WORKING PAPER: Solving Multistage Quantified Linear Optimization Problems with the Alpha-Beta Nested Benders Decomposition

Ulf Lorenz[1] and Jan Wolf[2]

[1]Fluid Systems Technology, Technische Universität Darmstadt, Germany
ulf.lorenz@fst.tu-darmstadt.de
[2]Institute of Mathematics, Technische Universität Darmstadt, Germany
wolf@mathematik.tu-darmstadt.de

Abstract. Quantified linear programs (QLPs) are linear programs (LPs) with variables being either existentially or universally quantified. QLPs are convex multistage decision problems on the one side, and two-person zero-sum games between an existential and a universal player on the other side. Solutions of feasible QLPs are so called winning strategies for the existential player that specify how to react on moves - certain fixations of universally quantified variables - of the universal player to certainly win the game. To find a certain best one among different winning strategies, we propose the extension of the QLP decision problem by an objective function. To solve the resulting QLP optimization problem, we exploit the problem's hybrid nature and combine linear programming techniques with solution techniques from game-tree search. As a result, we present an extension of the Nested Benders Decomposition algorithm by the $\alpha\beta$-heuristic and move-ordering, two techniques that are successfully used in game-tree search to solve minimax trees. We furthermore exploit solution information from QLP relaxations obtained by quantifier shifting. The applicability is examined in an experimental evaluation.

Keywords: Quantified Linear Programming, Optimization under Uncertainty, Nested Benders Decomposition, Game-Tree Search

1 Introduction

Nowadays, we are able to solve large mixed-integer programs (MIPs) of practical size, but companies observe an increasing danger of disruptions, which prevent them from acting as planned. One reason is that input data for a given problem is often assumed to be deterministic and exactly known when decisions have to be made, but in reality they are often afflicted with some kinds of uncertainties. Examples are flight and travel times, throughput-time, or arrival times of externally produced goods. Thus, there is a need for planning and deciding under uncertainty. Prominent solution paradigms for optimization under uncertainty are Stochastic Programming [8], Robust Optimization [3, 16], Dynamic Programming [2], Sampling [13], the use of Markov-Chains [26] and other LP-based techniques [15, 18, 19].

Uncertainty, however, often pushes the complexity of problems that are in P or NP, to the complexity class PSPACE. Also many optimization problems under uncertainty are PSPACE-complete [20] and therefore, NP-complete integer programs are not suitable to model these problems anymore. Relatively unexplored are the abilities of linear programming extensions for PSPACE-complete problems. In this context, Subramani introduced the notion of quantified linear programs (QLPs) [23–25]. While it is known that quantified linear integer programs (QIPs) are PSPACE-complete, the exact complexity class of their QLP-Relaxations is unknown in general. The idea of our research is to explore the abilities of linear programming techniques when being applied to PSPACE-complete problems and their combination with techniques from other fields. In this paper we show how the problem's hybrid nature of being a two-person zero-sum game on the one hand, and being a convex multistage decision problem on the other hand, can be utilized to combine linear programming techniques with techniques from game-tree search. To our best knowledge, a combination of techniques from these two fields has not been done before.

Solutions of feasible QLPs are so called winning strategies for the existential player that specify how to react on moves - certain fixations of universally quantified variables - of the universal player to certainly win the game. However, if there are several winning strategies, one might wish to find a certain (the best) one with respect to some kind of measure. We therefore propose an extension of the QLP decision problem by the addition of a linear objective function, which tries to minimize the objective function with respect to the maximum possible loss that can result from the universal player's possible decisions. To solve the resulting QLP optimization problem, we propose an extension of the Nested Benders Decomposition algorithm as presented in [11]. Solving a QLP with this algorithm can be interpreted as solving a tree of linear programs by passing information among nodes of the tree, and for the special case of QLPs with an objective function, the way these information are passed is similar to the minimax principle as it is known from game-tree search. This allows to integrate the concept of $\alpha\beta$-*cuts* in combination with *move-ordering*, as used in the $\alpha\beta$-Algorithm to solve minimax trees, which are used to represent two-person zero-sum games. We furthermore introduce a new cut based on the solution of

certain kinds of QLP relaxations resulting from quantifier shifts. The applicability of this approach is examined in an experimental evaluation, where we solve a set of QLPs that were generated from the well-known NETLIB test set. This test set contains real world LP instances of practical size, and the resulting QLPs can be interpreted as robust versions of the corresponding LP instances.

The rest of this paper is organized as follows. In Section 2, we formally describe the QLP optimization problem and its lower and upper bound relaxation based on quantifier shifting, followed by an explanation of the Nested Benders Decomposition approach in Section 3. Section 4 introduces the concepts of game-tree search and afterwards, in Section 5, we show how these techniques can be embedded into our existing algorithmic framework. We proceed with an experimental evaluation in Section 6 and end up with a conclusion in Section 7.

2 The Problem Statement: Quantified Linear Programs (QLPs)

Within this paper, we intend to concentrate on quantified linear programs (QLPs), as they were introduced in [25, 23], and in-depth analyzed in [17, 11]. We follow the notation as presented in [17] and briefly review the main results. Afterwards we show how a linear objective function can be incorporated into the concept of Quantifies Linear Programming and present certain kinds of QLP relaxations that result when the variable order is changed.

2.1 QLP Decision Problems

Definition 1 (Quantified Linear Program)**.** *Let there be a vector of n variables $x = (x_1, \ldots, x_n)^T \in \mathbb{Q}^n$, lower and upper bounds $l \in \mathbb{Z}^n$ and $u \in \mathbb{Z}^n$ with $l_i \leq x_i \leq u_i$, a coefficient matrix $A \in \mathbb{Q}^{m \times n}$, a right-hand side vector $b \in \mathbb{Q}^m$ and a vector of quantifiers $\mathcal{Q} = (\mathcal{Q}_1, \ldots, \mathcal{Q}_n)^T \in \{\forall, \exists\}^n$, let the term $\mathcal{Q} \circ x \in [l, u]$ with the component wise binding operator $\circ$ denote the* quantification vector *$(\mathcal{Q}_1 x_1 \in [l_1, u_1], \ldots, \mathcal{Q}_n x_n \in [l_n, u_n])^T$ such that every quantifier $\mathcal{Q}_i$ binds the variable x_i ranging over the interval $[l_i, u_i]$. We call $(\mathcal{Q}, l, u, A, b)$ with*

$$\mathcal{Q} \circ x \in [l, u] : Ax \leq b \tag{QLP}$$

a quantified linear program (QLP)*.*

Note that existential-only quantification results in conventional linear programs: The notation $\exists \circ x \in [l, u] : Ax \leq b$ can be verbalized as "There exist variables in the range which satisfy the given constraints".

We denote the quantification vector $\mathcal{Q} \circ x \in [l, u]$ as quantification sequence $\mathcal{Q}_1 x_1 \in [l_1, u_1] \ \ldots \ \mathcal{Q}_n x_n \in [l_n, u_n]$. In a similar manner, we denote $\mathcal{Q}$ as a quantifier sequence $\mathcal{Q}_1 \ \ldots \ \mathcal{Q}_n$ and x as a variable sequence $x_1 \ \ldots \ x_n$. Each maximal consecutive subsequence of $\mathcal{Q}$ consisting of identical quantifiers is called a *quantifier block* – the corresponding subsequence of x is called a *variable block*. The i-th quantifier block is denoted by $\mathcal{Q}^i \in \{\forall, \exists\}$, and likewise x^i denotes

the corresponding variable block. The total number of blocks less one is the number of *quantifier changes*. Let I be the (ordered) index set of x and let $I_\exists := \{i \in I : \mathcal{Q}_i = \exists\}$ and $I_\forall := \{i \in I : \mathcal{Q}_i = \forall\}$ be (ordered) subsets with respect to a specific quantifier. That is, $x_{I_\exists}$ and $x_{I_\forall}$ denote the (ordered) sets of existentially quantified, and respectively, universally quantified variables of the vector x. For convenience, we generally omit the letter I and write $x_\exists$ for $x_{I_\exists}$ and $x_\forall$ for $x_{I_\forall}$.

We interpret each QLP instance as a two-person zero-sum game between an *existential player* setting the $\exists$-variables and a *universal player* setting the $\forall$-variables. Each fixed vector $x \in [l, u]$, that is, when the existential player has fixed the existential variables and the universal player has fixed the universal variables, is called *a game*. If x satisfies the linear program $Ax \leq b$, we say *the existential player wins*, otherwise *he loses* and *the universal player wins*. The variables are set in consecutive order according to the quantification sequence. Consequently, we say that a player makes the move $x_k = z$, if he fixes the variable x_k to the value z. At each such move, the corresponding player knows the settings of $x_1, \ldots, x_{k-1}$ before taking his decision x_k. In the context of answering the question whether the existential player can certainly win the game, we use the term *policy*.

Definition 2 (Policy)**.** *Given a QLP* $(\mathcal{Q}, l, u, A, b)$ *with* $\mathcal{Q} \circ x \in [l, u] : Ax \leq b$*. An algorithm that fixes all existential variables* x_i *with the knowledge, how* $x_1, \ldots, x_{i-1}$ *have been set before, is called a* policy.

Observe that a policy can be represented as a set of computable functions of the form $x_i = f_i(x_1, \ldots, x_{i-1})$ for all existentially quantified variables x_i. A policy is called a *winning policy* if these functions ensure that the existential player wins all games that can result from this policy, independently of the universal player's moves.

Definition 3 (QLP Decision Problems)**.** *Given a QLP, the decision problem "Is there a winning policy for the existential player?" is called the* QLP Decision Problem.

In this paper we concentrate on QLPs since they are relaxations of QIPs (quantified integer programs with $x \in \mathbb{Z}$), which are known to be PSPACE-complete. It has been shown that the QLP problem with only one quantifier change is either in P (when the quantification begins with existential quantifiers and ends with universal ones) or coNP-complete (when the quantification begins with universal quantifiers and ends with existential ones) [25]. In [17] it was shown that the solution space of a QLP with n variables forms a polytope in $\mathbb{R}^n$, which is included in the polytope induced by the constraint set $Ax \leq b$ as shown in Figure 1. It was furthermore shown that it suffices to inspect the bounds of the universal quantified variables in order to check whether a *winning policy* does exist (cf. [25] and with a completely different proof in [17]).

Example 1. *The QLP*

$$\exists x_1 \in [1, 6]\ \forall x_2 \in [1, 2] : x_1 + x_2 \leq 6\ \wedge\ x_2 - x_1 \leq 0$$

has the following graphical representation of bounding box (dashed lines) and constraints (solid lines). We say a solution *to this problem is a move for the*

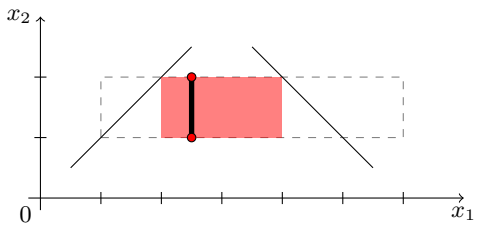


Fig. 1: Polyhedral QLP solution space

existential player such that he wins the game regardless of the universal player's reaction, or rather, the set of games $(x_1, x_2)^T$ *which can result from the existential player's decision (black line segment in the figure below), e.g. the output of a winning policy. The set of all solutions, i.e. the set of 'existential sticks' fitting in the specified trapezoid, is called the* solution space *(filled rectangle). In this example, we see that it indeed suffices to analyze discrete points (filled dots) to find a solution. Note that the order in the quantification sequence is crucial.*

The fact that it suffices to check the bounding values of the universally quantified variables in order to answer the question whether the existential player can certainly win the game, can be exploited in terms of asking whether a *winning strategy* for the existential player does exist.

Definition 4 (Strategy)**.** *A* strategy $S = (V, E, c)$ *is an edge-labeled finite arborescence with a set of nodes* $V = V_\exists \dot\cup V_\forall$*, a set of edges* E *and a vector of edge labels* $c \in \mathbb{Q}^{|E|}$*. Each level of the tree consists either of only nodes from* $V_\exists$ *or only of nodes from* $V_\forall$*, with the root node at level 0 being from* $V_\exists$*. The* i*-th variable of the QLP is represented by the inner nodes at depth* $i-1$*. Each edge connects a node in some level* i *to a node in level* $i+1$*. Outgoing edges represent moves of the player at the current node, the corresponding edge labels encode the variable allocations of the move. Each node* $v_\exists \in V_\exists$ *has exactly one child, and each node* $v_\forall \in V_\forall$ *has as two children, with the edge labels being the corresponding upper lower and upper bounds.*

A path from the root to a leaf represents a game of the QLP and the sequence of edge labels encodes its moves. A strategy is called a *winning strategy* if all paths from the root node to a leaf represent a vector x such that $Ax \leq b$. This terminology is also very similarly used in game-tree search [21].

Example 2. *The QLP*

$$\exists\, x_1 \in [0,1]\, \forall\, x_2 \in [0,1]\, \exists\, x_3 \in [0,1] :$$

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

has two quantifier changes. Figure 2 shows a visualization of the constraint polyhedron restricted to the unit cube. Since this example is rather small, we can guess a winning strategy for the existential player from the picture: 'Choose $x_1 \in [0, \frac{1}{2}]$, then choose x_3 apt to x_2, e.g. $x_3 = 1 - x_2$.' The highlighted solution space visualizes the set of all games with a definite winning outcome for the existential player. Figure 4 shows a winning-strategy for the existential player.

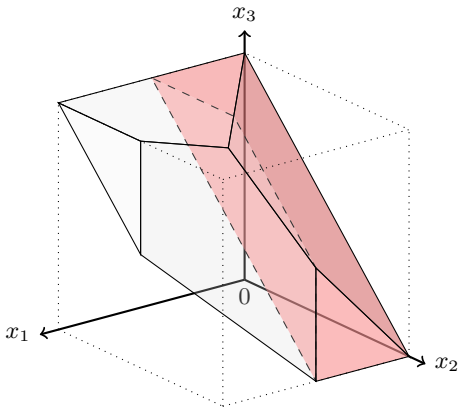


Fig. 2: Solution space of Example 2

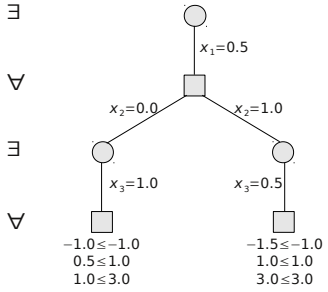


Fig. 3: Winning Strategy Example 2

Remark 1. *According to the results from [17] each QLP can be re-written to the effect that all universally quantified variables are binary. Since we can w.l.o.g. add dummy variables to the problem, we can assume a strictly alternating quantifier sequence with an existentially quantified variable at the start and at the end, when this is needed for simplicity.*

2.2 QLP Optimization Problems

If there is more than one winning strategy for the existential player, it can be reasonable to search for a certain (the 'best') one. We can therefore modify the problem to include a linear objective function as shown in the following (where we note that transposes are suppressed when they are clear from the context to avoid excessive notation).

Definition 5 (QLPs with Objective Function)**.** *Let $\mathcal{Q} \circ x \in [l, u] : Ax \leq b$ be given as in Definition 1 with the variable blocks being denoted by B_i. Let there*

also be a vector of objective coefficients $c \in \mathbb{Q}^n$*. We call*

$$z = \min_{B_1}(c^1x^1 + \max_{B_2}(c^2x^2 + \min_{B_3}(c^3x^3 + \max_{B_4}(\ \ldots \min_{B_m} c^m x^m))))) \\ \mathcal{Q} \circ x \in [l,u] : Ax \leq b \qquad \text{(QLP*)}$$

a QLP with objective function *(for a minimizing existential player (c.f. Remark 1)).*

Note that the variable vectors $x^1, ..., x^i$ are fixed when a player minimizes or maximizes over variable block B_{i+1}. Consequently, it is a dynamic multistage decision process, similar as it is also known from multistage stochastic programming [8]. However, whereas in the latter an expected value is minimized, in our case we try to minimize the possible worst case (maximum loss) scenario that can result from the universal player's decisions.

As we will show in Lemma 1, the above formulation is equivalent to the following formulation

$$\min\{k \,|\, (\exists, \mathcal{Q}) \circ (k, x) \in [l,u] : Ax \leq b \wedge c^T x \leq k\} \qquad \text{(QLP**)}$$

where the objective function is pushed into the constraint system and used to restrict an auxiliary existentially quantified variable $k \in \mathbb{Q}$, which is placed in front of the QLP and which shall be minimized. This means, that similar as it is the case in traditional linear programming, the objective function can be encoded in the constraint system.

Lemma 1. *A strategy S is a winning strategy for QLP* with objective function value z, if an only if S is also a winning strategy for QLP** where $k = z$ holds.*

Proof. Let S_0 be a winning strategy for QLP** with k being minimal. From this follows that for each possible game x^{S_0} of S_0 the inequality $c^T x^{S_0} \leq k$ is satisfied at all leafs of S_0. However, since k is minimal at least one leaf satisfies $c^T x^{S_0} = k$ as depicted in Figure 7. The universal player can chose the path in S_0, but whatever he does, the existential player will certainly win the game and he will achieve an objective function value of at least k. To reach exactly k, the universal player must chose the path that leads to a leaf of S_0 where $c^T x^{S_0} = k$. Obviously, the minimax value of a strategy is equal to the maximum value at the leafs of a min strategy (c.f. [21]), and therefore $k = z$.

Conversely let S_1 be a winning strategy for QLP* and let

$$z = \min_{B_1}(c^1 x^{S_1 1} + \max_{B_2}(c^2 x^{S_1 2} + \min_{B_3}(c^3 x^{S_1 3} + \max_{B_4}(\ \ldots \min_{B_m} c^{S_1} x^m)))))$$

Therefore S_1 consist of a set of games such that for each game x^{S_1} of S_1 $Ax^{S_1} \leq b$ and $c^T x^{S_1} \leq z$ at all leafs of S_1. Because the existential player minimizes the objective function, there must be at least one leaf in S_1 with $c^T x^{S_1} = z$. Therefore z is also minimal for QLP** and $z = k$.

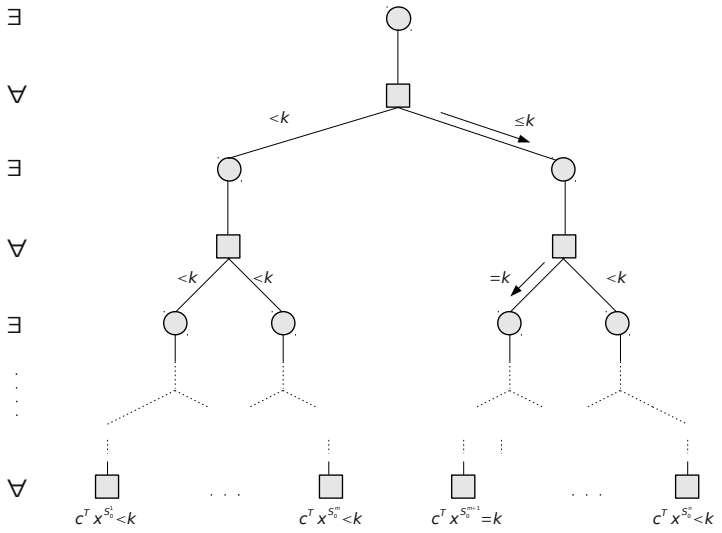


Fig. 4: Winning strategy S

In the following we use the abbreviation $\min c^T x$ for the objective function and denote by

$$\min\{c^T x \mid \mathcal{Q} \circ x \in [l, u] : Ax \leq b\}$$

a *quantified linear program with objective function.*

Definition 6 (QLP Optimization Problems)**.** *Given a QLP with objective function, the problem "Is it feasible? If yes, what is the best objective value of the existential player's winning strategies?" is called* QLP Optimization Problem.

Note, that to find the existential player's optimal objective value, it is not enough to fix the universal players variables to their worst-case values regarding the objective function. For clarification, consider the following program:

$$\min\{-x_1 - 2x_2 \mid \forall\, x_1 \in [0,1]\ \exists\, x_2 \in [0,1] : x_1 + x_2 \leq 1\}$$

Judging from the objective function the universal player should fix $x_1 = 0$, but this results in a better objective value for the existential player than forcing the existential player to fix $x_2 = 0$ in the constraint system. However, it still suffices to inspect only the bounding values of the universally quantified variables (c.f. Lemma 2).

Lemma 2. *For each universally quantified variable x_i in QLP**, it suffices to inspect the bounds l_i and u_i. Moves of the universal player in the range $l_i < x_i < u_i$ do not increase k.*

Proof. Let S be a winning strategy for

$$\mathcal{Q} \circ x \in [l, u] : Ax \leq b \wedge c^T x \leq k.$$

Let w.l.o.g. all universally quantified variables be binary. Let furthermore k be minimal for the QLP and let $c^T x = k$ in exactly one leaf of S as depicted in Figure 5.

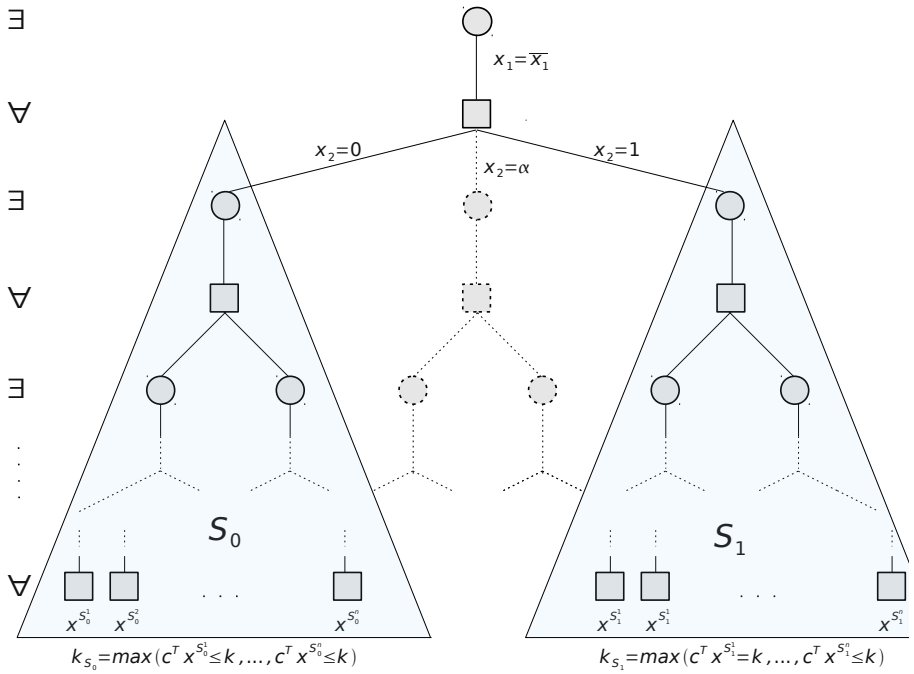


Fig. 5: Winning strategy S with sub strategies S_0 and S_1

After the first move of the existential player where he fixes the first variable to $x_0 = \overline{x}_0$, the winning strategy S splits up into two sub strategies S_0 and S_1. These are winning strategies for the remaining moves of the existential player when the universal player fixes the next universally quantified variable x_2 to the lower bound $x_2 = 0$ or to the upper bound $x_2 = 1$. Let k_{S_0} be the corresponding maximal value $c^T x \leq k$ that can result at the leafs from all possible games $x^{S_0} = (\overline{x}_0, 0, x_2, \ldots, x_n)$ of S_0 and k_{S_1} be the same value with respect to S_1 as depicted in Figure 5. Then it holds that $max\{k_{S_0}, k_{S_1}\} = k$. If the universal player is allowed to chose an $x_i = \alpha$ with $0 \leq \alpha \leq 1$ (e.g. $x_2 = \alpha$ as in Figure 5), we can construct a winning strategy S_α from S_0 and S_1 for any α and it holds that for all possible resulting games x^{S_α}, the corresponding values $c^T(\overline{x}_0, x_{S_\alpha})$ are less than k. To compute S_α, the existential player can compute

x^α for any possible move sequence of the universal player by setting variable $x_i^\alpha = \alpha x_i^{S_0} + (1-\alpha)x_i^{s_1}$. Then, for all possible games x^α in S_α holds:

$$A\overline{x}^\alpha = \alpha(Ax^0) + (1-\alpha)(Ax^1) \leq \alpha b + (1-\alpha b) = b$$
$$c^T\overline{x}^\alpha = \alpha(c^Tx^0) + (1-\alpha)(c^Tx^1) = \alpha k_0 + (1-\alpha)k_1 < k.$$

Lemma 2 follows using induction over the number of universally quantified variables.

2.3 QLP Relaxations by Quantifier Shifting

The order of the elements in the quantification vector $\mathcal{Q} \circ x \in [l,u]$ is crucial, because it determines the sequence in which the variables must be fixed by the two players. However, changing the order in the quantification sequence can yield interesting information about the original problem. In this context, the two special cases when the universal quantified variables are shifted to the end of the quantification sequence, and respectively, when they are shifted to the front are of high interest.

Given a general quantified linear program

$$z = \min\{c^Tx \mid \mathcal{Q} \circ x \in [l,u] : Ax \leq b\} \qquad (\text{QLP}_{\exists\forall\ldots\forall\exists})$$

with $\mathcal{Q}$ consisting of m quantifier blocks $\mathcal{Q} = (\exists^1, \forall^2, \exists^3, \ldots, \forall^{m-1}, \exists^m)$ and the first and last block consisting of existentially quantified variables (c.f. Remark 1). Accordingly the variable vector x consists of m variable blocks $x = (x^1, x^2, x^3, \ldots, x^{m-1}, x^m)$.

When all universally quantified variables are shifted to the front of the quantification vector

$$z_{\forall\exists} = \min\{c^Tx \mid \mathcal{Q}_{\forall\exists} \circ (x_\forall, x_\exists) \in [l,u] : Ax \leq b\} \qquad (\text{QLP}_{\forall\exists})$$

with $\mathcal{Q}_{\forall\exists} = (\forall^2, \ldots, \forall^{m-1}, \exists^1, \ldots, \exists^m)$ and $x_\exists = (x^1, x^3, \ldots, x^m)$ and $x_\forall = (x^2, \ldots, x^{m-1})$, then the universal player must fix all universally quantified variables $x_\forall$ before the existential player must fix the existentially quantified variables $x_\exists$.

Accordingly, in the case when all universal quantifiers are shifted to the end of the quantification vector

$$z_{\exists\forall} = \min\{c^Tx \mid \mathcal{Q}_{\exists\forall} \circ (x_\exists, x_\forall) \in [l,u] : Ax \leq b\} \qquad (\text{QLP}_{\exists\forall})$$

with $\mathcal{Q}_{\exists\forall} = (\exists^1, \ldots, \exists^m, \forall^2, \ldots, \forall^{m-1})$ and $x_\exists = (x^1, x^3, \ldots, x^m)$ and $x_\forall = (x^2, \ldots, x^{m-1})$, then the existential player must fix all existentially quantified variables $x_\exists$ before the universal player must fix all universally quantified variables $x_\forall$.

This means, in the case of $\text{QLP}_{\forall\exists}$ it suffices to find for any possible fixation $\overline{x}_\forall$ of the universally quantified variables, a fixation $\overline{x}_\exists$ of the existentially quantified variables such that $Ax \leq b$. In the case of $\text{QLP}_{\exists\forall}$ in contrast, we must find

one single fixation $\overline{x}_\exists$ of the existentially quantified variables such that $Ax \leq b$ holds *for all* possible fixations $\overline{x}_\forall$ of the universally quantified variables. Judging from the viewpoint of game playing, the latter is of course much harder to achieve for the existential player.

While the complexity of $\text{QLP}_{\exists\forall\ldots\forall\exists}$ is unknown in general, it was shown that $\text{QLP}_{\forall\exists}$ is in P and that $\text{QLP}_{\forall\exists}$ is coNP-complete [25]. Furthermore, the following relationships with respect to the feasibility and infeasibility of $\text{QLP}_{\exists\forall\ldots\forall\exists}$, $\text{QLP}_{\exists\forall}$, and $\text{QLP}_{\forall\exists}$ hold:

1. If $\text{QLP}_{\forall\exists}$ is infeasible for any fixed vector of universally quantified variables $\overline{x}_\forall$, then also $\text{QLP}_{\exists\forall\ldots\forall\exists}$ is infeasible and $\overline{x}_\forall$ is a *certificate of infeasibility*.
2. If $\text{QLP}_{\forall\exists}$ is feasible for every fixed vector of universally quantified variables $\overline{x}_\forall$ and $z^*_{\forall\exists}$ is the optimal value of the worst-case scenario, then for any feasible solution of $\text{QLP}_{\exists\forall\ldots\forall\exists}$ with objective function value z holds, that $z^*_{\forall\exists} \leq z$, and thus, $z^*_{\forall\exists}$ is a also lower bound for the optimal solution z^* of $\text{QLP}_{\exists\forall\ldots\forall\exists}$.
3. If $\text{QLP}_{\exists\forall}$ is feasible with solution $\overline{x}_\exists$, then also $\text{QLP}_{\exists\forall\ldots\forall\exists}$ is feasible, and $\overline{x}_\exists$ is a *certificate of feasibility*.
4. If $\text{QLP}_{\exists\forall}$ is feasible with $z^*_{\exists\forall}$ being the corresponding optimal objective function value, then for any feasible solution of $\text{QLP}_{\exists\forall\ldots\forall\exists}$ with objective function value z holds, that $z \leq z^*_{\exists\forall}$, and thus, $z^*_{\exists\forall}$ is also an upper bound for the optimal solution z^* of $\text{QLP}_{\exists\forall\ldots\forall\exists}$

The correctness of these relationships are easy to see and do not need an formal proof, it suffices to take the following into account:

1. If there is any move sequence $\overline{x}_\forall$ for the universal player in $\text{QLP}_{\forall\exists}$, such that $A(\overline{x}_\forall, x_\exists) \leq b$ is not feasible for any $x_\exists$, then the same move sequence will also lead to infeasibility of $\text{QLP}_{\exists\forall\ldots\forall\exists}$ (Note that the inversion of the argument is not correct).
2. If $\text{QLP}_{\forall\exists}$ is feasible with $z^*_{\forall\exists}$ being the optimal value of the worst-case scenario, then the optimal solution z^* of $\text{QLP}_{\exists\forall\ldots\forall\exists}$ can not get smaller than $z^*_{\forall\exists}$ because the corresponding worst-case scenario is also part of $\text{QLP}_{\exists\forall\ldots\forall\exists}$. Apart from that, z^* can become a sight worse than $z^*_{\forall\exists}$ because in $\text{QLP}_{\exists\forall\ldots\forall\exists}$ some variable fixations of the existential player most hold for more than one scenario, whereas for $\text{QLP}_{\forall\exists}$ the opposite is the case.
3. $\text{QLP}_{\exists\forall}$ is solved with the help of the gaming argument as presented in [25]. Therefore, the constraint system $Ax \leq b$ is converted into a modified constraint system where the universal player has replaced all variables x_i from the set $x_\forall$ by the worst case for the existential player between l_i and u_i *inequality by inequality* and also in the objective function. The modified constraint system does not longer contain any universally quantified variables and can be solved with standard linear programming algorithms to obtain the solution $\overline{x}_\exists$. This solution is also a solution for $\text{QLP}_{\exists\forall\ldots\forall\exists}$ because all variables x_i from the set $x_\forall$ must still be chosen between l_i and u_i, however

not worst case for the existential player inequality by inequality but rather with one single fixation for all constraints in $Ax \leq b$ for each variable x from the set $x_\forall$.

4. The optimal solution z^* of $\text{QLP}_{\exists\forall\ldots\forall\exists}$ cannot exceed $z^*_{\exists\forall}$ because all const raints of $\text{QLP}_{\exists\forall}$ are also part of $\text{QLP}_{\exists\forall\ldots\forall\exists}$. However, because the universal player may not choose his variables worst case for the existential player inequality by inequality but rather with one single fixation for each variable x_i from the set $x_\forall$ for all constraints in $Ax \leq b$, some of the constraints in $\text{QLP}_{\exists\forall\ldots\forall\exists}$ get weakened with respect to their counterparts in $\text{QLP}_{\exists\forall}$.

In Section 5 we describe how these information can be exploited in the solution process of the Nested Benders Decomposition algorithm.

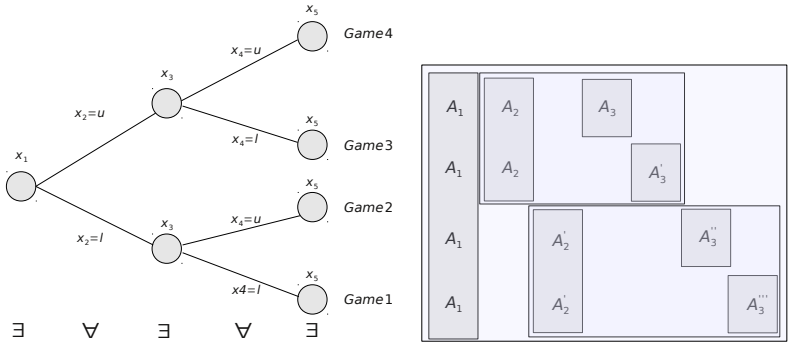
3 Nested Benders Decomposition (NBD)

The first algorithm to solve QLP decisions problems was proposed by Subramani [25] and relies on variable elimination techniques. Existentially quantified variables $x_\exists$ are eliminated using the well known Fourier-Motzkin elimination method, universally quantified variables $x_\forall$ are eliminated using a clever gaming argument. However, the algorithm has double-exponential runtime, because at each elimination step of an existentially quantified variable the number inequalities can grow from m to $(\frac{m}{2})^2$. Another approach based on decomposition techniques was proposed in[11] and tested in a detailed computational study. This algorithm is the basis for the algorithmic extension proposed in this paper.

The algorithm uses decomposition techniques to solve an implicit reformulation of a QLP, which we call a *deterministic equivalent problem* (DEP), and that does not longer contain universally quantified variables. The concept is similar to the notion of deterministic equivalence as it is known from stochastic programming (cf. [8]) in the context of multistage stochastic linear programs (MSSLPs). There, using the assumption of a finite time horizon and a discrete probability space, the resulting scenario tree is encoded into a DEP by replicating the LP for each possible scenario (possible path of events). Additionally, it is required that decisions must not depend on future events (*nonanticipativity*).

The DEP of a QLP instance can be constructed in a similar way, however, instead of encoding the scenario tree of randomly arising scenarios, we encode the decision tree of the universal player, which results from the series of all possible upper and lower bound combinations of the variables $x_\forall$ as determined by the quantification sequence $\mathcal{Q}_\forall \circ x_\forall \in [l_\forall, u_\forall]$. Nodes at stage t are decision points where the existential player has to fix variables, e.g. by solving a linear program, with respect to all previous moves $(x^1, \ldots, x^{t-1})$. Arcs of the tree represent moves of the universal player when he fixes his variables to the corresponding lower and upper bounds.

Figure 6 a) shows the universal player's decision tree for a QLP with alternating quantification sequence $\exists\, x_1 \in [l_1, u_1]\ \forall\, x_2 \in [l_2, u_2]\ \exists\, x_3 \in [l_3, u_3]\ \forall\, x_4 \in [l_4, u_4]\ \exists\, x_5 \in [l_5, u_5]$. The tree is similar to the strategy of a QLP where the moves of the existential player have not been fixed.



a) Universal Player Decision Tree b) DEP Matrix Structure

Fig. 6: Deterministic Equivalent Linear Program

Figure 6 b) shows the resulting DEP matrix structure using compact variable formulation, which implicitly satisfies the nonanticipativity property because all nodes in the three that share a common history also have the same set of decision variables up to that point. The resulting DEP grows exponentially with the number of universally quantified variables of the corresponding QLP, but the special block structure of the matrix can be exploited by the Nested Benders Decomposition (NBD) algorithm. The NBD algorithm is a recursive application of the well-known Benders Decomposition principle [6] and is widely used in the Stochastic Programming community to solve MSSLPs [22, 7].

To illustrate how the Benders Decomposition algorithm works to solve QLPs, we consider w.l.o.g. the DEP that results from a QLP with quantification sequence $\exists x_1 \in [0, u_1] \forall x_2 \in [1, 1] \exists x_3 \in [0, u_3]$ and an objective function $\min c_1^T x_1 + c_3^T x_3$. Let the constraint system $A_1 x_1 + A_2 x_2 + A_3 x_3 \leq b$ contain the upper bound u_1 of x_1 and u_3 of x_3. Since the $\forall$-variable $x_2 \in [1, 1]$ is a fixed variable, the DEP consists of a single game where the corresponding right hand side b' results from $b' = b - A_2 \overline{x}_2$ with $\overline{x}_2 = 1$. The resulting DEP looks as follows:

$$
\begin{aligned}
Z = \min\ & c_1^T x_1 + c_3^T x_3 \\
s.t.\quad & A_1 x_1 + A_3 x_3 \leq b' \\
& x_1 \geq 0, x_3 \geq 0
\end{aligned}
$$

Applying Benders Decomposition, the decision variables of the DEP are stage-wise partitioned and then decomposed into a *restricted master problem (RMP)* that contains the first-stage variable x_1, and one *subproblem (SP)* that contains the second-stage variable x_3. The corresponding *dual SP (DSP)* has the property that the solution space does not longer depend on the value of x_1, regardless whether it is feasible or infeasible for the SP. The SP and its DSP can be written as follows:

$$
\begin{array}{lll}
SP(x_1) = \min c_3^T x_3 & & DSP(x_1) = \max \pi^T (b' - A_1 x_1) \\
s.t. & A_3 x_3 \leq b' - A_1 x_1 & s.t. \quad A_3^T \pi \leq c_3 \\
& x_3 \geq 0 & \pi \leq 0
\end{array}
$$

For a non-optimal $\overline{x}_1$ obtained by the solution of the RMP, which can be empty at the beginning, the following two cases can happen. It the SP is feasible, the solution of the DSP is bounded and located at an extreme point of its solution space. If the SP is infeasible, the solution of the DSP is unbounded, which corresponds to an extreme ray of its solution space. Using these dual information, two different types of cutting planes - called *Benders cuts* - can be added to the RMP to cutoff the last $\overline{x}_1$ in the next solution of the RMP.

1. *feasibility cut*: $(\pi_r^j)(b' - A_1 x_1) \leq 0$, if the DSP($\overline{x}_1$) is unbounded, where π_r^j is the vector that corresponds to the extreme ray j.
2. *optimality cut*: $(\pi_p^i)(b' - A_1 x_1) \leq q$, if the DSP($\overline{x}_1$) is bounded, where π_p^i is the vector that corresponds to the extreme point i.

Since the DSP can only have finitely many extreme points and extreme rays, the RMP can be written as follows, where q is an auxiliary variable used to approximate the objective function value of the SP:

$$
\begin{array}{ll}
RMP = \min c_1^T x_1 + q & \\
s.t. & (\pi_r^j)(d - A_1 x_1) \leq 0 \ \forall j \in J \\
& (\pi_p^i)(d - A_1 x_1) \leq q \ \forall i \in I \\
& x_1 \geq 0
\end{array}
$$

This reformulation is equivalent to the initial DEP, however, there can be exponentially many extreme rays and extreme points and not all of them are needed to find the optimal solution. Therefore, the algorithm starts with I and J being empty, and computes cuts in an iterative process until an optimal solution is found or infeasibility is detected. In the latter case also the DEP and the corresponding QLP are infeasible. The optimal solution is found, if for a given candidate optimal solution (x_1^*, q^*), called proposal, also the SP(x_1^*) has an optimal solution with value $q(x_1^*)$ and the optimality condition $q(x_1^*) = q^*$ is satisfied. If this is the case, the algorithm stops. Otherwise a feasibility or optimality cut is added to the RMP, which is then re-solved again to obtain a new proposal. In each iteration where the SP is feasible, $c^T x_1^* + q^*$ yields a lower bound for the initial problem, while $c^T x_1^* + q(x_1^*)$ yields an upper bound. The difference between these bounds gets smaller, and if it becomes less than a predefined ϵ, the algorithm also terminates.

If there are k universally quantified variables, then there are 2^k games and therefore 2^k subproblems are solved in each iteration, each yielding a cut that is added to the RMP. The *min-max* property of the objective function is achieved, because all optimality cuts that result from the subproblems restrict the same

auxiliary variable q. For the computation of the upper bound, the maximum over all subproblems from the last iteration is used. For multistage QLPs resulting from a quantifier string $\exists\forall\exists\forall...\forall\exists$, Benders Decomposition can be recursively applied, which is known as Nested Benders Decomposition. Solving the DEP of a multistage QLP can be illustrated as solving a tree of linear programs that are attached to the nodes of the decision tree of the universal player. The tree is traversed forwards and backwards multiple times, with information being passed between adjoined nodes of the tree. A node at stage t passes proposals for the variables from the root up to stage t to its immediate descendants at stage $t+1$ and cuts to its immediate ancestor at stage $t-1$.

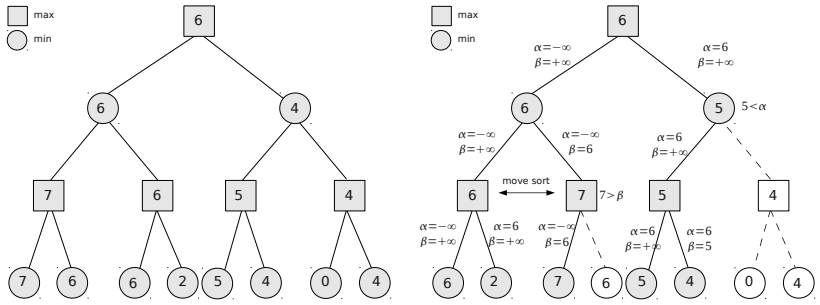
The algorithm has been implemented and tested in a detailed computational study with instances that were generated from existing LP and IP test sets [11].

4 Game-Tree-Search and the $\alpha\beta$-Algorithm

The term, *minimax tree* describes one of the most important data structures that allows computers to play two-person zero-sum games such as tic-tac-toe, checkers, chess, go, etc. Nodes of the tree are decision points for the players and are therefore subdivided in min and max nodes. Nodes from different stages are connected with branches, leaf nodes are end positions of the game and can be evaluated as a win, lose, or draw using the rules of the game. Often, a specific score from the max player's point of view is computed with the help of a weighting function and assigned to a leaf to represent how good or bad the sequence of moves from the root to the leaf is. With a complete game-tree, it is possible to solve the game with the *MiniMax*-Algorithm, which therefore fills the inner node values of the tree bottom-up starting with the evaluated values at the leafs. Nodes that belong to the max player get the maximum values of their successors, while nodes for the min player get the minimum. Figure 7a) illustrates this behavior.

While the MiniMax-Algorithm must evaluate the entire game-tree to compute the root value, the *$\alpha\beta$-Algorithm* [14, 1, 21] prunes away branches that cannot influence the final result. Therefore, it maintains two values, α and β, which represent the minimum score that the max player is sure to gain at least until that point in the tree, and the maximum score of the min player respectively. If the evaluation of a position where the min player has to move becomes less than α, the move needs not to be further explored, since a better move has already been found. The same holds, if at a position where the max player has to choose his move, the evaluation provides a value that is greater than β. Figure 7b) illustrates this behavior, the dashed subtrees were not visited. The left one due to a *β-cutoff*, the subtree on the right hand due to an *α-cutoff*.

Whereas the order in which the nodes of the tree are evaluated does not care for the MiniMax-Algorithm, it is of particular importance for the performance of the $\alpha\beta$-Algorithm. The *best moves* need to be evaluated first in order to find strong α and β values as soon as possible. Figure 7b) illustrates this, without swapping the subtrees under the first successor of the root on the left side, the β-



a) MiniMax Game-Tree b) $\alpha\beta$-Algorithm

Fig. 7: MiniMax Game-Tree and $\alpha\beta$-Algorithm

cutoff would not have occurred. If the best moves are searched first, the runtime of the $\alpha\beta$-Algorithm is only $O(\sqrt{b^d})$ where d is the depth of the tree and b is the number of possible moves at each node. The MiniMax-Algorithm has a runtime of $O(b^d)$.

5 The $\alpha\beta$-Nested Benders Decomposition ($\alpha\beta$-NBD)

Solving a multistage QLP with the NBD algorithm can be illustrated as solving a tree of linear programs that are attached to the nodes of the universal player's decision tree. The tree is traversed multiple times and information in form of proposals and cuts are passed between nodes of the tree.

If a node v_i at stage $t \in \{0, \ldots, T\}$ receives a new proposal $\overline{x}^{t-1}$ from its parent at stage $t-1$, the subtree rooted at v_i is solved to optimality, or until the nodal linear program attached to v_i becomes infeasible. After feasibility of the subtree is established, the upper and lower bounds of v_i converge at each iteration and for the optimal objective function value z_i holds $L_i^z \leq z_i \leq U_i^z$ until the values coincide at the end, and v_i passes $\overline{z}_i$ and the corresponding optimality cut to its parent at stage $t-1$. After an iteration where node v_i passed its current proposal $\overline{x}^t$ to its direct successors $v_j \in J$ at stage $t+1$, and all of them were feasible with $\overline{z}_j$ denoting the corresponding optimal objective function values, then v_i's upper bound computes as $U_i^z = c^T\overline{x}^t + \max_{j\in J}\overline{z}_j$. This is equal to the minimax principle, the existential player tries to *minimize* the nodal lp with respect to the worst-case move of the universal player, which is the subproblem with the *maximal* objective function value.

The current maximal value $\alpha_i = \max_{k\in K}\overline{z}_k$ of some successors $v_k \in K \subset J$ of node v_i can be used in a similar manner as α in the $\alpha\beta$-Algorithm, since it denotes the minimum value, the maximizing player (the universal one) will at least obtain at node v_i. α_i can be passed to the remaining nodes from the set $L = J\backslash K$ and each node v_l can stop computing its exact optimal objective function value after it determines feasibility and its upper bound U_l^z becomes

less than or equal to α_i. Nevertheless, if a proposal $\overline{x}^t$ and an α_i is passed from a node v_i to one of its direct successors $v_k \in K$, the entire subtree rooted at v_k must at least be traversed once to determine feasibility and thus to obtain a valid upper bound U_k^z. However, we can first solve the relaxation $\text{QLP}_{\exists\forall}$ where all universally quantified variables are shifted to the end of the quantification sequence and the variables are fixed accordingly to the current moves. If it is feasible with optimal objective function value $\overline{z}_{\exists\forall k}$ also the subtree rotted at v_k is feasible and $\overline{z}_{\exists\forall k}$ is an upper bound for z_k. It holds $L_k^z \leq z_k \leq \overline{z}_{\exists\forall k}$ and if $\overline{z}_{\exists\forall k}$ is less than or smaller than α_i the subtree rooted at node v_k needs not to be inspected, because one of v_k's siblings already passed back a worse solution.

We can also integrate a value analogously to β that depicts the maximum value the minimizing player will gain for sure at a specific node v_i at stage t. The value $\beta_i = U_i^z - c^{Tt}\overline{x}^t$ results from the upper bound from the last iteration minus the objective function part of the nodal lp after cuts have been added and the lp was resolved. β_i can be passed to all successors $v_j \in J$ at stage $t+1$ together with the new proposal $\overline{x}^t$. If a node v_j is feasible with respect to the current proposal $\overline{x}^t$ and its lower bound L_j^z is greater than or equal to β_i, it can stop computing its exact optimal objective function value because a better solution has already been found in the previous iteration. The lower bound $L_j^z = c^T\overline{x}^{t+1} + \overline{q}^{t+1}$ results from the objective function part $c^T\overline{x}^{t+1}$ of the nodal lp at node v_j, $\overline{q}^{t+1}$ is an approximation variable for the objective function part of the remaining stages.

Algorithm 1 illustrates the recursive application of the $\alpha\beta$-NBD algorithm in C-like style. The parameters of the function $\mathcal{Q}_j^t(v_j, \overline{x}_i^{t-1}, b^t(\omega_j), \alpha_i^{t-1}, \beta_i^{t-1})$ are a node v_j of the universal player decision tree at stage t, the set of proposals $\overline{x}^{t-1}$ of all existentially quantified variables from stage 0 to $t-1$ and the vector $b^t(\omega_j)$. The latter results from the original vector b^t of stage t after the universally quantified variables fixations of the universal player that correspond to node v_j are substituted into the constraint system and carried to the right-hand-side. We call the set of moves of the universal player that correspond to a node v_j a scenario and denote it by ω_j. At the root node the algorithm is initialized with an empty x-vector, the original right-hand-side b^0, α and β are set to negative infinity and positive infinity. Substituting the proposal $\overline{x}_i^{t-1}$ into the nodal lp of node v_j yields the restricted master problem of v_j, which is denoted by RMP_j^t. If v_j is a leaf node, the RMP_j^t is solved and depending on whether it is infeasible or feasible, the corresponding dual extreme ray $\overline{\pi}_{r,j}^t$ is passed back to its ancestor v_i at stage $t-1$, or the optimal objective function value $c^{Tt}\overline{x}_j^t + \overline{q}_j^t$ and the corresponding dual extreme point $\overline{\pi}_{p,j}^t$ respectively, as depicted in lines 2-6. If v_j is an inner node the relaxation $\text{QLP}_{\exists\forall}$, where the variables that correspond to the current moves of both player are accordingly fixed, is solved as shown in lines 7-10. If the relaxation is feasible and the optimal objective function value $z_{\exists\forall}$ is less then or equal to α_i^{t-1}, the algorithm directly passes back $z_{\exists\forall}$ and the corresponding dual extreme point $\overline{\pi}_{p,j}^t$. Otherwise, further nodes in the subtree must be visited as shown in lines 11-31. Therefore the RMP_j^t is solved and if it is infeasible, the corresponding dual extreme ray $\overline{\pi}_{r,j}^t$ is passed back to node v_i. If it is feasible, the solution yields a new lower bound L_j^z and a proposal $\overline{x}_j^t$ that

is passed to all direct successors $v_k \in K$ of node v_j as shown in lines 15-24. If a subproblem at a node v_k is infeasible, the corresponding dual extreme ray ($\overline{\pi}^{t+1}_{r,k}$) is used to compute a feasibility cut, otherwise the corresponding dual extreme point $\overline{\pi}^{t+1}_{p,k}$ is used to compute an optimality cut and α_i is updated. The cuts are added to RMP^t_j to cut off the current proposal $\overline{x}^t_j$ in the next iteration.

Algorithm 1: Function $\mathcal{Q}^t_j(v_j, \overline{x}^{t-1}_i, b^t(\omega_j), \alpha^{t-1}_i, \beta^{t-1}_i)$

1 $L^z_j = -\infty$, $U^z_j = +\infty, \alpha_j = -\infty$, $\beta_j = +\infty$;
2 **if** *v_j is leaf node* **then**
3 | Solve $\text{RMP}^t_j = min\{c^{Tt}x^t_j + q^t_j | A^t x^t_j \leq b^t(\omega_j) - T^{t-1}\overline{x}^{t-1}_i, x^t_j \geq 0\}$;
4 | **if** *RMP^t_j is infeasible* **then return** $\overline{\pi}^t_{r,j}$;
5 | **return** $c^{Tt}\overline{x}^t_j + \overline{q}^t_j$ and $\overline{\pi}^t_{p,j}$;
6 **end**
7 **if** *v_j is an inner node* **then**
8 | Solve the relaxation $z_{\exists\forall} = \text{QLP}_{\exists\forall}$ with $\overline{x}^{t-1}$ being fixed in $\text{QLP}_{\exists\forall}$;
9 | **if** *$QLP_{\exists\forall}$ is feasible and $z_{\exists\forall} \leq \alpha^{t-1}_i$* **then return** $z_{\exists\forall}$; `// deep` α`-cutoff`
10 **end**
11 **while** $|U^z_j - L^z_j| \geq \epsilon$ **do**
12 | Solve $\text{RMP}^t_j = min\{c^{Tt}x^t_j + q^t_j | A^t x^t_j \leq b^t(\omega_j) - T^{t-1}\overline{x}^{t-1}_i, x^t_j \geq 0\}$;
13 | **if** *RMP^t_j is infeasible* **then return** $\overline{\pi}^t_{r,j}$; `// infeasible master problem`
14 | $L^z_j = c^{Tt}\overline{x}^t_j + \overline{q}^t_j$; `// update lower bound`
15 | **forall the** *nodes $v_k \in K$ from the set of direct successors of node v_j* **do**
16 | | Solve $\text{SP}_k(\overline{x}^t_j) = \mathcal{Q}^{t+1}_k(v_k, \overline{x}^t_j, b^{t+1}(\omega_k), \alpha^t_j, \beta^t_j)$;
17 | | **if** *$SP_k(\overline{x}^t_j)$ is infeasible* **then**
18 | | | add feasibility cut: $(\overline{\pi}^{t+1}_{r,k})^T(b^{t+1}(\omega_k) - T^t x^t) \leq 0$ to RMP^t_j;
19 | | **end**
20 | | **if** *$SP_k(\overline{x}^t_j)$ is feasible with objective function value $q(\overline{x}^t)$* **then**
21 | | | add optimality cut: $(\overline{\pi}^{t+1}_{p,k})^T(b^{t+1}(\omega_k) - T^t x^t) \leq q^t$ to RMP^t_j;
22 | | | **if** $q(\overline{x}^t) > \alpha^{t-1}_i$ **then** $\alpha^t_j = q(\overline{x}^t)$
23 | | **end**
24 | **end**
25 | **if** *$SP_k(\overline{x}^t_j)$ is feasible for all $v_k \in K$* **then**
26 | | $U^z_j = c^t\overline{x}^t_j + max_{k\in K}\left[\mathcal{Q}^{t+1}(\overline{x}^t_j, b^{t+1}(\omega), \alpha^t_j, \beta^t_j)\right]$;
27 | | **if** $U^z_j \leq \alpha^{t-1}_i$ **then return** U^z_j and $\overline{\pi}_{p,j}$; `//` α`-cutoff`
28 | | **if** $L^z_j \geq \beta^{t-1}_i$ **then return** U^z_j and $\overline{\pi}_{p,j}$; `//` β`-cutoff`
29 | | $\beta^t_j = \text{U}^z_j - c^{Tt}\overline{x}^t_j$
30 | **end**
31 **end**
32 **return** $c^{Tt}\overline{x}^t_j + \overline{q}^t_j$ and $\overline{\pi}_{p,j}$;

If all nodes $v_k \in K$ were solved with respect to the current proposal and none of them was infeasible, v_j's upper bound is updated and the algorithms checks whether an α-cutoff or a β-cutoff occurred. In this case the current upper

bound U_j^z and the corresponding dual extreme point $\overline{\pi}_{p,j}^t$ of RMP_j^t is passed back. Otherwise β_j^t is updated and while the difference of the bounds does not exceed an predefined ϵ, the algorithm proceeds with the next iteration in line 13 by resolving RMP_j^t.

As in the case of the $\alpha\beta$-Algorithm, the order in which the nodes of the tree are solved is also an important issue in the $\alpha\beta$-NBD algorithm. Whereas the $\alpha\beta$-Algorithm uses heuristics, our algorithm organizes the order in which nodes are visited based on information from previous iterations. To obtain strong bounds as soon as possible, the successor of a node v_i that provided the worst-case sub solution in the previous iteration, is visited first in the next iteration, speculating that it will again provide a strong α-bound. However, many other sorting criterions are possible.

Since any NBD implementation has to deal with a number of details, which can substantially effect the overall performance, we briefly sketch the most important design decisions of our implementation. Before a subproblem is solved, it has to be loaded into the LP solver. As resolving is usually much faster than solving it from scratch, having one LP for each node of the tree might lower the computational time required. However, due to memory constraints, this is not possible for larger problems. We therefore decided to share LPs for all nodes at the same stage of the tree, since those problems only differ in their right-hand sides, apart from cuts being generated during the solution process.

To have a good warm-start behavior we store for each node all cuts from previous iterations and the last base, which is then used as starting base when the node is revisited in a later iteration. To benefit from the proposed game-tree techniques and because it is the most successful approach to traverse a game-tree, we decided to apply a depth-first-search to traverse instead of a breadth-first-search, which is more commonly used in the stochastic programming community. If a particular problem is feasible, it is an arbitrary decision in which direction to move, an optimality cut can be passed up the tree, or the new proposal down the tree. We use the *fast-forward* method [7, 12], where new proposals are generated as long as possible and cuts are only passed back, when an optimal solution for the nodal linear program is found or the problem becomes infeasible.

When solving a MSSLP, there are two common ways to create optimality cuts. One way is summing up the weighted dual variables for each subproblem forming one optimality cut, whereas the other way is to disaggregate optimality cuts by placing one cut for each subproblem in the corresponding master. The latter is called *multicut* and was suggested by Birge and Louveaux [9]. We use a hybrid scheme and only add the optimality cut of the current worst-case sub-solution to the master. Tests showed that this is clearly faster than adding all optimality cuts in an iteration because the nodal LPs stay small. Furthermore, before each new cut is added, we apply a check that detects if the cut is redundant or if it makes an existing cut redundant itself. To avoid iterations with slow progress at the beginning, we use an advanced start procedure to get a good proposal for the first iteration of the $\alpha\beta$-NBD algorithm by solving the QLP relaxation $\text{QLP}_{\forall\exists}$, where all scenario interdependencies are dropped.

6 Computational Results

In the following we present the results of our experimental evaluation. All tests were run on a Intel i7-3820 CPU with 3.60GHz and 64GB RAM, the algorithmic framework has been implemented in C++ using the LP Solver CPLEX 12.5 to solve nodal linear programs. Since yet no real-world QLP instances do exist in the literature, we decided to use existing LP instances from the well-known NETLIB Library [1] and convert them to QLPs. In the absence of real-world problems, this approach seems obvious since many NETLIB instances were used as basis in many computational studies in the context of Stochastic Programming[12, 10, 7] where they were converted to MSSLPs. Also in Robust Optimization they were studied [4, 5], and in the latter it was shown that even the feasibility properties can be severely affected by only small perturbations of the data. Randomly adding blocks of universally quantified variables as described in [11] can therefore be interpreted as inserting some kinds of uncertainties, that are controlled by an adversary.

For our tests we took LP instances with a maximum number of 500 variables and constraints and generated QLPs with 10, 15, and 18 universally quantified variables. For each new universally quantified variable $x_i \in [0, 1]$, we randomly added matrix coefficients from the interval $[-1, 1]$ with a density of 25%. We furthermore varied the number of $\forall$-quantifier blocks to 1, 2 and 5 and distributed them equally in the QLP. This results in twostage and multistage QLP instances and nine different test sets. Since our aim in this paper is to show the efficiency of the $\alpha\beta$-heuristic combined with move-ordering to find optimal worst-case solutions, we think that it makes sense to require QLP instances for these tests to be feasible, leading to the decision to drop infeasible QLP instances from the experimental tests. However, for each of the nine test sets, we were able to generate twelve QLP instances that accomplished to the properties as mentioned above, and that were feasible at the same time. To ensure comparability, the QLPs result from the same twelve original LP instances for each of the nine test sets. The original instances from the Netlib Library are: AFIRO, ISRAEL, KB2, RECIPELP, SC105, SC205, SC50A, SC50B, SCAGR25, SCAGR7, and SCTAP1. For each QLP instance, the corresponding DEP was computed using the compact-variable representation and stored in the CPLEX LP file format. Since our test set contains QLP optimization problems, the algorithm presented by Subramani was not used for the experimental tests, because its elimination procedure does not support an objective function.

Table 1 shows the summed up results solving each of the test sets with the standard NBD algorithm, the $\alpha\beta$-NBD algorithm, and when solving the corresponding DEPs with CPLEX. Column 1 contains the number of universally quantified variables followed by the number of blocks of universally quantified variables in column 2. Column 3 contains the solution times when the corresponding DEPs are solved with CPLEX running with standard settings and its preprocessor enabled. Columns 4 and 5 show the solution times and the number

[1] http://www.netlib.org/lp/

of LPs that were solved using the standard NBD algorithm. Column 6 and 7 show the same numbers when $\alpha\beta$-cuts and the move sort heuristic are used.

		CPLEX	NBD		$\alpha\beta$-NBD	
∀-Vars	∀-Blocks	Time (s)	Time (s)	LPs solved	Time (s)	LPs solved
10	1	84,00	23,99	238.598	9,00	136.211
10	2	55,00	17,99	243.513	11,99	97.925
10	5	32,99	36,00	591.495	15,12	194.668
15	1	16.326,00	501,74	5.544.158	214.00	3.129.341
15	2	15.251,00	444,03	5.875.887	113,79	1.181.629
15	5	11.085,11	745,78	12.733.826	196,99	2.253.480
18	1	>172800.00	3075,00	40.807.572	2557,24	35.805.290
18	2	>172800.00	3476,55	65.290.954	867,00	11.558.028
18	5	>172800.00	7028,00	138.049.182	2649,99	24.277.839

Table 1: Computational Results

The results show that the even the standard NBD algorithm implementation is clearly faster than solving the DEP in most cases, especially with an increasing number of universally quantified variables. This is due to the exponential growth of the DEP with an increasing number of universally quantified variables in the corresponding QLP. Note that for solving the DEP, the solution times decrease with an increasing number of quantifier blocks, because in split-variable formulation and equally distributed quantifier blocks, the resulting DEPs get smaller. For the NBD algorithm the opposite is the case because there is an increasing effort for tree traversal (e.g. saving bases and cuts, restoring bases and cuts, ...). When we additionally use the $\alpha\beta$-heuristic and move sort, we observe notable time savings up to about 75% compared to the standard implementation as we can e.g. see in the second last row of Table 1. The extended algorithm was able to reduce the number of subproblems that had to be solved from 65.290.954 to 11.558.028, resulting in a reduction of the solution time from 58 minutes to 15 minutes. The effect becomes stronger with an increasing number of stages and we observed time-savings of 63.5% on average for multistage QLPs. However, even in the twostage case, move-ordering alone leads to a performance gain of and 45%. These results show the high potential of combining techniques from game-tree search with the Nested Benders Decomposition approach and motivate a further research in this direction.

7 Summary

In the course of this paper we considered Quantified Linear Programs (QLPs) and proposed an extension by a linear objective function. To solve the resulting QLP optimization problem, we showed how its hybrid nature of being a two-person zero-sum game on the one side, and being a convex multistage decision

problem on the other side, can be used to combine linear programming techniques with solution techniques from game-tree search. We therefore extended the Nested Benders Decomposition algorithm by $\alpha\beta$-cuts and move-ordering, two techniques that are used in the $\alpha\beta$-Algorithm to evaluate minimax trees. We furthermore showed how QLP relaxations that result from quantifier shifting can be used to improve the cutting mechanism. We showed the applicability in an experimental evaluation, where we solved some QLP optimization problems, which were generated from the NETLIB test set. The results showed a speedup of up to 75% compared to the standard Nested Benders Decomposition implementation without these techniques. These results are promising and motivate a further research in this direction.

References

1. Field programmable logic and application, 14th international conference , fpl 2004, leuven, belgium, august 30-september 1, 2004, proceedings. In J. Becker, M. Platzner, and S. Vernalde, editors, *FPL*, volume 3203 of *Lecture Notes in Computer Science*. Springer, 2004.
2. R. Bellmann. Dynamic programming. *Princeton University Press*, 1957.
3. A. Ben-Tal, L. E. Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton University Press, 2009.
4. A. Ben-tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88:411–424, 2000.
5. A. Ben-Tal and A. Nemirovski. Robust optimization ? methodology and applications. *Mathematical Programming*, 92(3):453–480, 2002.
6. J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, Dec. 1962.
7. J. R. Birge, C. J. Donohue, D. F. Holmes, and O. G. Svintsitski. A parallel implementation of the nested decomposition algorithm for multistage stochastic linear programs. *Math. Program.*, 75:327–352, November 1996.
8. J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer Series in Operations Research and Financial Engineering. Springer, July 1997.
9. J. R. Birge and F. V. Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34(3):384–392, March 1988.
10. M. Dempster and R. Thompson. Parallelization and aggregation ofnested benders decomposition. *Annals of Operations Research*, 81:163–188, 1998. 10.1023/A:1018996821817.
11. T. Ederer, U. Lorenz, A. Martin, and J. Wolf. Quantified linear programs: A computational study. In *Proceedings of the 18th annual European conference on Algorithms: Part I*, ESA'11, pages 203–214, Berlin, Heidelberg, 2011. Springer-Verlag.
12. H. I. Gassmann. Mslip: A computer code for the multistage stochastic linear programming problem. *Mathematical Programming*, 47(1 - 3):407–423, May 1990.
13. A. Kleywegt, A. Shapiro, and T. Homem-De-Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Jour. of Opt.*, pages 479–502, 2001.
14. D. E. Knuth and R. W. Moore. An analysis of alpha-beta pruning. *Artif. Intell.*, 6(4):293–326, 1975.

15. F. König, M. Lübbecke, R. Möhring, G. Schäfer, and I. Spenke. Solutions to real-world instances of pspace-complete stacking. *Proc. ESAÃŢ07*, pages 729–740.
16. C. Liebchen, M. Lübbecke, R. Möhring, and S.Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. *Robust and online large-scale optimization*, pages 1–27, 2009.
17. 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, Berlin, Heidelberg, 2010. Springer-Verlag.
18. N. Megow and T. Vredeveld. Approximation results for preemtive stochastic online scheduling. *ESA 2006 14th Annual European Symposium on Algorithms*, 2006.
19. R. Möhring, A. Schulz, and M. Uetz. Approximation in stochastic scheduling: The power of lp-based priority schedules. *Journal of ACM*, 46(6):924–942, 1999.
20. C. Papadimitriou. Games against nature. *J. of Comp. and Sys. Sc.*, pages 288–301, 1985.
21. W. Pijls and A. de Bruin. Game tree algorithms and solution trees. *Theor. Comput. Sci.*, 252(1-2):197–215, 2001.
22. A. Ruszczyński. Parallel decomposition of multistage stochastic programming problems. *Math. Program.*, 58:201–228, February 1993.
23. K. Subramani. Analyzing selected quantified integer programs. *Springer, LNAI 3097*, pages 342–356, 2004.
24. K. Subramani. Tractable fragments of presburger arithmetic. *Theor. Comp. Sys.*, 38(5):647–668, Sept. 2005.
25. K. Subramani. On a decision procedure for quantified linear programs. *Annals of Mathematics and Artificial Intelligence*, 51(1):55–77, 2007.
26. L. Zhang, H. Hermanns, F. Eisenbrand, and D. Jansen. Flow faster: Efficient decision algorithms for probabilistic simulations. *Logical Methods in Computer Science*, 4(4), 2008.