does this, but it can only be used to a limited extend.

Definition 5.5.19 (IP-($\exists \wedge \forall$)-Relaxation of QIP$^{\text{ID}}$)**.**
For a QIP^{ID} $P = (A^{\exists}, A^{\forall}, b^{\exists}, b^{\forall}, c, \mathcal{L}, Q)$ its IP-($\exists \wedge \forall$)-relaxation *is given by*

$$\begin{aligned} \min\ & c^{\top}x \\ \text{s.t. } & A^{\exists}x \leq b^{\exists} \wedge A^{\forall}x \leq b^{\forall} \\ & x \in \mathcal{L}\,. \end{aligned}$$

In order to be able to use this relaxation the existence of a winning truncated universal strategy must be ensured.

Proposition 5.5.20. *Given a QIP^{ID} P and its IP-($\exists \wedge \forall$)-relaxation R. Let there be a winning truncated universal strategy for P, i.e. for the optimal value $\tilde{z}$ of P we know $\tilde{z}_P \in \mathbb{Q} \cup \{+\infty\}$.*

a) If R is infeasible, then P is also infeasible.

b) If R is feasible with optimal value $\tilde{z}_R$ then, $\tilde{z}_P \geq \tilde{z}_R$ with $\tilde{z}_P = minimax_e(r)$.

Proof. Let $G = (V, E, e)$ be the game tree corresponding to P.

a) If R is infeasible, then for each leaf $v \in V_L$ in G it holds $w(v) \in \{+\infty, -\infty, \pm\infty\}$. Since there is a winning truncated universal strategy for P the root node has the extended minimax value $+\infty$ and hence P is infeasible.

b) If R is feasible then $\tilde{z}_R = \min\{w(v) \mid v \in V_L \text{ and } w(v) \in \mathbb{Q}\}$ (see Definition 5.3.11). If P is infeasible $+\infty = \tilde{z}_P \geq \tilde{z}_R$. If P is feasible then $\tilde{z}_P \in \mathbb{Q}$, since there exists a winning truncated universal strategy for P. Therefore, with Theorem 5.3.16, it is $\tilde{z}_P \geq \tilde{z}_R$. □

By relaxing the integrality of the IP-($\exists \wedge \forall$)-relaxation we obtain the LP-($\exists \wedge \forall$)-relaxation, which has the same properties as described in Proposition 5.5.20.

All presented relaxations have advantages and disadvantages. Some can only be applied in a rather restricted setting, while others remain generally hard to solve in terms of complexity theory. Nevertheless, exploiting information gathered through relaxations can massively boost the search process. In Subsection 146 we discuss the relaxations implemented in our solver.

5.6. Solution Techniques for QIP$^{\text{ID}}$

5.6.1. Use of a Partial Deterministic Equivalent Program

In contrast to QIP, for QIP$^{\text{ID}}$ it is not straightforward possible to build a DEP. The problem is that the legality of each variable block (both existential and universal) depends on previous variable assignments (again both existential and universal). Further, there is only a set of

potential scenarios $\mathcal{L}_\forall$ and a scenario can only take place under certain conditions: it must be created by legal universal moves from $\mathcal{F}$. Due to the interdependence of universal and existential legal variable assignments it is also not sufficient to check a posteriori whether a scenario was legal. Furthermore, one cannot allow arbitrary existential variable assignment from $\mathcal{L}_\exists$ as their legality might depend on the present scenario and previous existential decisions. The detour via the reduced QIP and the DEP thereof, however, yields a legitimate equivalent. The reduced QIP assures that illegal existential moves immediately result in the unsatisfiability of the constraint system, whereas illegal universal moves result in the trivial satisfiability of the constraint system and an undesirable small objective objective. Thus, illegal moves are possible in principle, but are obviously detrimental compared to legal moves.

Definition 5.6.1 (Deterministic Equivalent Program of a QIP$^{\text{ID}}$)**.**
Let $P = (A^\exists, A^\forall, b^\exists, b^\forall, c, \mathcal{L}, Q)$ be a QIP^{ID} and let $R \in f^{ID}(P)$ be a QIP as given in Definition 5.4.2. We call the DEP of R (see Definition 3.1.2) the deterministic equivalent program *of P.*

The massive overhead of utilizing the reduction function and building the DEP of the resulting QIP prevents the practical applicability of this approach. However, it is possible to exploit a smaller partial deterministic equivalent program during a tree search for QIP$^{\text{ID}}$ by only considering unavoidable scenarios (see Definition 5.5.9).

Definition 5.6.2 (U-DEP-Relaxation)**.**
Let $P = (A^\exists, A^\forall, b^\exists, b^\forall, c, \mathcal{L}, Q)$ be a QIP^{ID} with $\mathcal{U} \neq \emptyset$. Let $\emptyset \neq U \subseteq \mathcal{U}$ be a non-empty subset of unavoidable scenarios. For a given unavoidable scenario $s \in U$ and existential variable block $i \in \mathcal{E}$ we call $\Sigma(s, i, U)$ the set of scenarios similar to s in U up to block i given by

$$\Sigma(s, i, U) = \left\{ \sigma \in U \mid \forall j \leq \sum_{k \in \mathcal{A}:\, k<i} |B_k| :\ s_j = \sigma_j \right\}.$$

The U-DEP-relaxation *of P is given as follows:*

$$\min\ k \tag{5.22}$$
$$\text{s.t. } c^\top x_s \leq k \qquad \forall\, s \in U \tag{5.23}$$
$$A^\exists x_s \leq b \qquad \forall s \in U \tag{5.24}$$
$$(x_s)_\forall = s \qquad \forall s \in U \tag{5.25}$$
$$(x_s)_\exists \in \mathcal{L}_\exists \qquad \forall\, s \in U \tag{5.26}$$
$$x_s^{(i)} = x_\sigma^{(i)} \qquad \forall\, i \in \mathcal{E},\ s, \sigma \in U,\ s \neq \sigma,\ \sigma \in \Sigma(s, i, U) \tag{5.27}$$

A U-DEP-relaxation is closely linked to the S-relaxation of a QIP (see Definition 3.2.7) where only a subset $S \subseteq \mathcal{L}_\forall$ is considered. For a QIP, i.e. a QIP$^{\text{ID}}$ with $A^\forall = 0$ and $b^\forall = 0$, obviously $\mathcal{U} = \mathcal{L}_\forall$ and the $\mathcal{L}_\forall$-DEP-relaxation of a QIP is simply its ordinary DEP. Further, for any subset $U \subseteq \mathcal{L}_\forall$ the U-DEP-relaxation is the DEP of its S-relaxation with $S = U$.

Proposition 5.6.3. *Let $P = (A^\exists, A^\forall, b^\exists, b^\forall, c, \mathcal{L}, Q)$ be a QIP^{ID} and let $\emptyset \neq U \subseteq \mathcal{U}$. Let R be the U-DEP-relaxation of P. If R is infeasible then P is infeasible.*

Proof. Consider the game tree $G = (V, E, e)$ of P and the subgraph $\hat{G} = (\hat{V}, \hat{E}, \hat{e})$ of G defined as follows: For any leaf node $v \in V_L$ of G with $(x_v)_\forall \in U$ the path from the root to v is part of $\hat{G}$. Hence, for any leaf $\hat{v} \in V_L \cap \hat{V}$ it is $A^\forall x_v \leq b^\forall$. Obviously, finding a solution to R is equal to finding a winning truncated existential strategy in $\hat{G}$. Since R is infeasible there exists no winning truncated existential strategy in $\hat{G}$ and thus there is a destructive universal strategy D in $\hat{G}$. Since for each existential node in $\hat{G}$ all successors as in G are present, D also constitutes a destructive strategy in G and P is infeasible with Corollary 5.3.8. □

Proposition 5.6.4. *Let $P = (A^\exists, A^\forall, b^\exists, b^\forall, c, \mathcal{L}, Q)$ be a QIP^{ID} and let $\emptyset \neq U \subseteq \mathcal{U}$. Let R be the U-DEP-relaxation of P and let R be feasible with objective value $\tilde{z}_R = c^\top \tilde{x}$. Then either P is infeasible or $\tilde{z}_R$ is a lower bound on the optimal value $\tilde{z}_P$ of P, i.e. $\tilde{z}_R \leq \tilde{z}_P$.*

Proof. Let $G = (V, E, e)$ be the game tree of P. If P is infeasible, then $minimax_e(G) = +\infty > \tilde{z}_R$. Therefore, let P be feasible. Since $\mathcal{U} \neq \emptyset$ it is $z_P \neq -\infty$ with Proposition 5.5.8. Consider the subgraph $\hat{G} = (\hat{V}, \hat{E}, \hat{e})$ of G as defined in the proof of Proposition 5.6.3. Since R is feasible there exists a winning truncated existential strategy in $\hat{G}$. A winning truncated existential strategy in G must have a winning truncated existential strategy of $\hat{G}$ as a subgraph, i.e. in order to win in G one must be able to win in $\hat{G}$. Thus, any winning truncated existential strategy in G has a leaf $v \in V \cap \hat{V}$ with x_v being the principal variation of the corresponding strategy in $\hat{G}$ with $w(v) \geq \tilde{z}_R$. Hence, with Theorem 5.3.16, it is $\tilde{z}_R \leq \tilde{z}_P$. □

By relaxing the integrality in Constraint (5.26) an LP arises that has grown linearly in the number of considered scenarios $|U|$ compared to the original existential constraint system. In particular, for $|U| = 1$ the LP-relaxed U-DEP-relaxation has the same size as the LP-$\exists$-relaxation with the benefit that universal variables are fixed to the scenario $s \in U$ (and therefore cannot be set as if they aim at minimizing the objective). Hence, the resulting objective value constitutes a better (or at least not worse) bound on the optimal value of the underlying QIP$^{\text{ID}}$. Therefore, if $\mathcal{U} \neq \emptyset$, fixing universal variables to one of the unavoidable scenarios always yields a not worse relaxation compared to solving the pure LP with potentially less runtime (since there are less open variables). In practice—similar to the observations for the S-relaxation of a QIP—one has to choose U wisely: a possibly better bound might arise if more scenarios are considered while simultaneously the runtime of the resulting IP (or further relaxed LP) increases with more incorporated scenarios. Furthermore, if the "wrong" scenarios are selected the bound does not even improve.

5.6.2. The Extended Minimax Algorithm and Alpha-Beta Pruning

As a QIP$^{\text{ID}}$ can be represented via a game tree as shown in Section 5.3 we want to present an extended minimax search capable of solving QIP$^{\text{ID}}$ instances. A minimax tree is a game tree in which each inner node—representing a decision for one of the players—is marked as a MIN or a MAX node. In a QIP$^{\text{ID}}$ the existential player is trying to minimize the objective and consequently nodes representing existential variable decisions are MIN nodes. MAX nodes represent decisions by the universal player. In contrast to a QIP we have to ensure a winning

truncated strategy for the existential player in order to show the feasibility of a QIP$^{\mathrm{ID}}$. Due to the special nature of the QIP$^{\mathrm{ID}}$ one must cope with leaves and inner nodes having the symbolic value $\pm\infty$, but note that the restriction $\{x \in \mathcal{L} \mid A^{\forall}x \leq b^{\forall}\} \neq \emptyset$ for any QIP$^{\mathrm{ID}}$ prohibits that the root note takes this value (see Corollary 5.3.13). The extended minimax value presented in Definition 5.3.11 already outlines the extended minimax algorithm (see Algorithm 4). This algorithm recursively computes the extended minimax value for each inner node, starting with the values of the leaves (given by the weighting function $w(v)$). Nodes with value $\pm\infty$ are omitted when calculating the minimum or maximum value of a node's successors. If all successors have the value $\pm\infty$ the node itself receives this value. Algorithm 4 is also applicable for the special

Algorithm 4: Extended minimax algorithm: ExtendedMinimax(v)

Input: node v

if $v \in V_L$ **then**
  return $w(v)$
end if
$\mathcal{P} := \emptyset$
for $v' \in \{u \in V \mid (v,u) \in E\}$ **do**
  if ExtendedMinimax(v') $\neq \pm\infty$ **then**
    $\mathcal{P} := \mathcal{P} \cup$ ExtendedMinimax(v') // `Only nodes not in` $V_{\pm\infty}$ `are added`
  end if
end for
if $\mathcal{P} = \emptyset$ **then**
  return $\pm\infty$
else if $v \in V_{\forall}$ **then**
  return $\max(\mathcal{P})$
else
  return $\min(\mathcal{P})$
end if

cases where the considered QIP$^{\mathrm{ID}}$ is in fact a QIP or a QIP$^{\mathrm{PU}}$. In those cases the possible values at the leaves are within $\mathbb{Q} \cup \{+\infty, -\infty\}$ and hence the set $\mathcal{P}$ always contains as many values as there are successors.

The extended minimax algorithm must traverse the entire game tree and evaluate every leaf in order to compute $minimax_e(r)$ of the root r. In mixed integer programming branch-and-cut procedures are used in order to prune subtrees that cannot contain the optimal solution [PR91]. For game tree search the alpha-beta algorithm can be used in order to prune away branches during the search [KM75, PdB01]. In this setting, subtrees are pruned that might be part of an optimal strategy but that cannot influence the minimax value, i.e. subtrees that do not contain the PV. This technique also can be used for QIPs [Wol15]. The values α and β, maintained during the search, have the following purpose at any given node $v \in V$ in the game tree:

- α is the maximum value of all MAX nodes above v that were already visited. Assume $v \in V_{\exists}$, i.e. v is a MIN node. If any child $v' \in \mathcal{L}(v)$ yields a value less than or equal to α the remaining children can be omitted, because v cannot be part of the PV since at an earlier stage the universal (MAX) player has a better move yielding the value α.

- β is the minimum value of all MIN nodes above v that were already visited. Assume $v \in V_\forall$, i.e. v is a MAX node. If any child $v' \in \mathcal{L}(v)$ yields a value larger than or equal to β the remaining children can be omitted, because v cannot be part of the PV since at an earlier stage the existential (MIN) player has a better move yielding the value β.

The alpha-beta algorithm is particular advantageous if the PV is visited first as the emerging values for α and β more frequently lead to cutoffs. Hence, the order in which the nodes are evaluated has an enormous impact on its performance. The search order used in Algorithm 4 on the other hand is completely irrelevant, since every leaf has to be visited anyway. Just like the minimax algorithm, the alpha-beta algorithm can be made applicable for QIP^{ID} by making a small change regarding the possible leaf value $\pm\infty$. We provide the basic framework in Appendix A.4 on page 176: Algorithm 11 is called as the main function, which results in a recursive call of Algorithm 12, where nodes having the value $\pm\infty$ are dealt with as if they were not visited at all.[11] If, however, only the legal successors $\mathcal{F}(v)$, instead of all successors $\mathcal{L}(v)$, of the current node v are explored (Lines 6 and 18 in Algorithm 12), no returned value will be $\pm\infty$. Additionally, querying $\mathcal{F}(v)$ immediately detects terminal nodes with value $+\infty$ or $-\infty$. This way a more compact alpha-beta algorithm can be stated in Algorithm 5, Algorithm 12.

Algorithm 5: Alpha-beta call for QIP^{ID} with legality check: $\text{AlphaBeta}_L(v,\alpha,\beta)$.

Input: node v, value α, value β

```
1: if v ∈ V_L then
2:    return w(v)          // weighting function w(v), see Definition 5.3.11
3: end if
4: if v ∈ V_∀ then          // v is a MAX node
5:    for all v' ∈ F(v) do      // only legal successors are considered
6:       α = max{α, AlphaBeta_L(v', α, β)}
7:       if α ≥ β then
8:          return β
9:       end if
10:   end for
11:   return α
12: end if
13: if v ∈ V_∃ then          // v is a MIN node
14:   for all v' ∈ F(v) do      // only legal successors are considered
15:      β = min{β, AlphaBeta_L(v', α, β)}
16:      if β ≤ α then
17:         return α
18:      end if
19:   end for
20:   return β
21: end if
```

This algorithm is very similar to the standard alpha-beta algorithm with the main difference that for some inner nodes $v \in V \setminus V_L$ no successors need to be evaluated. If in such a case $v \in V_\exists$,

[11] Algorithm 11 is only given for the sake of completeness and Algorithm 12 is replaced in the following lines, which is why both are only listed in the appendix.

Algorithm 5 returns the current β value. Note that for a yet unmodified β this is consistent with the actual value of this terminal node, as β is initialized to $+\infty$.

The efficiency of the legality checks in Lines 5 and 14 is crucial to the performance of Algorithm 5. To verify the legality of a particular variable assignment an IP-satisfiability problem has to be solved (see Definition 5.2.1), which is NP-complete in general. In Chapter 6 it is discusses in which cases the distinction between legal and illegal assignments can be done efficiently.

Example 5.6.5. *Consider a QIP^{ID} with four binary variables and $Q = (\exists, \forall, \exists, \forall)$. The leaf values in the game tree in Figure 5.2 are deliberately chosen to show possible effects of Algorithms 5 and 12. If Algorithm 5 is utilized dotted lines (illegal moves) are not considered. Algorithm 12 traverses dotted lines as usual but does never takes children that return $\pm\infty$ into account when updating α or β values. For example the second leaf (counting from the left) is visited and evaluated if Algorithm 12 is used but is not even considered in Algorithm 5. The selected move order has an enormous impact on the performed cutoffs. If, for example, the two suggested nodes at the bottom left were visited in reverse order, no cutoff would occur in this subtree.*

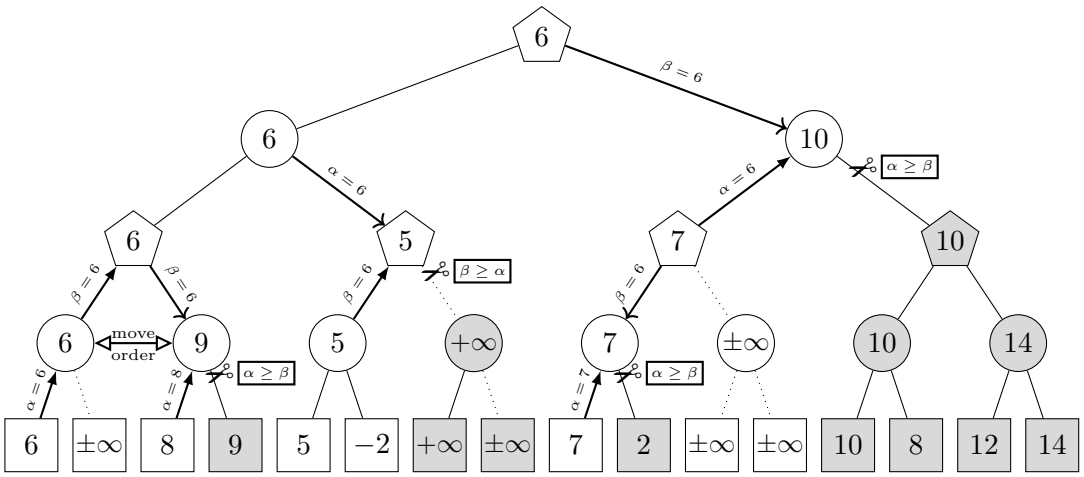

Figure 5.2.: Game tree with rectangular leaves, circular MAX nodes, and pentagonal MIN nodes. Values in the nodes are the actual extended minimax values. Gray nodes are not visited by the alpha-beta algorithm if the search order always picks the left successor first. Dotted lines indicate illegal moves. The most important updates(→) and transfers(⟶) of α and β values are indicated next to the respective edges.

5.6.3. Strategic Copy-Pruning for QIP$^{\text{ID}}$

The strategic copy-pruning mechanism presented in Subsection 3.1.3 can be used in QIPs in order to quickly verify the existence and optimality of (sub)strategies. For QIP$^{\text{ID}}$ this idea can be adopted, given that the optimal extended minimax value $\tilde{z}$ of the already examined subtree is rational and in particular not $-\infty$, $+\infty$ or $\pm\infty$. The question is whether the existential variable assignments as in the corresponding PV can be used in the neighboring subtree, immediately resulting in a strategy for the existential player that is better than the already found optimal substrategy in the first searched subtree. For QIP this is done by verifying the fulfillment of each

(existential) constraint using the constraint-specific worst-case universal variable assignments and checking that the objective value at each leaf of the resulting strategy is less than or equal to $\tilde{z}$. Similarly, we can check the fulfillment of the existential constraint system in order to verify the existence of a winning truncated substrategy of a QIP$^{\text{ID}}$.

Proposition 5.6.6. *Given a binary QIP^{ID} P and let $v \in V$ be an inner node of the corresponding game tree $G = (V, E, e)$, reached via the partial variable assignment $(\tilde{x}_1, \dots, \tilde{x}_k)$ with $k \in \mathcal{I}$. Consider a leaf below v corresponding to $\hat{x} = (\tilde{x}_1, \dots, \tilde{x}_k, \hat{x}_{k+1}, \dots, \hat{x}_\beta) \in \mathcal{L}$ with $A^\exists \hat{x} \leq b^\exists$. If*

$$\sum_{j=1}^{k} A^\exists_{i,j}\tilde{x}_j + \sum_{\substack{j\in\mathcal{I}_\exists:\\ j>k}} A^\exists_{i,j}\hat{x}_j + \sum_{\substack{j\in\mathcal{I}_\forall:\\ j>k\wedge A^\exists_{i,j}>0}} A^\exists_{i,j} \leq b^\exists_i \qquad \forall i \in \{1,\dots,m_\exists\} \tag{5.28}$$

the substrategy below v arising when always setting future existential variables according to $\hat{x}$ is a winning existential substrategy and

$$minimax_e(v) \leq \sum_{j=1}^{k} c_j\tilde{x}_j + \sum_{\substack{j\in\mathcal{I}_\exists:\\ j>k}} c_j\hat{x}_j + \sum_{\substack{j\in\mathcal{I}_\forall:\\ j>k}} c^+_j \tag{5.29}$$

with $c^+_j = \max(c_j, 0)$ being the positive part of c_j.

Proof. Let $T = (V', E', e')$ be the arising substrategy. In each leaf $\ell \in V_L \cap V'$ it is $A^\exists x_\ell \leq b^\exists$ with (5.28) and therefore $minimax_e(\ell) = w(\ell) \in \{c^\top x_\ell, -\infty\}$. Therefore, T is a winning strategy. With Corollary 5.3.10 T implies a winning truncated strategy T' for the existential player, which is a subgraph of T without parts reached via illegal universal variable assignments. With Theorem 2.1.16 $minimax^T(v)$ is equal to the largest value at the leaves and hence

$$minimax_e(v) \leq minimax^{T'}_e(v) \leq minimax^T(v) = \sum_{j=1}^{k} c_k\tilde{x}_j + \sum_{\substack{j\in\mathcal{I}_\exists:\\ j>k}} c_j\hat{x}_j + \sum_{\substack{j\in\mathcal{I}_\forall:\\ j>k}} c^+_j\,. \qquad \square$$

In the above proposition the universal constraint system is neglected and therefore leaves having the value $-\infty$ are not recognized as such. Therefore, the bound on $minimax_e(v)$ can be rather conservative, as illegal universal variable assignments are not prohibited in T. Further, some winning truncated existential substrategies are not detected, as illegal universal variable assignments can lead to a violation of Condition (5.28). However, as we are aiming for a computationally quick verification of a substrategy, we accept that Condition (5.28) is only sufficient. This result can be used in order to show the applicabilty of SCP in QIP$^{\text{ID}}$.

Theorem 5.6.7 (Strategic Copy-Pruning (SCP) for QIP$^{\text{ID}}$)**.**
Consider a binary QIP^{ID} and its game tree $G = (V, E, e)$. Let $k \in \mathcal{I}_\forall$ and let $(\tilde{x}_1, \dots, \tilde{x}_{k-1}) \in \{0,1\}^{k-1}$ be a fixed legal variable assignment of the variables $x_1, \dots, x_{k-1}$. Let $v \in V_\forall$ be the corresponding node in the game tree. Let $\tilde{w} \in V$ and $\hat{w} \in V$ be the two children of v corresponding to the variable assignment $\tilde{x}_k$ and $\hat{x}_k = 1 - \tilde{x}_k$ of the universal variable x_k, respectively.

Let there be an optimal winning truncated existential strategy for the subtree below $\tilde{w}$ with extended minimax value $\tilde{z} \in \mathbb{Q}$ defined by the variable assignment $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_n) \in \{0,1\}^n$, i.e. $\tilde{z} = c^\top \tilde{x}$. If the value of the copied strategy for the subtree below $\hat{w}$—obtained by adoption of future existential variable assignments as in $\tilde{x}$—is not larger than $\tilde{z}$ and if this copied strategy constitutes a winning strategy[12] *then $minimax_e(v) = \tilde{z}$. Formally: If both*

$$c_k(\hat{x}_k - \tilde{x}_k) + \sum_{\substack{j \in \mathcal{I}_\forall: \\ j>k \wedge c_j \geq 0}} c_j(1 - \tilde{x}_j) - \sum_{\substack{j \in \mathcal{I}_\forall: \\ j>k \wedge c_j < 0}} c_j \tilde{x}_j \leq 0 \tag{5.30}$$

and

$$\sum_{\substack{j \in \mathcal{I}: \\ j \in \mathcal{I}_\exists \vee j<k}} A^\exists_{i,j} \tilde{x}_j + A^\exists_{i,k} \hat{x}_k + \sum_{\substack{j \in \mathcal{I}_\forall: \\ j>k \wedge A^\exists_{i,j}>0}} A^\exists_{i,j} \leq b^\exists_i \qquad \forall i \in \{1, \ldots, m_\exists\} \tag{5.31}$$

then $minimax_e(v) = \tilde{z}$.

Proof. Similar reasoning as in the proof of Proposition 5.6.6 is used. If (5.31) is fulfilled a winning strategy exists below $\hat{w}$, which implies a winning truncated strategy. The extended minimax value of this winning truncated strategy is less than or equal to the minimax value of the winning strategy. Hence, if Condition (5.30) is fulfilled, the value of this winning strategy is less than or equal to $\tilde{z}$ and therefore $minimax_e(\hat{w}) \leq \tilde{z}$. Consequently, $minimax_e(v) = \tilde{z}$. □

Similar to results obtained in Theorem 3.1.8, SCP can also be adapted for general QIP$^{\mathrm{ID}}$ that are not restricted to binary variables. The implementation of SCP as presented in Algorithm 1 can also be used for QIP$^{\mathrm{ID}}$, since the universal constraint system is neglected and the same conditions must be checked at each universal node. In particular, Conditions (5.30) and (5.31) in the above theorem exactly match Conditions (3.7) and (3.8) of Theorem 3.1.7. We refer to page 37 for implementation details. However, depending on the used data structure for storing the current path in the search tree and the corresponding partial variable assignment, one must keep in mind that Line 3 of Algorithm 1 is based on the actual game tree. In particular, only considering the branching variables can be problematic for the use of SCP in a QIP$^{\mathrm{ID}}$: In order to ensure the fulfillment of the universal constraint system—and thus the legality of universal moves—the assignment of a universal variable can be implied in a QIP$^{\mathrm{ID}}$ if the fulfillment of some universal constraint is only possible with a certain fixation of this variable. This is never the case for QIP, where each universal variable assignment within $\mathcal{L}_\forall$ can occur. Line 3 must refer to the parent node within the actual game tree, rather than only the previous branching variable. Omitting implied universal variables in Line 3 leads to incorrect results, if in Line 5 only those constraints are checked in which the universal variable corresponding to the current node is present (cf. Proposition 3.1.9).

Example 5.6.8. *Consider a QIP^{ID} with four variables with $\mathcal{L} = \{0,1\}^4$. The parameters describing the instance can be found in Table 5.1. and its game tree is given in Figure 5.3.*

[12] Not a truncated winning strategy.

Table 5.1.: Parameters of Example 5.6.8.

$c^\top x$			$Q \circ x$				$A^\exists x \leq b^\exists$						$A^\forall x \leq b^\forall$			
min	$-3x_3$	$-x_4$	$\forall x_1$	$\forall x_2$	$\exists x_3$	$\exists x_4$	x_1	$-x_2$	$+x_3$	$-x_4$	≤ 1	(a)				
								$-x_2$	$+x_3$	$+x_4$	≤ 1	(b)	x_1	$+x_2$	$= 1$	
								x_2		$-x_4$	≤ 0	(c)				

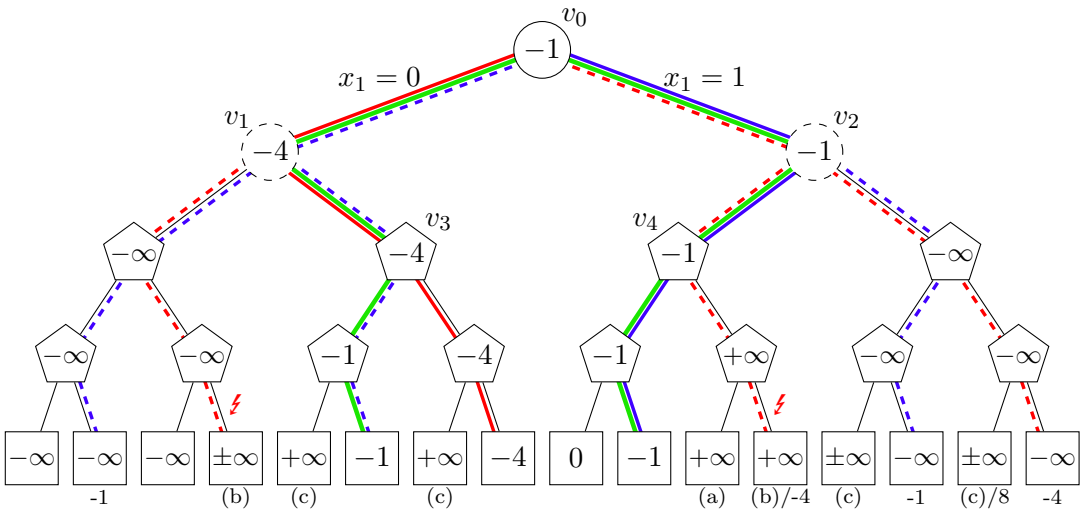

Figure 5.3.: Example for the use of SCP in a QIP$^{\text{ID}}$. Game tree with rectangular leaves, circular universal (MAX) nodes, and pentagonal existential (MIN) nodes. Edges to the left and right represent setting the corresponding variable to 0 and 1, respectively. Values in the nodes are the actual extended minimax values. Green lines indicate an optimal winning truncated strategy. Writing beneath certain leaves indicate the violated existential constraint and/or the value of $c^\top x$ at this node.

a) Assume the search first traverses the solid red path. After setting $x_1 = 0$ immediately $x_2 = 1$ can be implied—as this is the only way to fulfill $A^\forall x \leq b^\forall$—and no explicit branching step might be executed for x_2. In this case setting $x_3 = 1$ and $x_4 = 1$ is optimal resulting in the objective value -4. Hence, SCP might be applicable and Algorithm 1 traverses the tree towards the root. If, however, $parent(v_3) = v_0$—i.e. if implied variables are not explicitly stored as nodes—and if in Line 5 only those (existential) constraints are checked in which the variable corresponding to v_0, i.e. x_1, occurs, SCP produces an incorrect result: x_1 is only present in Constraint (a) and for $\tilde{x}_3 = 1$ and $\tilde{x}_4 = 1$ this constraint is fulfilled regardless of the assignment of the universal variables. Further, the worst-case objective remains -4. Therefore, the two Conditions (5.30) and (5.31) seem to be fulfilled. Thus, there appears to be a strategy for the right-hand side subtree (dashed red lines) with objective -4 and the use of SCP falsely yields $minimax_e(v_0) = -4$. There are two options to bypass this mistake:

1. *Always check the entire existential constraint system in Line 5 of Algorithm 1 in order to ensure Condition (5.31). Then the violation of Constraint (b) is detected while checking Line 5 at the root v_0.*

2. *Ensure that in Line 3 implied variables are not omitted. If* $parent(v_3) = v_1$ *then the query of Condition (5.31) in Line 5 checks the worst-case fulfillment of Constraints (a), (b) and (c), since* x_2 *is present in each of them. Hence, the violation of Constraint (b) is detected and no node is marked as finished.*

b) Assume the search first follows the solid blue path, which indeed is the PV of the instance. For v_2 *the query in Line 5 returns* `TRUE`*, as setting* $\tilde{x}_3 = 0$ *and* $\tilde{x}_4 = 1$ *is also applicable in the subtree corresponding to* $x_1 = 1$ *and* $x_2 = 1$ *with objective value* -1*. Thus, SCP is checked at the above universal root node* $v_0 = parent(v_2)$*. Again* $\tilde{x}_3 = 0$ *and* $\tilde{x}_4 = 1$ *yield a winning strategy in the left subtree (see dashed blue lines) with objective value equal to* -1*. This winning strategy implies a winning truncated strategy (solid green lines).*

Note that the optimal solution indicated be green lines is not unique: In the left-hand subtree setting $x_3 = x_4 = 1$ *yields a better (local) payoff of* -4*. However, as the (global) optimal value solely depends on the PV, an optimal winning truncated strategy does not have to consist of all (locally) optimal substrategies.*

SCP as presented in Theorem 5.6.7 and implemented as in Algorithm 1 demands a winning strategy, instead of a winning *truncated* strategy, which is a rather strong condition for QIP$^{\text{ID}}$. If the two outer dashed blue lines would not yield a fulfilled existential system or an objective larger than -1 SCP would not be applicable; even though these leaves belong to subtrees only reachable via illegal moves by the universal player. The reason for this is that Conditions (5.30) and (5.31) examine the worst-case universal variable assignment according to $\mathcal{L}$, which in case of QIP$^{\text{ID}}$ does not necessarily reflect legal play by the universal player. However, incorporating the worst-case *legal* scenario in these conditions would require the evaluation of several subproblems, thwarting the polynomial evaluation time (see Proposition 3.1.9). We are able to show in Chapter 7, and in particular on pages 153ff., that the use of SCP has an extraordinary impact on the runtime of our solver on certain instances.

5.6.4. Monotone Variables

The monotonous occurrence of a variable in the existential constraint system and the objective can be exploited in a QIP as presented in Subsection 3.1.2. In the presence of universal constraints, however, this procedure is no longer valid and must be adapted. Again, we restrict ourselves to the case that only binary variables are present. The basic idea remains that certain fixations of a monotone variable can be omitted during the search as setting it the other way never yields a worse optimal value.

Definition 5.6.9 (Monotonicity in a QIP$^{\text{ID}}$)**.**
In a QIP^{ID} a variable x_k *is called* positive (negative) monotone *if it occurs with only positive (negative) sign in the existential constraint system and objective and with only negative (positive) sign in the universal constraint system i.e. if the entries of* $A^{\exists}$ *and* c *belonging to* x_k *(*$A^{\exists}_{\star,k}$ *and* c_k*) are all non-negative (non-positive) and the entries in* $A^{\forall}_{\star,k}$ *are all non-positive (non-negative).*

First we point out the relation between two leaves in the game tree of a QIP$^{\text{ID}}$ corresponding to completely assigned variable vectors that only differ in the entry of such a monontone variable.

Corollary 5.6.10. *Consider a QIP^{ID} and its corresponding game tree $G = (V, E, e)$. Let variable x_k, $k \in \mathcal{I}$, be positive monotone. Consider any two leaves $\bar{v}, \tilde{v} \in V_L$ with $(x_{\tilde{v}})_k = 0$,$(x_{\bar{v}})_k = 1$ and $(x_{\tilde{v}})_j = (x_{\bar{v}})_j$ for any other index $j \neq k$. Then $A^{\forall} x_{\tilde{v}} \geq A^{\forall} x_{\bar{v}}$ and $A^{\exists} x_{\tilde{v}} \leq A^{\exists} x_{\bar{v}}$ and $c^{\top} x_{\tilde{v}} \leq c^{\top} x_{\bar{v}}$.*

This corollary already shows the main idea of monotonicity in QIP$^{\text{ID}}$: Setting $x_k = 0$ is "better" for the existential system and the objective value and it is to the detriment of the universal constraint system. Thus, the existential player would set it to 0 and the universal player would set it to 1.

Proposition 5.6.11. *Consider a QIP^{ID} and its corresponding game tree $G = (V, E, e)$. Let variable x_k, $k \in \mathcal{I}_{\exists}$, be positive monotone. For any node $v \in V_{\exists}$ with level$(v) = k - 1$ and its two successors $v^{(0)}$ and $v^{(1)}$ representing the assignment of $x_k = 0$ and $x_k = 1$, respectively, it holds: $minimax_e(v) = minimax_e(v^{(0)})$ or $\mathcal{F}(v) = \emptyset$.*

Proof. Let $\mathcal{F}(v) \neq \emptyset$. We distinguish the four options for the extended minimax value of $v^{(1)}$.

a) $minimax_e(v^{(1)}) = \pm\infty$, i.e. $v^{(1)} \in V_{\pm\infty}$.
With Corollary 5.3.12 for any leaf $\ell \in V_L$ below $v^{(1)}$ it is $A^{\exists} x_{\ell} \not\leq b^{\exists}$. Thus, since $\mathcal{F}(v) \neq \emptyset$, there must exist a leaf $\bar{\ell} \in V_L$ below $v^{(0)}$ with $A^{\exists} x_{\bar{\ell}} \leq b^{\exists}$. With Corollary 5.3.12 it is $minimax_e(v^{(0)}) \neq \pm\infty$ and with Definition 5.3.11 $minimax_e(v) = minimax_e(v^{(0)})$.

b) $minimax_e(v^{(1)}) = +\infty$.
Since $\mathcal{F}(v) \neq \emptyset$, there exists a leaf $\bar{\ell} \in V_L$ below $v^{(0)}$ with $A^{\exists} x_{\bar{\ell}} \leq b^{\exists}$ and thus it is $minimax_e(v^{(0)}) \neq \pm\infty$. Therefore, $minimax_e(v^{(0)}) \leq minimax_e(v^{(1)})$ and hence $minimax_e(v) = minimax_e(v^{(0)})$.

c) $minimax_e(v^{(1)}) = -\infty$.
Then there exists an optimal winning truncated existential strategy $\bar{T}$ below $v^{(1)}$ and for each terminal node $\bar{v} \in \mathcal{T}(\bar{T})$ it is $minimax_e(\bar{v}) = -\infty$ (due to Theorem 5.3.16). Consider the truncated strategy $\tilde{T}$ below $v^{(0)}$ which results from selecting the edges with the same labels as in $\bar{T}$. Hence, for each terminal node $\tilde{v} \in \mathcal{T}(\tilde{T})$ of this strategy there exists a terminal node $\bar{v} \in \mathcal{T}(\bar{T})$ with $(x_{\tilde{v}})_j = (x_{\bar{v}})_j$ for any index $j \neq k$. If $\bar{v} \in V_L$ it is $A^{\forall} x_{\bar{v}} \not\leq b^{\forall}$ and $A^{\exists} x_{\bar{v}} \leq b^{\exists}$ and with Corollary 5.6.10 this is also true for the corresponding terminal node $\tilde{v}$ and therefore $minimax_e(\tilde{v}) = -\infty$. If $\bar{v} \in V_{\forall}$ then $\mathcal{F}(\bar{v}) = \emptyset$. Again, with Corollary 5.6.10 for the corresponding terminal node $\tilde{v} \in \mathcal{T}(\tilde{T})$ also $\mathcal{F}(\tilde{v}) = \emptyset$ and thus $minimax_e(\tilde{v}) = -\infty$. Hence, $\tilde{T}$ is a winning truncated existential strategy below $v^{(0)}$ and $minimax_e(v^{(0)}) = minimax_e(v^{(1)}) = minimax_e(v)$.

d) $minimax_e(v^{(1)}) \in \mathbb{Q}$.
Then there exists an optimal winning truncated existential strategy $\bar{T}$ below $v^{(1)}$ and for each terminal node $\bar{v} \in \mathcal{T}(\bar{T}) \subseteq V_L$ it is $minimax_e(\bar{v}) = c^{\top} x_{\bar{v}}$ and in particular $A^{\exists} x_{\bar{v}} \leq b^{\exists}$ and $A^{\forall} x_{\bar{v}} \leq b^{\forall}$. Analogous to case c), a strategy $\tilde{T}$ below $v^{(0)}$ can be build. With

Corollary 5.6.10 for each terminal node $\tilde{v} \in \mathcal{T}(\tilde{T})$ we know $A^{\exists} x_{\tilde{v}} \leq b^{\exists}$ and $c^{\top} x_{\tilde{v}} \leq c^{\top} x_{\bar{v}}$. Hence, $\tilde{T}$ is a winning (potentially truncated) existential strategy and with Theorem 5.3.16 $minimax_e(v^{(1)}) \geq minimax_e(v^{(0)}) = minimax_e(v)$. □

Remark 5.6.12. *In Proposition 5.6.11 the case $\mathcal{F}(v) = \emptyset$ is treated separately. If $\mathcal{F}(v) = \emptyset$ for $v \in V_{\exists}$ then $minimax_e(v) \in \{+\infty, \pm\infty\}$. In case $minimax_e(v) = \pm\infty$ this node v cannot be reached via legal play. If $minimax_e(v) = +\infty$ the claim $minimax_e(v) = minimax_e(v^{(0)})$ is not always true, as $minimax_e(v^{(0)}) = \pm\infty$ and $minimax_e(v^{(1)}) = +\infty$ can occur.*

By applying Proposition 5.6.11 certain subtrees can be omitted a priori when solving QIP$^{\mathrm{ID}}$. Obviously, similar results can be achieved for negative monotone variables and furthermore Proposition 5.6.11 can be restated for monotone universal variables. Additionally, in case of integer variables similar results can be achieved by only considering the variable's bounds.

6. Simply Restricted QIP$^{\mathrm{ID}}$

The general QIP$^{\mathrm{ID}}$ has the drawback that determining the set of legal moves $\mathcal{F}^{(i)}$ at any stage constitutes an NP-complete problem. In order not to slow down the search excessively, a quick evaluation of $\mathcal{F}^{(i)}$ would be advantageous. In this chapter we motivate why certain restrictions on the universal constraint system are reasonable and lessen the hardness of evaluating $\mathcal{F}^{(i)}$. We present a weakened version of the QIP$^{\mathrm{ID}}$ and discuss the consequences both for the modeling possibilities as well as the solution process.

6.1. Motivation

In Section 5.2 the definition of the set of legal variable block assignments $\mathcal{F}^{(i)}$ was established. The evaluation of $\mathcal{F}^{(i)}$ requires the solution of several IP-satisfiability problems. Intuitively, however, the resulting theoretical complexity appears unnatural and artificial: In terms of game playing determining the set of legal actions defined by the rules of the game is mostly a result of evaluating some if-then-else rules. Hence, after reading and understanding the rulebook a move by a player can instantly be classified as legal or illegal. This is certainly true in classic games like chess or go and a fast classification is possible in all—or in at least most—two-person zero-sum games, regardless of the game type. What remains hard is to evaluate the quality of a move.

Another argument why the general QIP$^{\mathrm{ID}}$ might be too comprehensive can be found in real-world planning tasks under uncertainty. Here the "uncertainty" is the occurrence of a scenario selected by an often intangible decision-maker, represented by universal variables. Thus, the universal player can be considered to be the one who decides which uncertain event occurs, the scope of which depends on previous planning decisions. We—as the planner (represented by existential variables)—do not have the ability to make planning decisions that obliterate uncertainty in the sense that there is no further legal assignment for universal variables such that uncertainty "loses". Hence, even though in a general QIP$^{\mathrm{ID}}$ it can occur that the universal player has no strategy in order to ensure the fulfillment of $A^{\forall}x \leq b^{\forall}$, this might never be necessary in operations research. In particular, for $i \in \mathcal{A}$ and legal variable assignments $\hat{x}^{(1)}, \ldots, \hat{x}^{(i-1)}$ we assume that there always exists a legal move for the universal player, i.e. $\mathcal{F}^{(i)} \neq \emptyset$.

Whether the above claim $\mathcal{F}^{(i)} \neq \emptyset$ for each universal variable block $i \in \mathcal{A}$ is reasonable for a two-person zero-sum game highly depends on the modeling approach. An intuitive way when modeling a game is to interpret the two constraint systems $A^{\exists}x \leq b^{\exists}$ and $A^{\forall}x \leq b^{\forall}$ as the "rulebook" for each player and the objective $c^{\top}x$ as the quality measure of a completed match, e.g. 0 represents a win for the existential player and 1 a loss. But some games end because one

player has no legal move left (e.g. chess) and most games have strongly varying match lengths as they terminate when a specific event occurs (e.g. Tic Tac Toe or again chess). There are two straightforward modeling approaches for the end of a play in a QIP$^{\mathrm{ID}}$: As soon as the game terminates (due to the lack of legal moves or due to the occurrence of the game ending event)

a) the losing player is determined (via special constraints and variables) resulting in an empty set of legal moves for the losing player in her next (auxiliary) turn.

b) the game score is calculated and fed to the objective value. The remaining variables (present for potential longer matches) can be set trivially, not affecting the objective value.

Both approaches have their advantages and disadvantages. Certainly approach b) should be used if the outcome of a play is not only "win or loss", but a payoff for the players, since in a) the result of the QIP$^{\mathrm{ID}}$ is $+\infty$ or $-\infty$ if the game ends due to the lack of legal moves. In particular, when using b), the existence of a winning universal (truncated) strategy is ensured a priori. Keep in mind that "winning strategy" only means, that there is always a way to play legally and hence ensure $A^{\forall}x \leq b^{\forall}$; it does not ensure a "win" with respect to the game. The remaining question is which one is the best strategy regarding the objective function.

Therefore, we argue that the assumption that the universal player always has a winning (truncated) strategy can be realized in models of robust multistage OR problems as well as models of two-person zero-sum games. Furthermore, in both settings, determining whether a move by the universal player is legal is not NP-complete but rather trivial.

6.2. The Simply Restricted QIP$^{\mathsf{ID}}$ and its Properties

We want to specify which simplifications make sense for a weakened, but highly applicable QIP$^{\mathrm{ID}}$. In this section, we examine the consequences of applying the following requirements:

a) there always exists a winning truncated universal strategy.

b) the legality of a universal variable assignment can be checked by separately examining the universal constraints.

c) illegal existential moves do not have to be detected explicitly, but are uncovered by the search as such.

If these requirements are fulfilled, the alpha-beta algorithm as presented in Algorithm 5 can be simplified: The verification of the legality of universal moves can be done in polynomial time by checking the universal constraints separately (cf. Line 5 of Algorithm 5) and existential variables assignments can be chosen from $\mathcal{L}$ rather than $\mathcal{F}$ (cf. Line 14).

To elaborate the requirements needed in order to achieve the mentioned goals, we start from the basic idea that the universal player should always have a legal move. Hence, for $i \in \mathcal{A}$ and each partial variable assignment $\hat{x}^{(1)} \in \mathcal{F}^{(1)}, \ldots, \hat{x}^{(i-1)} \in \mathcal{F}^{(i-1)}$ it should be $\mathcal{F}^{(i)}(\hat{x}^{(1)}, \ldots, \hat{x}^{(i-1)}) \neq \emptyset$. This requirement also immediately ensures the existence of a winning truncated universal strategy.

Lemma 6.2.1. *Given a QIP^{ID} P. If $\mathcal{F}^{(i)}(\hat{x}^{(1)},\ldots,\hat{x}^{(i-1)}) \neq \emptyset$ for every universal variable block $i \in \mathcal{A}$ and any partial assignment of the previous variable blocks resulting from legal play $\hat{x}^{(1)} \in \mathcal{F}^{(1)},\ldots,\hat{x}^{(i-1)} \in \mathcal{F}^{(i-1)}$, then there exists a winning truncated universal strategy.*

Proof. Let $G = (V, E, e)$ be the game tree of P. We explicitly construct a winning truncated universal strategy $S = (V', E', e')$ via Algorithm 6. The selection conducted in Line 8 is always

Algorithm 6: Building a truncated universal strategy.

Input: QIP$^{\mathrm{ID}}$ $(A^{\exists}, A^{\forall}, b^{\exists}, b^{\forall}, c, \mathcal{L}, Q)$, game tree $G = (V, E, e)$

```
1:  V' = {r}          // r is the unique root node of the game tree
2:  N = {r}
3:  E' = ∅
4:  T' = ∅
5:  while N ≠ ∅ do
6:    v = Extract(N)          // Select and Remove any element out of N
7:    if v ∈ V_∀ then
8:      Select any w ∈ F(v)
9:      E' = E' ∪ {(v, w)}        // Add edge leading to w
10:     V' = V' ∪ {w}          // Add w to the set of nodes
11:     N = N ∪ {w}
12:   end if
13:   if v ∈ V_∃ then
14:     if F(v) = ∅ then
15:       T = T ∪ {v}
16:     end if
17:     for all w ∈ F(v) do
18:       E' = E' ∪ {(v, w)}
19:       V' = V' ∪ {w}
20:       N = N ∪ {w}
21:     end for
22:   end if
23:   if v ∈ V_L then
24:     T = T ∪ {v}
25:   end if
26: end while
```

valid, as $\mathcal{F}(v) \neq \emptyset$ due to the Lemma's condition. With values in $e' \in \mathbb{Q}^{|E'|}$ set according to the edges' original values in e, the constructed graph $S = (V', E', e')$ is obviously a truncated universal strategy. The constructed set T is the set of terminal nodes of S, as no edges are added that originate from elements of T. Hence, for any terminal node $v \in T$ it is either $v \in V_{\exists}$ or $v \in V_L$. If $v \in V_L$, this node was reached via an edge (v', v) starting from a universal node $v' \in V_{\forall}$ (as the final variable block is a universal block, as demanded in Definition 5.2.3). In particular, $v \in \mathcal{F}(v')$ and thus $A^{\forall}x_v \leq b^{\forall}$. Therefore, S is also a winning truncated universal strategy. □

Therefore, if the legality of existential variable assignments is ensured, there always exists a legal universal variable assignment. But as mentioned in the goals, we do not want to explicitly

keep track of the legality of existential variable assignments, since this would also require the frequent solution of NP-complete subproblems. Therefore, we strengthen the condition in Lemma 6.2.1 and demand that there always should be a legal assignment for universal variables even if the existential player selects illegal moves from $\mathcal{L}$.

Corollary 6.2.2. *Given a QIP^{ID} P. If $\mathcal{F}^{(i)}(\hat{x}^{(1)},\dots,\hat{x}^{(i-1)}) \neq \emptyset$ for every universal variable block $i \in \mathcal{A}$ and any partial assignment of the previous variable blocks resulting from legal play by the universal player $\hat{x}^{(1)} \in \mathcal{L}^{(1)}, \hat{x}^{(2)} \in \mathcal{F}^{(2)},\dots,\hat{x}^{(i-2)} \in \mathcal{F}^{(i-2)}, \hat{x}^{(i-1)} \in \mathcal{L}^{(i-1)}$, then there exists a winning truncated universal strategy.*

As the requirements of Lemma 6.2.1 are also fulfilled in Corollary 6.2.2 we omit the specific proof. In order to reach all the goals we still want to make sure that the legality of universal variable assignments can be decided in polynomial time. In particular, we do not want to solve an IP in order to ensure $x^{(i)} \in \mathcal{F}^{(i)}$ but rather examine the universal constraints one by one and deduce legality from this local information. Therefore, we demand that an illegal universal variable assignment always has the consequence that at least one universal constraint can no longer be fulfilled.

Definition 6.2.3 (Simply Restricted QIP$^{\mathrm{ID}}$)**.**
Let $P = (A^{\exists}, A^{\forall}, b^{\exists}, b^{\forall}, c, \mathcal{L}, Q)$ be a QIP^{ID}. If the two following conditions are fulfilled, we call P a simply restricted QIP$^{\mathrm{ID}}$.

a) For each universal variable block $i \in \mathcal{A}$ we demand

$$\forall \left(\hat{x}^{(1)} \in \mathcal{L}^{(1)}, \hat{x}^{(2)} \in \mathcal{F}^{(2)},\dots,\hat{x}^{(i-2)} \in \mathcal{F}^{(i-2)}, \hat{x}^{(i-1)} \in \mathcal{L}^{(i-1)}\right) : \mathcal{F}^{(i)} \neq \emptyset\,. \tag{6.1}$$

b) Let $i \in \mathcal{A}$ and let $\hat{x}^{(1)} \in \mathcal{L}^{(1)},\dots,\hat{x}^{(i-1)} \in \mathcal{L}^{(i-1)}$ be a partial variable assignment up to block i. If $\tilde{x}^{(i)} \notin \mathcal{F}^{(i)}(\hat{x}^{(1)},\dots,\hat{x}^{(i-1)})$, then

$$\exists\, k \in \{1,\dots,m_{\forall}\} : \sum_{j<i} A^{\forall}_{k,(j)}\hat{x}^{(j)} + A^{\forall}_{k,(i)}\tilde{x}^{(i)} + \sum_{j>i} \min_{x^{(j)}\in\mathcal{L}^{(j)}} A^{\forall}_{k,(j)}x^{(j)} \not\leq b^{\forall}_{k}\,. \tag{6.2}$$

Condition (6.1) requests, that there always exists a legal move for the universal player, even if the existential player does not play legally. In particular, previous variable assignments—although they can restrict the set of legal moves—can never make $A^{\forall}x \leq b^{\forall}$ unfulfillable. In Condition (6.2) we demand that a universal variable assignment $\tilde{x}^{(i)} \in \mathcal{L}^{(i)}$ is illegal, if there is a universal constraint that cannot be fulfilled, even in the best case. Therefore, it is sufficient to separately check the constraints in which variables of block i are present in order to ensure $\tilde{x}^{(i)} \in \mathcal{F}^{(i)}$, as outlined in Algorithm 7. If Condition (6.2) is satisfied, Algorithm 7 always returns `FALSE` if $\tilde{x}^{(i)}$ is illegal and `TRUE` if $\tilde{x}^{(i)} \in \mathcal{F}^{(i)}$. Hence, in case of a simply restricted QIP$^{\mathrm{ID}}$, there always exists a strategy for the universal player to fulfill $A^{\forall}x \leq b^{\forall}$ (due to (6.1) and Corollary 6.2.2) and further checking $x^{(i)} \in \mathcal{F}^{(i)}$ can be done in polynomial time (due to (6.1) and (6.2)) for universal variable blocks. The legality of existential variable assignments does not have to be checked immediately and can be left to the search, as illegal moves by the existential player can never yield a loss for the universal player (due to Condition (6.1)).

Algorithm 7: Legality check of variable block assignment for simply restricted QIP$^{\mathrm{ID}}$.

Input: universal variable block assignment $\tilde{x}^{(i)}$, partial assignment $\hat{x}$ up to block $i-1$

1: **for all** $k \in \{1,\dots,m_\forall\}$ with $A^\forall_{k,(i)} \neq 0$ **do**
2: **if** $\left(\sum_{j<i} A^\forall_{k,(j)}\hat{x}^{(j)} + A^\forall_{k,(i)}\tilde{x}^{(i)} + \sum_{j>i} \min_{x^{(j)}\in\mathcal{L}^{(j)}} A^\forall_{k,(j)}x^{(j)}\right) \not\leq b^\forall$ **then**
3: **return** `FALSE`
4: **end if**
5: **end for**
6: **return** `TRUE`

Remark 6.2.4. *Condition* (6.2) *only ensures that the legality of an entire universal variable block can be ensured by checking the corresponding constraints. A stronger requirement would be that for each single universal variable such a violation can be detected this way. Consider for example two binary universal variables* x_1 *and* x_2 *which occur in the same variable block and assume there are two universal constraints* $x_1 + x_2 \leq 1$ *and* $x_1 - x_2 \leq 0$*. For each assignment of* $(x_1, x_2) \in \{0,1\}^2$ *it is easy to verify its legality, i.e. whether the mentioned constraints are satisfied or not, and Condition* (6.2) *holds. However, if* x_2 *is not yet assigned, the verification of the legality of* $x_1 = 1$ *cannot be guaranteed this way, since both constraints still can be fulfilled separately, although no solution of the universal constraint* system *exists.*

Since our implementation, as well as the previously presented variants of the alpha-beta algorithm, branch on variables and not entire variable blocks, Algorithm 7 cannot be called at each search node. Therefore, we introduce Algorithm 8 which checks, whether a single universal variable assignment immediately violates one of the universal constraints. As mentioned in

Algorithm 8: Preliminary legality check of a single variable assignment for simply restricted QIP$^{\mathrm{ID}}$: SeemsLegal($\tilde{x}_\ell$, $\hat{x}$).

Input: universal variable assignment $\tilde{x}_\ell$, partial assignment $\hat{x}$ up to variable $\ell-1$

1: **for all** $k \in \{1,\dots,m_\forall\}$ with $A^\forall_{k,\ell} \neq 0$ **do**
2: **if** $\left(\sum_{j<\ell} A^\forall_{k,j}\hat{x}_j + A^\forall_{k,\ell}\tilde{x}_\ell + \sum_{j>\ell} \min_{x_j\in\mathcal{L}_j} A^\forall_{k,j}x_j\right) \not\leq b^\forall$ **then**
3: **return** `FALSE`
4: **end if**
5: **end for**
6: **return** `TRUE`

Remark 6.2.4 this does not guarantee that all illegal universal variable assignment are detected. However, due to the following Lemma, we still can employ this local check.

Proposition 6.2.5. *Let* $i \in \mathcal{A}$ *be a universal variable block and* $\hat{x} \in \mathcal{L}^{(1)} \times \cdots \times \mathcal{L}^{(i-1)}$ *a partial variable assignment up to block* $(i-1)$*. For* $\tilde{x}^{(i)} \in \mathcal{L}^{(i)}$ *the following holds: Algorithm 7 with input* $(\tilde{x}^{(i)}, \hat{x})$ *returns* `TRUE`*, if and only if Algorithm 8 returns* `TRUE` *for each of the inputs* $(\tilde{x}^{(i)}_1, \hat{x})$*,* $(\tilde{x}^{(i)}_2, (\hat{x}, \tilde{x}^{(i)}_1))$*,* $\dots$*,*$(\tilde{x}^{(i)}_\ell, (\hat{x}, \tilde{x}^{(i)}_1, \dots, \tilde{x}^{(i)}_{\ell-1}))$*,*$\dots$*,* $(\tilde{x}^{(i)}_{|B_i|}, (\hat{x}, \tilde{x}^{(i)}_1, \dots, \tilde{x}^{(i)}_{|B_i|-1}))$*.*

Proof. Assume Algorithm 7 returns `TRUE`. Then for each constraint in which variables of block i are present, fulfilling assignments of future variable blocks exist after the variables of block i

are assigned according to $\tilde{x}^{(i)}$. Assume there exists an $\ell \in \{1, \dots, |B_i|\}$ for which at the input of $(\tilde{x}_\ell^{(i)}, (\hat{x}, \tilde{x}_1^{(i)}, \dots, \tilde{x}_{\ell-1}^{(i)}))$ Algorithm 8 returns `FALSE`. Then there exists a constraint in which variables of block i are present, which cannot be fulfilled even in the best case and even if only a portion of the variables of block i are assigned according to $\tilde{x}^{(i)}$. Hence, this constraint can also not be fulfilled after the entire variable block is assigned according to $\tilde{x}^{(i)}$, which contradicts the assumption that Algorithm 7 returns `TRUE`. Now assume Algorithm 7 returns `FALSE`. Then there exists a constraint $k \in \{1, \dots, m_\forall\}$ which can no longer be fulfilled after the variables of block i are assigned according to $\tilde{x}^{(i)}$. Let $\ell \in \{1, \dots, |B_i|\}$ be the index of the entry in $\tilde{x}^{(i)}$ that corresponds to the last variable of block i present in the violated constraint k. Then

$$\begin{aligned}\sum_{\substack{j<\mu(i,\ell)\\ j\notin B_i}} A^\forall_{k,j}\hat{x}_j + \sum_{j<\ell} A^\forall_{k,\mu(i,j)}\tilde{x}_j^{(i)} + A^\forall_{k,\mu(i,\ell)}\tilde{x}_\ell^{(i)} + \sum_{j>\mu(i,\ell)} \min_{x_j\in\mathcal{L}_j} A^\forall_{k,j}x_j \\ = \sum_{j<i} A^\forall_{k,(j)}\hat{x}^{(j)} + A^\forall_{k,(i)}\tilde{x}^{(i)} + \sum_{j>i} \min_{x^{(j)}\in\mathcal{L}^{(j)}} A^\forall_{k,(j)}x^{(j)} \not\leq b^\forall\,.\end{aligned}$$

Hence, for input $(\tilde{x}_\ell^{(i)}, (\hat{x}, \tilde{x}_1^{(i)}, \dots, \tilde{x}_{\ell-1}^{(i)}))$ Algorithm 8 returns `FALSE`. □

Therefore, Algorithm 7 can be imitated by consecutive calls of Algorithm 8: Instead of checking the assignment of an entire universal variable block via Algorithm 7 it suffices to call Algorithm 8 for each single universal variable assignment in order to ensure that no illegal universal variable block assignment is investigated further. In Algorithm 9 the requirements of a simply restricted QIP$^{\mathrm{ID}}$ and the result of Proposition 6.2.5 are exploited in the alpha-beta framework.

Theorem 6.2.6. *For a simply restricted QIP^{ID} Algorithm 9 yields the correct extended minimax value.*

Proof. Consider a universal node $w \in V_\forall$ with $parent(w) \in V_\exists$ and $(\mathrm{level}(w)+1) \in B_2$, i.e. w represents the first universal variable within the second variable block, which is by Definition 5.2.3 a universal variable block. Due to Condition (6.1) $\mathcal{F}^{(2)}(x_w) \neq \emptyset$, i.e. there exists a legal assignment for the universal variable block. Consider any universal node $v \in V_\forall$ below w with $v' \in \mathcal{L}(v) \subseteq V_\exists \cup V_L$ and $(\mathrm{level}(v)+1) \in B_2$, i.e. v represents the final universal variable within the second variable block. If $\mathcal{F}(v) = \emptyset$, due to Proposition 6.2.5 and Condition (6.2), either SeemsLegal$(e(v,v'),x_v)$ returns `FALSE` for each successor $v' \in \mathcal{L}(v)$ or there is a node y within the path $(w, \dots, y, y', \dots, v)$ from w to v for which SeemsLegal$(e(y,y'),x_y)$ returns `FALSE`. Consequently, if $\mathcal{F}(v) = \emptyset$, Algorithm 9 never visits any of the nodes $v' \in \mathcal{L}(v)$. Therefore, paths corresponding to illegal universal variable block assignments are detected due to Proposition 6.2.5 and Condition (6.2) within the same universal variable block and at the same time have no effect on the returned values.

By induction over the universal variable blocks, any node $z \in V_\exists \cup V_L$ reached by Algorithm 9 represents a partial variable assignment x_z consisting of only legal universal variable assignments. Furthermore, illegal existential variable assignments selected in Line 14 always yield $\beta = +\infty$ in Line 15, as for each leaf $\ell \in V_L$ reached during the search the universal constraint system is fulfilled. Condition (6.1) ensures that for any reached universal block (even via illegal existential

Algorithm 9: Alpha-beta call for simply restricted QIP$^{\mathrm{ID}}$: AlphaBeta$_S(v,\alpha,\beta)$.

Input: node v, value α, value β

```
1: if v ∈ V_L then
2:   return w(v)        // weighting function w(v), see Definition 5.3.11
3: end if
4: if v ∈ V_∀ then      // v is a MAX node
5:   for all v' ∈ L(v) with SeemsLegal(e(v,v'), x_v) do
6:     α = max{α, AlphaBeta_S(v', α, β)}
7:     if α ≥ β then
8:       return β
9:     end if
10:  end for
11:  return α
12: end if
13: if v ∈ V_∃ then      // v is a MIN node
14:  for all v' ∈ L(v) do      // also illegal successors are considered
15:    β = min{β, AlphaBeta_S(v', α, β)}
16:    if β ≤ α then
17:      return α
18:    end if
19:  end for
20:  return β
21: end if
```

variable assignments) there exists a path that represents a legal assignment for this universal variable block, which will be traversed by the search.

Therefore, illegal universal variable block assignments are detected and omitted during the search and illegal existential variable assignments result in the correct extended minimax value of $+\infty$, while paths representing legal variable assignments by both players are freely traversed. □

Consquently, in order to solve simply restricted QIP$^{\mathrm{ID}}$ the alpha-beta Algorithm 5 can be adapted by querying Algorithm 8 as shown in Algorithm 9. Since our solver uses such an alpha-beta algorithm as shown in [EH+17] we can adapt it to solve simply restricted QIP$^{\mathrm{ID}}$.

Remark 6.2.7. *In the proof of Theorem 6.2.6 Condition (6.1) is needed to ensure that illegal existential variable assignments do not have to be detected right away. Instead it is ensured, that a path to an underlying leaf will be traversed yielding $+\infty$. Hence, if it is ensured that in Line 14 only legal existential assignments are selected (i.e. $v' \in \mathcal{F}(v)$ rather than $v' \in \mathcal{L}(v)$) Condition (6.1) either can be dropped, or weakened by demanding for any universal variable block $i \in \mathcal{A}$*

$$\forall \left(\hat{x}^{(1)} \in \mathcal{F}^{(1)}, \hat{x}^{(2)} \in \mathcal{F}^{(2)}, \ldots, \hat{x}^{(i-2)} \in \mathcal{F}^{(i-2)}, \hat{x}^{(i-1)} \in \mathcal{F}^{(i-1)}\right) : \mathcal{F}^{(i)} \neq \emptyset\,,$$

in order to still comply with the requirement that there always should be a winning truncated universal strategy.

In addition to the applicability of a slightly adapted alpha-beta algorithm in the solver Yasol there are several other properties that can be used in order to solve QIP$^{\mathrm{ID}}$ more efficiently within the Yasol framework. Due to Condition (6.1) in Definition 6.2.3 the LP-relaxation, which only contains $A^{\exists}x \leq b^{\exists}$, is a valid relaxation at each existential search node. This is not true for general QIP$^{\mathrm{ID}}$, as discussed starting on page 103.

Proposition 6.2.8. *Consider a simply restricted QIP^{ID}. At any node $v \in V_{\exists}$ reached via Algorithm 9 the result of the LP-$\exists$-relaxation $\tilde{z} = \min\{c^{\top}x \mid (l \leq x \leq u) \wedge (A^{\exists}x \leq b^{\exists}) \wedge (\forall j \in \{1, \ldots, level(v)\} : x_j = (x_v)_j)\}$ is a lower bound on the extended minimax value of v.*

Proof. With Corollary 6.2.2 there is a winning truncated universal strategy. Algorithm 9 only visits existential nodes that are part of a winning truncated universal strategy as shown in the proof of Theorem 6.2.6. Therefore, at each visited existential node $v \in V_{\exists}$ it is $minimax_e(v) \notin \{-\infty, \pm\infty\}$ and hence there exists a leaf ℓ below v with $w(\ell) \in \{c^{\top}x_{\ell}, +\infty\}$ and consequently $\tilde{z} \leq w(\ell)$. With Theorem 5.3.16 $\tilde{z} \leq minimax_e(v)$. □

Remark 6.2.9. *The result of Proposition 6.2.8 is not entirely satisfactory as it only holds for existential nodes. Universal nodes still could be reached via illegal universal moves and thus the LP-$\exists$-relaxation does not provide a valid bound (see Example 6.2.10).*

For QIP$^{\mathrm{ID}}$ in general one cannot arbitrarily assign universal variables within the LP-$\exists$-relaxation and still obtain a valid relaxation, which turned out to be very useful and important for solving QIP (see Definition 3.2.4). In order to use a universal variable assignment in the LP-relaxation one must find unavoidable scenarios (see Definition 5.5.9) and even if $\mathcal{U} \neq \emptyset$ the available unavoidable scenarios are often not very useful as they are likely some kind of trivial best case assignments that ensure the existence of a winning universal strategy as demanded in Condition (6.1): For example in a machine scheduling problem in which maintenance prevents the failure of machines the only unavoidable scenario might be that no machine fails, which is likely the resulting universal variable assignment in the plain LP-$\exists$-relaxation. However, the IP-$(\exists \wedge \forall)$-relaxation as well as the LP-$(\exists \wedge \forall)$-relaxation can be used, since, the conditions of Proposition 5.5.20 are fulfilled, as the existence of a winning truncated universal strategy is ensure at the root by Corollary 6.2.2 and at each existential node $v \in V_{\exists}$ of the game tree when applying Algorithm 9. Hence, by adding the universal constraints to the relaxation the interdependence of universal and existential variables is partially included.

Example 6.2.10. *For universal nodes, in general, it is not guaranteed, that they are always reached via legal universal variable assignments in Algorithm 9. Hence, the LP-$\exists$-relaxation can yield misleading results when applied at universal nodes reached via illegal universal variable assignments. Consider a QIP^{ID} with quantification sequence*

$$\exists\, x_1 \in \{0,1\}\ \forall\, x_2 \in \{0,1\}\ \forall\, x_3 \in \{0,1\}\ \exists\, x_4 \in \{0,1\}\,, \tag{6.3}$$

universal constraint system

$$x_2 + x_3 \leq 1 \tag{6.4}$$
$$x_2 - x_3 \leq 0\,, \tag{6.5}$$

existential constraint system

$$x_1 + x_2 + x_4 \leq 1\,, \tag{6.6}$$

and objective function

$$\min\ -2x_1 - x_4\,. \tag{6.7}$$

Note that in this case a closing universal variable block is not needed, as the existential variable x_4 is not present in the universal constraint system. This is a simply restricted QIP^{ID} as a) a winning (truncated) universal strategy exists (regardless of any existential variable assignment) and b) the legality of an assignment of the entire universal variable block can be checked by separately checking the constraints. Assume the search first traverses the edge corresponding to $x_1 = 1$. Note that in this case $x_2 = 1$ is illegal. Algorithm 8, however, returns `TRUE`*, for the input $(1, (1))$ since Constraint (6.4) can still be fulfilled with $x_3 = 0$ and Constraint (6.4) with $x_3 = 1$. The existential constraint system, on the other hand, is violated and the corresponding LP-∃-relaxation is infeasible. However, the extended minimax value of the current node is $\pm\infty$ and therefore it is wrong to conclude that a destructive winning universal strategy exists if x_1 is set to 1. In particular, $x_1 = 1$ is the optimal first-stage solution.*

Corollary 6.2.11. *Consider a simply restricted QIP^{ID}. Assume only legal universal successors are visited during Algorithm 9, i.e. if in Line 5 $v' \in \mathcal{F}(v)$ is ensured. Then for any given node $v \in V$ reached during this search the result of the LP-∃-relaxation $\tilde{z} = \min\{c^\top x \mid (l \leq x \leq u) \wedge (A^\exists x \leq b^\exists) \wedge (\forall j \in \{1, \dots, level(v)\} : x_j = (x_v)_j)\}$ is a lower bound on the extended minimax value of v, i.e. $\tilde{z} \leq minimax_e(v)$.*

Proof. Each visited node is part of a winning truncated universal strategy, due to Condition (6.1) and the requirement that only legal universal variable assignments are considered. Hence, by adopting the proof of Proposition 6.2.8, the conclusion follows immediately. □

6.3. Structural Requirements

Our solver utilizes the LP-∃-relaxation at every search node in order to asses the quality of a branching variable (e.g. [AKM05]), the satisfiability of the existential constraint system in the current subtree or for the generation of bounds. In order to use those mechanisms for simply restricted QIP$^{\text{ID}}$ we are interested in ensuring that the requirements of Corollary 6.2.11 are fulfilled, i.e. only legal successors of universal nodes should be visited during the search. There are two obvious options:

a) Ensure that the routine SeemsLegal() in Line 5 of Algorithm 9 returns `FALSE`, if and only if the universal variable assignment is illegal.

b) Ensure that $v' \in \mathcal{F}(v)$ by solving the IP-$\forall$-relaxation in Line 5 of Algorithm 9.

The former highly depends on the structure of the universal constraint system and might cause considerable restrictions to the modeling option. However, we show that universal constraint systems in many relevant cases possess this characteristic. In this section we provide insight into the structural requirements of $A^{\forall}x \leq b^{\forall}$ in order to ensure Conditions (6.1) and (6.2) and further we present properties which guarantee that Algorithm 9 only traverses legal universal nodes. First, we provide further results on the connection of the structure of $A^{\forall}x \leq b^{\forall}$ and the Conditions (6.1) and (6.2).

Proposition 6.3.1. *Let P be a QIP^{ID} and let Conditions (6.1) and (6.2) be fulfilled, i.e. P is a simply restricted QIP^{ID}. Any universal constraint $k \in \{1, \dots, m_{\forall}\}$ can be rewritten such that $Q_\ell = \forall$ for the variable with highest index $\ell = \max\{j \in \mathcal{I} | A^{\forall}_{k,j} \neq 0\}$ present in the constraint, without changing the optimal solution found by Algorithm 9.*

Proof. Assume there is a universal constraint $k \in \{1, \dots, m_{\forall}\}$ in which the largest index $\ell = \max\{j \in \mathcal{I} | A^{\forall}_{k,j} \neq 0\}$ belonging to a non-zero entry corresponds to an existential variable, i.e. $\ell = \mathcal{I}_{\exists}$. Let $i \in \{1, \dots, \beta\}$ be the last variable block present in constraint k, i.e. $\ell \in B_i$ and $Q^{(i)} = \exists$. If $i = 1$, this universal constraint consists of only first-stage existential variables and must be trivially fulfilled due to Condition (6.1). Hence, constraint k can be excluded from the universal constraint system without changing the optimal solution of P, since it does not constitute a restriction. Let $i \geq 3$. Constraint k has the form $\sum_{j=1}^{i-2} A^{\forall}_{k,(j)} x^{(j)} + A^{\forall}_{k,(i-1)} x^{(i-1)} + A^{\forall}_{k,(i)} x^{(i)} \leq b^{\forall}_k$. Let $\hat{x}^{(1)} \in \mathcal{L}^{(1)}, \hat{x}^{(2)} \in \mathcal{F}^{(2)}, \dots, \hat{x}^{(i-3)} \in \mathcal{F}^{(i-3)}, \hat{x}^{(i-2)} \in \mathcal{L}^{(i-2)}$ be any partial variable assignment visited during Algorithm 9 and let $\hat{b} = b^{\forall}_k - \sum_{j=1}^{i-2} A^{\forall}_{k,(j)} \hat{x}^{(j)}$. With Condition (6.1) $\mathcal{F}^{(i-1)}(\hat{x}^{(1)}, \dots, \hat{x}^{(i-2)}) \neq \emptyset$.

a) Let $\hat{x}^{(i-1)} \in \mathcal{F}^{(i-1)}$. Then for any $x^{(i)} \in \mathcal{L}^{(i)}$ it is $A^{\forall}_{k,(i-1)} \hat{x}^{(i-1)} + A^{\forall}_{k,(i)} x^{(i)} \leq \hat{b}$ since there cannot exist an assignment $x^{(i)} \in \mathcal{L}^{(i)}$ resulting in a terminal universal node, due to Condition (6.1).

b) Let $\hat{x}^{(i-1)} \in \mathcal{L}^{(i-1)} \setminus \mathcal{F}^{(i-1)}$, i.e. $\hat{x}^{(i-1)}$ is an illegal universal variable block assignment. Then there exists a universal constraint $\hat{k} \in \{1, \dots, m_{\forall}\}$ that can no longer be fulfilled due to Condition (6.2). If $k = \hat{k}$, it is $A^{\forall}_{k,(i-1)} \hat{x}^{(i-1)} + A^{\forall}_{k,(i)} x^{(i)} \not\leq \hat{b}$ for any $x^{(i)} \in \mathcal{L}^{(i)}$.

Hence, when deciding the legality of $\hat{x}^{(i-1)}$ with regard to constraint k it always suffices to consider any particular $x^{(i)} \in \mathcal{L}^{(i)}$: If, on the one hand, $\hat{x}^{(i-1)}$ is legal then constraint k is fulfilled *for any* $x^{(i)} \in \mathcal{L}^{(i)}$. If, on the other hand, $\hat{x}^{(i-1)}$ is illegal then there either exists another universal constraint indicating $\hat{x}^{(i-1)} \notin \mathcal{F}^{(i-1)}$ or constraint k is violated *for any* $x^{(i)} \in \mathcal{L}^{(i)}$. Thus, changing constraint k to $\sum_{j=1}^{i-2} A^{\forall}_{k,(j)} x^{(j)} + A^{\forall}_{k,(i-1)} x^{(i-1)} \leq \hat{b}^{\forall}_k$ with $\hat{b}^{\forall}_k = b^{\forall}_k - \min_{x^{(i)} \in \mathcal{L}^{(i)}} A^{\forall}_{k,(i)} x^{(i)}$ does not change the legality of $x^{(i-1)}$, if the legality of previous universal variable assignments is ensured. Further, constraint k has no influence on the legality of future variable blocks $j \in \{i, \dots, \beta\}$ and for previous universal variable blocks $\mathcal{A} \ni j < i - 2$ it suffices to consider the chosen minimizing assignment due to Condition (6.2). □

The above proposition shows that if the modeler created a universal constraint that closes with an existential variable block this constraint either can be rewritten such that the closing variable block is universal, or the modeled QIP$^{\text{ID}}$ is not a simply restricted QIP$^{\text{ID}}$.

Proposition 6.3.2. *Let $P = (A^{\exists}, A^{\forall}, b^{\exists}, b^{\forall}, c, \mathcal{L}, Q)$ be a QIP^{ID}. If for each universal constraint $k \in \{1, \dots, m_{\forall}\}$ the two conditions*

a) $\exists i \in \mathcal{A} : A^{\forall}_{k,(i)} \neq 0 \wedge \forall s > i : A^{\forall}_{k,(s)} = 0$

b) $\exists i \in \mathcal{A}\ \forall s \in \mathcal{A} \setminus \{i\} : A^{\forall}_{k,(s)} = 0$

are fulfilled, then Condition (6.2) is fulfilled for P.

The two requirements in Proposition 6.3.2 demand that in each universal constraint only variables of one universal variable block are present and that this universal variable block is the last present variable block.

Proof. For each universal variable block $i \in \mathcal{A}$ there exists a subset $\mu_i \subseteq \{1, \dots, m_{\forall}\}$ of constraints in which variable block i is present. Due to Condition b) those subsets are distinct, i.e. $\mu_i \cap \mu_j = \emptyset$ for each $i, j \in \mathcal{A}$, $i \neq j$. Hence, the legality of a universal variable block assignment $x^{(i)}$ can be checked using the constraints in μ_i and due to Condition a) the fulfillment of the constraints in μ_i is immediately obvious. Furthermore, if $x^{(i)} \notin \mathcal{F}^{(i)}$ there exists a constraint $k \in \mu_i$ with $\sum_{j<i} A^{\forall}_{k,(j)} \hat{x}^{(j)} + A^{\forall}_{k,(i)} \tilde{x}^{(i)} \not\leq b^{\forall}$. □

Note that in Example 6.2.10 the requirements of Proposition 6.3.2 are fulfilled, but since the legality check via Algorithm 8 does not detect every illegal universal variable assignment the LP-∃-relaxation is not applicable at universal nodes. In the following proposition we present several conditions under which only legal universal variable assignments are visited during Algorithm 9. In such a case the requirements of Corollary 6.2.11 are fulfilled and therefore the LP-∃-relaxation is applicable at general nodes.

Proposition 6.3.3. *Consider a simply restricted binary QIP^{ID}. If $A^{\forall} x \leq b^{\forall}$ fulfills one of the following conditions, then for each universal node $v \in V_{\forall}$ visited by Algorithm 9 it holds:*

$$\textit{SeemsLegal}(e(v, v'), x_v) \text{ in Line 5 returns } \texttt{TRUE}, \text{ if and only if } v' \in \mathcal{F}(v).$$

a) $A^{\forall} \geq 0$, *i.e. each entry is non-negative.*

b) $A^{\forall} \leq 0$, *i.e. each entry is non-positive.*

c) For each universal variable $\ell \in \mathcal{I}_{\forall}$, at most one universal constraint $k \in \{1, \dots, m_{\forall}\}$ exists with $A^{\forall}_{k,\ell} \neq 0$.

d) For each universal constraint $k \in \{1, \dots, m_{\forall}\}$ at most one universal variable $\ell \in \mathcal{I}_{\forall}$, exists with $A^{\forall}_{k,\ell} \neq 0$.

e) For each universal constraint $k \in \{1, \dots, m_{\forall}\}$ there exists another unique constraint $\bar{k} \in \{1, \dots, m_{\forall}\}$ with $0 \leq b^{\forall}_{k} = -b^{\forall}_{\bar{k}}$ and $0 \leq A^{\forall}_{k,\star} = -A^{\forall}_{\bar{k},\star}$ with $A^{\forall}_{k,\star} \in \{0,1\}^n$. Further, with $S_k = \{\ell \in \mathcal{I} \mid A^{\forall}_{k,\ell} \neq 0\}$ being the set of variables present in constraint k, we demand

$S_k \cap S_{\hat{k}} = \emptyset$ for each $\hat{k} \in \{1, \dots, m_\forall\}$, $\hat{k} \neq k$, $\hat{k} \neq \bar{k}$ and for each entry $\ell \in S_k$ it is $Q_\ell = \forall$. Hence, the constraint system consists of $\frac{m_\forall}{2}$ equations of the form $\sum_{\ell \in S_k} x_\ell = b_k^\forall$ and each variable with non-zero coefficient is universally quantified and only present in at most one equation (i.e. two constraints).

Proof. Keep in mind that $\{x \in \mathcal{L} \mid A^\forall x \leq b^\forall\} \neq \emptyset$. Assume the search has reached node $v \in V_\forall$ with x_v being composed of $\hat{x}^{(1)} \in \mathcal{L}^{(1)}, \hat{x}^{(2)} \in \mathcal{F}^{(2)}, \dots, \hat{x}^{(i-2)} \in \mathcal{F}^{(i-2)}, \hat{x}^{(i-1)} \in \mathcal{L}^{(i-1)}$, with $i \in \mathcal{A}$. Let $\ell = level(v) + 1 = \min B_i$, $\ell \in \mathcal{I}_\forall$, be the index of the universal variable represented by v. Due to Condition (6.1) $\mathcal{F}(x_v) \neq \emptyset$. Obviously, SeemsLegal($e(v, v')$,$x_v$) returns `TRUE` for each legal successor $v' \in \mathcal{F}(x_v)$. What remains to be shown is that SeemsLegal($e(v, v')$,x_v) returns `FALSE` if $v' \notin \mathcal{F}(x_v)$.

a) Assume there is $\tilde{x}_\ell \in \mathcal{L}_\ell \setminus \mathcal{F}(x_v)$ for which SeemsLegal($\tilde{x}_\ell$,x_v) returns `TRUE`. Hence, for each universal constraint $k \in \{1, \dots, m_\forall\}$ with $A^\forall_{k,\ell} \neq 0$ it is

$$\sum_{j<\ell} A^\forall_{k,j} \hat{x}_j + A^\forall_{k,\ell} \tilde{x}_\ell + \sum_{j>\ell} \min_{x_j \in \mathcal{L}_j} A^\forall_{k,j} x_j \leq b^\forall_k \,.$$

Due to $A^\forall \geq 0$ each considered constraint can be fulfilled by setting upcoming variables with index $j > \ell$ to their lower bound l_j. Due to $\mathcal{F}(x_v) \neq \emptyset$ and $A^\forall \geq 0$ any other universal constraint also can be fulfilled by setting $x_j = l_j$ for each future variable $j > \ell$. Therefore, a single assignment of future variables exists after setting $x_\ell = \tilde{x}_\ell$, that fulfills the universal constraint *system* and thus $\tilde{x}_\ell \in \mathcal{F}(x_v)$, which contradicts the assumption.

Inductively, for any other variable $\tilde{\ell} \in B_i$, $\ell < \tilde{\ell}$, the search has reached the corresponding node via legal variable assignments. Therefore, $\mathcal{F} \neq \emptyset$ and by using the same arguments as above the claim is proven.

b) Same as the proof of a) with the difference that future variables are all set to their upper bound in order to prove the legality.

c) Assume there is $\tilde{x}_\ell \in \mathcal{L}_\ell \setminus \mathcal{F}(x_v)$ for which SeemsLegal($\tilde{x}_\ell$,x_v) returns `TRUE`. If $A^\forall_{k,\ell} = 0$, for each universal constraint it is $\tilde{x}_\ell \in \mathcal{F}(x_v)$ due to $\mathcal{F}(x_v) \neq \emptyset$, which contradicts the assumption. Hence, let $k \in \{1, \dots, m_\forall\}$ be the unique universal constraint with $A^\forall_{k,\ell} \neq 0$ and due to the return of SeemsLegal($\tilde{x}_\ell$,x_v) we know

$$\sum_{j<\ell} A^\forall_{k,j} \hat{x}_j + A^\forall_{k,\ell} \tilde{x}_\ell + \sum_{j>\ell} \min_{x_j \in \mathcal{L}_j} A^\forall_{k,j} x_j \leq b^\forall_k \,. \tag{6.8}$$

We now construct a legal variable *block* assignment $\bar{x}^{(i)}$ that contains $\tilde{x}_\ell$. Let $y^{(i)} \in \mathcal{F}^{(i)}(x_v) \neq \emptyset$ be any legal variable block assignment. Let $\bar{x}^{(i)} \in \mathcal{L}^{(i)}$ be given by

$$\bar{x}^{(i)}_j = \begin{cases} \tilde{x}_\ell & , \text{if } j = 1 \\ \arg\min_{x_{\mu(i,j)} \in \mathcal{L}_{\mu(i,j)}} A^\forall_{k,\mu(i,j)} x_{\mu(i,j)} & , \text{if } j \neq 1 \text{ and } A^\forall_{k,\mu(i,j)} \neq 0 \\ y^{(i)}_j & , \text{else} \,, \end{cases}$$

i.e. $\bar{x}^{(i)}$ contains the variable block assignment needed in order to fulfill constraint k and the remaining universal variables are set similar as $y^{(i)}$. Since $\tilde{x}_\ell \notin \mathcal{F}(x_v)$ we know $\bar{x}^{(i)} \notin \mathcal{F}^{(i)}(x_v)$. With Condition (6.2) there must exist a constraint $\bar{k} \in \{1, \dots, m_\forall\}$ for which

$$\sum_{j<i} A^\forall_{\bar{k},(j)} \hat{x}^{(j)} + A^\forall_{\bar{k},(i)} \bar{x}^{(i)} + \sum_{j>i} \min_{x^{(j)} \in \mathcal{L}^{(j)}} A^\forall_{\bar{k},(j)} x^{(j)} \not\leq b^\forall_{\bar{k}} \,. \tag{6.9}$$

Due to the selection of $\bar{x}^{(i)}$ it is $\bar{k} \neq k$, as the universal variables present in constraint k were set such that Constraint (6.8) is fulfilled. Since all universal variables not present in constraint k are set as in $y^{(i)} \in \mathcal{F}^{(i)}(x_v)$, Condition (6.9) is also not fulfilled for any other universal constraints due to Condition (6.2), the selection of $\bar{x}^{(i)}$ and the empty intersection of the constraints regarding the presence universal variables. Thus, the assumption is false.

Inductively, for any other variable $\tilde{\ell} \in B_i$, $\ell < \tilde{\ell}$, the search has reached this node via legal variable assignments. Therefore, $\mathcal{F} \neq \emptyset$ and by using the same arguments as above the statement holds.

d) Let $k \in \{1, \dots, m_\forall\}$ be any of the universal constraints. With Proposition 6.3.1 we can assume that *exactly* one universal variable ℓ is present in k and that for each variable $j > \ell$ with higher index $A^\forall_{k,j} = 0$. Thus, the conditions of Proposition 6.3.2 are fulfilled even more restrictively. Hence, by adapting the proof of Proposition 6.3.2 we see that if $\tilde{x}_\ell \notin \mathcal{F}$ for $\ell \in \mathcal{I}_\forall$ it is $\sum_{j<\ell} A^\forall_{k,j} \hat{x}_j + A^\forall_{k,\ell} \tilde{x}_\ell \not\leq b^\forall_k$ for its unique universal constraint k.

e) If variable ℓ is not present in any universal constraint, each assignment from $\mathcal{L}_\ell$ is legal. Let k and $\bar{k}$ be the two constraints representing the equation in which variable ℓ is present and let $K = b^\forall_k = |b^\forall_{\bar{k}}|$. Due to Condition (6.1) $K \in \{0, \dots, |S_k|\}$. Assume there is $\tilde{x}_\ell \in \mathcal{L}_\ell \setminus \mathcal{F}(x_v)$. Since each variable $j \in S_k = S_{\bar{k}}$ has coefficient zero in all the other universal equations the illegality must be due to the unique equation in which variable ℓ is present. If the number of variables set to 1 exceeds K, i.e. $\sum_{j \in S_k, j<\ell} \hat{x}_j + \tilde{x}_\ell \not\leq K$, Algorithm 8 returns `FALSE` due to constraint k. If the number of yet unassigned variables present in the equation are not enough to reach K, i.e. if $\sum_{j \in S_{\bar{k}}, j<\ell} \hat{x}_j + \tilde{x}_\ell + |\{j \in S \mid j > \ell\}| \not\geq K$, Algorithm 8 returns `FALSE` due to constraint $\bar{k}$. Further, if both $\sum_{j \in S_k, j<\ell} \hat{x}_j + \tilde{x}_\ell \leq K \leq \sum_{j \in S_{\bar{k}}, j<\ell} \hat{x}_j + \tilde{x}_\ell + |\{j \in S \mid j > \ell\}|$, i.e. if SeemsLegal() returns `TRUE`, there exists an actual assignment fulfilling both constraints, due to the binary coefficients and variables. □

Remark 6.3.4. *The stated cases in Proposition 6.3.3 obviously describe only a small subset of structural properties that ensure the desired behavior of the subroutine. For example the restriction that only universal variables are allowed in case e) can be softened, as Condition (6.1) must be ensured, anyway. Hence, existential variables present in such an equation can be seen as modifiable right-hand side value. But note that the condition of only boolean variables and boolean coefficients in case e) is crucial: the universal equation $x_1 + 2x_2 = 2$ with two boolean universal variables is represented by the two universal constraints $x_1 + 2x_2 \leq 2$ and $-x_1 - 2x_2 \leq -2$. Setting $x_1 = 1$ is obviously illegal, but for each constraint a fulfilling assignment of x_2 exists and hence this illegal assignment is not detected.*

Furthermore, in case e) the demand that universal variables appear in at most one universal equation is crucial, e.g. for the two universal constraints $x_1 + x_2 = 1$, $x_2 = 1$ setting x_1 to 1 is not detected as illegal.

To avoid unnecessary computational overhead by repeatedly solving the IP-∀-relaxation in order to ensure the legality of each universal assignment, we propose to check whether the universal constraint system fulfills either of the conditions in Proposition 6.3.3. If this is the case, the quick check SeemsLegal() suffices to ensure only legal universal variable assignments. If no known structure is found the IP-∀-relaxation has to be called in order to ensure the legality of each single universal variable assignment.

Remark 6.3.5. *All examples in Section 4.4, except the first one, fulfill at least one of the stated conditions. Thus, the LP-∃-relaxation can be applied at every search node during Algorithm 9 when solving such instances.*

6.4. Examples

6.4.1. Tic-Tac-Toe

From a didactic point of view we first want to consider the simple game tic-tac-toe. This game can be modeled very compactly as QIP^{ID} and it is intended to serve as illustrative example for the structure of a QIP^{ID} and the interdependence of the legal domains.

We consider the classic tic-tac-toe game on a 3×3 board in which two players alternate placing pieces trying to have three own pieces in a row (horizontally, vertically or diagonally). It is well known that the second player always can achieve a tie [HAL63, BB08], i.e. if both players play optimally neither one is able to place three own pieces in a row.

In [EL$^+$11b] the rules of the closely related game Gomoku (5 in a row on a 19×19-board) were modeled via linear constraints and the connection to quantified integer programming was drawn. However, it was neglected how exactly a QIP looks like, which forces the universal player to follow the rules. Instead they stated that auxiliary existential variables (and constraints) could be added in order to achieve this. Due to the introduction of a universal constraint system we are now able to provide a very compact formulation of tic-tac-toe, which can be extended to several connection games.

The basic playing rule of tic-tac-toe can be stated as follows: "When it is your turn, place exactly one of your pieces in an empty cell". The game ends if either one player manages to put three own pieces in a row (resulting in a win for this player), or if all cells are filled (resulting in a draw). In order to model tic-tac-toe as a QIP^{ID} we interpret the existential player as the starting player and the universal player as the second player. Let $k \in \{1, \ldots, 9\}$ indicated the number of the current turn. If k is odd it is an existential player's turn, and if k is even it is the universal player's turn. The existential player assigns the variables $A^k \in \{0,1\}^{3\times 3}$ and the universal player assigns $B^k \in \{0,1\}^{3\times 3}$ in the k-th turn. $A^k_{i,j} = 1$ indicates that cell (i,j) is

filled with a piece from the existential player after turn k. $B^k_{i,j} = 1$ indicates a universal player's piece at (i,j) after turn k. Thus, the quantification sequence already can be outlined:

$$\exists A^1 \in \{0,1\}^{3\times 3}\ \forall B^2 \in \{0,1\}^{3\times 3}\ \exists A^3 \in \{0,1\}^{3\times 3}\ \cdots\ \forall B^8 \in \{0,1\}^{3\times 3}\ \exists A^9 \in \{0,1\}^{3\times 3} \tag{6.10}$$

Let $P = \{1,2,3\} \times \{1,2,3\}$ be the set of board positions. According to the game's rules the following constraints must be observed by the existential player:

$$B^{k-1}_{i,j} + A^k_{i,j} \leq 1 \qquad \forall\,(i,j) \in P,\ k \in \{3,5,7,9\} \tag{6.11}$$

$$A^{k-2}_{i,j} - A^k_{i,j} \leq 0 \qquad \forall\,(i,j) \in P,\ k \in \{3,5,7,9\} \tag{6.12}$$

$$\sum_{(i,j)\in P} A^k_{i,j} = \left\lfloor \frac{k}{2} \right\rfloor + 1 \qquad \forall\, k \in \{1,3,5,7,9\} \tag{6.13}$$

The universal player must obey similar constraints:

$$A^{k-1}_{i,j} + B^k_{i,j} \leq 1 \qquad \forall\,(i,j) \in P,\ k \in \{2,4,6,8\} \tag{6.14}$$

$$B^{k-2}_{i,j} - B^k_{i,j} \leq 0 \qquad \forall\,(i,j) \in P,\ k \in \{4,6,8\} \tag{6.15}$$

$$\sum_{(i,j)\in P} B^k_{i,j} = \frac{k}{2} \qquad \forall\, k \in \{2,4,6,8\} \tag{6.16}$$

Constraints (6.11) and (6.14) ensure that at most one piece is placed in each cell. In particular, it is not allowed to place a piece in an already occupied cell. Constraints (6.12) and (6.15) ensure that previously placed pieces remain on the board. Constraints (6.13) and (6.16) demand that at each turn exactly one own piece is added to the board. With the above universal and existential constraints and the preliminary quantification sequence as in (6.10) the filling of the board according to the basic rules is ensured. However, the detection of a win must be incorporated, which we illustrate using the example of horizontal lines: After a new piece is placed, auxiliary existential variables are used to check whether this piece created a row. If three pieces of the existential player form such a row, the variable w_k can be set to 1 indicating the win. If the universal player manages to place three in a row, the existential constraint system is violated resulting in a loss for the existential player. Consider any row $i \in \{1,2,3\}$ of the board. For each row we add the auxiliary variable h^k_i with $h^k_i = 1$ indicating that after turn k row i contains three pieces of the existential player. This link is established with the following constraint:

$$\sum_{(i,j)\in P} A^k_{i,j} \geq 3h^k_i \qquad \forall\, i \in \{1,2,3\},\ k \in \{1,3,5,7,9\} \tag{6.17}$$

This is also done for vertical rows (via auxiliary variables v^k_j) and diagonal rows (via d^k_j). If either of those variables indicate three in a row for the existential player, the variable w_k is allowed to be set to 1:

$$\sum_{i=1}^{3} h^k_i + \sum_{j=1}^{3} v^k_j + d^k_1 + d^k_2 \geq w_k \qquad \forall\, k \in \{1,3,5,7,9\} \tag{6.18}$$

For the detection of a win for the universal player, no additional variables are used. Instead, we add constraints to the existential constraint system that are automatically violated if the universal player *first* manages to place three pieces in a row:

$$\sum_{(i,j)\in P} B^k_{i,j} \leq 2 + w_{k-1} \qquad \forall\, i \in \{1,2,3\},\ k \in \{2,4,6,8\} \tag{6.19}$$

This is also done for the vertical and diagonal rows. Thus, by merging the auxiliary variables w_k, h^k, v^k and d^k into the symbolic variable C^k the actual quantification sequence (without variable domains) is given by:

$$\exists\, A^1\ \exists C^1\ \forall\, B^2\ \exists\, A^3\ \exists C^3\ \cdots\ \forall\, B^8\ \exists\, A^9\ \exists C^9 \tag{6.20}$$

The only variable present in the maximizing objective function is w_9. The complete QIP$^{\mathrm{ID}}$ is given in Appendix A.3. The outcome of this QIP$^{\mathrm{ID}}$ can be interpreted in the following way:

a) If the instance is infeasible, the second player can ensure a win.

b) If the instance is feasible with value 1, the starting player can ensure a win.

c) If the instance is feasible with value 0, the second player can always achieve a tie.

Hence, the solution of this QIP$^{\mathrm{ID}}$ is expected to be "feasible with objective value 0". Indeed, our solver is able to solve this QIP$^{\mathrm{ID}}$ within less than a second with the anticipated result. Note that the first-stage solution differs from run to run as for any first move by the existential player the universal player can ensure a tie. Hence, the first existential player's move and thus the first-stage solution, does not change the result.

When expanding this model for general board sizes, even three-dimensional ones (e.g. see [BB08]), and various sizes of the winning row (e.g. 5 in Gomoku), one has to deal more specifically with the detection of a win, as is does no longer suffice to count pieces in a row, but their connection must also be ensured. We refer to [EL$^+$11b] for a nice approach.

Note that this QIP$^{\mathrm{ID}}$ is not simply restricted according to Definition 6.2.3: Condition (6.1) is not fulfilled since illegal existential moves can result in an inevitable violation of the universal constraint system, e.g. if the existential player fills the entire board with own pieces in his first turn the universal player cannot fulfill (6.16) due to (6.14). Furthermore, even though Condition (6.2) is fulfilled the assignment of a single universal variable cannot be checked via Algorithm 8. In particular, none of the conditions mentioned in Lemma 6.3.3 are fulfilled. But there is a surprising twist. If the existential variables are always assigned legally, the fulfillment of Constraints (6.14)–(6.16) can always be achieved. Due to the rather simple existential and universal constraint system an implication procedure in our implemented search immediately detects existential (universal) variables that can be fixed to particular values, in order to maintain the satisfiability of the existential (universal) constraint system. For more details on the implication of variables we refer to pages 142–143. As a result, existential variables are always assigned legally and each illegal universal variable assignment is immediately detected by Algorithm 8:

Example 6.4.1. *Consider the game position given in Figure 6.1 right after turn 7. It is now the universal player's turn to place an O at one of the empty cells. The relevant variable assignments*

O	X	O
X	X	
X	O	

Figure 6.1.: Game position with upcoming move by second player placing an O.

in order to asses the legality of the upcoming universal variable block, consisting of the variables B^8, are B^6 and A^7, which represent the placement of Os and Xs, respectively, after turn 7. The assignment of 7 universal variables in B^8 can be implied immediately, due to universal Constraints (6.14) and (6.15):

$$B^6 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \qquad A^7 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \qquad \Longrightarrow \qquad B^8 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & B^8_{2,3} \\ 0 & 1 & B^8_{3,3} \end{pmatrix}$$

Hence, the universal player only can decide on the assignment of $B^8_{2,3}$ and $B^8_{3,3}$. With the implied variables already assigned, the universal constraints in which those variables are present simplify to:

$$\begin{aligned} 0 \leq B^8_{2,3} \leq 1 \quad & \text{originating from Constraints (6.14) and (6.15)} \\ 0 \leq B^8_{3,3} \leq 1 \quad & \text{originating from Constraints (6.14) and (6.15)} \\ B^8_{2,3} + B^8_{3,3} = 1 \quad & \text{originating from Constraint (6.16)} \end{aligned}$$

Obviously, only the last constraint restricts the set of legal assignments. Since this matches the case e) in Lemma 6.3.3 Algorithm 8 immediately detects illegal universal moves.

For earlier universal variable blocks the resulting constraint system basically looks similarly with additional constraints that influence the legality of future universal variables (Constraint (6.15)). Nevertheless, if all implied variables are properly assigned, the only constraint relevant regarding the legality of the current variable block always simplifies to to to a single equation demanding to "place exactly one O in exactly one of the empty cells".

The above example shows that, even if none of the conditions of Lemma 6.3.3 are fulfilled for the basic instance, in some cases applying simple implication techniques can result in a structure of a locally valid universal constraint system, for which Algorithm 9 only visits legal universal successors, without calling the IP-∀-relaxation. For the tic-tac-toe QIP$^{\text{ID}}$ this is always the case due to the very simple structure of both constraint systems, which allows bypassing illegal existential and universal variable assignments by fixing implied variables.

6.4.2. Explorable Uncertainty in the Multistage Selection Problem

The research area of explorable uncertainty has recently received more attention (see Subsection 2.2.3), and we believe that QIP^{ID} can play an important role therein. In order to provide insights into how QIP^{ID} can be employed for such problems we modify the selection problem as stated in Subsection 4.4.4, where the universal player must select exactly one cost scenario at each iteration and the existential player has to anticipate all of the scenarios. Here we discuss the option of querying the upcoming scenario, in which case the universal player must disclose the scenario of a specific future iteration and the existential player can incorporate knowledge of future events in his decision-making. Such a query, however, is associated with costs that increase the objective value. Hence, a query might not always be part of a cost-optimal strategy.

In contrast to the cheapest set problem [Kah91, EHK16] we are considering a multistage optimization problem and do not assume that there is a pre-fixed (unknown) cost for each item. In particular, rather than discovering the cheapest set with as few queries as possible, we are interested in a strategy to select a fixed number of items, which costs depend on the cost scenario selected by the universal player, while minimizing the arising costs consisting of query- and buying-costs.

Example 6.4.2. *Assume there are four items to choose from, of which two must be selected. The cost for selecting one of the items right away is* 20€ *per item. After this initial selection option, the universal player chooses one of the cost scenarios listed in Table 6.1 and the existential player can select further items according to the scenario costs.*

Table 6.1.: Cost scenarios for Example 6.4.2.

	cost of item 1	cost of item 2	cost of item 3	cost of item 4
scenario 1	1€	100€	100€	100€
scenario 2	100€	1€	100€	100€
scenario 3	100€	100€	1€	100€
scenario 4	100€	100€	100€	1€

For this instance the overall costs solely depends on the number of selected items in the first decision stage:

a) If two items are selected, the overall costs are 40€*.*

b) If one item is selected, the worst-case scenario is the one in which the already selected item costs 1€ *and the remaining items cost* 100€*, resulting in overall costs of* 120€*.*

c) If no items are selected in the initial selection phase, the overall costs are 101€*.*

Now let us assume that the universal player can be forced to disclose the future scenario for 10€ *before making the initial selection option. Then the optimal solution is to query the future scenario for* 10€*, buy one of the items that will cost* 100€ *in the disclosed future scenario right away for* 20€*, and finally buy the* 1€*-item. Hence, although costs were incurred due to the query, the total costs of* 31€ *are smaller in comparison with the optimal solution without query.*

Based on problem SELQ$^{\text{PU}}$ (see page 66) we model this problem with explorable uncertainty as QIP$^{\text{ID}}$. We adopt the existential variables x_i^s indicating the selection of item i and the universal variables q_k^s indicating the occurrence of scenario k in iteration s. We add boolean existential query variables a_s and boolean universal variables d_k^s for the disclosure of the future scenarios. In addition to the requirement that exactly one scenario must be selected (see Constraint (6.21)), the universal constraint system now also contains constraints that enforce the disclosure of one of the scenarios if demanded by the existential player (Constraint (6.22)), and constraints that ensure the compliance with the previously disclosed scenario (Constraint (6.23)).

$$\sum_{k=1}^{N} q_k^s = 1 \qquad \forall\, s \in \{1,\dots,S\} \tag{6.21}$$

$$\sum_{k=1}^{N} d_k^s = a_s \qquad \forall\, s \in \{1,\dots,S\} \tag{6.22}$$

$$d_k^s \leq q_k^s \qquad \forall\, s \in \{1,\dots,S\},\ k \in \{1,\dots,N\} \tag{6.23}$$

The query and disclosure variables are added to the quantification sequence as follows:

$$\begin{array}{llll} \exists\, a_1 \in \{0,1\} & \forall\, d^1 \in \{0,1\}^N & \exists\, x^0 \in \{0,1\}^n & \forall\, q^1 \in \{0,1\}^N \\ \exists\, a_2 \in \{0,1\} & \forall\, d^2 \in \{0,1\}^N & \exists\, x^1 \in \{0,1\}^n & \forall\, q^2 \in \{0,1\}^N \\ \cdots & \forall\, q^S \in \{0,1\}^N & \exists\, x^S \in \{0,1\}^n & \exists\, z \in \mathbb{R}_+^S \end{array}$$

Note that instead of the legal variable domain $\mathcal{F}$, the basic variable domain $\mathcal{L}$ is given in the above order for clarity. The objective function is extended by the costs α_s for a performed query.

$$c^\top x = \sum_{i=1}^{n} c_i^0 x_i^0 + \sum_{s=1}^{S} (z_s + \alpha_s a_s) \tag{6.24}$$

The existential constraint system remains unaffected:

$$\sum_{i=1}^{n} \sum_{s=0}^{S} x_i^s = p \tag{6.25}$$

$$\sum_{s=0}^{S} x_i^s \leq 1 \qquad \forall\, i \in \{1,\dots,n\} \tag{6.26}$$

$$\sum_{i=1}^{n} c_{i,k}^s x_i^s \leq z_s + M(1 - q_k^s) \qquad \forall\, k \in \{1,\dots,N\},\ s \in \{1,\dots,S\}\,. \tag{6.27}$$

Hence, if the existential player decides to query the scenario of the upcoming iteration s by setting $a_s = 1$, the universal player is forced to set exactly one of her variables d_k^s to 1 to disclose which scenario she selects later on. The existential player then can incorporate this information when selecting items in iteration $s - 1$.

Note that the requirements of a simply restricted QIP$^{\text{ID}}$ are fulfilled, since a) the universal player always has a legal move in order to fulfill her constraint system (6.21)–(6.23) and b) an

illegal variable block assignment would be immediately apparent due to the simply nature of $A^{\forall}x \leq b^{\forall}$. Following a similar explanation as in Example 6.4.1 in the previous subsection, the application of simple implication techniques results in a structure of the remaining universal constraint system, for which Algorithm 9 only visits legal universal successors, without calling the IP-$\forall$-relaxation.

One obvious variant would be to allow the existential player to query the scenario of any future iteration rather than only the upcoming one. Further, the same concept of explorable uncertainty within the QIP$^{\mathrm{ID}}$ framework can be applied to various other optimization problems, e.g. the assignment problem (see Subsection 4.4.5), the robust knapsack problem (cf. [GG+15]) or shortest path problems with uncertainty (cf. [FM+07]).

6.4.3. Further Examples

We briefly describe a few other examples where QIP$^{\mathrm{ID}}$ can be used in order to grasp the relevance of robust optimization with decision-dependent uncertainty. We do not explicitly specify how the described problems can be translated into linear constraints, but note that all the upcoming examples can be modeled as QIP$^{\mathrm{ID}}$ while meeting the requirements of a simply restricted QIP$^{\mathrm{ID}}$. Further, keep in mind that QIP$^{\mathrm{ID}}$ is a multistage optimization framework. Therefore, rather than adhering to adjustable robust programming with only a single response stage, planning problems with multiple decision stages are realizable.

Maintenance Reduces Downtime Consider a job shop problem with several tasks and machines. One is interested in a robust schedule as machines can fail for a certain amount of time. Universal variables indicate which machines fail and how long they fail. The basic problem can be enhanced by adding maintenance work to the set of jobs: the maintenance of a machine prevents its failure for a certain amount of time at the expense of the time required for maintenance and the maintenance costs. Therefore, the universal constraint system contains constraints describing the relationship between maintenance and failure: With existential variable $m_{i,t}$ indicating the maintenance of machine i at time t and universal variable $f_{i,t}$ indicating the failure of machine i at time t the (universal) constraint $f_{i,t+j} \leq 1 - m_{i,t}$ prohibits the failure of machine i for each of the $j \in \{0, \dots, K\}$ subsequent time periods. We also refer to [GP+06] where a related stochastic program is discussed with the aim of minimizing the costs resulting from planned maintenance and unplanned repair work.

The universal constraint system also could contain further restrictions regarding the number of machines allowed to fail at the same time, similar to (variable) budget constraints (e.g. [Pos13]. This budget amount also can depend on other previous planning decisions, e.g. the overall machine utilization. Furthermore, reduced operation speed might reduce wear and therefore increase durability and lessen the risk of failure.

Workers' Skills Affect Sources of Uncertainty The assignment of employees to various tasks may have significant impact on potential occurring failures, processing times and the quality of the product. For example, it might be cheaper to have a trainee carry out a task, but the

risk of error is higher and the processing time might increase. Further, some workers might work slower but with more diligence—resulting in a long processing-time but a high quality output—than other faster, but sloppier, workers. Hence, the decision which worker performs a particular task has an impact on the anticipated uncertain events. In a more global perspective staff training and health-promoting measures affect the skills and availability of a worker, and thereby affecting potential risks.

Road Maintenance for Disaster Control In order to mitigate the impact of a disaster, road rehabilitation can improve traveling time as the damage of such roads can be reduced (see [NS18]). Again, a budget for the deterioration of travel times for all roads could be implemented, whereat the budget amount could be influenced by the number of emergency personal, emergency vehicles and technical equipment made available.

Time-Dependent Factors in Process Scheduling In [LG16] the authors present a process scheduling approach with uncertain processing-times of the jobs, whereat the range of the uncertain processing-time parameters depend on the scheduling time of the job itself. The selection of specific scheduling times therefore actively determines the range in which the uncertain processing-times are expected. For a QIP this influence on uncertainty could be achieved by adding universal constraints as follows: Let $x_{i,t}$ be the existential binary indicator whether task i is scheduled to start at time t and let y_i be the universal variable indicating the occuring processing-time of task i. Let $l_{i,t}$ and $u_{i,t}$ indicate the range of the processing-time of task i if scheduled at t. Adding $\sum_t l_{i,t}x_{i,t} \leq y_i \leq \sum_t u_{i,t}x_{i,t}$ to the universal constraint system would establish the intended interdependence.

7. Implementation Details and Experimental Results

7.1. QLP File Format for QIP$^{\mathsf{ID}}$ and General Input Requirements for Yasol

The QLP file format used for QMIPs in our solver was presented in [Wol15]. It extends the CPLEX LP-file format[13] by adding the keywords ALL, EXISTS and ORDER, which enable the identification of universal and existential variables as well as the order of the variables. So far, universal variables were only restricted by their bounds and their type (binary, general integer, or continuous variable). In order to allow universal constraints we introduce the convention that such a constraint must be explicitly named and its name must begin with "U_".

Example 7.1.1. *Consider a binary QIP^{ID} given as follows:*

$$
\begin{aligned}
c^\top x: &\quad \min\ x_1 - 2x_2 + 2x_3 + x_4 \\
Q \circ \mathcal{L}: &\quad \exists x_1 \in \{0,1\}\ \exists x_2 \in \{0,1\}\ \forall x_3 \in \{0,1\}\ \exists x_4 \in \{0,1\} \\
A^\exists x \leq b^\exists: &\quad \begin{pmatrix} 1 & -2 & 1 & -1 \\ 1 & 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \leq \begin{pmatrix} 1 \\ 2 \end{pmatrix} \\
A^\forall x \leq b^\forall: &\quad \begin{pmatrix} 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \leq \begin{pmatrix} 2 \end{pmatrix}
\end{aligned}
$$

[13] http://lpsolve.sourceforge.net/5.1/CPLEX-format.htm (accessed May 3, 2020)

The optimal value of this instance is -1 *with* $PV\ (1,1,0,0)$*. The corresponding QLP file looks as follows:*

MINIMIZE

$x1 - 2x2 + 2x3 + x4$

SUBJECT TO

Constraint1: $x1 - 2x2 + x3 - x4 <= 1$

Constraint2: $x1 + x2 + x3 - x4 <= 2$

U_Constraint1: $x1 + x2 + x3 <= 2$

BOUNDS

$0 <= x1 <= 1$

$0 <= x2 <= 1$

$0 <= x3 <= 1$

$0 <= x4 <= 1$

BINARIES

$x1\ x2\ x3\ x4$

EXISTS

$x1\ x2\ x4$

ALL

$x3$

ORDER

$x1\ x2\ x3\ x4$

END

No name is required for existential constraints, and in particular unnamed constraints are considered part of $A^{\exists}x \leq b^{\exists}$. A QIP$^{\text{ID}}$ must be a simply restricted QIP$^{\text{ID}}$ (see Definition 6.2.3) for Yasol to solve it correctly at this point in time. We are currently not able to explicitly check whether the given QIP$^{\text{ID}}$ indeed is a simply restricted one. We refer to Section 6.3, where structural requirements are discussed.

Currently, the general QLP-file has to fulfill the following properties in order to be solvable with our solver:

- The first and final variable block must be existential, i.e. $Q_1 = Q_n = \exists$ or rather $Q^{(1)} = Q^{(\beta)} = \exists$.
- Continuous variables, i.e. variables not listed below the BINARIES or GENERAL keyword, have to be part of the final (existential) variable block.
- The right-hand side of each constraint may only contain a single parameter, i.e. all variables have to be on the left-hand side.

Demanding $Q^{(\beta)} = \exists$ seems to contradict the definition of a QIP$^{\text{ID}}$. However, by adding an existential dummy variable at the end of the quantification sequence, a QIP$^{\text{ID}}$ instance can be transformed without changing the outcome (cf. discussion starting on page 78). Additionally, for simply restricted QIP$^{\text{ID}}$ a final existential variable cannot lead to a violation of the universal constraint system anyway: as the final existential variable x_4 in Example 7.1.1 is not present in the universal constraint system nothing has to be done.

It should also be mentioned, that internally general integer variables are first transformed to have the lower bound 0 and are subsequently binarized. Therefore, upper bound constraints for such binarized variables are explicitly added to the constraint system. For universal variables this was not possible until universal constraints were added to the solver. Now, general universal variables are also permitted, as after a binarization the upper bound constraint can be added to the universal constraint system.

7.2. Main Modifications and Enhancements of Yasol

7.2.1. Implementation of the Reduction Function

The reduction function presented in Section 5.4 does not only explicitly prove the PSPACE-completeness of QIP$^{\text{ID}}$, but also provides a solution approach and checking routine: a QIP$^{\text{ID}}$ can be transformed into a QIP and either be solved directly using Yasol, or the resulting QIP can be transformed into its DEP and solved via standard MIP solvers.

A first implementation draft of the reduction was implemented by Tobias Marx within the scope of a seminar [Mar18] and subsequently served as the basis for further refinements and adjustments. The reduction is implemented in our solver and can be executed be adding the keyword “Reduce”:

```
.\Yasol <instance_name> Reduce
```

The reduced QIP is then written to the file `instance_name.reduced`. The QIP$^{\text{ID}}$ instance must meet the following conditions:

- *The first variable block has to be an existential variable block.* Otherwise the reduced QIP also has a leading universal variable block, which currently prohibits the solution via Yasol.
- *The final variable block does not necessarily have to be a universal block.* Otherwise an artificial final check of the universal constraint system is added; similar to adding an auxiliary universal variable at the end.
- *Continuous variables are only allowed in a final existential stage.* The reduced QIP would also have continuous variables in inner blocks prohibiting the solution via Yasol.
- *Continuous variables must not be present within universal constraints.* The reduction would become invalid, since Constraint (5.8) is not designed to handle continuous variables.

Note that the orientation of the objective in the resulting reduced QIP is always ’MINIMIZE’. For the selection of R_k^{LCD} of row $k \in \{1, \dots, m_\forall\}$ the coefficient with the most decimal places in $A^\forall_{k,\star}$ and $b_k^\forall$ is detected. Let p_k be the largest number of decimal places and hence $p_k = \min\{\tilde{p} \in$

$\mathbb{N}_0 \mid \forall d \in \{b_k^\forall, A_{k,1}^\forall, \ldots, A_{k,n}^\forall\} : d \cdot 10^{\tilde{p}} \in \mathbb{Z}\}$. Then we select $R_k^{LCD} = 10^{-p_k}$, as this value underestimates the reciprocal of the lowest common multiple of the coefficients denominators (see Lemma 4.3.1). The remaining parameters L, M and $\tilde{M}$ are set to the threshold values given by (5.11), (5.12) and (5.14), respectively.

7.2.2. Deduction Techniques in the Presence of Universal Constraints

Our solver Yasol is specialized on solving QMIPs in which universal variables were only restricted to lie within their bounds. The addition of universal constraints therefore requires a modification of various routines. We outline the most important changes, using only the term QIP^{ID}, which of course also includes QIP^{PU}.

Monotonicity of Variables The monotonicity of a variable as presented in Definition 3.1.4 is no longer valid for a QIP^{ID}: the universal constraint system must also be taken into account (see Subsection 5.6.4). If the solver detects a QIP^{ID} the check for monotone variables also traverses the universal constraint system (see Definition 5.6.9) and during the search Proposition 5.6.11 is applied, i.e. only one fixation of such a variable must be investigated. The case that there is no legal move when trying to set a monotone variable poses no problem (cf. Remark 5.6.12) due to Condition (6.1) of simply restricted QIP^{ID}.

Implication of Variables due to Existential Constraints After making an assignment decision for a variable within the alpha-beta search of a QIP an implication procedure is invoked. This procedure takes already assigned variables into account and detects unassigned variables that are implicitly fixed to a particular value, as all other values within the variable's domain result in the inevitable violation of at least one existential constraint. In particular, if this procedure detects a universal variable that must be assigned to a certain value, the search does not need to traverse this subtree as the universal player would immediately have the ability to violate the existential constraint system.

Example 7.2.1. *Consider the following QIP*

$$\min \ -3x_1 - 2x_3$$

$$\text{s.t. } \exists x_1 \in \{0,1\} \ \forall x_2 \in \{0,1\} \ \exists x_3 \in \{0,1\} \ \forall x_4 \in \{0,1\} :$$

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \leq \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

After setting $x_1 = 1$ the implication $x_4 = 0$ is found. Since x_4 is a universal variable and no universal constraints are present this immediately leads to the conclusion that $x_1 = 1$ cannot be part of a winning existential strategy. The PV is $(0,0,1,1)$ with objective value -2.

If universal constraints are present this implication might be invalid as shown in Example 7.2.2.

Example 7.2.2. *Consider the QIP^{ID} which arises by adding the universal constraint $x_3+x_4 \leq 1$ to the QIP in Example 7.2.1. Setting $x_1 = 1$ is legal and regardless of x_2 the existential player can set $x_3 = 1$. This, however, renders $x_4 = 1$ illegal. Therefore, the implication $x_1 = 1 \Rightarrow x_4 = 0$ does not allow the conclusion that the subtree corresponding to $x_1 = 1$ cannot contain a winning truncated strategy. Indeed the PV of this QIP^{ID} is $(1, 1, 1, 0)$ with objective value -5.*

Therefore, the following adaptations must be made within this implication procedure if universal constraints are present: the implication of a universal variable does not lead to the abortion of the search, if the universal variable is present within a universal constraint and a) belongs to a higher variable block or b) setting it the other way would immediately violate the universal constraint system. The implication procedure for existential variables does not need to be altered since we demand a simply restricted QIP$^{\text{ID}}$ and therefore implied higher block existential variable assignments cannot alter the legality of earlier universal variables.

Implication of Variables due to Universal Constraints The assignment of variables present within $A^{\forall}x \leq b^{\forall}$ can lead to the implication of yet unassigned universal variables, if the fulfillment of a universal constraint can only be achieved with a particular fixation of a yet unassigned universal variable. Such an implication already anticipates the detection of illegal universal variable assignments via Algorithm 8 and thus leads to a restriction of the subtrees to be examined. As we only consider simply restricted QIP$^{\text{ID}}$, a universal constraint can never cause the implication of an existential variable: Condition (6.1) demands that no existential variable assignment can result in a violation of a universal constraint if universal variables are assigned legally.

Let us suppose variable x_k with a non-zero entry in $A^{\forall}_{\star,k}$ is assigned legally during the search. Then each universal constraint in which x_k is present is examined, in order to check whether its fulfillment depends on a particular assignment of another (universal) variable. If a yet unassigned universal variable is found that has to take on a certain value in order to ensure the fulfillment of the universal constraint system, it is added to a queue. If this implied variable is in the same variable block as the originally assigned variable the fixation is executed immediately. If the implied variable is part of a later variable block the assignment is postponed, until the search has reached this variable block. This postponement is in fact not necessary for *simply restricted* QIP$^{\text{ID}}$: Assigning future universal variables in the way the universal player intends to do it anyway does not change the optimal course of play. Due to Condition (6.1) this only remaining option for this universal variable cannot be prohibited by existential variable assignments.

Simplification of Constraints In a QIP a constraint can be simplified if the variables with the highest block present in this constraint are universal variables:

Example 7.2.3. *Consider a QIP with the SAT-constraint $x_1 + x_3 \geq 1$ and let x_1 be an existential and x_3 be a universal binary variable. If x_3 succeeds x_1 in the quantification sequence, the existential player must anticipate the local worst-case assignment of x_3, which is $x_3 = 0$. Therefore, this constraint can be simplified by locally assigning $x_3 = 0$, which results in the constraint $x_1 \geq 1$. Hence, in any winning strategy, x_1 must be set to 1.*

This simplification is based on the fact that the universal player only must obey the variable bounds. In case of a QIP$^{\text{ID}}$ such a simplification leads to false results, as the assumption that universal variables are unrestricted is no longer true. In particular, in the above example, a universal constraint could allow the existential player to force $x_3 = 1$, e.g. $x_2 + x_3 \geq 1$ with existential variable x_2. Therefore, this simplification procedure is deactivated in case of a QIP$^{\text{ID}}$.

Strategic Copy-Pruning In Subsection 5.6.3 we have shown that SCP can be applied to QIP$^{\text{ID}}$ straightforward. However, in the implementation of Algorithm 1 the selection of the parent node in Line 3 is sensitive: nodes within the game tree that belong to implied variables are not nodes of the search tree and therefore cannot be marked as finished or unfinished. However, not checking Conditions (5.30) and (5.31) for implied variables leads to incorrect results, as shown in Example 5.6.8. Therefore, Conditions (5.30) and (5.31) are also checked for implied universal variable, i.e. in Line 3 of Algorithm 1 the parent with respect to the game tree is selected, rather than the last branching variable (cf. Example 5.6.8).

Learning Conflict Clauses As outlined in [EH+17] our solver uses backjumping techniques when a contradiction is encountered during the search. Furthermore, a constraint—a conflict clause—is learned containing the variables causing this conflict (e.g. see[GNT03, GN+08]). This conflict clause, however, is not necessarily valid in every part of the search tree if universal constraints are present, as the assumption that universal variables are free to take any value within their bounds is no longer true: a conflict can arise due to a certain assignment of previous universal variables which allow (or prohibit) a particular assignment of future universal variables.

Remark 7.2.4. *In this paragraph the term "clause" is used frequently. A clause can also be represented via a linear constraint, e.g. the clause* $(a \vee \neg b \vee \neg c)$ *is equivalent to* $a - b - c \geq -1$ *with binary variables* a*,* b *and* c*. The right-hand side of the equivalent constraint results automatically from the number of negated variables* η *and is given by* $(1 - \eta)$*. Therefore, it suffices to state the left-hand side of such a constraint if it is known to represent a clause.*

Example 7.2.5. *Consider a QIP^{ID} satisfiability instance with existential constraint system*

$$\begin{aligned} x_1 - y_1 &\leq 0 && (7.1)\\ x_2 - y_2 &\leq 0 && (7.2)\\ x_3 - y_3 &\leq 0 && (7.3)\\ -x_0 - x_1 - x_2 - x_3 &\leq -2 && (7.4) \end{aligned}$$

and a universal constraint system with the single constraint

$$-y_1 - y_2 - y_3 \leq -1\,. \qquad (7.5)$$

All variables are boolean and the quantification sequence is given by

$$\exists\, x_0 \quad \forall\, y_1\ y_2 \quad \exists\, x_1\ x_2 \quad \forall\, y_3 \quad \exists\, x_3\,.$$

The instance is feasible, as $x_0 = 1$ *and* $x_i = y_i$ *for* $i = 1, 2, 3$ *constitutes a winning truncated strategy, since at least one* y_i *must be set to* 1. *Note that without the universal Constraint* (7.5) $y_i = 0$ *for* $i = 1, 2, 3$ *would be a legal universal assignment which would ultimately result in the violation of Constraint* (7.4). *Now assume the search takes the following course, where "*$\Rightarrow$*" indicates the implication of variables:*

$$
\begin{aligned}
&\textit{Assign } x_0 = 1 \\
&\textit{Assign } y_1 = 0 \;\Rightarrow\; x_1 = 0 \textit{ due to Constraint } (7.1) \\
&\textit{Assign } y_2 = 1 \\
&\textit{Assign } x_2 = 0 \;\Rightarrow\; x_3 = 1 \textit{ due to Constraint } (7.4) \\
&\qquad\qquad\;\; \Rightarrow\; y_3 = 1 \textit{ due to Constraint } (7.3) \\
&\qquad\qquad\;\; \Rightarrow\; \textit{found conflict due to implied universal variable } y_3
\end{aligned}
$$

The conflict found is valid as y_3 *is not restricted by the universal Constraint* (7.5) *in this subtree and it would be a legal universal move to set* $y_3 = 0$. *Hence, this implication indeed constitutes a conflict to the universal player's intention. The conflict clause* $(y_1 \vee x_2)$ *is generated similiar to the conflict-driven clause learning method as described in [Zha03, ELW13]. This conflict clauses however, is not globally valid for the* QIP^{ID}.

In the conflict-driven clause learning the information "which branching variables contributed to the conflict" is gathered. For QIP^{ID}, *however, the question "in what cases is the universal player allowed to do this" has to be considered. In the presented case, the conflict clause is only valid, if the implication* $y_3 = 1$ *constitutes a conflict to the universal player's intention. In particular, if the implication* $y_3 = 1$ *does not restrict the set of legal assignments of this universal variable, no conflict would arise. For our example this is the case if* $y_1 = y_2 = 0$ *and therefore this conflict clause is only valid if* $y_1 = 1$ *or* $y_2 = 1$. *Since the assignment of* y_1 *to* 0 *is involved in this specific conflict, solely adding* $\neg y_2$ *to the conflict clause yields the globally valid restriction*

$$(y_1 \vee \neg y_2 \vee x_2)\,, \tag{7.6}$$

for which the following statements holds:

a) If $y_1 = y_2 = 0$, *Clause* (7.6) *is fulfilled (due to the added literal).*

b) If $y_1 = 0$ *and* $y_2 = 1$, *Clause* (7.6) *is only fulfilled if* $x_2 = 1$.

c) If $y_1 = 1$ *and* $y_2 = 0$, *Clause* (7.6) *is fulfilled (even without the added literal).*

d) If $y_1 = y_2 = 1$, *Clause* (7.6) *is fulfilled (even without the added literal).*

Hence, by adding $y_1 - y_2 + x_2 \geq 0$ *to the existential constraint system the search can not revisit the conflicting path (due to case b)) and the search is not restricted if the originally detected conflict is no longer attainable (due to case a)).*

In general, however, it is not that simple to determine the cases in which a conflict is invalid. Therefore, if a clause is generated for which either

a) a universal variable x_k with $A^{\forall}_{\star,k} \neq 0$ is the conflicting variable, or

b) a universal variable x_k with $A^{\forall}_{\star,k} \neq 0$ is present in the clause,

it is modified by Algorithm 10. Note that for the initially found conflict clause in Example

Algorithm 10: Adapting a found conflict clause.

Input: Conflict clause $\sum_{i=1}^{n} a_i x_i \geq \eta$ with $a_i \in \{-1, 0, +1\}$ for each $i \in \mathcal{I}$.
Partial variable assignment $\tilde{x}$ representing the selected path in the game tree.
Variable x_c that caused the conflict.

```
1:  for k = 1 to n with ((a_k ≠ 0 and Q_k = ∀) or (k = c and Q_c = ∀)) do
2:    for j = 1 to m_∀ with A^∀_{j,k} ≠ 0 do
3:      for i = 1 to n with A^∀_{j,i} ≠ 0 and i ≠ k do
4:        if A^∀_{j,i} > 0 and x̃_i = 0 then
5:          a_i = +1
6:        else if A^∀_{j,i} < 0 and x̃_i = 1 then
7:          a_i = −1
8:        end if
9:      end for
10:   end for
11: end for
```

7.2.5, Algorithm 10 results in the proposed Clause (7.6). This algorithm alters the learned conflict clause by adding literals that make the clause automatically fulfilled if the validity of the underlying conflict cannot be guaranteed. In particular, a variable is added to the clause, if it is currently set beneficial for a universal constraint (see Line 4 and 6) in which conflicting universal variables are present: if such a binary variable was set the other way the legal domain of a conflicting variable might become more restricted and thereby preventing the conflict. By adding such a variable to the original conflict clause as in Line 5 and 7, the arising clause will be trivially fulfilled in subtrees where the occurrence of the original conflict is not guaranteed: this clause is only active within certain subtrees, in which the validity of the original conflict clause is ensured. Hence, if the search later on enters a subtree in which the legal domain of the conflicting variable *might* be more restricted (and thereby making the conflict non-existent), this clause becomes fulfilled due to the flipped assignment of an added variable affecting this legal domain. Therefore, Algorithm 10 is rather conservative as the original conflict clause might in fact be valid in several other subtrees.

Note that the conflict-based backjumping mechanism is not affected, as the backjumping itself solely ensures that the (valid) conflict in the current subtree is evaded.

7.2.3. Used Relaxations

During the search process in our solver at multiple occasions a relaxation is called in order to asses the quality of a branching variable (e.g. [AKM05]), the satisfiability of the existential constraint system in the current subtree or for the generation of bounds. The dilemma arises of having to choose between the quality of the solution and the runtime for solving the relaxation. In Yasol

two relaxation techniques are used for QIPs: the LP-relaxation with fixed scenario (see Definition 3.2.4) and the S-relaxation (see Definition 3.2.7). For the latter the corresponding DEP is built (see Example 3.2.8) and the integrality condition is relaxed. For both used techniques a set of scenarios is required for which an improvement of the relaxation's quality can be expected. For this purpose universal scenarios $S \subseteq \mathcal{L}_\forall$ are stored, that either frequently led to the violation of the existential constraint system or that contributed to the current PV. Among other heuristic approaches, the killer heuristic [AN77] is utilized to assemble such scenarios.

The S-relaxation is rebuilt each time the search restarts at the root node. The objective function of this relaxation reflects the worst-case objective value regarding the considered scenarios, as shown in Example 3.2.8. Since new information about (supposedly) crucial scenarios is continuously generated during the search, the set S containing the scenarios considered decisive, changes dynamically over time. By rebuilding the S-relaxation, this gained knowledge is also incorporated into the relaxation, which (hopefully) increases its benefit. For the considered number of scenarios we currently demand $|S| \leq \min\{\frac{n_\forall}{\log(n_\forall+1)}, 16\}$. This bound, however, has been adapted several times and is part of ongoing research.

The S-relaxation is only used in the very first variable block: as soon as the search dives deeper and universal variables are assigned along the search path, it is not ensured that the scenarios in S are still relevant in the current subtree. Therefore, as soon as a variable within the second variable block (or higher) is reached, the LP-relaxation with fixed scenario is applied, for which the incorporated scenario is adapted constantly.

If universal constraints are present, the problem arises that one cannot simply select scenarios $S \subseteq \mathcal{L}_\forall$, but must ensure that the scenarios are unavoidable scenarios (see Definition 5.5.9) and hence a U-DEP-relaxation is required (Definition 5.6.2). The generation of unavoidable scenarios is divided into two cases: a) the instance is a QIP^{PU}, i.e. universal constraints consist only of universal variables, and b) the instance is a QIP^{ID}, i.e. existential variables are present in universal constraints.

Generation of Unavoidable Scenarios for QIP^{PU} For the generation of unavoidable scenarios we highly benefit from the demand that the instance has to be simply restricted. We consider a scenario $\tilde{x}_\forall \in S \subseteq \mathcal{L}_\forall$ proposed by our heuristics and check whether it fulfills the universal constraint system. Note that such scenarios are not necessarily unavoidable, as their entries are sometimes collected independently during the search process. If $A^\forall \tilde{x}_\forall \not\leq b^\forall$ we utilize Algorithm 7 in order to incrementally find legal variable block assignments starting with the first universal variable block: in a local search we flip entries and detect resulting implied variables until Algorithm 7 returns `TRUE` and then move on to the next universal variable block and repeat. We try to stick as close as possible to the proposed universal variable assignment $\tilde{x}_\forall$. Of course there is still much room for improvement here, but for the universal constraint systems we have considered thus far, this worked very well.

Generation of Unavoidable Scenarios for QIP^{ID} For a general QIP^{ID} the same procedure cannot be used, as existential variable assignments can be crucial for the legality of universal

variable assignments. Hence, if there still are unassigned existential variables at the current search node, we generally have no chance of finding unavoidable scenarios within reasonable time. But in the following cases we can apply the same procedures as used for QIP^{PU}, to generate an unavoidable scenario during the search for QIP^{ID}:

a) All existential variables x_j with $A^{\forall}_{\star,j} \neq 0$ are already assigned. Hence, the local subproblem is a QIP^{PU}.

b) All unassigned universal variables x_j do not occur in any universal constraint, i.e. $A^{\forall}_{\star,j} = 0$. In this case the remaining universal variables can be assigned as in any scenario $\tilde{x}_{\forall} \in S$.

In order to improve the general LP-relaxation—which must be applied cautiously even for simply restricted QIP^{ID} (see Corollary 6.2.11)—universal variables x_j with $A^{\forall}_{\star,j} = 0$ can be assigned arbitrarily within $\mathcal{L}_j$ in the relaxation. Hence, such universal variables can be fixed similarly as it is done for general QIP. Furthermore, implied, but not yet assigned, universal variables (see page 143) can be fixed in the relaxation.

7.3. Computational Experiments

In this section we present computational studies carried out on several test problems. With these experiments we

- validate the effectiveness and correctness of the techniques presented in Section 3.1 and Section 5.6.
- compare the performance of our open-source solver Yasol solving quantified programs with a standard MIP solver solving the corresponding robust counterparts.
- investigate how well our solver performs on QIP^{ID} instances[14] compared to equivalent QIP instances.

All experiments were run on an Intel(R) Core(TM) i7-4790 with 3.60 GHz and 32 GB RAM. Our solver Yasol uses CPLEX (12.6.1) as LP solver. The runtime of Yasol is measured using wall-clock time rounded to seconds. We use CPLEX (12.9.0) as MIP solver in order to solve robust counterparts. Even though CPLEX provides more accurate options to measure time we used wall-clock time and rounded it to seconds in order to improve comparability. Furthermore, CPLEX was restricted to only one thread, as our algorithm is not parallelized. Additional information on this subject can be found on page 155.

The use of CPLEX (12.6.1)—rather than CPLEX (12.9.0)—as LP solver is due to the fact that a better comparability with our older results can be guaranteed and that the main algorithmic framework was built around CPLEX (12.6.1). In principle, however, the use of CPLEX (12.9.0) as LP solver is supported.

[14] Even though all investigates instances in this section are QIP^{PU}, i.e. a special type of QIP^{ID}, we use the term QIP^{ID} as the solution techniques presented for the general QIP^{ID} are applied.

7.3.1. Runway Scheduling

In this subsection, we investigate the multistage runway scheduling problem as introduced in Subsection 2.3.1 in which the universal player decides on the time windows in which airplanes must land. When selecting the lengths of the time windows the universal player is restricted as described in Subsection 4.4.3 on page 64. We compare the performance of our solver on the QIP^{ID} and the equivalent QIP. As no compact robust counterpart was built, we only briefly compare the performance of our solver with the performance of CPLEX on the DEP. We also refer to [HL19a] where we utilized the corresponding QIP to show the effectiveness of SCP and the impact of monotonicity in a small study.

Generation of Instances A multistage runway scheduling instance is defined via the number of airplanes $|A|$, the number of time slots $|S|$, the number of runways b, the initial planning costs $c_{i,j}$, the costs of rescheduling an airplane $c(x_{i,\star}, y_{i,\star})$ and the sets S and D describing the starting time slots and length of the time window the universal player is allowed to choose from. Furthermore, if the time windows of the airplanes do not become known all at the same time, an order of the airplanes is needed. Instances are created using a Java application and each random value is generated by the function `Java.util.Random.nextInt()`. The cost when an airplane needs to be rescheduled is set to $c(x_{i,\star}, y_{i,\star}) = R|\sum_{j\in S} j x_{i,j} - \sum_{j\in S} j y_{i,j}|$, where $R \in \{1, \dots, 5\}$ is a random value, i.e. for each slot that the airplane is moved away from its originally planned time slot, a cost of R is incurred. For fixed $|A|$ and $|S|$ the initial planning costs are generated as follows:

1. For each airplane $i \in A$ an original time slot $o_i \in S$ is randomly selected and the initial planning cost of this time slot is set to 0, i.e. $c_{i,o_i} = 0$.
2. The initial planning costs of the remaining time slots are generated using $c_{i,j} = |j - o_i| + U$ with $U \in \{-1, 0, 1\}$ randomly selected.

In order to specify the set of starting time slots S_i for each airplane $i \in A$, a basic starting time slot $b_i \in \{o_i - 2, o_i - 1, o_i, o_i + 1, o_i + 2\} \in S$ is randomly selected. Then one of the following eight cases is chosen at random for the universal domain (+ and − are different cases) :

a) $\mathsf{S}_i = \{b_i, b_i \pm 1\}$

b) $\mathsf{S}_i = \{b_i, b_i \pm 1, b_i \pm 2\}$

c) $\mathsf{S}_i = \{b_i, b_i \pm 1, b_i \pm 2, b_i \pm 3\}$

d) $\mathsf{S}_i = \{b_i, b_i \pm 2\}$

Similarly, for the duration of the time window a base length $\ell_i \in \{0, 1, 2\}$ is randomly selected. Then one of the following four cases is randomly chosen for the corresponding universal domain:

a) $D_i = \{\ell_i, \ell_i + 1\}$

b) $D_i = \{\ell_i, \ell_i + 1, \ell_i + 2\}$

c) $D_i = \{\ell_i, \ell_i + 1, \ell_i + 2, \ell + 3\}$

d) $D_i = \{\ell_i, \ell_i + 2\}$

The airplanes are then ordered with regard to their original time slot o_i and randomly split up into as many stages as required. The sorting is intended to ensure that the actual time windows for airplanes originally expected at similar times are revealed in chronological order. Finally, the universal constraint—restricting the set of lengths the universal player is allowed to choose—is given by

$$\sum_{i \in A} d_i \geq \lceil r \cdot |A| \rceil \ . \tag{7.7}$$

For each instance, r is selected at random from the open interval $[1.5, 2[$. The QIP$^{\text{ID}}$ is built by adding Constraint (7.7) as a universal constraint. The corresponding QIP is built as outlined in Subsection 4.4.3, i.e. we do not explicitly utilize the reduction function, but only add the required variables and constraints that detect and punish the universal player's misconduct, and only relax Constraint (2.7) to ensure a winning existential strategy in case of a violation of the universal constraint system.

QIP$^{\text{ID}}$ vs. DEP For building and solving the DEP of a robust runway scheduling QIP we can use the built-in option of Yasol, which internally builds the DEP and directly passes it to CPLEX. For each airplane 2 to 4 universal binary variables are required to represent the demanded time window. Therefore, the DEP grows rapidly for an increasing number of airplanes and consequently CPLEX has little chance to compete with the compact QIP. In Table 7.1 we

Table 7.1.: Comparsion of CPLEX (12.9.0) solving the DEP with our solver solving the QIP$^{\text{ID}}$ for randomly selected instances. Runtimes are given in seconds.

QIP$^{\text{ID}}$ instance data							runtime QIP$^{\text{ID}}$	runtime DEP
$\|A\|$	$\|S\|$	b	β	$n_\forall$	$n_\exists$	$m_\exists$		
3	6	2	3	9	49	37	1	2
4	8	2	3	12	81	49	1	63
5	6	2	3	13	81	53	1	355
4	10	3	7	13	97	53	1	130
5	7	2	5	13	91	53	3	1257
4	5	3	3	14	57	43	1	623
5	7	3	3	14	91	55	3	1532
5	7	3	3	18	91	55	13	> 3600

present the results for a very small number of randomly chosen (not handpicked) instances. As the DEP is solved by a call to CPLEX within our solver, we compiled our solver with CPLEX (12.9.0) for these computations. The results show that solving the corresponding DEP is considerably slower even for rather small instances. For instances with even more universal variables ($n_\forall$) the results indicate that only very few instances can be solved in reasonable time and therefore no further detailed experiments concerning the DEP are carried out. But note that an explicitly stated compact robust counterpart might be better manageable for CPLEX,

since building the DEP does not take domain-specific information into account, which could be done for an explicit MIP model.

QIP^{ID} vs. QIP In order to assess the benefit of using a universal constraint system instead of the equivalent QIP we created 3360 instances: for various numbers of airplanes $|A| \in \{2, \ldots, 8\}$, available timeslots $|S| \in \{3, \ldots, 10\}$ and runways $b \in \{2, 3, 4\}$ 20 instance for each constellation were created. We demanded $\beta = 3$, i.e. the time windows of all airplanes are revealed simultaneously in those instance. This resulted in 3360 QIP and QIP^{ID} instances, which we solved with a time limit of 200 seconds. The number of solved instances according to the used model, the average runtimes of those instances for which both models were solved as well as the resulting runtime difference in seconds (runtime of QIP^{ID} − runtime of QIP), are presented in Tables 7.2 and 7.3. As the infeasibility of an instance often can be detected significantly faster (in less than one second on average), compared to finding and proving the optimal solution, we subdivided the instances. Overall, for growing $|A|$ or $|S|$ more QIP^{ID} than QIP instances are found to be

Table 7.2.: Number of solved instances (within 200 seconds), the average runtime and runtime difference (in seconds) of solved instances with two to eight airplanes.

	$\|A\| = 2$	$\|A\| = 3$	$\|A\| = 4$	$\|A\| = 5$	$\|A\| = 6$	$\|A\| = 7$	$\|A\| = 8$
solved as infeasible							
solved QIP^{ID}s	0	12	29	81	112	192	230
solved QIPs	0	12	29	81	112	192	230
time difference	-	0	.0.3	0.06	-0.07	-0.14	0.03
solved as feasible							
solved QIP^{ID}s	480	468	451	399	367	272	179
solved QIPs	480	468	451	399	365	226	115
runtime QIP^{ID}	0.12	0.31	0.65	2.71	8.80	19.35	26.27
runtime QIP	0.14	0.26	0.93	4.53	19.31	42.80	51.13
time difference	-0.02	0.04	-0.28	-1.81	-10.51	-23.45	-24.87

Table 7.3.: Number of solved instances (within 200 seconds), the average runtime and runtime difference (in seconds) of solved instances with three to ten available time slots.

	$\|S\| = 3$	$\|S\| = 4$	$\|S\| = 5$	$\|S\| = 6$	$\|S\| = 7$	$\|S\| = 8$	$\|S\| = 9$	$\|S\| = 10$
solved as infeasible								
solved QIP^{ID}s	221	138	80	59	53	41	35	29
solved QIPs	221	138	80	59	53	41	35	29
time difference	0.01	-0.22	-0.03	0.07	0	0.22	-0.26	0.07
solved as feasible								
solved QIP^{ID}s	199	281	335	353	351	363	365	369
solved QIPs	199	280	330	346	333	336	340	340
runtime QIP^{ID}	0.38	1.88	4.68	7.81	5.74	4.02	5.91	5.94
runtime QIP	0.48	3.26	7.60	12.93	12.26	10.53	12.82	14.62
time difference	-0.1	-1.38	-2.92	-5.12	-6.52	-6.51	-6.91	-8.68

feasible. The number of instances found to be infeasible are the same for both models. Due to

the smaller average runtime for finding infeasibilities, the time difference fluctuates around zero. Note, however, that we observed a slight superiority of the QIP model on (other) larger instances when it comes to infeasible instances. This is probably due to the adapted S-relaxation and deduction techniques (see Subsection 7.2.3 and 7.2.2) that had to be weakened for general QIP$^{\text{ID}}$ instances. For feasible instances, however, more QIP$^{\text{ID}}$ instances were solved in significantly less time: for those instances solved in both cases, the runtime of the QIP$^{\text{ID}}$ is about half the runtime of the equivalent QIP.

We also compared the QIP$^{\text{ID}}$ model with the QIP on actual multistage instances, i.e. on instances where the time windows for the airplanes are not disclosed simultaneously. For $b = 3$, $|A| \in \{4, \dots, 8\}$, $|S| \in \{5, \dots, 12\}$ and $\beta \in \{5, 7, 9\}$ (i.e. two to four universal variable blocks) we created five instances for each configuration resulting in 600 instances. With a time limit of 600 seconds per instance our solver was used for both models. Only 14 instances were (detected to be) infeasible, which is mainly due to the better balance of airplanes and available time slots. For a concise comparison, the performance profiles [DM02] of these tests are displayed in Figure 7.1. It can be seen that the QIP$^{\text{ID}}$ model is solved fastest for about 83% of the instances.

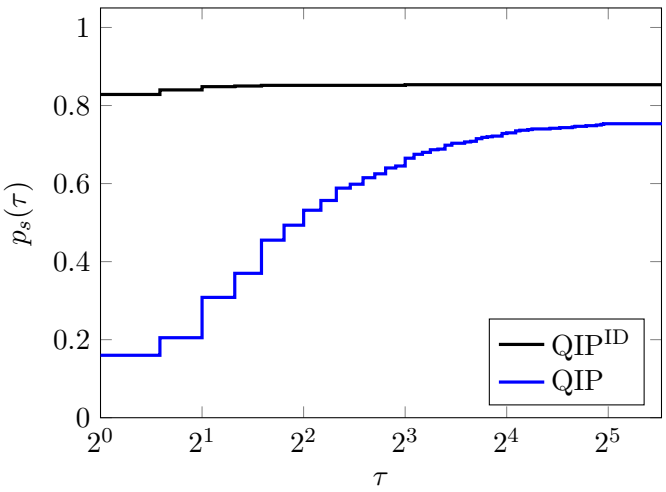

Figure 7.1.: Performance profiles for tests on multistage runway scheduling instances modeled as QIP$^{\text{ID}}$ and QIP with logarithmic τ-axis scale.

Overall, 85% of the QIP$^{\text{ID}}$ and 75% of the QIP instances are solved within the time limit (10 minutes). The time to solve the QIP is less than four times the runtime of the corresponding QIP$^{\text{ID}}$ in about 50% of all instances. Only 18 QIP instances are solved in *less* time than the corresponding QIP$^{\text{ID}}$.

Remark 7.3.1 (Performance profiles as introduced in [DM02])**.**
Let S be the set of considered solvers, P the set of instances and $t_{p,s}$ the runtime of solver s on instance p. The performance ratio of solver s on instance p is given by

$$r_{p,s} = \frac{t_{p,s}}{\min\limits_{\hat{s} \in S} t_{p,\hat{s}}} .$$

We assume $t_{p,s}$ is set to infinity (or large enough) if solver s does not solve instance p within the time limit. The percentage of instances for which the performance ratio of solver s is within a factor $\tau \geq 1$ of the best ratio of all solvers is given by

$$p_s(\tau) = \frac{1}{|P|} \left| \{ p \in P : \ r_{p,s} \leq \tau \} \right| .$$

Hence, the function p_s can be viewed as the distribution function for the performance ratio, which is plotted in a performance profile for each solver. For each future performance profiles in this thesis we calculate $p_s(\tau)$ for $\tau = 1 + 0.5t$ with $t \in \mathbb{N}_0$ and t large enough until $p_s(\tau)$ is constant.

Impact of SCP For the robust runway scheduling problem SCP can be interpreted as follows: Assume for a given realization of universal variables, i.e. the demanded time windows, a valid schedule is found. SCP is then used to check whether the same schedule can be reused for other realizations of the universal variables, i.e. other time windows. Here it becomes intuitively clear that in some cases a schedule is still valid even if several universal variables are changed, as long as the specified time windows contain the planned time slot for each airplane. We restrict ourselves to the case with $|A| = 7$ airplanes, $|S| = 12$ time slots and $b = 3$ runways, as for this setting the results of the previous paragraph showed that insightful results could be obtained in reasonable time. For each instance the time limit is set to 1800 seconds. We created 300 instances with one universal variable block, i.e. for all airplanes the realization of their demanded time window is revealed at the same time. In Table 7.4 a performance comparison is given for those instances. The immense benefit SCP brings to QIP$^{\text{ID}}$ instances is astonishing, as not only

Table 7.4.: Number of instances solved, overall runtime of the experiments and the average runtime for those instances solved in all four setting (170 instances).

	number of solved instances	overall computation time in hours	average runtime in seconds
QIP$^{\text{ID}}$ with SCP	300	3.00	10.42
QIP$^{\text{ID}}$ without SCP	244	52.87	172.64
QIP with SCP	232	46.41	124.28
QIP without SCP	176	86.37	477.98

all instances are solved, but the runtime is significantly reduced to less than a tenth compared to the runs without SCP. Furthermore, we can see again that the QIP$^{\text{ID}}$ is far superior on these runway scheduling instances compared to the equivalent QIP, as 68 more instances are solved and solving the QIP also requires more than ten times as much time: the average runtimes on all 232 instances solved in these two cases are 16 seconds for QIP$^{\text{ID}}$ and 193 seconds for QIP. As already observed in previous tests [HL19a], using SCP to solve such QIPs results in a reduction of the runtime by a factor of about 4. Figure 7.2 shows the progression of the number of instances solved during the four tests as well as the performance profiles. Note that only the endpoints of the progression curves are significant, since their course depends strongly on the

order of the instances. The very steep increase for the number of solved QIP^{ID} instances with SCP is quite remarkable compared to the other three tests and the performance profile shows a clear superiority. The performance profile for solving QIP^{ID} without SCP proceeds very similar to the one for QIP with SCP, while more QIP^{ID} instances are solved this way but in more time. With SCP the QIP^{ID} was solved first in all but three instances and never took more than a

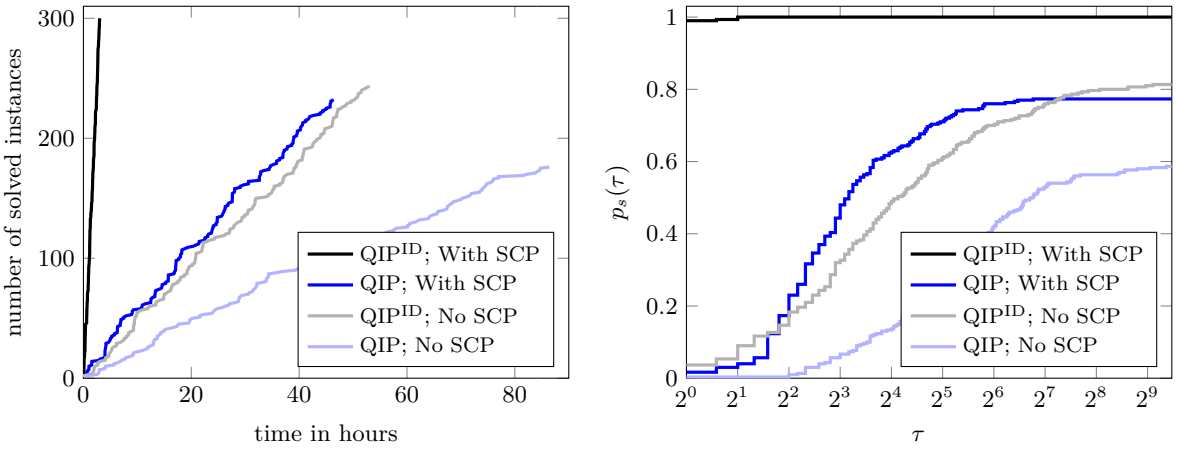

Figure 7.2.: Progression of the number of instances solved (left) and the performance profiles (right) for solving the QIP or QIP^{ID} model with or without the use of SCP.

factor of 2 longer. In fact in two of those cases the runtime of the QIP^{ID} was 2 seconds and the fastest runtime was 1 second, which both are rounded values. The number of QIP^{ID}-instances solved first without SCP slightly exceeds the number of QIPs solved fastest with SCP.

These results show that SCP has a massive impact if a single universal variable block is present. We now want to examine the impact of SCP if there is more than one universal variable block. We randomly divided the seven airplanes into groups, for which the time windows are disclosed simultaneously. We build 300 instances with each 1, 2, 3 and 4 groups, i.e. each 300 instances with overall 3, 5, 7 and 9 variable blocks. For each setting we compare the performance of our solver with and without SCP on the arising QIP^{ID}. In Table 7.5 the results are displayed. For

Table 7.5.: Number of instances solved and average runtime for instances solved in both cases.

		number of solved instances	average runtime (in seconds)
$\beta = 3$	QIP^{ID} with SCP	300	20.28
	QIP^{ID} without SCP	244	366.95
$\beta = 5$	QIP^{ID} with SCP	276	139.60
	QIP^{ID} without SCP	232	403.73
$\beta = 7$	QIP^{ID} with SCP	260	180.63
	QIP^{ID} without SCP	210	383.91
$\beta = 9$	QIP^{ID} with SCP	252	237.43
	QIP^{ID} without SCP	221	373.85

$\beta = 3$ we used the results of the previous comparison with the QIP. For increasing β the number of solved instances tends to decrease while our solver is always able to solve more instances, when SCP is enabled. The runtimes on the instances for which a solution is found with and without SCP seem to converge for increasing β. Overall, SCP also contributes massively to the optimization process if multiple stages are considered, while its impact is most impressive on instances with few stages.

7.3.2. Multistage Selection Problem

In this subsection, we investigate the multistage selection problem as introduced in Subsection 4.4.4 in which p out of n items must be selected. We compare the performance of our solver on the quantified models SELQ$^{\text{PU}}$ and SELQ with the performance of CPLEX on the robust counterpart SELRC.

Generation of Instances An instance is given by the number of available items n, the number of items p to be selected, the number of iterations S and the number of scenarios N per iteration. We limit ourselves to instances with $n = 2p$ and thus, the value of p is omitted from now on. The remaining parameters of an instance are the costs $c^s_{i,k}$ of each each item i in scenario k of iteration s, which are randomly selected from the range $0, 1, \ldots, 99$. Those values are created using the `C++` function `rand()` from the standard general utilities library and the modulo operator.

CPLEX as Solver for the Robust Counterpart We use CPLEX (12.9.0) as MIP solver in order to solve the robust counterpart. Since Yasol currently only uses a single thread we also wanted to restrict CPLEX to a single thread in order to obtain a more fair comparison. However, CPLEX turned out to be even faster on our instances if restricted to a single thread. This is illustrated in Table 7.6 for multistage selection instances with $N = 4$, $S = 6$ and various numbers of items $n \in \{10, 20, 30, 40, 50\}$ with a time limit of 1800 seconds per instance. Even

Table 7.6.: Number of instances solved and average runtime for those instances solved with both settings for CPLEX (12.9.0) restricted to one thread and with default settings.

	number of solved instances		average runtime (in seconds)		number of instances solved first	
	1 thread	default	1 thread	default	1 thread	default
$n = 10$	50	50	13.28	15.70	35	13
$n = 20$	50	50	54.74	70.66	41	9
$n = 30$	48	50	101.04	159.02	44	6
$n = 40$	50	50	232.29	263.37	39	11
$n = 50$	49	50	241.57	293.36	42	8

though default CPLEX was able to solve three more instances, the runtimes on those instances solved by both configurations is significantly larger in comparison to CPLEX on a single thread. Overall, default CPLEX could only solve less than a fifth of the instances faster than CPLEX restricted to a single thread. Similar behavior was observed for other instances of the multistage

selection problem as well as the multistage assignment problem. We have thus unintentionally somewhat fine tuned CPLEX for those instances.

Memory Usage The first thing one notices when creating multistage selection instances is the enormous difference in the file sizes of instances of the three presented models. This is certainly not surprising, but it is quite impressive when a few instance of the robust counterpart blow up the hard drive, whereas the same instances as QIP or QIP^{ID} do not even require one megabyte. For example, for an instance with $n = 10$, $S = 5$ and $N = 32$ the QIP^{ID} requires about 28 KB, the QIP requires 36 KB and the robust counterpart LP requires more than 91 GB. Thus, it becomes obvious that solving the robust counterpart of such instances may already fail when trying to import the LP-file into the solver. We refer to Appendix B.2 on pages 178ff. for a data table of selection instances with ten items, which also contains information about the file size of such instances.

In addition to this very compact problem description of the SELQ^{PU} and SELQ models, our solver does not excessively make use of memory during the search process. The same cannot be said about CPLEX when solving SELRC. This is partially due to the sheer number of variables and constraints in the robust counterpart but also due to the fact that our solver does not explicitly store the search tree. In Figure 7.3 the average maximum RAM used during the entire

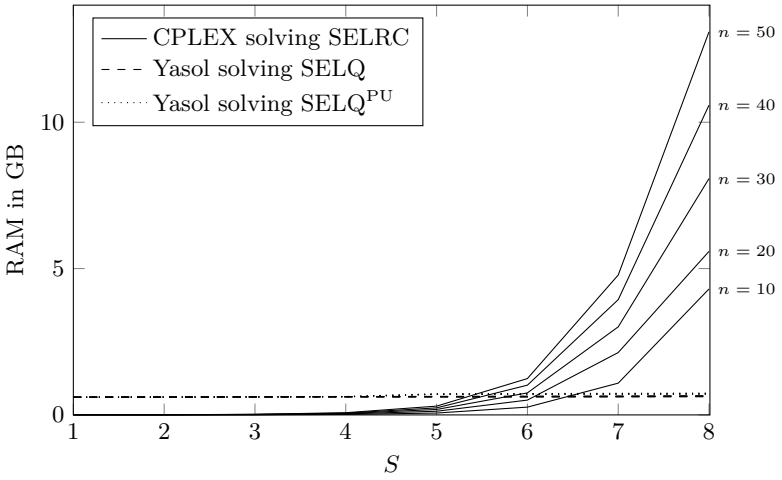

Figure 7.3.: Average of the maximum RAM required during the solution process (time limit 1800 seconds) of the different models for $N = 4$. 250 instances for each $S \in \{1, \ldots, 8\}$.

search process is depicted. Similar to the increasing size of the instance itself, the memory usage of CPLEX solving the robust counterpart increases dramatically for increasing number of iterations and scenarios, while the memory usage of our solver only slightly increases. The data can be found in Table B.1 on page 177. The memory usage when solving SELQ^{PU} is slightly higher compared to the solution process for SELQ. This is partially due to the overhead of having to maintain the universal constraint system $A^{\forall}x \leq b^{\forall}$. Furthermore, as described in Subsection 7.2.3, the size of the built S-relaxation depends on the number of universal variables, which is larger for SELQ^{PU} and hence the required memory for the relaxation is increased.

Fixed Number of Scenarios, Various S and n We now fix the number of scenarios in each iteration to $N = 4$ and vary the number of item $n \in \{10, 20, 30, 40, 50\}$ and the number of iterations $S \in \{1, \dots, 8\}$. For each setting 50 instances of each of the three models SELQ$^{\text{PU}}$, SELQ and SELRC are created. The quantified programs are solved using our solver and the robust counterpart is solved using CPLEX with a maximum time limit of 1800 seconds. We are interested in the number of instances solved within the time limit and the runtimes. In Table 7.7 the number of solved instances for each setting is display.

Table 7.7.: Number of solved multistage selection instances for fixed number of scenarios $N = 4$ and various n and S within 1800 seconds.

	model	$S=1$	$S=2$	$S=3$	$S=4$	$S=5$	$S=6$	$S=7$	$S=8$
$n=10$	SELRC	50	50	50	50	50	50	49	30
	SelQ	50	50	50	50	50	50	50	50
	SELQ$^{\text{PU}}$	50	50	50	50	50	50	50	50
$n=20$	SELRC	50	50	50	50	50	50	32	0
	SelQ	50	50	50	50	50	50	50	50
	SELQ$^{\text{PU}}$	50	50	50	50	50	50	50	50
$n=30$	SELRC	50	50	50	50	50	48	21	0
	SelQ	50	50	50	50	50	50	50	49
	SELQ$^{\text{PU}}$	50	50	50	50	50	50	50	50
$n=40$	SELRC	50	50	50	50	50	50	15	0
	SelQ	50	50	50	50	50	50	49	34
	SELQ$^{\text{PU}}$	50	50	50	50	50	49	50	41
$n=50$	SELRC	50	50	50	50	50	49	7	0
	SelQ	50	50	50	50	50	46	31	15
	SELQ$^{\text{PU}}$	50	50	50	50	50	48	33	20

As expected, the number of solved instances within the time limit tends to decrease for increasing n and S. With the exception of $S = 6$, $n \geq 40$ we can observe that a) the number of solved robust counterparts is never higher than the number of solved quantified programs and b) the number of solved instances with universal constraints (SELQ$^{\text{PU}}$) is always the highest. The number of solved robust counterparts in particular tends to decrease significantly faster for an increasing S. In most cases all 50 instances are solved and thus the average runtimes—of those instances for which all three models are solved—are shown in Table 7.8.

The values with asterisk are less significant as they are based on fewer instances, as not all three models were solved to optimality. Keep in mind that all runtimes have been rounded to seconds and hence very small average runtimes may also not be too significant. For most settings the average runtime of the quantified program with universal constraints SELQ$^{\text{PU}}$ is lower than for the pure QIP SELQ. For fixed S and increasing number of items n the runtime of the robust counterpart does not grow as quick as the runtime of the quantified programs. In particular, for $S = 6$ SELQ and SELQ$^{\text{PU}}$ are solved faster on average up to $n = 40$. For $n = 50$ CPLEX can display its strength and the (average) runtime even decreases compared to the average runtime for instances with $n = 40$. This decrease, however, is owed to the only 44 instances for which all three models with $n = 50$ are solved. Note that for $S = 6$ and $n = 50$ the

Table 7.8.: Average runtime (in seconds) of multistage selection instances with $N = 4$ for which each model was solved. Values marked with an asterisk resulted from less than 20 solved instances. A hypen indicates that for no instance all three models were solved.

	model	$S = 1$	$S = 2$	$S = 3$	$S = 4$	$S = 5$	$S = 6$	$S = 7$	$S = 8$
$n = 10$	SELRC	0.0	0.0	0.0	0.0	1.5	13.3	186.6	1016.7
	SELQ	0.1	0.1	0.2	0.3	0.7	2.6	7.5	32.1
	SELQ$^{\text{PU}}$	0.1	0.1	0.1	0.2	0.6	1.2	4.2	9.7
$n = 20$	SELRC	0.0	0.0	0.0	0.7	4.3	54.7	587.5	-
	SELQ	0.1	0.2	0.7	1.4	5.5	12.1	40.9	-
	SELQ$^{\text{PU}}$	0.1	0.2	0.5	0.9	2.6	7.5	26.2	-
$n = 30$	SELRC	0.0	0.0	0.0	1.1	8.6	101.0	859.8	-
	SELQ	0.2	0.5	2.7	7.9	15.5	38.4	88.5	-
	SELQ$^{\text{PU}}$	0.2	0.4	1.7	8.1	14.3	36.5	65.9	-
$n = 40$	SELRC	0.0	0.0	0.0	1.6	11.6	232.3	1029.2*	-
	SELQ	1.3	1.8	5.6	28.3	35.1	136.5	177.6*	-
	SELQ$^{\text{PU}}$	1.3	1.9	3.7	14.3	46.9	113.5	162.9*	-
$n = 50$	SELRC	0.0	0.0	0.0	2.1	17.0	206.4	1538.2*	-
	SELQ	1.4	3.0	13.9	55.8	112.6	350.3	290.6*	-
	SELQ$^{\text{PU}}$	1.4	2.6	8.0	35.1	94.6	281.6	462.2*	-

SELRC model already has more than a quarter of a million variables and more than 200,000 constraints, while SELQ$^{\text{PU}}$ only needs 380 variables and 81 constraints to represent the same instance. For fixed n and increasing S, however, there always exists a threshold S' for which the quantified programs outperform the robust counterpart: the vertical lines indicate in which area the robust counterpart is solved faster (on average) than SELQ$^{\text{PU}}$. The conjecture that this dominance remains true for even larger S is strongly supported by the pure growth of the instance itself and the resulting difficulty of CPLEX to manage the needed RAM or even load the LP-file. In summary, CPLEX is able to solve the robust counterpart faster for a large number of items and our solver can better handle a large number of iterations in the quantified programs. For a better comparison of the perfomances of our solver on SELQ$^{\text{PU}}$ and SELQ we refer to Table B.2 in the appendix on page 177, which shows the average runtimes of those instances for which both quantified models were solved. The main observation remains, that instances with universal constraints are solved faster on average than the pure QIPs.

Furthermore, for each model we consider the number of instances for which this model was solved the fastest. Figure 7.4 shows the percentage of instances where the other models were solved slower. Note that the numbers do not necessarily add up to 100% due to instances with two or more models with the same runtime and due to unsolved instances.

These graphs further support the claim that the higher the number of items, the better CPLEX performs compared to our solver on the same instance. On the other hand, the more iterations are considered within an instance, the better our solver performs compared to CPLEX.

Finally we provide the performance profiles (see Remark 7.3.1) for all three models in Figure 7.5. We also refer to Appendix B.3 for further very insightful performance profiles, where the instances are separated according to n and S. Note that for each instance with runtime of 0 seconds we used the runtime of 1 second in order to be able to generate useful performance

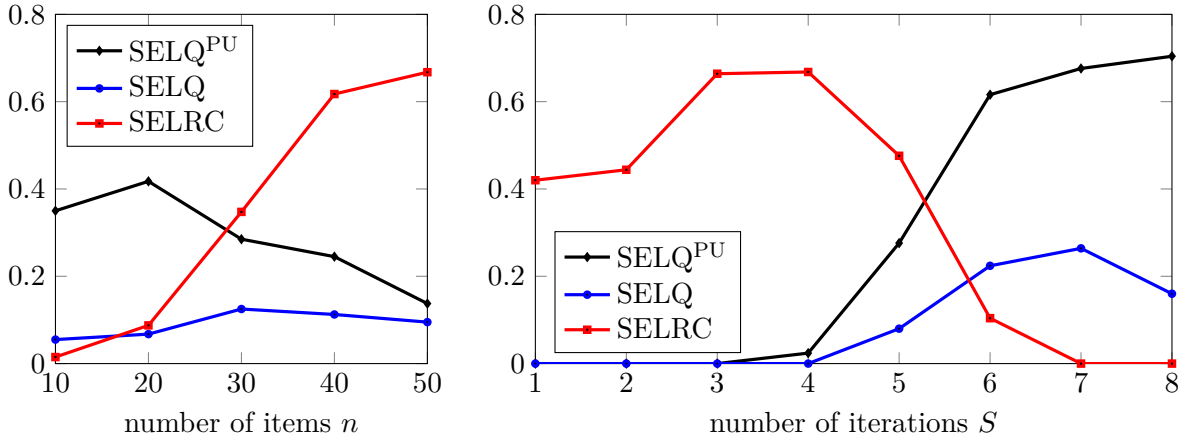

Figure 7.4.: Percentage of instances per model where it was solved the fastest, i.e. if both other models took *more* time to solve. Plot for each number of items n (left-hand side) and each number of iterations S (right-hand side).

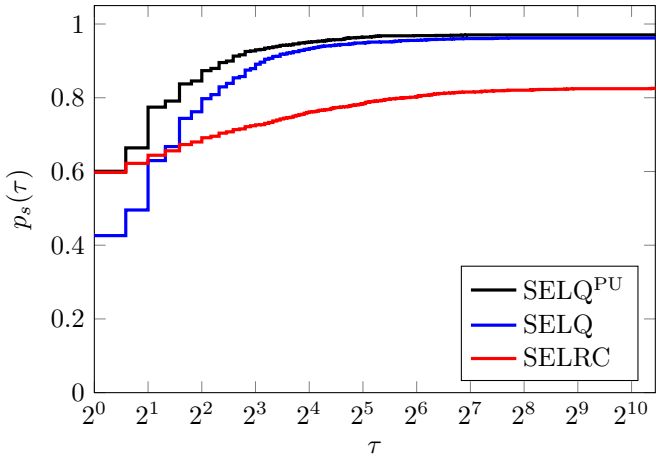

Figure 7.5.: Performance profile for all examined multistage selection instances with $N = 4$ and each modeling variant.

profiles. The robust counterpart as well as the QIP with universal constraints is solved fastest on about 60% of all instances. Furthermore, the performance profile for SELQ$^{\text{PU}}$ remains above the other two profiles and our solver is able to solve more than 87% of the SELQ$^{\text{PU}}$ instances within a factor of 4 compared to the fastest solved instance. In the performance profiles provided in the Appendix B.3 the deterioration of CPLEX solving the robust counterpart for increasing number of iterations as well as the deterioration of our solver on the quantified programs for increasing number of items is clearly visible.

Comparison to Simple Heuristics In order to emphasize the benefit of solving the robust multistage optimization problem we now compare the worst-case outcome of the three online strategies (see pages 68ff.) with the optimal solution of the multistage selection problem. As a reminder: In strategy 1 the p cheapest items in the initial stage are bought. In strategy 2 the lowest guaranteed future price for an item is compared to its current price. In strategy 3 an

item is bought only if there exists no future iteration in which the remaining number of items can be bought for a cheaper price.

We use the instances as above with fixed number of $N = 4$ scenarios and various constellations of $n \in \{10, 20, 30, 40, 50\}$ and $S \in \{1, \ldots, 8\}$ with 50 instances per constellation resulting in 2000 instances. In order to detect the worst-case outcome when using one of the presented heuristics a tree search is implemented in Python. For any instance it took only seconds to determine the worst-case outcome of any strategy, which is the obvious advantage of a heuristic. Determining the worst-case outcome of strategy 1 was the fastest ($\ll 1$ second per instance), followed by strategy 2 (< 1 second per instance) and strategy 3 (≈ 1 second per instance). The computational times for the optimal solution was discussed in the previous paragraph and in general significantly exceed these times.

Note that for 45 of the 2000 instances no optimal solution was found or ensured by either model. All three heuristic strategies managed to outperform the *best known* value for 7 instances, in which case only a (very trivial) starting solution was found during the optimization process of each of the three models SELQ$^{\text{PU}}$, SELQ and SELRC. From now on we disregard the 45 instances of which we do not know the optimal solution. Strategy 1 never resulted in the optimal value, while strategy 2 reached the optimal value in 19 cases and strategy 3 in 143 cases. Those 143 optimally solved instances by using Strategy 3, however, are either instances with a single iteration, or with at most 20 items. In general we can say that the more items and the more iterations are considered, the larger is the relative deviation from the optimum for all three heuristics. In Appendix B.4 we provide box plots of the relative deviation of strategy 2 and 3 for each number of iterations and items. In Table 7.9 the average relative deviation from

Table 7.9.: Average relative deviation of the worst-case outcome when applying each strategy from the instance's optimal value. 50 instances per cell with the exception of ($n = 40$, $S = 8$: 43 instances), ($n = 50$, $S = 7$: 38 instances) and ($n = 50$, $S = 8$: 24 instances).

	strat.	$S = 1$	$S = 2$	$S = 3$	$S = 4$	$S = 5$	$S = 6$	$S = 7$	$S = 8$
	1	0.66	0.80	1.16	1.28	1.38	1.34	1.71	1.80
$n = 10$	2	0.29	0.32	0.50	0.55	0.56	0.62	0.66	0.68
	3	0.04	0.08	0.14	0.22	0.20	0.23	0.30	0.31
	1	0.68	1.04	1.38	1.60	2.13	2.06	2.06	2.56
$n = 20$	2	0.33	0.49	0.72	0.80	1.02	1.00	0.98	1.21
	3	0.05	0.13	0.26	0.27	0.38	0.41	0.45	0.57
	1	0.64	1.18	1.69	2.01	2.30	2.57	2.75	2.96
$n = 30$	2	0.35	0.59	0.85	1.02	1.16	1.32	1.43	1.48
	3	0.06	0.14	0.24	0.34	0.45	0.58	0.65	0.77
	1	0.78	1.32	1.79	2.30	2.53	2.94	3.22	3.42
$n = 40$	2	0.40	0.69	0.99	1.14	1.30	1.48	1.64	1.70
	3	0.06	0.18	0.31	0.42	0.54	0.69	0.74	0.88
	1	0.80	1.38	1.90	2.25	2.79	2.99	3.60	3.94
$n = 50$	2	0.41	0.71	1.00	1.23	1.41	1.63	1.82	2.02
	3	0.06	0.16	0.30	0.47	0.63	0.81	0.87	1.01

the optimum are shown to give a rough idea. Strategy 3 is always closest to the optimal value

on average and the average relative deviation almost always increases for increasing number of items and iterations. We expect similar behavior for a growing number of scenarios. Note that even though strategy 3 is the best on average there are 7 instances in which strategy 2 results in a better worst-case outcome. Obviously, further improvements are possible, and other online strategies could lead to even smaller deviations from the optimal value, but with potentially higher computing time. Due to the negligible computational effort and the good results, strategy 3 is suitable as a generator of good starting solutions for a domain-specific solver for the multistage selection problem.

Impact of SCP For the robust runway scheduling problem we have already shown that SCP significantly benefits the optimization process. For the multistage selection problem we do not expect such a positive impact since a different assignment of a universal variable block represents a totally different cost scenario. In particular, keeping the assignments of existential variables—indicating the bought items—unchanged for several scenarios of a single iteration cannot result in an optimal winning strategy in the vast majority of cases. For SELQ we can even a priori rule out that SCP has a positive effect: different assignments of the universal variable ℓ_s—indicating the number of the selected scenario in iteration s—call for different assignments of the subsequent existential indicator variables q_k^s due to Constraint (4.51). Hence, for any two different universal variable assignments the subsequent existential variables must not be assigned to the same value making SCP inapplicable. We are therefore curious to what extent SCP has a positive effect on the QIP$^{\text{ID}}$ SELQ$^{\text{PU}}$ and whether the constant querying of SCP without any prospect of success is unfavorable for the QIP SELQ. For $n = 50$, $S = 4$ and $N = 4$ we created 1000 instances and solved SELQ and SELQ$^{\text{PU}}$ using our solver with SCP turned on and off with a time limit of 1800 seconds. The results in Table 7.10 show surprisingly clearly what we

Table 7.10.: Number of instances solved, overall runtime of the experiments and the average runtime for those multistage selection instances solved with and without SCP for each model.

	number of solved instances	overall computation time (in hours)	average runtime (in seconds)
SELQ$^{\text{PU}}$ with SCP	997	10.52	32.32
SELQ$^{\text{PU}}$ without SCP	996	12.43	37.62
SELQ with SCP	998	16.60	56.27
SELQ without SCP	998	16.51	55.95

expected: the use of SCP has a positive impact on SELQ$^{\text{PU}}$, but not as impressive as for the runway scheduling instances. For SELQ the use of SCP does not significantly affect the runtime, which is also a very positive result, as it shows that the computational overhead of repeatedly checking the applicability of SCP is negligible.

Fixed Number of Items, Various S and N So far the number of scenarios was fixed to $N = 4$, but we also want to examine how the different approaches deal with instances with various

numbers of scenarios. It is expected that with an increasing number of scenarios the approach of solving the quantified program is superior to solving the robust counterpart. We fix the number of items to $n = 10$, which is quite small but necessary in order to allow large values of S and N and still be able to find the optimal solution for many instances in reasonable time. For various $S \in \{1, \ldots, 8\}$ and $N \in \{2^1, 2^2, \ldots, 2^8\}$ again 50 instances are created for most constellations. Note that not all constellations are considered, since the prospects of finding the optimal solution within the time limit of 1800 seconds is very small, if both S and N are large. The following Table 7.11 shows the number of instances solved. For cells marked with

Table 7.11.: Number of solved multistage selection instances for fixed number of items $n = 10$ and various N and S within 1800 seconds.

	model	$S=1$	$S=2$	$S=3$	$S=4$	$S=5$	$S=6$	$S=7$	$S=8$
$N=2^1$	SELRC	50	50	50	50	50	50	50	50
	SELQ	50	50	50	50	50	50	50	50
	SELQ$^{\text{PU}}$	50	50	50	50	50	50	50	50
$N=2^2$	SELRC	50	50	50	50	50	50	49	26
	SELQ	50	50	50	50	50	50	50	50
	SELQ$^{\text{PU}}$	50	50	50	50	50	50	50	50
$N=2^3$	SELRC	50	50	50	50	48	4	0	0*
	SELQ	50	50	50	50	50	50	47	0
	SELQ$^{\text{PU}}$	50	50	50	50	50	50	50	38
$N=2^4$	SELRC	50	50	50	44	0	0*	0*	-
	SELQ	50	50	50	50	50	0	0	-
	SELQ$^{\text{PU}}$	50	50	50	50	50	36	8	-
$N=2^5$	SELRC	50	50	50	3	0*	0*	-	-
	SELQ	50	50	50	50	0	0	-	-
	SELQ$^{\text{PU}}$	50	50	50	50	23	1	-	-
$N=2^6$	SELRC	50	50	32	0*	0*	-	-	-
	SELQ	50	50	50	0	0	-	-	-
	SELQ$^{\text{PU}}$	50	50	50	27	2	-	-	-
$N=2^7$	SELRC	50	50	0	0*	-	-	-	-
	SELQ	50	50	14	0	-	-	-	-
	SELQ$^{\text{PU}}$	50	50	48	3	-	-	-	-
$N=2^8$	SELRC	50	48	0*	-	-	-	-	-
	SELQ	50	50	0	-	-	-	-	-
	SELQ$^{\text{PU}}$	50	50	11	-	-	-	-	-

a hyphen no experiments were conducted as it is expected that none (or very few) instances would be solved in the given time limit. The 0 entries with an asterisk indicate that we tried to create and solve these robust counterparts, but they exceeded our available RAM, simply due to their enormous file size (see Table B.3 in Appendix B.2). For most entries with largest N for each S—not listed in Table B.3—the instance sizes of SELRC (well) exceeded 200GB in which case we stopped the file creation. As expected, for increasing N and S the number of quantified programs solved by Yasol tends to be larger than the number of robust counterpart solved by CPLEX. For each configuration, the number of solved SELQ$^{\text{PU}}$ models is highest and at least one SELQ$^{\text{PU}}$ instance is always solved. On 243 of the 303 instances where no model

was solved to optimality, SELQ$^{\text{PU}}$ resulted in the best incumbent solution. On 175 of those instances SELQ$^{\text{PU}}$ was the only model for which any solution was found at all. On additional 23 instances the optimal solution of SELQ$^{\text{PU}}$ was found while for the other models not even an incumbent solution was found.

Furthermore, we are interested in the runtimes of CPLEX and Yasol on the robust counterpart and the quantified programs, respectively. In Table 7.12 we present the average runtimes on instances of which all models were solved. For configurations with too few or no instances solved

Table 7.12.: Average runtime (in seconds) of multistage selection instances with $n = 10$ for which each model type was solved. Values marked with an asterisk show the average runtime on all solved instances of that model type.

	model	$S=1$	$S=2$	$S=3$	$S=4$	$S=5$	$S=6$	$S=7$	$S=8$
$N=2^1$	SELRC	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
	SELQ	0.1	0.1	0.1	0.2	0.2	0.3	0.4	0.4
	SELQ$^{\text{PU}}$	0.1	0.1	0.1	0.1	0.0	0.2	0.3	0.5
$N=2^2$	SELRC	0.0	0.0	0.0	0.1	1.5	10.8	191.1	1023.7
	SELQ	0.1	0.1	0.2	0.2	0.7	2.2	8.7	28.6
	SELQ$^{\text{PU}}$	0.0	0.1	0.2	0.2	0.5	1.1	4.1	6.3
$N=2^3$	SELRC	0.0	0.0	0.3	7.3	306.3	1261*	-	-
	SELQ	0.0	0.2	0.4	2.0	13.9	103.9*	848.2*	-
	SELQ$^{\text{PU}}$	0.1	0.1	0.4	1.0	3.9	26.2*	146.8*	678.8*
$N=2^4$	SELRC	0.0	0.0	6.3	396.3	-	-	-	-
	SELQ	0.1	0.2	2.2	30.8	557.5*	-	-	-
	SELQ$^{\text{PU}}$	0.0	0.3	1.1	7.3	111.7*	816.9*	895*	-
$N=2^5$	SELRC	0.0	0.8	102.4	1667.7*	-	-	-	-
	SELQ	0.1	0.8	19.7	600*	-	-	-	-
	SELQ$^{\text{PU}}$	0.2	0.5	5.3	129.3*	644.6*	970*	-	-
$N=2^6$	SELRC	0.0	4.2	807.5	-	-	-	-	-
	SELQ	0.3	3.6	222.7	-	-	-	-	-
	SELQ$^{\text{PU}}$	0.2	1.4	38.7	800.9*	1490.5*	-	-	-
$N=2^7$	SELRC	0.0	25.4	-	-	-	-	-	-
	SELQ	0.4	20.0	1221.9*	-	-	-	-	-
	SELQ$^{\text{PU}}$	0.4	6.8	358.3*	398.7*	-	-	-	-
$N=2^8$	SELRC	0.0	187.6	-	-	-	-	-	-
	SELQ	1.0	147.5	-	-	-	-	-	-
	SELQ$^{\text{PU}}$	1.0	45.0	621.1*	-	-	-	-	-

for all models, we use the average runtime of all solved instances of that model type. Those value are marked with an asterisk. We highlight the modest increase of the runtime of SELQ$^{\text{PU}}$ models for increasing S and N, even compared to SELQ. These two tables show in an impressive manner, that a) for instances with many iterations and scenarios the use of quantified programs is far superior to solving the robust counterpart, and b) utilizing universal constraints rather than standard QIPs is of surprisingly great advantage in this setting.

7.3.3. Multistage Assignment Problem

In this subsection we briefly investigate the multistage assignment problem as introduced in Subsection 4.4.5 in which a perfect matching in a bipartite graph with minimal costs has to be determined. We compare the performance of our solver on the quantified models ASSQ$^{\text{PU}}$ and ASSQ with the performance of CPLEX on the robust counterpart ASSRC.

Generation of Instances Each instance is given by the size n of each partition, the number of iterations S and the number of scenarios N per iteration. The remaining parameters of an instance are the cost $c^s_{i,j,k}$ for each edge (i,j) in scenario k of iteration s which are randomly selected from the range $0, 1, \ldots, 99$. Those values are created using the `C++` function `rand()` from the standard general utilities library and the modulo operator.

QIP$^{\text{ID}}$ vs. QIP vs. Robust Counterpart For each $n \in \{4, \ldots, 10\}$, $S \in \{1, \ldots, 4\}$ and $N \in \{2^1, 2^2, 2^3\}$ we created 50 instances for each model type ASSQ$^{\text{PU}}$, ASSQ and ASSRC. The quantified models are solved with our solver and the robust counterpart is solved with CPLEX with a time limit of 1800 seconds. We examine how the realization of n, S and N affects the runtime and which model-solver combination is best suited for the different instances. Based on the results of the multistage selection problem we expect that the runtime for ASSQ$^{\text{PU}}$ is lower than for ASSQ. Furthermore, for increased S and N we expect a growing advantage for the quantified models, while for increasing n CPLEX on ASSRC is expected to become more competitive. In Table 7.13 for instances with $N = 4$ scenarios both the number of solved

Table 7.13.: Number of solved multistage assignment instances with $N = 4$ and average runtime.

	model	$S=1$		$S=2$		$S=3$		$S=4$	
$n=4$	ASSRC	50	(0.0)	50	(0.0)	50	(0.0)	50	(3.1)
	ASSQ	50	(0.1)	50	(0.1)	50	(0.3)	50	(0.7)
	ASSQ$^{\text{PU}}$	50	(0.1)	50	(0.1)	50	(0.2)	50	(0.5)
$n=5$	ASSRC	50	(0.0)	50	(0.0)	50	(0.6)	50	(14.5)
	ASSQ	50	(0.0)	50	(0.3)	50	(1.0)	50	(2.6)
	ASSQ$^{\text{PU}}$	50	(0.1)	50	(0.2)	50	(0.5)	50	(1.8)
$n=6$	ASSRC	50	(0.0)	50	(0.0)	50	(2.0)	50	(46.9)
	ASSQ	50	(0.2)	50	(1.0)	50	(3.9)	50	(13.3)
	ASSQ$^{\text{PU}}$	50	(0.2)	50	(0.6)	50	(1.9)	50	(6.4)
$n=7$	ASSRC	50	(0.0)	50	(0.0)	50	(8.9)	48	(152.7)
	ASSQ	50	(0.2)	50	(5.3)	50	(19.4)	50	(72.7)
	ASSQ$^{\text{PU}}$	50	(0.2)	50	(1.7)	50	(6.0)	50	(25.2)
$n=8$	ASSRC	50	(0.0)	50	(0.1)	50	(26.1)	33	(470.7)
	ASSQ	50	(0.4)	50	(14.4)	50	(76.3)	50	(257.5)
	ASSQ$^{\text{PU}}$	50	(0.3)	50	(3.5)	50	(26.0)	50	(117.0)
$n=9$	ASSRC	50	(0.0)	50	(0.7)	50	(155.5)	15	(615.9)
	ASSQ	50	(0.6)	50	(59.9)	50	(365.2)	33	(1070.9)
	ASSQ$^{\text{PU}}$	50	(0.4)	50	(11.2)	50	(102.9)	49	(528.7)
$n=10$	ASSRC	50	(0.0)	50	(1.0)	47	(224.3)	8	(912.3)
	ASSQ	50	(1.4)	50	(192.9)	36	(990.9)	1	(1440.0)
	ASSQ$^{\text{PU}}$	50	(0.8)	50	(43.4)	48	(529.9)	12	(1339.3)

instances as well as their average runtime is presented. We refer to Appendix B.5 on page 186 for similar tables for 2 and 8 scenarios. We would like to point out again that the average runtime is not too revealing if only a few instances have been solved. For $S = 2$ all instances for each model and configuration are solved and the expected strength of CPLEX is clearly visible: the runtime barely increases for increasing n. For the quantified models the runtime increases significantly faster, whereat ASSQ$^{\text{PU}}$ models are solved considerably faster than ASSQ models. For increasing S and N (cf. Appendix B.5), the ASSQ$^{\text{PU}}$ model tends to yield the best results but CPLEX (on ASSRC) remains highly competitive and is able to catch up with increasing n. In Figure 7.6 the performance profile on all 4200 instances is given. It can be seen that

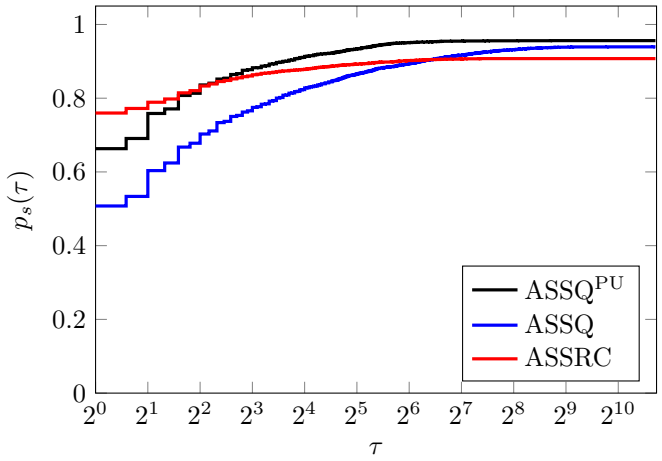

Figure 7.6.: Performance profile for all examined assignment instances and each model.

CPLEX is significantly faster on most ASSRC instances and is fastest on more than 75% of the instances. However, overall more of the ASSQ$^{\text{PU}}$ and ASSQ instances are solved. In comparison with Figure 7.5, the benefit of using universal constraints instead of the standard QIP seems to be even greater for this assignment problem than for the selection problem on the tested instances, as the blue and black curves approach each other much more slowly.

Impact of SCP Just like for the multistage selection problem, the use of SCP can only affect the solution process of ASSQ$^{\text{PU}}$ because for ASSQ a change in the universal variable assignment requires the change of subsequent existential variables. Hence, we only examined ASSQ$^{\text{PU}}$ by creating 500 instances with $n = 8$, $S = 4$ and $N = 8$ and solved them with and without the use of SCP with a time limit of 1800 seconds. With the use of SCP 436 instances were solved. 428 instance were solved without SCP. The average runtime for those instances solved with and without the use of SCP was 785.45 and 820.29 seconds, respectively. These results are very similar to the ones obtained for multistage selection instances and show a positive impact of SCP on the solution process.

8. Conclusion and Outlook

8.1. Conclusion

In this thesis we studied the concept of quantified integer programming which is a framework for multistage optimization under uncertainty. Our main contributions are the theoretical substantiation of two extensions QIP^{PU} and QIP^{ID}, and the development and implementation of solution techniques for standard QIP as well as the extensions.

We adapted well-known solution techniques to QIP and developed strategic copy-pruning (SCP), which makes it possible to omit certain subtrees during the search process by implicitly verifying the existence of a strategy in linear time. Additionally, we introduced novel relaxations that incorporate enhanced information regarding potential realizations of uncertain variables.

In our first extension universally quantified variables must obey a second constraint system resulting in a QIP with polyhedral uncertainty set QIP^{PU}. We presented a polynomial-time reduction function and several examples illustrated how the pure modeling benefits from having access to such explicit restrictions.

Our second novel extension QIP^{ID} allows multistage optimization under decision-dependent uncertainty for which we derived a well-defined problem statement. Based on this we examined the effects regarding the game tree interpretation and introduced the concept of a truncated strategy. Then we established the `PSPACE`-completeness of the extension and developed a polynomial-time reduction function. We further introduced several relaxation techniques, adapted game tree search methods and also showed that pruning mechanism, and in particular SCP, are also applicable for QIP^{ID}.

Due to a tremendous computational overhead when solving general QIP^{ID} we proposed a restricted version that allows a straightforward integration into our solver. We pointed out advantages of the made assumptions for the solution process and illustrated examples where they naturally apply.

Finally, we described how the general solver was adapted in order to be able to cope with the presented extensions. In a detailed computational study we investigated the advantages and disadvantages and compared the performance of models utilizing the explicit polyhedral uncertainty set with the equivalent standard QIP and the robust counterpart, which we solved using CPLEX. We demonstrated that our solver shows its strengths when there are multiple stages and scenarios where the sheer size of the robust counterpart is no longer manageable for CPLEX. In particular, the required RAM during the solution process of CPLEX increased exponentially while our solver's RAM usage only slightly increased for growing instances. However, for an increased size of the underlying deterministic problem (e.g. more items to select from)

the results showed that the IP abilities of CPLEX are currently more powerful. The comparison of the solution processes of QIP^{PU} and QIP were very promising: The more compact formulation of the polyhedral uncertainty set in the QIP^{PU} resulted in a (generally) decreased runtime compared to the corresponding QIP. This was not necessarily expected, as the handling of a second constraint system called for several cut backs in the implemented deduction techniques. Furthermore, experiments showed that utilizing our developed pruning techniques resulted in a massive improvement in both the number of solved instances and the runtime on several test sets, while no significant negative effect regarding the computational overhead was observed.

With the development and theoretical substantiation of the extensions QIP^{PU} and QIP^{ID}, the implementation of corresponding solution techniques in our open-source solver, and the algorithmic advances in the solution of QIP, this thesis paves the way for practical use of quantified programs.

8.2. Outlook

The intuitive and compact problem description of the general QIP^{ID} as well as the promising computational results indicate that quantified programs can be successfully applied for a variety of optimization problems. In particular, problems in a multistage setting can easily be modeled as QIP, QIP^{PU} or QIP^{ID}, for example combinatorial optimization problems and problems from the OR research area, where the multistage nature of underlying real-world decision-making processes is apparent. It will be interesting to see further applications of quantified programming to multistage optimization problems under decision-dependent uncertainty. In particular, we see great potential in the areas of explorable uncertainty and scheduling.

One major task when modeling problems under uncertainty is to find a description of possible scenarios that one wishes to protect against. A balance must be struck between the benefit of the achieved robustness and an increasing conservatism. Data-driven robust optimization techniques can be adapted in order to construct relevant uncertainty sets for quantified programs, which can be modeled using the universal constraint system introduced in this thesis. In times of big data a multitude of real-world data sets is available that can be utilized in research oriented multistage models in order to form the foundation for real-world applications.

The promising computational results of our solver on problems with several decision stages demonstrate the power of compact modeling via quantified programs and our developed solution methods. However, more development is needed to increase our solver's IP abilities in order to be able to solve even larger instances. Furthermore, research regarding the structural requirements of the introduced "simply restricted QIP^{ID}" is necessary. Results in these areas will further open up possibilities towards modeling and solving real-world problems under decision-dependent uncertainty.

Additional approaches to speed up the solution process through QIP-specific solution techniques and heuristics are now realizable such as the use of cutting planes in the universal constraint system that implicitly prune irrelevant universal variable assignments. Furthermore, we plan to apply and adapt techniques from other research areas to improve our handling of quan-

tified programs, e.g. learning techniques can be implemented to gain a better understanding of worst-case scenarios in order to speed up the detection of the principal variation. In addition, there are certainly a variety of methods for QBF, QCSP, MIP and RO that can be used profitably in our game tree search with only slight modifications.

The exploration and testing of further algorithmic and heuristic methods specifically dealing with QIP^{PU} and QIP^{ID} is necessary. Both problem types already benefited from crafty techniques implemented for QIP but we showed that not all of them are applicable. In particular, we see potential for improvement with regard to implication and conflict learning techniques, as we have adapted the existing implementation rather conservatively. Further extensive computational studies of problems under decision-dependent uncertainty, e.g. the mentioned selection problem with explorable uncertainty, will need to be done.

Another interesting aspect is the use of relaxations. In this thesis we theoretically substantiated several relaxations that can be applied during a tree search. However, currently our solver uses a more sophisticated relaxation only in the very first variable block and utilizes an only slightly refined standard LP-relaxation in subsequent blocks. Practical implementations have to be engineered, that allow the use of sophisticated relaxations, e.g. the S-relaxation, without a significant computational overhead for adapting or rebuilding the relaxation in order to keep it applicable. In combination with an improved understanding of important scenarios, our current approach has great potential to be improved even further.

The development of domain-specific heuristics or algorithms for multistage problems can be assisted by our open-source solver. On the one hand our solver can be adapted in order to exploit problem-specific knowledge within the game tree search. On the other hand, we envision the use of artificial neural networks that learn decision rules for multistage problems, where optimal solutions of quantified programs can be used in the learning process. The multistage combinatorial problems examined in this thesis could be used as a basis for this approach.

A. Examples and Algorithms

A.1. Example of Reduction $\text{QIP}^{\text{ID}} \leq_p \text{QIP}$

In order to compute the corresponding QIP of the QIP^{ID} as given in Example 5.3.19 we need to compute the bounds for the values L, M, $\tilde{M}$ and R^{LCD}. It is $L_1 \leq -2$, $M_1 \geq 2$, $M_2 \geq 3$, $\tilde{M} > 5$ and $R_1^{LCD} \leq 1$. Hence, a possible equivalent QIP corresponding to the QIP^{ID} is given as follows:

$$
\begin{aligned}
z = \quad & \min_{x_1 \in \{1,2,3\}} \left(-x_1 + \max_{x_2, x_3 \in \{0,1\}} \left(-x_2 - 2x_3 - \min_{p \in \{0,1\}} 10p \right) \right) \\
\text{s.t.} \quad & \exists x_1 \in \{1,2,3\}\ \exists v_2^{(1)} \in \{0,1\}\ \exists v_3^{(1)} \in \{0,1\} \\
& \forall\, x_2 \in \{0,1\}\ \forall\, x_3 \in \{0,1\}\ \exists\, y_1^{(2)} \in \{0,1\}\ \exists\, t_2 \in \{0,1\}\ \exists\, p \in \{0,1\} : \\
& \begin{array}{llllll}
x_1 & +v_2^{(1)} & +v_3^{(1)} & & & \leq 3 \\
2x_1 & -3v_2^{(1)} & & & & \leq 3 \\
x_1 & +x_2 & +x_3 & -2p & & \leq 3 \\
2x_1 & -3x_2 & & -3p & & \leq 3 \\
-x_1 & +2x_2 & +x_3 & +3y_1^{(2)} & & \leq 2 \\
p & -t_2 & & & & \leq 0 \\
t_2 & -y_1^{(2)} & & & & \leq 0
\end{array}
\end{aligned}
$$

A.2. Heuristic Strategies for Example 4.4.2

The three heuristic strategies and the optimal strategy with their worst-case outcome explored in Example 4.4.2 are presented. The investigated problem is a multistage selection problem with 6 items of which 3 have to be selected. There are 2 additional selection phases (which we called iterations) with each 2 scenarios. For better comprehensibility the costs in the initial stage and each scenario as given in Table 4.1 on page 70 are again displayed in the following Table A.1.

Table A.1.: Cost scenarios for an instance of the multistage selection problem.

i	1	2	3	4	5	6
c_i^0	84	14	76	61	31	45
$c_{i,1}^1$	40	24	29	41	90	71
$c_{i,2}^1$	45	30	15	18	44	44
$c_{i,1}^2$	13	25	12	11	75	50
$c_{i,2}^2$	80	10	29	32	64	30

Tables A.2-A.5 describe the constructed strategies. Values in parentheses are the item's cost, when bough in the proposed iteration and scenario. Gray cells indicate that this cell represents the second scenario of this iteration. The tables should be read from left to right, and can be interpreted as following the paths in the scenario tree. Cells with a hyphen indicate that nothing should be done. The costs in the worst-case scenario are printed in bold.

Table A.2.: Selection strategy according to "Buy All Now".

iteration 0	iteration 1	iteration 2	costs
select item 2(14) select item 5(31) select item 6(45)	-	-	**90**
		-	**90**
	-	-	**90**
		-	**90**

When buying the cheapest items in the initial stage, the resulting costs obviously do not depend on the scenarios that occur in upcoming iterations.

Table A.3.: Selection strategy according to "Buy Now, If Never Cheaper in Worst Case".

iteration 0	iteration 1	iteration 2	costs
select item 2(14) select item 5(31)	-	select item 4(11)	56
		select item 3(29)	**74**
	select item 3(15)	-	60
		-	60

The optimal solution as shown in Table A.5, is to select no items in the initial stage and wait for the scenario in iteration 1. This was surprising, as buying item 2 in the initial stage seemed

Table A.4.: Selection strategy according to “Buy Now, If Few are Cheaper”.

iteration 0	iteration 1	iteration 2	costs
select item 2(14)	select item 3(29)	select item 4(11)	54
		select item 6(30)	**73**
	select item 3(15) select item 4(18)	-	47
		-	47

Table A.5.: Optimal selection strategy for Example 4.4.2.

iteration 0	iteration 1	iteration 2	costs
-	-	4(11), 3(12) and 1(13)	36
		2(10), 3(29) and 6(30)	**69**
	select item 3(15) select item 4(18)	select item 1(13)	46
		select item 2(10)	43

inevitable due to the low cost of 14. In this example the costs after each realization of the scenarios in each iteration are smaller when applying the optimal winning strategy compared to the other heuristic strategies. Note that this does not have to be the case, as the sole aim of the optimization is minimize the worst-case costs.

A.3. Tic-Tac-Toe as QIP$^{\mathsf{ID}}$

Table A.6.: Variables of the QIP$^{\mathrm{ID}}$ model of tic-tac-toe.

name	domain	blocks / quantifier	description
A^k	$\{0,1\}^{3\times 3}$	$\{1,3,5,7,9\}$ / $\exists$	placement indicator of Xs after turn k
B^k	$\{0,1\}^{3\times 3}$	$\{2,4,6,8\}$ / $\forall$	placement indicator of Os after turn k
h^k	$\{0,1\}^3$	$\{1,3,5,7,9\}$ / $\exists$	indicator for horizontal rows of Xs after turn k
v^k	$\{0,1\}^3$	$\{1,3,5,7,9\}$ / $\exists$	indicator for vertical rows of Xs after turn k
d^k	$\{0,1\}^2$	$\{1,3,5,7,9\}$ / $\exists$	indicator for diagonal rows of Xs after turn k
w_k	$\{0,1\}$	$\{1,3,5,7,9\}$ / $\exists$	winning indicator for the starting player after turn k

Objective: $\max w_9$

Variable order: $\exists A^1\ h^1\ v^1 d^1 w_1\ \forall B^2\ \cdots\ \exists A^7\ h^7\ v^7 d^7 w_7\ \forall B^8\ \exists A^9\ h^9\ v^9 d^9 w_9$

Existential constraint system $A^{\exists} x \leq b^{\exists}$:

$$
\begin{aligned}
B^{k-1}_{i,j} + A^k_{i,j} &\leq 1 && \forall\,(i,j) \in P,\ k \in \{3,5,7,9\}\\
A^{k-2}_{i,j} - A^k_{i,j} &\leq 0 && \forall\,(i,j) \in P,\ k \in \{3,5,7,9\}\\
\sum_{(i,j)\in P} A^k_{i,j} &= \left\lfloor \frac{k}{2} \right\rfloor + 1 && \forall\, k \in \{1,3,5,7,9\}\\
\sum_{(i,j)\in P} A^k_{i,j} &\geq 3h^k_i && \forall\, i \in \{1,2,3\},\ k \in \{1,3,5,7,9\}\\
\sum_{(i,j)\in P} A^k_{i,j} &\geq 3v^k_i && \forall\, j \in \{1,2,3\},\ k \in \{1,3,5,7,9\}\\
\sum_{i=1}^{3} A^k_{i,i} &\geq 3d^k_1 && \forall\, k \in \{1,3,5,7,9\}\\
\sum_{i=1}^{3} A^k_{i,4-i} &\geq 3d^k_2 && \forall\, k \in \{1,3,5,7,9\}\\
\sum_{i=1}^{3} h^k_i + \sum_{j=1}^{3} v^k_j + d^k_1 + d^k_2 &\geq w_k && \forall\, k \in \{1,3,5,7,9\}\\
\sum_{(i,j)\in P} B^k_{i,j} &\leq 2 + w_{k-1} && \forall\, i \in \{1,2,3\},\ k \in \{2,4,6,8\}\\
\sum_{(i,j)\in P} B^k_{i,j} &\leq 2 + w_{k-1} && \forall\, j \in \{1,2,3\},\ k \in \{2,4,6,8\}\\
\sum_{i=1}^{3} B^k_{i,i} &\leq 2 + w_{k-1} && \forall\, k \in \{2,4,6,8\}\\
\sum_{i=1}^{3} B^k_{i,4-i} &\leq 2 + w_{k-1} && \forall\, k \in \{2,4,6,8\}
\end{aligned}
$$

Universal constraint system $A^{\forall}x \leq b^{\forall}$:

$$\begin{aligned} A^{k-1}_{i,j} + B^{k}_{i,j} &\leq 1 && \forall\,(i,j) \in P,\ k \in \{2,4,6,8\} \\ B^{k-2}_{i,j} - B^{k}_{i,j} &\leq 0 && \forall\,(i,j) \in P,\ k \in \{4,6,8\} \\ \sum_{(i,j)\in P} B^{k}_{i,j} &= \frac{k}{2} && \forall\, k \in \{2,4,6,8\} \end{aligned}$$

A.4. Alpha-Beta Algorithm for QIP$^{\mathsf{ID}}$

Algorithm 11: Initial call of the alpha-beta algorithm for $\mathrm{QIP}^{\mathrm{ID}}$.

Input: $\mathrm{QIP}^{\mathrm{ID}}$ $(A^{\exists}, A^{\forall}, b^{\exists}, b^{\forall}, c, \mathcal{L}, Q)$, game tree $G = (V, E, e)$

$\alpha = -\infty$
$\beta = +\infty$
for all $v' \in \mathcal{L}(r)$ **do**
  $value =$AlphaBeta(v', α, β)
  if $value \neq \pm\infty$ **then**
    $\beta = \min\{\beta, value\}$
  end if
end for
return β

Algorithm 12: Basic alpha-beta call for $\mathrm{QIP}^{\mathrm{ID}}$: AlphaBeta($v$,$\alpha$,$\beta$).

Input: node v, value α, value β

if $v \in V_L$ **then**
  return $w(v)$ // `weighting function` $w(v)$`, see Definition 5.3.11`
end if
$NodeValue = \pm\infty$ // `assume` $v \in V_{\pm\infty}$
if $v \in V_{\forall}$ **then** // v `is a MAX node`
  for all $v' \in \mathcal{L}(v)$ **do**
    if AlphaBeta$(v', \alpha, \beta) \neq \pm\infty$ **then**
      $\alpha = \max\{\alpha,$ AlphaBeta$(v', \alpha, \beta)\}$
      $NodeValue = \alpha$ // `in particular` $v \notin V_{\pm\infty}$
      if $\alpha \geq \beta$ **then**
        return β
      end if
    end if
  end for
  return $NodeValue$
end if
if $v \in V_{\exists}$ **then** // v `is a MIN node`
  for all $v' \in \mathcal{L}(v)$ **do**
    if AlphaBeta$(v', \alpha, \beta) \neq \pm\infty$ **then**
      $\beta = \min\{\beta,$ AlphaBeta$(v', \alpha, \beta)\}$
      $NodeValue = \beta$ // `in particular` $v \notin V_{\pm\infty}$
      if $\beta \leq \alpha$ **then**
        return α
      end if
    end if
  end for
  return $NodeValue$
end if

B. Supplemental Data

B.1. Additional Data on Multistage Selection Experiments

Table B.1.: Average of the maximum RAM required during the solution process of the three models with $N = 4$. Average of 50 instances per data point.

	model	$S = 1$	$S = 2$	$S = 3$	$S = 4$	$S = 5$	$S = 6$	$S = 7$	$S = 8$
$n = 10$	SELRC	4.37	3.82	2.33	16.99	65.19	272.39	1111.5	4406.75
	SELQ	619.76	621.78	624.59	628.19	630.27	629.87	632.44	637.24
	SELQ$^{\text{PU}}$	619.84	621.85	623.61	629.85	718.47	719.76	720.53	721.99
$n = 20$	SELRC	3.02	3.81	6.34	34.63	128.7	518.2	2181.64	5722.53
	SELQ	620.61	623.68	627.62	633.57	632.83	634.99	637.21	643.46
	SELQ$^{\text{PU}}$	620.57	623.51	626.55	633.47	722.77	724.0	727.36	729.31
$n = 30$	SELRC	4.65	3.45	13.19	48.43	190.32	770.31	3080.61	8268.61
	SELQ	621.43	624.92	629.56	635.3	636.88	639.44	642.93	654.28
	SELQ$^{\text{PU}}$	621.34	625.3	628.65	635.3	726.4	729.8	733.83	739.23
$n = 40$	SELRC	3.69	3.09	19.31	62.12	245.99	1040.35	4032.11	10838.5
	SELQ	622.2	626.36	631.65	639.44	641.87	647.24	652.92	661.08
	SELQ$^{\text{PU}}$	621.81	626.36	630.29	640.44	732.58	737.58	743.9	753.36
$n = 50$	SELRC	4.66	4.05	22.1	77.03	307.31	1271.37	4893.63	13402.8
	SELQ	622.57	628.27	634.18	642.17	648.45	654.5	659.28	671.43
	SELQ$^{\text{PU}}$	622.54	627.73	634.13	646.89	739.74	747.18	757.86	762.53

Table B.2.: Average runtime (in seconds) of multistage selection instances with $N = 4$ for which both quantified model were solved. An asterisk indicates less than 20 solved instances.

	model	$S = 1$	$S = 2$	$S = 3$	$S = 4$	$S = 5$	$S = 6$	$S = 7$	$S = 8$
$n = 10$	SELQ	0.1	0.1	0.2	0.3	0.7	2.6	7.5	35.4
	SELQ$^{\text{PU}}$	0.1	0.1	0.1	0.2	0.6	1.2	4.2	10.8
$n = 20$	SELQ	0.1	0.2	0.7	1.4	5.5	12.1	43.1	149.7
	SELQ$^{\text{PU}}$	0.1	0.2	0.5	0.9	2.6	7.5	28.8	86.0
$n = 30$	SELQ	0.2	0.5	2.7	7.9	15.5	43.7	159.7	448.9
	SELQ$^{\text{PU}}$	0.2	0.4	1.7	8.1	14.3	41.6	97.9	368.1
$n = 40$	SELQ	1.3	1.8	5.6	28.3	35.1	136.5	332.2	783.1
	SELQ$^{\text{PU}}$	1.3	1.9	3.7	14.3	46.9	113.5	304.0	643.4
$n = 50$	SELQ	1.4	3.0	13.9	55.8	112.6	377.5	643.5	1021.0*
	SELQ$^{\text{PU}}$	1.4	2.6	8.0	35.1	94.6	281.7	601.8	831.7*

B.2. Data Table of Multistage Selection Instances

In Table B.3 we provide information on multistage selection instances with 10 items of which 5 must be selected. For each setting (number of iterations S, number of scenarios N) we provide the data on a *representative* instance for the models SELQ$^{\text{PU}}$, SELQ and SELRC (see Subsection 4.4.4). Note that for *any* instance with fixed number of items, N and S, all entries in the table are equal except for the file size: the file sizes vary only marginally due to the different random cost parameters. The ambiguity of n, which was used in this local context as the number of items and as the number of variables in the global context, is avoided by only using n as the number of variables. Furthermore, the overall number of constraints m, as well as the number of existential ($m_\exists$) and universal constraints ($m_\forall$) is shown. The number of variable blocks β is explicitly given, even though $\beta = 1 + 2S$ in all cases.

Table B.3.: Data table of multistage selection instances with 10 items.

items	N	S	model	n	$n_\forall$	$n_\exists$	m	$m_\forall$	$m_\exists$	β	file size
			SELRC	30	–	–	24	–	–	–	2 KB
10	2	1	SELQ	24	1	23	16	0	16	3	1 KB
			SELQ$^{\text{PU}}$	23	2	21	14	1	13	3	1 KB
			SELRC	50	–	–	48	–	–	–	3 KB
10	4	1	SELQ	27	2	25	20	0	20	3	2 KB
			SELQ$^{\text{PU}}$	25	4	21	16	1	15	3	2 KB
			SELRC	90	–	–	96	–	–	–	5 KB
10	8	1	SELQ	32	3	29	28	0	28	3	3 KB
			SELQ$^{\text{PU}}$	29	8	21	20	1	19	3	2 KB
			SELRC	170	–	–	192	–	–	–	10 KB
10	16	1	SELQ	41	4	37	44	0	44	3	4 KB
			SELQ$^{\text{PU}}$	37	16	21	28	1	27	3	3 KB
			SELRC	330	–	–	384	–	–	–	21 KB
10	32	1	SELQ	58	5	53	76	0	76	3	8 KB
			SELQ$^{\text{PU}}$	53	32	21	44	1	43	3	6 KB
			SELRC	650	–	–	768	–	–	–	43 KB
10	64	1	SELQ	91	6	85	140	0	140	3	15 KB
			SELQ$^{\text{PU}}$	85	64	21	76	1	75	3	11 KB
			SELRC	1290	–	–	1536	–	–	–	87 KB
10	128	1	SELQ	156	7	149	268	0	268	3	29 KB
			SELQ$^{\text{PU}}$	149	128	21	140	1	139	3	21 KB
			SELRC	2570	–	–	3072	–	–	–	178 KB
10	256	1	SELQ	285	8	277	524	0	524	3	60 KB
			SELQ$^{\text{PU}}$	277	256	21	268	1	267	3	42 KB
			SELRC	70	–	–	48	–	–	–	5 KB
10	2	2	SELQ	38	2	36	21	0	21	5	2 KB
			SELQ$^{\text{PU}}$	36	4	32	17	2	15	5	2 KB

Extension of Table B.3: Data table of multistage selection instances.

items	N	S	model	n	$n_\forall$	$n_\exists$	m	$m_\forall$	$m_\exists$	β	file size
			SELRC	210	–	–	192	–	–	–	17 KB
10	4	2	SELQ	44	4	40	29	0	29	5	3 KB
			SELQ$^{\text{PU}}$	40	8	32	21	2	19	5	3 KB
			SELRC	730	–	–	768	–	–	–	64 KB
10	8	2	SELQ	54	6	48	45	0	45	5	5 KB
			SELQ$^{\text{PU}}$	48	16	32	29	2	27	5	4 KB
			SELRC	2730	–	–	3072	–	–	–	262 KB
10	16	2	SELQ	72	8	64	77	0	77	5	8 KB
			SELQ$^{\text{PU}}$	64	32	32	45	2	43	5	6 KB
			SELRC	10570	–	–	12288	–	–	–	1 MB
10	32	2	SELQ	106	10	96	141	0	141	5	15 KB
			SELQ$^{\text{PU}}$	96	64	32	77	2	75	5	11 KB
			SELRC	41610	–	–	49152	–	–	–	4 MB
10	64	2	SELQ	172	12	160	269	0	269	5	29 KB
			SELQ$^{\text{PU}}$	160	128	32	141	2	139	5	21 KB
			SELRC	165130	–	–	196608	–	–	–	18 MB
10	128	2	SELQ	302	14	288	525	0	525	5	58 KB
			SELQ$^{\text{PU}}$	288	256	32	269	2	267	5	42 KB
			SELRC	657930	–	–	786432	–	–	–	76 MB
10	256	2	SELQ	560	16	544	1037	0	1037	5	120 KB
			SELQ$^{\text{PU}}$	544	512	32	525	2	523	5	83 KB
			SELRC	150	–	–	96	–	–	–	13 KB
10	2	3	SELQ	52	3	49	26	0	26	7	3 KB
			SELQ$^{\text{PU}}$	49	6	43	20	3	17	7	3 KB
			SELRC	850	–	–	768	–	–	–	92 KB
10	4	3	SELQ	61	6	55	38	0	38	7	4 KB
			SELQ$^{\text{PU}}$	55	12	43	26	3	23	7	4 KB
			SELRC	5850	–	–	6144	–	–	–	716 KB
10	8	3	SELQ	76	9	67	62	0	62	7	7 KB
			SELQ$^{\text{PU}}$	67	24	43	38	3	35	7	6 KB
			SELRC	43690	–	–	49152	–	–	–	6 MB
10	16	3	SELQ	103	12	91	110	0	110	7	12 KB
			SELQ$^{\text{PU}}$	91	48	43	62	3	59	7	9 KB
			SELRC	338250	–	–	393216	–	–	–	50 MB
10	32	3	SELQ	154	15	139	206	0	206	7	22 KB
			SELQ$^{\text{PU}}$	139	96	43	110	3	107	7	17 KB
			SELRC	$> 2 \cdot 10^6$	–	–	$> 3 \cdot 10^6$	–	–	–	409 MB
10	64	3	SELQ	253	18	235	398	0	398	7	43 KB
			SELQ$^{\text{PU}}$	235	192	43	206	3	203	7	32 KB

Extension of Table B.3: Data table of multistage selection instances.

items	N	S	model	n	$n_\forall$	$n_\exists$	m	$m_\forall$	$m_\exists$	β	file size
			SELRC	$> 21 \cdot 10^6$	–	–	$> 25 \cdot 10^6$	–	–	–	3 GB
10	128	3	SELQ	448	21	427	782	0	782	7	87 KB
			SELQ$^{\text{PU}}$	427	384	43	398	3	395	7	62 KB
			SELRC	$> 168 \cdot 10^6$	–	–	$> 201 \cdot 10^6$	–	–	–	29 GB
10	256	3	SELQ	835	24	811	1550	0	1550	7	180 KB
			SELQ$^{\text{PU}}$	811	768	43	782	3	779	7	124 KB
			SELRC	310	–	–	192	–	–	–	33 KB
10	2	4	SELQ	66	4	62	31	0	31	9	4 KB
			SELQ$^{\text{PU}}$	62	8	54	23	4	19	9	4 KB
			SELRC	3410	–	–	3072	–	–	–	489 KB
10	4	4	SELQ	78	8	70	47	0	47	9	6 KB
			SELQ$^{\text{PU}}$	70	16	54	31	4	27	9	5 KB
			SELRC	46810	–	–	49152	–	–	–	8 MB
10	8	4	SELQ	98	12	86	79	0	79	9	9 KB
			SELQ$^{\text{PU}}$	86	32	54	47	4	43	9	7 KB
			SELRC	699050	–	–	786432	–	–	–	130 MB
10	16	4	SELQ	134	16	118	143	0	143	9	15 KB
			SELQ$^{\text{PU}}$	118	64	54	79	4	75	9	12 KB
			SELRC	$> 10 \cdot 10^6$	–	–	$> 12 \cdot 10^6$	–	–	–	2 GB
10	32	4	SELQ	202	20	182	271	0	271	9	29 KB
			SELQ$^{\text{PU}}$	182	128	54	143	4	139	9	22 KB
			SELRC	$> 170 \cdot 10^6$	–	–	$> 201 \cdot 10^6$	–	–	–	36 GB
10	64	4	SELQ	334	24	310	527	0	527	9	57 KB
			SELQ$^{\text{PU}}$	310	256	54	271	4	267	9	42 KB
			SELRC	630	–	–	384	–	–	–	84 KB
10	2	5	SELQ	80	5	75	36	0	36	11	5 KB
			SELQ$^{\text{PU}}$	75	10	65	26	5	21	11	5 KB
			SELRC	13650	–	–	12288	–	–	–	2 MB
10	4	5	SELQ	95	10	85	56	0	56	11	7 KB
			SELQ$^{\text{PU}}$	85	20	65	36	5	31	11	6 KB
			SELRC	374490	–	–	393216	–	–	–	78 MB
10	8	5	SELQ	120	15	105	96	0	96	11	11 KB
			SELQ$^{\text{PU}}$	105	40	65	56	5	51	11	9 KB
			SELRC	$> 11 \cdot 10^6$	–	–	$> 12 \cdot 10^6$	–	–	–	3 GB
10	16	5	SELQ	165	20	145	176	0	176	11	19 KB
			SELQ$^{\text{PU}}$	145	80	65	96	5	91	11	15 KB
			SELRC	$> 346 \cdot 10^6$	–	–	$> 402 \cdot 10^6$	–	–	–	91 GB
10	32	5	SELQ	250	25	225	336	0	336	11	36 KB
			SELQ$^{\text{PU}}$	225	160	65	176	5	171	11	28 KB

Extension of Table B.3: Data table of multistage selection instances.

items	N	S	model	n	$n_\forall$	$n_\exists$	m	$m_\forall$	$m_\exists$	β	file size
			SELRC	1270	–	–	768	–	–	–	207 KB
10	2	6	SELQ	94	6	88	41	0	41	13	6 KB
			SELQ^{PU}	88	12	76	29	6	23	13	5 KB
			SELRC	54610	–	–	49152	–	–	–	12 MB
10	4	6	SELQ	112	12	100	65	0	65	13	8 KB
			SELQ^{PU}	100	24	76	41	6	35	13	7 KB
			SELRC	$> 2 \cdot 10^6$	–	–	$> 3 \cdot 10^6$	–	–	–	775 MB
10	8	6	SELQ	142	18	124	113	0	113	13	13 KB
			SELQ^{PU}	124	48	76	65	6	59	13	11 KB
			SELRC	$> 178 \cdot 10^6$	–	–	$> 201 \cdot 10^6$	–	–	–	54 GB
10	16	6	SELQ	196	24	172	209	0	209	13	23 KB
			SELQ^{PU}	172	96	76	113	6	107	13	18 KB
			SELRC	2550	–	–	1536	–	–	–	499 KB
10	2	7	SELQ	108	7	101	46	0	46	15	7 KB
			SELQ^{PU}	101	14	87	32	7	25	15	6 KB
			SELRC	218450	–	–	196608	–	–	–	60 MB
10	4	7	SELQ	129	14	115	74	0	74	15	9 KB
			SELQ^{PU}	115	28	87	46	7	39	15	8 KB
			SELRC	$> 23 \cdot 10^6$	–	–	$> 25 \cdot 10^6$	–	–	–	8 GB
10	8	7	SELQ	164	21	143	130	0	130	15	15 KB
			SELQ^{PU}	143	56	87	74	7	67	15	12 KB
			SELRC	5110	–	–	3072	–	–	–	1 MB
10	2	8	SELQ	122	8	114	51	0	51	17	7 KB
			SELQ^{PU}	114	16	98	35	8	27	17	7 KB
			SELRC	873810	–	–	786432	–	–	–	286 MB
10	4	8	SELQ	146	16	130	83	0	83	17	11 KB
			SELQ^{PU}	130	32	98	51	8	43	17	9 KB
			SELRC	$> 191 \cdot 10^6$	–	–	$> 201 \cdot 10^6$	–	–	–	72 GB
10	8	8	SELQ	186	24	162	147	0	147	17	17 KB
			SELQ^{PU}	162	64	98	83	8	75	17	14 KB

B.3. Performance Profiles for Multistage Selection Experiments

We provide performance profiles (see Remark 7.3.1) for experiments on multistage selection instances with $N = 4$ scenarios (see Subsection 7.3.2). We first compare the performance on all instances with fixed number of items n and then for fixed number of iterations S. In Figure B.1 we see that for increasing n CPLEX is more often the fastest method (solving SELRC) but our solver is able to solve more instances using the quantified models. In Figure B.2 we see that for large S CPLEX cannot keep up with our solver on SELQ^{PU} and SELQ instances. For fewer iterations ($S \leq 5$) we cannot outperform CPLEX solving SELRC but always solve all instances within the time limit.

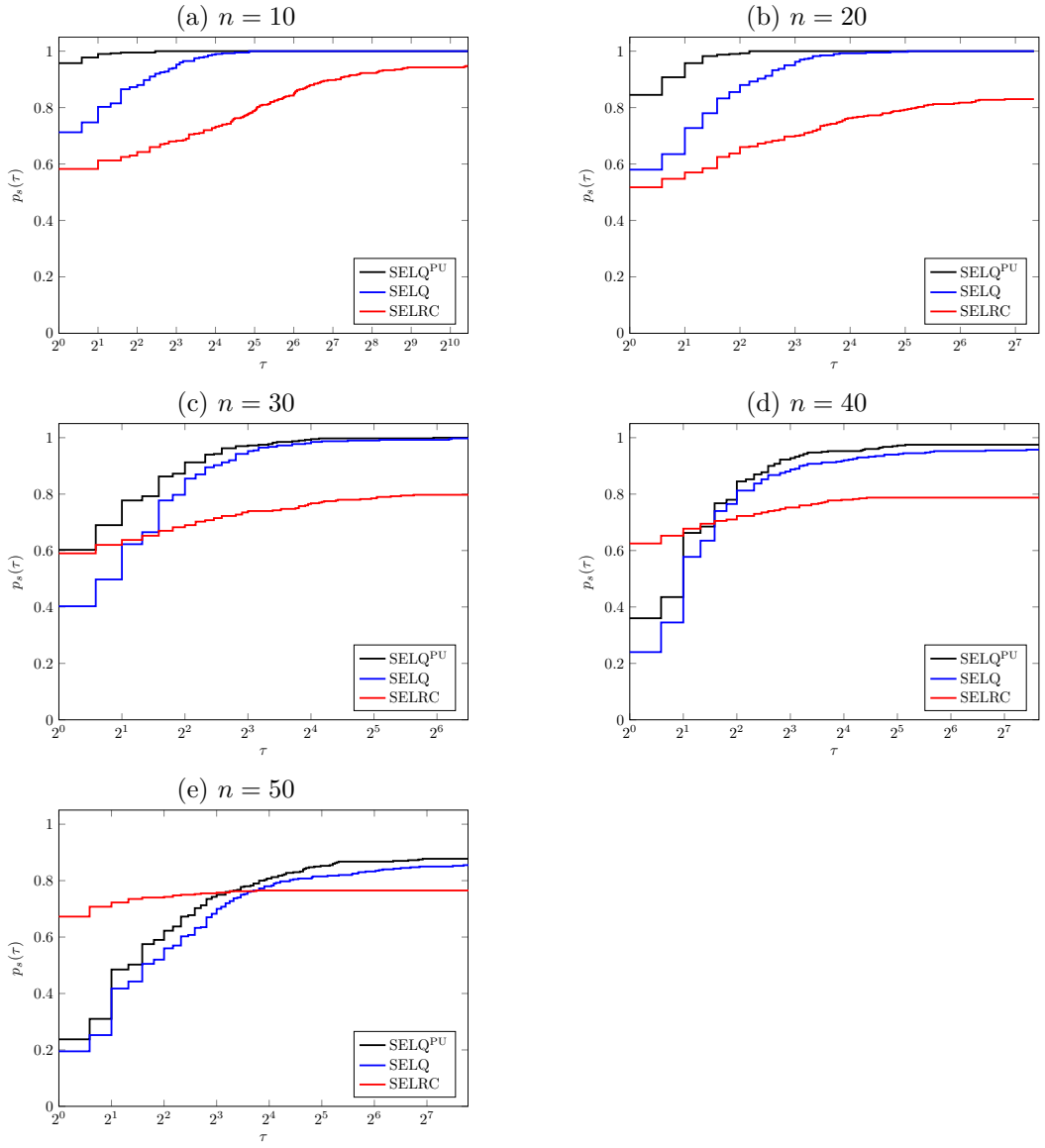

Figure B.1.: Performance profiles for multistage selection models for various n.

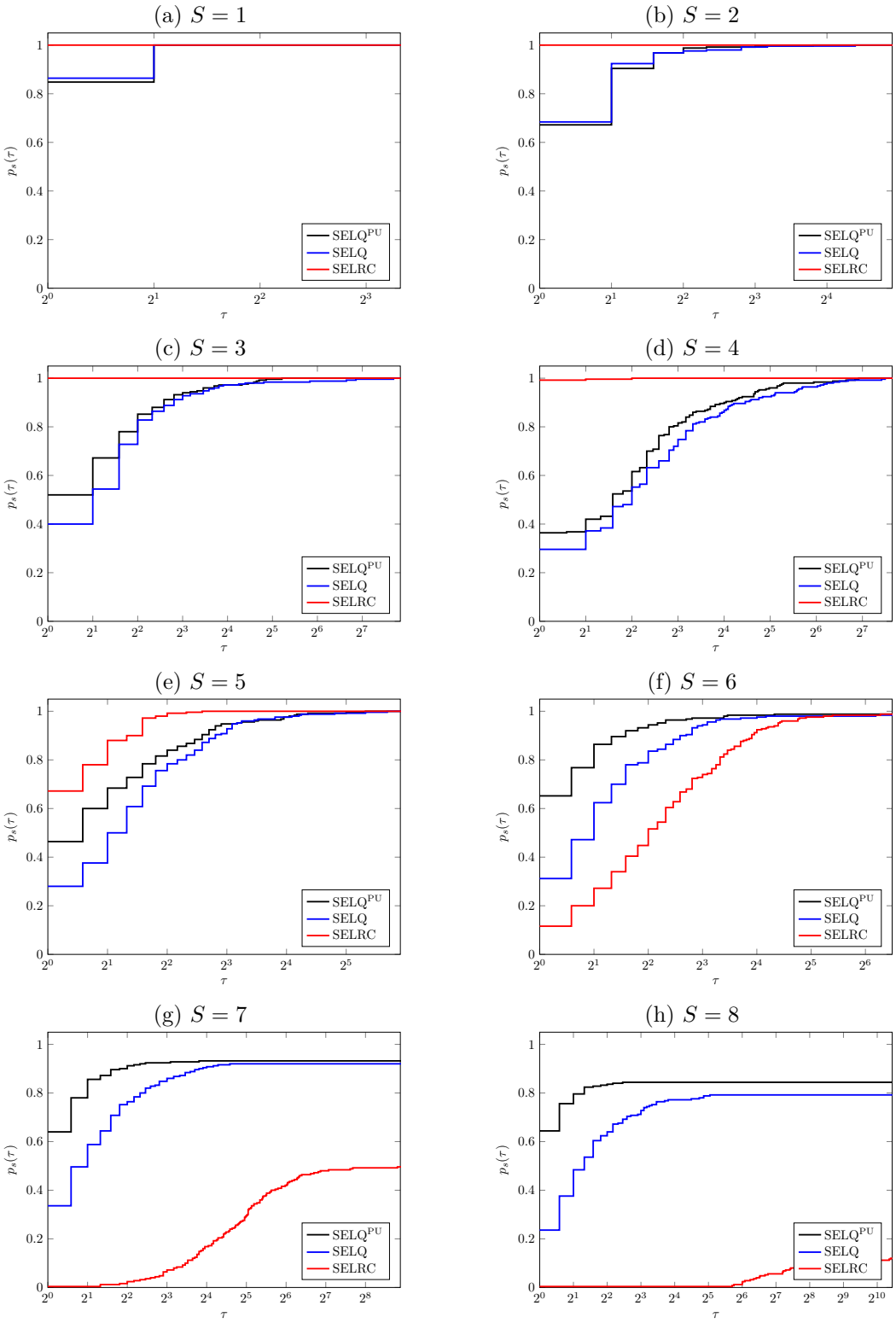

Figure B.2.: Performance profiles for multistage selection models for various S.

B.4. Optimal Robust Strategy vs. Heuristic Selection Strategy

For the heuristic selection strategies 2 and 3, presented in Subsection 4.4.4, box plots displaying the relative deviation of the worst-case outcome when applying those heuristics from the optimal value are shown for instances with various number of items and stages with a fixed number of $N = 4$ scenarios per stage. Strategy 1 is not considered as it would distort the graphs without gaining new insights, as strategy 1 is (obviously) significantly worse. Box plots (e.g. [MTL78]) are created using the macro `psboxplot` of the LaTeX package `pst-plot`[15]. The interquantile range factor, defining the area of outliers, is set to 1.5 by default. Each plot comprises of 50 instances, except for $n = 40$ with $S = 8$ as well as $n = 50$ with $S = 7$ and $S = 8$. The maximum range of each plot is set to 3.5 in order to allow a better comparability for the various n. Light gray box plots represent strategy 2 and dark gray box plots represent strategy 3.

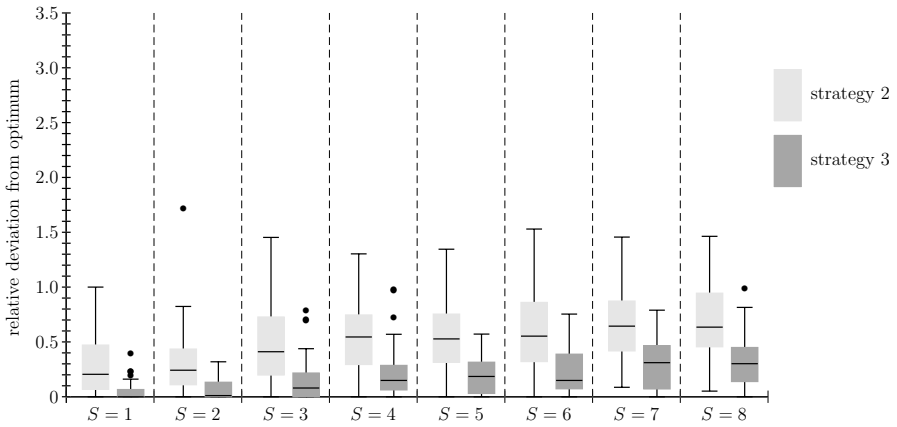

Figure B.3.: Relative deviation from the optimal value of the heuristic selection strategies 2 and 3 for $n = 10$ items.

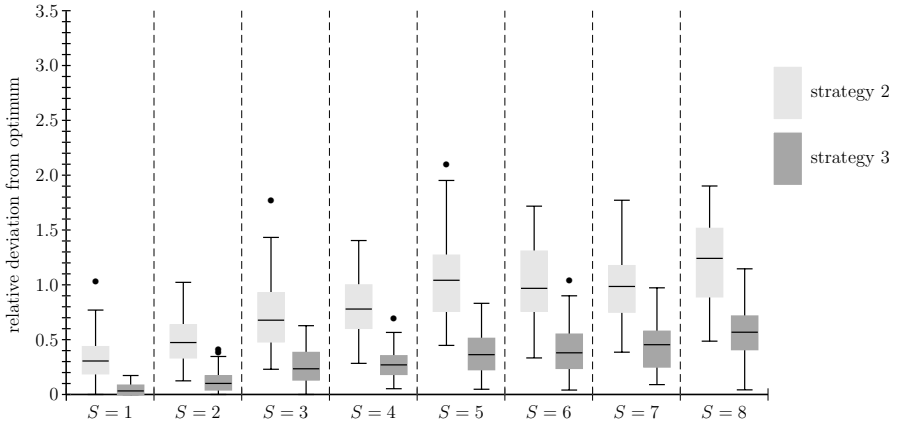

Figure B.4.: Relative deviation from the optimal value of the heuristic selection strategies 2 and 3 for $n = 20$ items.

[15] http://ctan.org/pkg/pst-plot (accessed May 3, 2020)

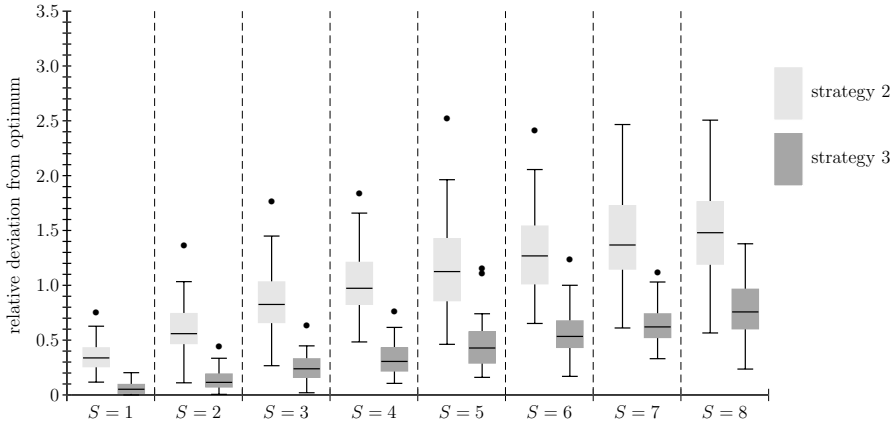

Figure B.5.: Relative deviation from the optimal value of the heuristic selection strategies 2 and 3 for $n = 30$ items.

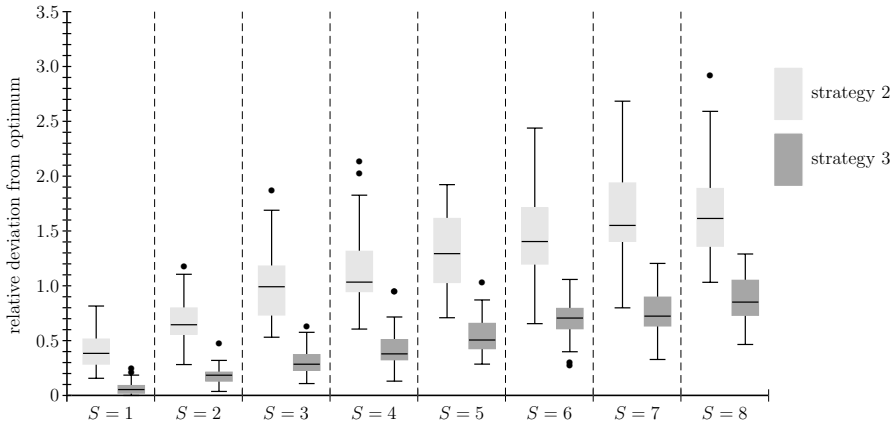

Figure B.6.: Relative deviation from the optimal value of theheuristic selection strategies 2 and 3 for $n = 10$ items. Only 43 instances for $S = 8$.

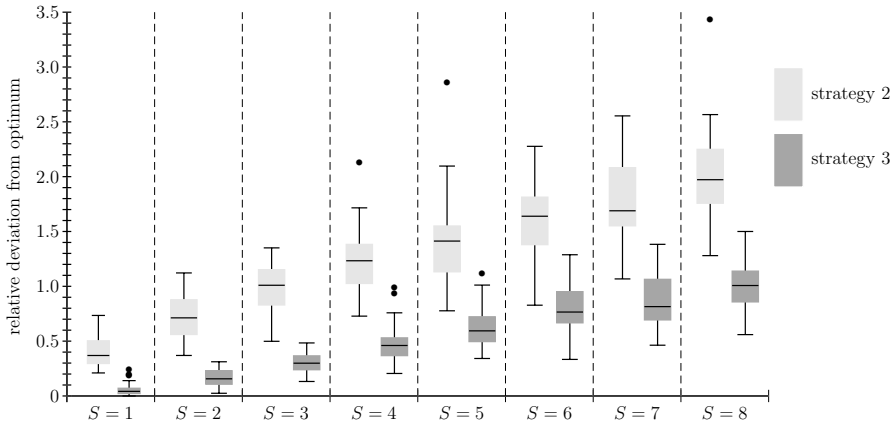

Figure B.7.: Relative deviation from the optimal value of the heuristic selection strategies 2 and 3 for $n = 10$ items. Only 38 instances for $S = 7$ and 24 instances for $S = 8$.

B.5. Additional Data on Multistage Assignment Experiments

Table B.4.: Number of solved multistage assignment instances with $N = 2$ and average runtime.

	model	$S=1$		$S=2$		$S=3$		$S=4$	
$n=4$	ASSRC	50	(0.0)	50	(0.0)	50	(0.0)	50	(0.0)
	ASSQ	50	(0.0)	50	(0.1)	50	(0.2)	50	(0.2)
	ASSQ$^{\mathrm{PU}}$	50	(0.1)	50	(0.1)	50	(0.2)	50	(0.2)
$n=5$	ASSRC	50	(0.0)	50	(0.0)	50	(0.0)	50	(0.0)
	ASSQ	50	(0.1)	50	(0.2)	50	(0.3)	50	(0.7)
	ASSQ$^{\mathrm{PU}}$	50	(0.1)	50	(0.1)	50	(0.2)	50	(0.4)
$n=6$	ASSRC	50	(0.0)	50	(0.0)	50	(0.0)	50	(0.0)
	ASSQ	50	(0.1)	50	(0.3)	50	(0.7)	50	(1.7)
	ASSQ$^{\mathrm{PU}}$	50	(0.1)	50	(0.2)	50	(0.5)	50	(0.8)
$n=7$	ASSRC	50	(0.0)	50	(0.0)	50	(0.0)	50	(0.1)
	ASSQ	50	(0.2)	50	(0.7)	50	(2.7)	50	(7.6)
	ASSQ$^{\mathrm{PU}}$	50	(0.1)	50	(0.4)	50	(0.9)	50	(2.1)
$n=8$	ASSRC	50	(0.0)	50	(0.0)	50	(0.0)	50	(0.5)
	ASSQ	50	(0.2)	50	(2.0)	50	(11.6)	50	(20.1)
	ASSQ$^{\mathrm{PU}}$	50	(0.2)	50	(0.6)	50	(2.6)	50	(5.1)
$n=9$	ASSRC	50	(0.0)	50	(0.0)	50	(0.0)	50	(1.1)
	ASSQ	50	(0.2)	50	(5.1)	50	(22.6)	50	(116.3)
	ASSQ$^{\mathrm{PU}}$	50	(0.2)	50	(1.7)	50	(7.7)	50	(21.4)
$n=10$	ASSRC	50	(0.0)	50	(0.0)	50	(0.0)	50	(2.1)
	ASSQ	50	(0.6)	50	(10.0)	50	(98.0)	50	(351.6)
	ASSQ$^{\mathrm{PU}}$	50	(0.3)	50	(4.6)	50	(22.9)	50	(84.3)

Table B.5.: Number of solved multistage assignment instances with $N = 8$ and average runtime.

	model	$S=1$		$S=2$		$S=3$		$S=4$	
$n=4$	ASSRC	50	(0.0)	50	(0.0)	50	(3.2)	50	(118.6)
	ASSQ	50	(0.1)	50	(0.2)	50	(0.8)	50	(4.0)
	ASSQ$^{\mathrm{PU}}$	50	(0.1)	50	(0.2)	50	(1.3)	50	(5.8)
$n=5$	ASSRC	50	(0.0)	50	(0.2)	50	(11.8)	43	(349.0)
	ASSQ	50	(0.1)	50	(0.6)	50	(2.5)	50	(13.9)
	ASSQ$^{\mathrm{PU}}$	50	(0.2)	50	(0.4)	50	(2.4)	50	(18.4)
$n=6$	ASSRC	50	(0.0)	50	(0.9)	50	(70.0)	25	(662.8)
	ASSQ	50	(0.3)	50	(2.5)	50	(13.6)	50	(63.6)
	ASSQ$^{\mathrm{PU}}$	50	(0.2)	50	(1.7)	50	(8.4)	50	(70.0)
$n=7$	ASSRC	50	(0.0)	50	(1.4)	49	(215.2)	7	(762.7)
	ASSQ	50	(0.4)	50	(11.3)	50	(58.8)	49	(310.7)
	ASSQ$^{\mathrm{PU}}$	50	(0.3)	50	(4.1)	50	(33.7)	50	(248.0)
$n=8$	ASSRC	50	(0.0)	50	(4.8)	41	(445.8)	2	(496.5)
	ASSQ	50	(1.2)	50	(47.4)	50	(251.1)	39	(854.5)
	ASSQ$^{\mathrm{PU}}$	50	(0.5)	50	(13.5)	50	(110.8)	44	(869.4)
$n=9$	ASSRC	50	(0.0)	50	(21.9)	24	(470.5)	3	(991.3)
	ASSQ	50	(2.9)	50	(228.8)	31	(1117.9)	7	(898.4)
	ASSQ$^{\mathrm{PU}}$	50	(1.5)	50	(45.8)	50	(598.5)	4	(1131.0)
$n=10$	ASSRC	50	(0.0)	50	(56.4)	15	(517.3)	1	(388.0)
	ASSQ	50	(8.1)	43	(633.9)	7	(675.9)	1	(173.0)
	ASSQ$^{\mathrm{PU}}$	50	(3.0)	50	(266.5)	8	(1056.0)	1	(1154.0)

Bibliography

[AB09] S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009. (cited on page 54)

[ABV05] H. Aissi, C. Bazgan, and D. Vanderpooten. Complexity of the min–max and min–max regret assignment problems. *Operations Research Letters*, 33(6):634–640, 2005. (cited on page 71)

[AE+17] U. Aldasoro, L. F. Escudero, M. Merino, and G. Perez. A parallel branch-and-fix coordination based matheuristic algorithm for solving large sized multistage stochastic mixed 0–1 problems. *European Journal of Operational Research*, 258(2):590–606, 2017. (cited on page 19)

[AGV18] S. Akshay, B. Genest, and N. Vyas. Distribution-based objectives for markov decision processes. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 36–45, 2018. (cited on page 15)

[AKM05] T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005. (cited on pages 14, 125, and 146)

[All94] V. Allis. *Searching for Solutions in Games and Artificial Intelligence*. PhD thesis, Maastricht, Maastricht, The Netherlands, 1994. (cited on pages 29 and 86)

[Alt88] I. Althöfer. On the complexity of searching game trees and other recursion trees. *Journal of Algorithms*, 9(4):538–567, 1988. (cited on page 29)

[AN77] S. Akl and M. Newborn. The principal continuation and the killer heuristic. In *Proceedings of the 1977 annual conference, ACM '77, Seattle, Washington, USA*, pages 466–473, 1977. (cited on pages 14, 34, 47, and 147)

[AP19a] S. Avraamidou and E. N. Pistikopoulos. Adjustable robust optimization through multiparametric programming. *Optimization Letters*, pages 1–15, 2019. (cited on page 20)

[AP19b] S. Avraamidou and E. N. Pistikopoulos. Multi-parametric global optimization approach for tri-level mixed-integer linear optimization problems. *Journal of Global Optimization*, 74(3):443–465, 2019. (cited on page 20)

[Apa17] R. M. Apap. *Models and Computational Strategies for Multistage Stochastic Programming under Endogenous and Exogenous Uncertainties*. PhD thesis, Carnegie Mellon University, 2017. (cited on page 21)

[Bal01] E. Balas. Projection and lifting in combinatorial optimization. In M. Jünger and D. Naddef, editors, *Computational Combinatorial Optimization: Optimal or Provably Near-Optimal Solutions*, pages 26–56. Springer Berlin Heidelberg, 2001. (cited on page 43)

[BB08] J. Beck and J. Beck. *Combinatorial games: tic-tac-toe theory*, volume 114. Cambridge University Press, 2008. (cited on pages 130 and 132)

[BB09] D. Bertsimas and D. B. Brown. Constructing uncertainty sets for robust linear optimization. *Operations Research*, 57(6):1483–1495, 2009. (cited on pages 49 and 50)

[BBC11] D. Bertsimas, D. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011. (cited on pages 1, 17, 18, and 29)

[BC09] M. Bodirsky and H. Chen. Relatively quantified constraint satisfaction. *Constraints*, 14(1):3–15, 2009. (cited on page 17)

[BC10] D. Bertsimas and C. Caramanis. Finite adaptability in multistage linear optimization. *IEEE Transactions on Automatic Control*, 55(12):2751–2766, 2010. (cited on page 18)

[BC+16] O. Beyersdorff, L. Chew, R. A. Schmidt, and M. Suda. Lifting qbf resolution calculi to dqbf. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 490–499. Springer International Publishing, Cham, 2016. (cited on page 16)

[BCJ14] V. Balabanov, H.-J. K. Chiang, and J.-H. R. Jiang. Henkin quantifiers and boolean formulae: A certification perspective of dqbf. *Theoretical Computer Science*, 523:86–100, 2014. (cited on page 16)

[BD16] D. Bertsimas and I. Dunning. Multistage robust mixed-integer optimization with adaptive partitions. *Operations Research*, 64(4):980–998, 2016. (cited on page 18)

[BDN19] H. Bakker, F. Dunke, and S. Nickel. A structuring review on multi-stage optimization under uncertainty: Aligning concepts from theory and practice. *Omega*, 2019. (cited on page 17)

[Bea55] E. M. Beale. On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society: Series B (Methodological)*, 17(2):173–184, 1955. (cited on pages 1 and 17)

[Bel57] R. E. Bellman. *Dynamic Programming*. Courier Dover Publications, 1957. (cited on pages 1 and 17)

[Ben62] J. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962. (cited on page 19)

[Ber01] D. P. Bertsekas. Dynamic programming and optimal control, two volume set, 2001. (cited on page 17)

[BF+05] D. Bustan, A. Flaisher, O. Grumberg, O. Kupferman, and M. Y. Vardi. Regular vacuity. In D. Borrione and W. Paul, editors, *Correct Hardware Design and Verification Methods*, pages 191–206. Springer Berlin Heidelberg, 2005. (cited on page 51)

[BG15] D. Bertsimas and A. Georghiou. Design of near optimal decision rules in multistage adaptive mixed-integer optimization. *Operations Research*, 63(3):610–627, 2015. (cited on page 18)

[BGS11] D. Bertsimas, V. Goyal, and X. A. Sun. A geometric characterization of the power of finite adaptability in multistage stochastic and adaptive optimization. *Mathematics of Operations Research*, 36(1):24–54, 2011. (cited on page 19)

[BH92] J. R. Birge and D. F. Holmes. Efficient solution of two-stage stochastic linear programs using interior point methods. *Computational Optimization and Applications*, 1(3):245–276, 1992. (cited on page 30)

[BH+05] R. Bruce, M. Hoffmann, D. Krizanc, and R. Raman. Efficient update strategies for geometric computing with uncertainty. *Theory of Computing Systems*, 38(4):411–423, 2005. (cited on page 21)

[Bie08] A. Biere. Adaptive restart strategies for conflict driven sat solvers. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 28–33. Springer Berlin Heidelberg, 2008. (cited on page 14)

[BIP10] D. Bertsimas, D. A. Iancu, and P. A. Parrilo. Optimality of affine policies in multistage robust optimization. *Mathematics of Operations Research*, 35(2):363–394, 2010. (cited on page 18)

[Bir85] J. R. Birge. Decomposition and partitioning methods for multistage stochastic linear programs. *Operations Research*, 33(5):989–1007, 1985. (cited on page 19)

[BJT16] B. Bogaerts, T. Janhunen, and S. Tasharrofi. Solving qbf instances with nested sat solvers. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016. (cited on page 16)

[BL11] J. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer US, New York, NY, 2nd edition, 2011. (cited on pages 1, 19, and 29)

[BL18] M. Bodur and J. R. Luedtke. Two-stage linear decision rules for multi-stage stochastic programming. *Mathematical Programming*, pages 1–34, 2018. (cited on page 19)

[BLV08] M. Benedetti, A. Lallouet, and J. Vautard. Quantified constraint optimization. In *International Conference on Principles and Practice of Constraint Programming*, pages 463–477. Springer Berlin Heidelberg, 2008. (cited on page 17)

[BP+12] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012. (cited on page 17)

[BPS04] D. Bertsimas, D. Pachamanova, and M. Sim. Robust linear optimization under general norms. *Operations Research Letters*, 32(6):510–516, 2004. (cited on page 49)

[BS04] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004. (cited on pages 18 and 49)

[BS07] H.-G. Beyer and B. Sendhoff. Robust optimization - a comprehensive survey. *Computer Methods in Applied Mechanics and Engineering*, 196(33-34):3190–3218, 2007. (cited on page 17)

[BS14] V. Barichard and I. Stéphan. The cut tool for qcsp. In *26th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'14)*, pages 883–890. IEEE, 2014. (cited on page 17)

[BSZ19] D. Bertsimas, M. Sim, and M. Zhang. Adaptive distributionally robust optimization. *Management Science*, 65(2):604–618, 2019. (cited on pages 18 and 19)

[BTBB10] A. Ben-Tal, D. Bertsimas, and D. B. Brown. A soft robust model for optimization under ambiguity. *Operations Research*, 58(4):1220–1234, 2010. (cited on page 18)

[BTG+04] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376, 2004. (cited on page 18)

[BTG+05] A. Ben-Tal, B. Golany, A. Nemirovski, and J.-P. Vial. Retailer-supplier flexible commitments contracts: A robust optimization approach. *Manufacturing & Service Operations Management*, 7(3):248–271, 2005. (cited on page 18)

[BTGN09] A. Ben-Tal, L. E. Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton University Press, 2009. (cited on pages 1, 17, and 18)

[BTGS09] 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. (cited on page 18)

[BTN98] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805, 1998. (cited on pages 10 and 79)

[BTN99] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–13, 1999. (cited on pages 29, 49, and 50)

[BTN00] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88(3):411–424, 2000. (cited on page 18)

[BTN02] A. Ben-Tal and A. Nemirovski. Robust optimization - methodology and applications. *Mathematical Programming*, 92(3):453–480, 2002. (cited on page 17)

[BV17] D. Bertsimas and P. Vayanos. Data-driven learning in dynamic pricing using adaptive optimization. *Optimization Online*, 2017. (cited on page 20)

[BW19] N. Bhargava and B. C. Williams. Complexity bounds for the controllability of temporal networks with conditions, disjunctions, and uncertainty. *Artificial Intelligence*, 271:1–17, 2019. (cited on page 15)

[CC+98] S. Ceria, C. Cordier, H. Marchand, and L. A. Wolsey. Cutting planes for integer programs with general integer variables. *Mathematical programming*, 81(2):201–214, 1998. (cited on page 14)

[CC18] B. Christian and S. Cremaschi. A branch and bound algorithm to solve large-scale multistage stochastic programs with endogenous uncertainty. *AIChE Journal*, 64(4):1262–1271, 2018. (cited on page 21)

[CF+93] A. Condon, J. Feigenbaum, C. Lund, and P. Shor. Probabilistically checkable debate systems and approximation algorithms for pspace-hard functions. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing*, STOC '93, pages 305–314, 1993. (cited on page 17)

[CF12] C. Carlsson and R. Fuller. *Fuzzy reasoning in decision making and optimization*, volume 82. Physica, 2012. (cited on page 17)

[CG16] A. Chassein and M. Goerigk. Performance analysis in robust optimization. In *Robustness Analysis in Decision Aiding, Optimization, and Analytics*, pages 145–170. Springer International Publishing, Cham, 2016. (cited on page 18)

[CG+18] A. Chassein, M. Goerigk, A. Kasperski, and P. Zieliński. On recoverable and two-stage robust selection problems with budgeted uncertainty. *European Journal of Operational Research*, 265(2):423–436, 2018. (cited on page 65)

[CH17] D. Chistikov and C. Haase. On the complexity of quantified integer programming. In I. Chatzigiannakis, P. Indyk, F. Kuhn, and A. Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 94:1–94:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. (cited on page 15)

[Che14] H. Chen. Beyond q-resolution and prenex form: a proof system for quantified constraint satisfaction. *Logical Methods in Computer Science (LMCS)*, 10(4), 2014. (cited on page 17)

[CHH99] M. Campbell, A. Hoane, and F.-H. Hsu. Search control methods in deep blue. In *AAAI Spring Symposium on Search Techniques for Problem Solving under Uncertainty and Incomplete Information*, pages 19–23, 1999. (cited on page 33)

[CK04] H. M. Chen and D. Kozen. *The computational complexity of quantified constraint satisfaction*. Cornell University, 2004. (cited on page 17)

[CKS81] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981. (cited on page 54)

[CM83] M. Campbell and T. Marsland. A comparison of minimax tree search algorithms. *Artificial Intelligence*, 20(4):347–367, 1983. (cited on pages 13 and 33)

[CM16] H. Chen and P. Mayr. Quantified constraint satisfaction on monoids. In *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. (cited on page 17)

[Con17] E. Conde. A minimum expected regret model for the shortest path problem with solution-dependent probability distributions. *Computers and Operations Research*, 77:11–19, 2017. (cited on page 21)

[Con19] E. Conde. Robust minmax regret combinatorial optimization problems with a resource–dependent uncertainty polyhedron of scenarios. *Computers and Operations Research*, 103:97–108, 2019. (cited on page 20)

[CRT90] J. D. Camm, A. S. Raturi, and S. Tsubakitani. Cutting big m down to size. *Interfaces*, 20(5):61–66, 1990. (cited on page 56)

[CS99] C. C. CarøE and R. Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24(1–2):37–45, 1999. (cited on page 19)

[CS+02] M. Cadoli, M. Schaerf, A. Giovanardi, and M. Giovanardi. An algorithm to evaluate quantified boolean formulae and its experimental evaluation. *Journal of Automated Reasoning*, 28(2):101–142, 2002. (cited on page 31)

[CS14] G. Chu and P. J. Stuckey. Nested constraint programs. In *International Conference on Principles and Practice of Constraint Programming*, pages 240–255. Springer International Publishing, Cham, 2014. (cited on page 17)

[CVB01] X. Chen and P. Van Beek. Conflict-directed backjumping revisited. *Journal of Artificial Intelligence Research*, 14:53–81, 2001. (cited on page 14)

[CZ09] X. Chen and Y. Zhang. Uncertain linear programs: Extended affinely adjustable robust counterparts. *Operations Research*, 57(6):1469–1482, 2009. (cited on page 18)

[Dan55] G. B. Dantzig. Linear programming under uncertainty. *Management Science*, 1(3-4):197–206, 1955. (cited on pages 1 and 17)

[DBL20] M. Daryalal, M. Bodur, and J. R. Luedtke. Lagrangian dual decision rules for multistage stochastic mixed integer programming. *arXiv preprint arXiv:2001.00761*, 2020. (cited on page 19)

[DE+18] C. Dürr, T. Erlebach, N. Megow, and J. Meißner. Scheduling with explorable uncertainty. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. (cited on page 22)

[DI15] E. Delage and D. A. Iancu. Robust multistage decision making. In *The Operations Research Revolution*, pages 20–46. INFORMS, 2015. (cited on page 18)

[DIN11] DIN standard 1988-500. *Codes of practice for drinking water installations - Part 500: Pressure boosting stations with RPM-regulated pumps*, german technical and scientific association for gas and water, 2011. (cited on page 24)

[DL04] C. Donninger and U. Lorenz. The chess monster hydra. In *Field Programmable Logic and Application*, pages 927–932. Springer Berlin Heidelberg, 2004. (cited on pages 14 and 29)

[DLL62] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962. (cited on page 29)

[DM02] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002. (cited on page 152)

[dRBT+17] F. J. de Ruiter, A. Ben-Tal, R. C. Brekelmans, and D. den Hertog. Robust optimization of uncertain multistage inventory systems with inexact data in decision rules. *Computational Management Science*, 14(1):45–66, 2017. (cited on page 18)

[DS06] M. Dyer and L. Stougie. Computational complexity of stochastic programming problems. *Mathematical Programming*, 106(3):423–432, 2006. (cited on page 19)

[DW+06] V. G. Deı, G. J. Woeginger, et al. On the robust assignment problem under a fixed number of cost scenarios. *Operations Research Letters*, 34(2):175–179, 2006. (cited on page 71)

[EG+12] L. F. Escudero, M. A. Garín, M. Merino, and G. Pérez. An algorithmic framework for solving large-scale multistage stochastic mixed 0–1 problems with nonsymmetric scenario trees. *Computers and Operations Research*, 39(5):1133–1144, 2012. (cited on page 19)

[EG+20] L. F. Escudero, M. A. Garín, J. F. Monge, and A. Unzueta. Some matheuristic algorithms for multistage stochastic optimization models with endogenous uncertainty and risk management. *European Journal of Operational Research*, 2020. (cited on page 21)

[EGU16] L. F. Escudero, M. A. Garín, and A. Unzueta. Cluster lagrangean decomposition in multistage stochastic optimization. *Computers and Operations Research*, 67:48–62, 2016. (cited on page 19)

[EH+08] T. Erlebach, M. Hoffmann, D. Krizanc, M. Mihalák, and R. Raman. Computing minimum spanning trees with uncertainty. In *25th International Symposium on Theoretical Aspects of Computer Science*, pages 277–288, 2008. (cited on page 21)

[EH+15] T. Erlebach, M. Hoffmann, et al. Query-competitive algorithms for computing with uncertainty. *Bulletin of EATCS*, 2(116), 2015. (cited on page 22)

[EH+17] T. Ederer, M. Hartisch, U. Lorenz, T. Opfer, and J. Wolf. Yasol: An open source solver for quantified mixed integer programs. In *15th International Conference on Advances in Computer Games, ACG 2017*, pages 224–233. Springer International Publishing, Cham, 2017. (cited on pages 2, 14, 29, 123, and 144)

[EHK16] T. Erlebach, M. Hoffmann, and F. Kammer. Query-competitive algorithms for cheapest set problems under uncertainty. *Theoretical Computer Science*, 613:51–64, 2016. (cited on pages 21 and 134)

[EL+11a] T. Ederer, U. Lorenz, A. Martin, and J. Wolf. Quantified linear programs: A computational study. In *Proceedings of the 19th European Conference on Algorithms*, ESA'11, pages 203–214. Springer Berlin Heidelberg, 2011. (cited on pages 2, 7, and 15)

[EL+11b] T. Ederer, U. Lorenz, T. Opfer, and J. Wolf. Modeling games with the help of quantified integer linear programs. In *13th International Conference on Advances in Computer Games, ACG 2011*, pages 270–281. Springer Berlin Heidelberg, 2011. (cited on pages 15, 130, and 132)

[ELO14] T. Ederer, U. Lorenz, and T. Opfer. Quantified combinatorial optimization. In *Operations Research Proceedings 2013*, pages 121–128. Springer International Publishing, Cham, 2014. (cited on page 49)

[ELW13] U. Egly, F. Lonsing, and M. Widl. Long-distance resolution: Proof generation and strategy extraction in search-based qbf solving. In K. McMillan, A. Middeldorp, and A. Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 291–308. Springer Berlin Heidelberg, 2013. (cited on page 145)

[EPH20] T. Erfani, K. Pachos, and J. J. Harou. Decision-dependent uncertainty in adaptive real-options water resource planning. *Advances in Water Resources*, 136:103490, 2020. (cited on page 21)

[ER+12] P. Eirinakis, S. Ruggieri, K. Subramani, and P. Wojciechowski. Computational complexities of inclusion queries over polyhedral sets. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2012. (cited on pages 15 and 16)

[ER+14] P. Eirinakis, S. Ruggieri, K. Subramani, and P. Wojciechowski. On quantified linear implications. *Annals of Mathematics and Artificial Intelligence*, 71(4):301–325, 2014. (cited on page 16)

[Erl18] T. Erlebach. Computing and scheduling with explorable uncertainty. In F. Manea, R. G. Miller, and D. Nowotka, editors, *Sailing Routes in the World of Computation*, pages 156–160. Springer International Publishing, Cham, 2018. (cited on pages 21 and 22)

[FM+00] T. Feder, R. Motwani, R. Panigrahy, C. Olston, and J. Widom. Computing the median with uncertainty. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, STOC '00, pages 602–607, 2000. (cited on page 21)

[FM+07] T. Feder, R. Motwani, L. O'Callaghan, C. Olston, and R. Panigrahy. Computing shortest paths with uncertainty. *Journal of Algorithms*, 62(1):1–18, 2007. (cited on pages 21 and 136)

[FM09] M. Fischetti and M. Monaci. Light robustness. In *Robust and Online Large-Scale Optimization*, pages 61–84. Springer Berlin Heidelberg, 2009. (cited on page 18)

[FMM92] R. Feldmann, P. Mysliwietz, and B. Monien. Distributed game tree search on a massively parallel system. In *Data structures and efficient algorithms*, pages 270–288. Springer Berlin Heidelberg, 1992. (cited on page 29)

[FO07] A. Ferguson and B. O'Sullivan. Quantified constraint satisfaction problems: from relaxations to explanations. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 74–79, 2007. (cited on page 17)

[GG05] V. Goel and I. E. Grossmann. A lagrangean duality based branch and bound for solving linear stochastic programs with decision dependent uncertainty. In *Computer Aided Chemical Engineering*, volume 20, pages 55–60. Elsevier, 2005. (cited on page 21)

[GG06] V. Goel and I. Grossmann. A class of stochastic programs with decision dependent uncertainty. *Mathematical Programming*, 108(2):355–394, 2006. (cited on page 21)

[GG11] V. Gupta and I. E. Grossmann. Solution strategies for multistage stochastic programming with endogenous uncertainties. *Computers & Chemical Engineering*, 35(11):2235–2247, 2011. (cited on page 21)

[GG14] V. Gupta and I. Grossmann. A new decomposition algorithm for multistage stochastic programs with endogenous uncertainties. *Computers & Chemical Engineering*, 62:62–79, 2014. (cited on pages 20 and 21)

[GG+15] M. Goerigk, M. Gupta, J. Ide, A. Schöbel, and S. Sen. The robust knapsack problem with queries. *Computers and Operations Research*, 55:12–22, 2015. (cited on pages 21 and 136)

[GH+16] D. Gade, G. Hackebeil, S. M. Ryan, J.-P. Watson, R. J.-B. Wets, and D. L. Woodruff. Obtaining lower bounds from the progressive hedging algorithm for stochastic mixed-integer programs. *Mathematical Programming*, 157(1):47–67, 2016. (cited on page 19)

[GK+15] G. Gamrath, T. Koch, A. Martin, M. Miltenberger, and D. Weninger. Progress in presolving for mixed integer programming. *Mathematical Programming Computation*, 7(4):367–398, 2015. (cited on page 40)

[GKW19] A. Georghiou, D. Kuhn, and W. Wiesemann. The decision rule approach to optimization under uncertainty: methodology and applications. *Computational Management Science*, 16(4):545–576, 2019. (cited on page 18)

[GMT14] V. Gabrel, C. Murat, and A. Thiele. Recent advances in robust optimization: An overview. *European Journal of Operational Research*, 235(3):471–483, 2014. (cited on pages 1 and 17)

[GN+02] E. Giunchiglia, M. Narizzano, A. Tacchella, et al. Learning for quantified boolean logic satisfiability. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 649–654, 2002. (cited on page 14)

[GN+08] I. P. Gent, P. Nightingale, A. Rowley, and K. Stergiou. Solving quantified constraint satisfaction problems. *Artificial Intelligence*, 172(6-7):738–771, 2008. (cited on pages 33 and 144)

[Gna16] S. Gnad. Quantifizierte ganzzahlige Programmierung angewandt auf Flugplanungsprobleme. Bachelor's thesis, University of Siegen, 2016. (cited on page 22)

[GNT03] E. Giunchiglia, M. Narizzano, and A. Tacchella. Backjumping for quantified boolean logic satisfiability. *Artificial Intelligence*, 145(1):99–120, 2003. (cited on pages 14, 29, 33, and 144)

[GP+04] A. Gupta, M. Pál, R. Ravi, and A. Sinha. Boosted sampling: Approximation algorithms for stochastic optimization. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pages 417–426, 2004. (cited on page 17)

[GP+06] A. Galenko, E. Popova, E. Kee, and R. Grantom. Optimal preventive maintenance under decision dependent uncertainty. In *14th International Conference on Nuclear Engineering*, pages 23–29. American Society of Mechanical Engineers Digital Collection, 2006. (cited on pages 21 and 136)

[GS10] J. Goh and M. Sim. Distributionally robust optimization and its tractable approximations. *Operations Research*, 58(4-part-1):902–917, 2010. (cited on page 19)

[GS16] M. Goerigk and A. Schöbel. Algorithm engineering in robust optimization. In *Algorithm Engineering*, pages 245–279. Springer International Publishing, Cham, 2016. (cited on page 18)

[GSS11] M. Gupta, Y. Sabharwal, and S. Sen. The update complexity of selection and related problems. *Leibniz International Proceedings in Informatics, LIPIcs*, 13, 2011. (cited on page 21)

[GW+20] J. Gao, J. Wang, K. Wu, and R. Chen. Solving quantified constraint satisfaction problems with value selection rules. *Frontiers of Computer Science*, 14(5):1–11, 2020. (cited on page 17)

[GYdH15] B. L. Gorissen, İ. Yanıkoğlu, and D. den Hertog. A practical guide to robust optimization. *Omega*, 53:124–137, 2015. (cited on page 18)

[HAL63] A. HALES. Regularity and positional games. *Trans. Amer. Math. Soc.*, 106:222–229, 1963. (cited on page 130)

[HAW14] J. Hu, M. Aminzadeh, and Y. Wang. Searching feasible design space by solving quantified constraint satisfaction problems. *Journal of Mechanical Design*, 136(3), 2014. (cited on page 17)

[HBT18] L. Hellemo, P. I. Barton, and A. Tomasgard. Decision-dependent probabilities in stochastic programs with recourse. *Computational Management Science*, 15(3-4):369–395, 2018. (cited on page 21)

[HdL19] M. M. Halldórsson and M. S. de Lima. Query-competitive sorting with uncertainty. In *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019. (cited on page 21)

[HE+16] M. Hartisch, T. Ederer, U. Lorenz, and J. Wolf. Quantified integer programs with polyhedral uncertainty set. In *Computers and Games - 9th International Conference, CG 2016*, pages 156–166. Springer International Publishing, Cham, 2016. (cited on page 49)

[Hel11] H. Helmke. Scheduling algorithms for atm applications tools and toys. In *Digital Avionics Systems Conference, 2011 IEEE/AIAA 30th*, pages 3C2–1. IEEE, 2011. (cited on page 22)

[HH+16] A. Heidt, H. Helmke, M. Kapolke, F. Liers, and A. Martin. Robust runway scheduling under uncertain conditions. *Journal of Air Transport Management*, 56:28–37, 2016. (cited on page 22)

[HH+18] M. Hartisch, A. Herbst, U. Lorenz, and J. B. Weber. Towards resilient process networks-designing booster stations via quantified programming. In *Applied Mechanics and Materials*, volume 885, pages 199–210. Trans Tech Publ, 2018. (cited on pages 24 and 25)

[HJ+15] M. Heule, M. Järvisalo, F. Lonsing, M. Seidl, and A. Biere. Clause elimination for sat and qsat. *Journal of Artificial Intelligence Research*, 53:127–168, 2015. (cited on page 16)

[HK65] L. Henkin and C. R. Karp. Some remarks on infinitely long formulas. *Journal of Symbolic Logic*, 30(1):96–97, 1965. (cited on page 16)

[HKM16] F. Hooshmand Khaligh and S. MirHassani. A mathematical model for vehicle routing problem under endogenous uncertainty. *International Journal of Production Research*, 54(2):579–590, 2016. (cited on page 21)

[HL19a] M. Hartisch and U. Lorenz. Game tree search in a robust multistage optimization framework: Exploiting pruning mechanisms. *arXiv preprint arXiv:1811.12146;* To appear in: *16th International Conference on Advances in Computer Games, ACG 2019*, 2019. (cited on pages 32, 40, 149, and 153)

[HL19b] M. Hartisch and U. Lorenz. Mastering uncertainty: Towards robust multistage optimization with decision dependent uncertainty. In *Pacific Rim International Conference on Artificial Intelligence*, pages 446–458. Springer International Publishing, Cham, 2019. (cited on page 7)

[Hla12] M. Hladık. Interval linear programming: A survey. *Linear Programming - New Frontiers in Theory and Applications*, pages 85–120, 2012. (cited on page 17)

[HM16] F. Hooshmand and S. MirHassani. Efficient constraint reduction in multistage stochastic programming problems with endogenous uncertainty. *Optimization Methods and Software*, 31(2):359–376, 2016. (cited on page 21)

[HR09] H. Heitsch and W. Römisch. Scenario tree modeling for multistage stochastic programs. *Mathematical Programming*, 118(2):371–406, 2009. (cited on page 19)

[ISS13] D. A. Iancu, M. Sharma, and M. Sviridenko. Supermodularity and affine policies in dynamic robust optimization. *Operations Research*, 61(4):941–956, 2013. (cited on page 18)

[Jan18] M. Janota. Towards generalization in qbf solving via machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. (cited on page 16)

[JK+16] M. Janota, W. Klieber, J. Marques-Silva, and E. Clarke. Solving qbf with counterexample guided refinement. *Artif. Intell.*, 234:1 – 25, 2016. (cited on pages 16 and 33)

[JMS15] M. Janota and J. Marques-Silva. Solving qbf by clause selection. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 325–331, 2015. (cited on page 16)

[Jon98] T. Jonsbråten. Oil field optimization under price uncertainty. *Journal of the Operational Research Society*, pages 811–818, 1998. (cited on page 21)

[JWW98] T. Jonsbråten, R. J.-B. Wets, and D. Woodruff. A class of stochastic programs withdecision dependent random elements. *Annals of Operations Research*, 82(0):83–106, 1998. (cited on page 21)

[KA19] A. J. Keith and D. K. Ahner. A survey of decision making and optimization under uncertainty. *Annals of Operations Research*, pages 1–35, 2019. (cited on page 17)

[Kah91] S. Kahan. A model for data in motion. In *Proceedings of the Twenty-third Annual ACM Symposium on Theory of computing*, STOC '91, pages 265–277, 1991. (cited on pages 21 and 134)

[Kaw96] Y. Kawano. Using similar positions to search game trees. *Games of No Chance*, 29:193–202, 1996. (cited on page 34)

[KB+19] A. Khassiba, F. Bastin, B. Gendron, S. Cafieri, and M. Mongeau. Extended aircraft arrival management under uncertainty: A computational study. *Journal of Air Transportation*, pages 1–13, 2019. (cited on page 23)

[KLV12] R. Kalaı, C. Lamboray, and D. Vanderpooten. Lexicographic α-robustness: An alternative to min–max criteria. *European Journal of Operational Research*, 220(3):722–728, 2012. (cited on page 18)

[KM75] D. Knuth and R. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6(4):293–326, 1975. (cited on pages 14, 29, 54, 86, and 107)

[KT01] S. Khanna and W.-C. Tan. On computing functions with uncertainty. In *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 171–182, 2001. (cited on page 21)

[KV18] B. Korte and J. Vygen. Spanning trees and arborescences. In *Combinatorial Optimization*, pages 133–157. Springer Berlin Heidelberg, 2018. (cited on page 11)

[KWG11] 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. (cited on page 18)

[KY97] P. Kouvelis and G. Yu. *Robust Discrete Optimization and Its Applications*. Springer US, Boston, MA, 1997. (cited on pages 10 and 18)

[KZ16] A. Kasperski and P. Zieliński. Robust discrete optimization under discrete and interval uncertainty: A survey. In *Robustness Analysis in Decision Aiding, Optimization, and Analytics*, pages 113–143. Springer International Publishing, Cham, 2016. (cited on pages 10 and 18)

[LB+13] A. Lallouet, L. Bordeaux, et al. Constraint games: Framework and local search solver. In *25th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'13)*, pages 963–970. IEEE, 2013. (cited on page 17)

[LB+15] F. Lonsing, F. Bacchus, A. Biere, U. Egly, and M. Seidl. Enhancing search-based qbf solving by dynamic blocked clause elimination. In *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 418–433. Springer Heidelberg Berlin, 2015. (cited on page 16)

[LDF11] Z. Li, R. Ding, and C. A. Floudas. A comparative theoretical and computational study on robust counterpart optimization: I. robust linear optimization and robust mixed integer linear optimization. *Industrial & Engineering Chemistry Research*, 50(18):10567–10603, 2011. PMID: 21935263. (cited on page 29)

[LE17] F. Lonsing and U. Egly. Depqbf 6.0: A search-based qbf solver beyond traditional qcdcl. In *International Conference on Automated Deduction*, pages 371–384. Springer International Publishing, Cham, 2017. (cited on page 16)

[LE19] F. Lonsing and U. Egly. Qratpre+: Effective qbf preprocessing via strong redundancy properties. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 203–210. Springer International Publishing, Cham, 2019. (cited on page 16)

[LG16] N. Lappas and C. Gounaris. Multi-stage adjustable robust optimization for process scheduling under uncertainty. *AIChE Journal*, 62(5):1646–1667, 2016. (cited on pages 18, 20, and 137)

[LG18a] N. H. Lappas and C. E. Gounaris. Robust optimization for decision-making under endogenous uncertainty. *Computers & Chemical Engineering*, 111:252–266, 2018. (cited on page 20)

[LG18b] N. H. Lappas and C. E. Gounaris. Theoretical and computational comparison of continuous-time process scheduling models for adjustable robust optimization. *AIChE Journal*, 64(8):3055–3070, 2018. (cited on page 20)

[LL+09] C. Liebchen, M. Lübbecke, R. Möhring, and S. Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In *Robust and online large-scale optimization*, pages 1–27. Springer Berlin Heidelberg, 2009. (cited on page 18)

[LL16] F. Liers and U. Lorenz. personal communication, 2016. (cited on page 22)

[LM+20] S. Leyffer, M. Menickelly, T. Munson, C. Vanaret, and S. M. Wild. A survey of nonlinear robust optimization. *INFOR: Information Systems and Operational Research*, pages 1–32, 2020. (cited on page 17)

[LMS19] R. Levi, T. Magnanti, and Y. Shaposhnik. Scheduling with testing. *Management Science*, 65(2):776–793, 2019. (cited on page 21)

[LMW10] U. Lorenz, A. Martin, and J. Wolf. Polyhedral and algorithmic properties of quantified linear programs. In *Proceedings of the 18th Annual European Conference on Algorithms: Part I*, ESA'10, pages 512–523. Springer Berlin Heidelberg, 2010. (cited on pages 12 and 15)

[LOW13] U. Lorenz, T. Opfer, and J. Wolf. Solution techniques for quantified linear programs and the links to gaming. In *Computers and Games - 8th International Conference, CG 2013*, pages 110–124. Springer International Publishing, Cham, 2013. (cited on page 15)

[LRS+19] N. H. Lappas, L. A. Ricardez-Sandoval, R. Fukasawa, and C. E. Gounaris. Adjustable robust optimization for multi-tasking scheduling with reprocessing due to imperfect tasks. *Optimization and Engineering*, 20(4):1117–1159, 2019. (cited on page 20)

[LS+16] Á. Lorca, X. A. Sun, E. Litvinov, and T. Zheng. Multistage adaptive robust optimization for the unit commitment problem. *Operations Research*, 64(1):32–51, 2016. (cited on page 18)

[LSvG16] F. Lonsing, M. Seidl, and A. van Gelder. The qbf gallery: Behind the scenes. *Artificial Intelligence*, 237:92–114, 2016. (cited on page 16)

[LW15] U. Lorenz and J. Wolf. Solving multistage quantified linear optimization problems with the alpha–beta nested benders decomposition. *EURO Journal on Computational Optimization*, 3(4):349–370, 2015. (cited on page 15)

[Mar17] B. Martin. Quantified constraints in twenty seventeen. In *Dagstuhl Follow-Ups*, volume 7. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. (cited on page 17)

[Mar18] T. Marx. Optimierung unter Unsicherheit — Grundlagen und Dokumentation der Implementierung einer parametrisierten Reduktionsfunktion für den QMIP-Solver Yasol. student research project, University of Siegen, 2018. (cited on page 141)

[MBD19] D. C. McElfresh, H. Bidkhori, and J. P. Dickerson. Scalable robust kidney exchange. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1077–1084, 2019. (cited on page 21)

[MGK19] L. R. Matthews, C. E. Gounaris, and I. G. Kevrekidis. Designing networks with resiliency to edge failures using two-stage robust optimization. *European Journal of Operational Research*, 279(3):704–720, 2019. (cited on page 20)

[ĐMM17] P. Đapić, P. Marković, and B. Martin. Quantified constraint satisfaction problem on semicomplete digraphs. *ACM Transactions on Computational Logic (TOCL)*, 18(1):1–47, 2017. (cited on page 17)

[MMS17] N. Megow, J. Meißner, and M. Skutella. Randomization helps computing a minimum spanning tree under uncertainty. *SIAM Journal on Computing*, 46(4):1217–1240, 2017. (cited on page 21)

[MNL19] F. Motamed Nasab and Z. Li. Multistage adaptive optimization using hybrid scenario and decision rule formulation. *AIChE Journal*, 65(12):e16764, 2019. (cited on page 18)

[MOQ15] D. Mehta, B. O'Sullivan, and L. Quesada. Extending the notion of preferred explanations for quantified constraint satisfaction problems. In *International Colloquium on Theoretical Aspects of Computing*, pages 309–327. Springer International Publishing, Cham, 2015. (cited on page 17)

[MSLM09] J. Marques-Silva, I. Lynce, and S. Malik. Conflict-driven clause learning sat solvers. In *Handbook of Satisfiability*, pages 131–153. ios Press, 2009. (cited on page 14)

[MSU99] R. Möhring, A. Schulz, and M. Uetz. Approximation in stochastic scheduling: The power of lp-based priority policies. *J. ACM*, 46(6):924–942, 1999. (cited on page 17)

[MTL78] R. McGill, J. W. Tukey, and W. A. Larsen. Variations of box plots. *The American Statistician*, 32(1):12–16, 1978. (cited on page 184)

[MV06] N. Megow and T. Vredeveld. Approximation in preemptive stochastic online scheduling. In Y. Azar and T. Erlebach, editors, *Algorithms – ESA 2006*, pages 516–527. Springer Berlin Heidelberg, 2006. (cited on page 17)

[MVZ95] J. M. Mulvey, R. J. Vanderbei, and S. A. Zenios. Robust optimization of large-scale systems. *Operations Research*, 43(2):264–281, 1995. (cited on page 49)

[Nem94] G. L. Nemhauser. The age of optimization: Solving large-scale real-world problems. *Operations Research*, 42(1):5–13, 1994. (cited on page 2)

[NP20] D. Nguyen and I. Pak. The computational complexity of integer programming with alternations. *Mathematics of Operations Research*, 45(1):191–204, 2020. (cited on page 15)

[NR17] O. Nohadani and A. Roy. Robust optimization with time-dependent uncertainty in radiation therapy. *IISE Transactions on Healthcare Systems Engineering*, 7(2):81–92, 2017. (cited on page 21)

[NRL18] N. Noyan, G. Rudolf, and M. Lejeune. Distributionally robust optimization with decision-dependent ambiguity set. http://www.optimization-online.org/DB_HTML/2018/09/6821.html (accessed May 3, 2020), 2018. (cited on page 21)

[NS18] O. Nohadani and K. Sharma. Optimization under decision-dependent uncertainty. *SIAM Journal on Optimization*, 28(2):1773–1795, 2018. (cited on pages 20 and 137)

[NW88] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, USA, 1988. (cited on pages 17, 29, 31, and 33)

[NY17] C. Ning and F. You. A data-driven multistage adaptive robust optimization framework for planning and scheduling under uncertainty. *AIChE Journal*, 63(10):4343–4369, 2017. (cited on page 18)

[NY19] C. Ning and F. You. Optimization under uncertainty in the era of big data and deep learning: When machine learning meets mathematical programming. *Computers & Chemical Engineering*, 125:434–448, 2019. (cited on page 17)

[OW00] C. Olston and J. Widom. Offering a precision-performance tradeoff for aggregation queries over replicated data. In *Proceedings of the 26th International Conference on Very Large Data Bases*, pages 144–155. Morgan Kaufmann Publishers Inc., 2000. (cited on page 22)

[Pal19] A. Palmieri. *Constraint games revisited*. PhD thesis, Normandy University, Caen, France, 2019. (cited on page 17)

[Pap85] C. Papadimitriou. Games against nature. *Journal of Computer and System Sciences*, 31(2):288–301, 1985. (cited on pages 2, 19, and 60)

[PdB01] W. Pijls and A. de Bruin. Game tree algorithms and solution trees. *Theoretical Computer Science*, 252(1):197–215, 2001. (cited on pages 7, 13, 84, and 107)

[PdH16] K. Postek and D. d. den Hertog. Multistage adjustable robust mixed-integer optimization via iterative splitting of the uncertainty set. *INFORMS Journal on Computing*, 28(3):553–574, 2016. (cited on page 18)

[Pfl90] G. C. Pflug. On-line optimization of simulated markovian processes. *Mathematics of Operations Research*, 15(3):381–395, 1990. (cited on page 21)

[Pos13] M. Poss. Robust combinatorial optimization with variable budgeted uncertainty. *4OR*, 11(1):75–92, 2013. (cited on pages 20 and 136)

[Pos14] M. Poss. Robust combinatorial optimization with variable cost uncertainty. *European Journal of Operational Research*, 237(3):836–845, 2014. (cited on page 20)

[Pos18] M. Poss. Robust combinatorial optimization with knapsack uncertainty. *Discrete Optimization*, 27:88–102, 2018. (cited on page 20)

[PR91] M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1):60–100, 1991. (cited on page 107)

[PRA01] G. Peterson, J. Reif, and S. Azhar. Lower bounds for multiplayer noncooperative games of incomplete information. *Computers & Mathematics with Applications*, 41(7-8):957–992, 2001. (cited on page 16)

[PS+96] A. Plaat, J. Schaeffer, W. Pijls, and A. de Bruin. Best-first fixed-depth minimax algorithms. *Artificial Intelligence*, 87(1):255–293, 1996. (cited on page 34)

[PS+10] S. Peeta, F. S. Salman, D. Gunnec, and K. Viswanath. Pre-disaster investment decisions for strengthening a highway network. *Computers and Operations Research*, 37(10):1708–1719, 2010. (cited on page 21)

[PS19] L. Pulina and M. Seidl. The 2016 and 2017 qbf solvers evaluations (qbfeval'16 and qbfeval'17). *Artificial Intelligence*, 274:224–248, 2019. (cited on page 16)

[PSS19a] T. Peitl, F. Slivovsky, and S. Szeider. Dependency learning for qbf. *Journal of Artificial Intelligence Research*, 65:180–208, 2019. (cited on page 16)

[PSS19b] T. Peitl, F. Slivovsky, and S. Szeider. Long-distance q-resolution with dependency schemes. *Journal of Automated Reasoning*, 63(1):127–155, 2019. (cited on page 16)

[PWB19] A. Philpott, F. Wahid, and J. Bonnans. Midas: A mixed integer dynamic approximation scheme. *Mathematical Programming*, pages 1–32, 2019. (cited on page 19)

[QS17] Y. Qi and S. Sen. The ancestral benders' cutting plane algorithm with multi-term disjunctions for mixed-integer recourse decisions in stochastic programming. *Mathematical Programming*, 161(1-2):193–235, 2017. (cited on page 19)

[Qui54] W. V. Quine. Quantification and the empty domain. *The Journal of Symbolic Logic*, 19(3):177–179, 1954. (cited on page 51)

[Rab17] M. N. Rabe. A resolution-style proof system for dqbf. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 314–325. Springer International Publishing, Cham, 2017. (cited on page 16)

[Rei83] A. Reinefeld. An improvement to the scout tree search algorithm. *ICGA Journal*, 6(4):4–14, 1983. (cited on page 34)

[RT15] M. N. Rabe and L. Tentrup. Caqe: a certifying qbf solver. In *2015 Formal Methods in Computer-Aided Design (FMCAD)*, pages 136–143. IEEE, 2015. (cited on page 16)

[Rus97] A. Ruszczyński. Decomposition methods in stochastic programming. *Mathematical Programming*, 79(1-3):333–353, 1997. (cited on page 19)

[RW91] R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16(1):119–147, 1991. (cited on pages 19, 31, and 49)

[RW17] J. O. Royset and R. J.-B. Wets. Variational theory for optimization under stochastic ambiguity. *SIAM Journal on Optimization*, 27(2):1118–1149, 2017. (cited on page 21)

[SB+19] A. Shukla, A. Biere, L. Pulina, and M. Seidl. A survey on applications of quantified boolean formulas. In *31st IEEE International Conference on Tools with Artificial Intelligence (ICTAI'19)*, pages 78–84. IEEE, 2019. (cited on page 16)

[SC+10] S. Solak, J.-P. B. Clarke, E. L. Johnson, and E. R. Barnes. Optimization of r&d project portfolios under endogenous uncertainty. *European Journal of Operational Research*, 207(1):420–433, 2010. (cited on page 21)

[Sch83] J. Schaeffer. The history heuristic. *ICGA Journal*, 6(3):16–19, 1983. (cited on page 47)

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986. (cited on page 17)

[Sch89] J. Schaeffer. The history heuristic and alpha-beta search enhancements in practice. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(11):1203–1212, 1989. (cited on page 34)

[Sch03] R. Schultz. Stochastic programming with integer variables. *Mathematical Programming*, 97(1-2):285–309, 2003. (cited on page 19)

[Sch13] J. Schaeffer. *One jump ahead: challenging human supremacy in checkers*. Springer US, New York, NY, 2013. (cited on page 33)

[SDR09] A. Shapiro, D. Dentcheva, and A. Ruszczyński. Lectures on stochastic programming: Modeling and theory, 2009. (cited on page 19)

[Sen05] S. Sen. Algorithms for stochastic mixed-integer programming models. *Handbooks in Operations Research and Management Science*, 12:515–558, 2005. (cited on page 19)

[SH+16] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016. (cited on page 29)

[Sha08] A. Shapiro. Stochastic programming approach to optimization under uncertainty. *Mathematical Programming*, 112(1):183–220, 2008. (cited on pages 1 and 19)

[Sha11] A. Shapiro. A dynamic programming approach to adjustable robust optimization. *Operations Research Letters*, 39(2):83–87, 2011. (cited on page 18)

[SJ+19] C. Scholl, J.-H. R. Jiang, R. Wimmer, and A. Ge-Ernst. A pspace subclass of dependency quantified boolean formulas and its effective solving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1584–1591, 2019. (cited on page 16)

[SN05] A. Shapiro and A. Nemirovski. On complexity of stochastic programming problems. In *Continuous Optimization*, pages 111–146. Springer US, Boston, MA, 2005. (cited on page 19)

[Söl12] G. Sölveling. *Stochastic programming methods for scheduling of airport runway operations under uncertainty*. PhD thesis, Georgia Institute of Technology, 2012. (cited on page 24)

[SS+17] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017. (cited on page 17)

[ST08] J. C. Smith and Z. C. Taskin. A tutorial guide to mixed-integer programming models and solution techniques. *Optimization in Medicine and Biology*, pages 521–548, 2008. (cited on page 56)

[Sto76] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976. (cited on page 16)

[Sto79] G. Stockman. A minimax algorithm better than alpha-beta? *Artificial Intelligence*, 12(2):179–196, 1979. (cited on pages 13, 34, and 84)

[Sub03] K. Subramani. An analysis of quantified linear programs. In *Discrete Mathematics and Theoretical Computer Science*, pages 265–277. Springer Berlin Heidelberg, 2003. (cited on pages 7 and 10)

[Sub04] K. Subramani. Analyzing selected quantified integer programs. In D. Basin and M. Rusinowitch, editors, *Automated Reasoning: Second International Joint Conference, IJCAR 2004*, volume 3097, pages 342–356. Springer Berlin Heidelberg, 2004. (cited on pages 2, 7, and 10)

[SW+12] S. A. Spacey, W. Wiesemann, D. Kuhn, and W. Luk. Robust software partitioning with multiple instantiation. *INFORMS Journal on Computing*, 24(3):500–515, 2012. (cited on page 20)

[SZ14] S. Sen and Z. Zhou. Multistage stochastic decomposition: a bridge between stochastic programming and approximate dynamic programming. *SIAM Journal on Optimization*, 24(1):127–153, 2014. (cited on page 19)

[Ten19] L. Tentrup. Caqe and quabs: Abstraction based qbf solvers. *Journal on Satisfiability, Boolean Modeling and Computation*, 11(1):155–210, 2019. (cited on page 16)

[TGG09] B. Tarhan, I. E. Grossmann, and V. Goel. Stochastic programming approach for the planning of offshore oil or gas field infrastructure under decision-dependent uncertainty. *Industrial & Engineering Chemistry Research*, 48(6):3078–3097, 2009. (cited on page 21)

[TGG13] B. Tarhan, I. E. Grossmann, and V. Goel. Computational strategies for non-convex multistage minlp models with decision-dependent uncertainty and gradual uncertainty resolution. *Annals of Operations Research*, 203(1):141–166, 2013. (cited on page 21)

[TTE09] A. Thiele, T. Terry, and M. Epelman. Robust linear optimization with recourse. *Rapport Technique*, pages 4–37, 2009. (cited on page 18)

[vdHNL05] H. van den Herik, J. Nunn, and D. Levy. Adams outclassed by hydra. *ICGA Journal*, 28(2):107–110, 2005. (cited on page 33)

[VGM16] R. Vujanic, P. Goulart, and M. Morari. Robust optimization of schedules affected by uncertain events. *Journal of Optimization Theory and Applications*, 171(3):1033–1054, 2016. (cited on page 20)

[VKR11] P. Vayanos, D. Kuhn, and B. Rustem. Decision rules for information discovery in multistage stochastic programming. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 7368–7373. IEEE, 2011. (cited on page 21)

[VSW69] R. M. Van Slyke and R. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663, 1969. (cited on page 19)

[WES17] P. Wojciechowski, P. Eirinakis, and K. Subramani. Erratum to: Analyzing restricted fragments of the theory of linear arithmetic. *Annals of Mathematics and Artificial Intelligence*, 79(4):371–392, 2017. (cited on page 15)

[Wet74] R.-B. Wets. Stochastic programs with fixed recourse: The equivalent deterministic program. *SIAM Review*, 16(3):309–339, 1974. (cited on pages 19, 29, and 30)

[Wil76] H. P. Williams. Fourier-motzkin elimination extension to integer programming problems. *Journal of Combinatorial Theory, Series A*, 21(1):118–123, 1976. (cited on page 51)

[WK+17] R. Wimmer, A. Karrenbauer, R. Becker, C. Scholl, and B. Becker. From dqbf to qbf by dependency elimination. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 326–343. Springer International Publishing, Cham, 2017. (cited on page 16)

[WKS14] W. Wiesemann, D. Kuhn, and M. Sim. Distributionally robust convex optimization. *Operations Research*, 62(6):1358–1376, 2014. (cited on page 19)

[Wol15] J. Wolf. Quantified linear programming. In *Forschungsberichte zur Fluidsystemtechnik*, Bd. 7. Shaker, 2015. (cited on pages 2, 7, 8, 10, 15, 17, 19, 25, 29, 30, 31, 43, 44, 86, 99, 107, and 139)

[WR+17] R. Wimmer, S. Reimer, P. Marin, and B. Becker. Hqspre–an effective preprocessor for qbf and dqbf. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 373–390. Springer Berlin Heidelberg, 2017. (cited on page 16)

[WSE16] P. J. Wojciechowski, K. Subramani, and P. Eirinakis. On the computational complexities of quantified integer programming variants. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2016. (cited on pages 15 and 16)

[WSP12] M. Webster, N. Santen, and P. Parpas. An approximate dynamic programming framework for modeling global climate policy under decision-dependent uncertainty. *Computational Management Science*, 9(3):339–362, 2012. (cited on page 21)

[WvdW+04] M. Winands, E. van der Werf, H. van den Herik, and J. Uiterwijk. The relative history heuristic. In *Computers and Games, 4th International Conference, CG 2004*, pages 262–272. Springer International Publishing, Cham, 2004. (cited on page 34)

[WZ05] S. W. Wallace and W. T. Ziemba. *Applications of stochastic programming*. SIAM, 2005. (cited on page 19)

[YGdH19] İ. Yanıkoğlu, B. L. Gorissen, and D. den Hertog. A survey of adjustable robust optimization. *European Journal of Operational Research*, 277(3):799–813, 2019. (cited on page 18)

[ZAS19] J. Zou, S. Ahmed, and X. A. Sun. Stochastic dual dynamic integer programming. *Mathematical Programming*, 175(1-2):461–502, 2019. (cited on page 19)

[ZC19] Z. Zeng and S. Cremaschi. A general primal bounding framework for large-scale multistage stochastic programs under endogenous uncertainties. *Chemical Engineering Research and Design*, 141:464–480, 2019. (cited on page 21)

[ZDHS18] J. Zhen, D. Den Hertog, and M. Sim. Adjustable robust optimization via fourier–motzkin elimination. *Operations Research*, 66(4):1086–1100, 2018. (cited on page 18)

[Zha03] L. Zhang. *Searching for Truth: Techniques for Satisfiability of Boolean Formulas*. PhD thesis, Princeton University, Princeton, NJ, USA, 2003. (cited on pages 33 and 145)

[Zha07] Y. Zhang. General robust-optimization formulation for nonlinear programming. *Journal of Optimization Theory and Applications*, 132(1):111–124, 2007. (cited on page 17)

[Zha16] J.-Y. Zhang. A novel schedulability test algorithm for preemptive real-time scheduling problem in qcsp. In *2016 International Conference on Information System and Artificial Intelligence (ISAI)*, pages 120–123. IEEE, 2016. (cited on page 17)

[ZK+17] X. Zhang, M. Kamgarpour, A. Georghiou, P. Goulart, and J. Lygeros. Robust optimal control with adjustable uncertainty sets. *Automatica*, 75:249–259, 2017. (cited on pages 20 and 21)

[ZY19] S. Zhao and F. You. Resilient supply chain design and operations with decision-dependent uncertainty using a data-driven robust optimization approach. *AIChE Journal*, 65(3):1006–1021, 2019. (cited on page 20)

[ZZ18] Y. Zhan and Q. P. Zheng. A multistage decision-dependent stochastic bilevel programming approach for power generation investment expansion planning. *IISE Transactions*, 50(8):720–734, 2018. (cited on page 21)