Theory Comput. Systems **38**, 647–668 (2005)
DOI: 10.1007/s00224-004-1220-0

Theory of Computing Systems

Tractable Fragments of Presburger Arithmetic

K. Subramani

LDCSEE,
West Virginia University,
Morgantown, WV 26506, USA
ksmani@csee.wvu.edu

Abstract. In this paper we introduce a problem called Quantified Integer Programming, which generalizes the Quantified Satisfiability problem (QSAT). In a Quantified Integer Program (QIP) the program variables can assume arbitrary integral values, as opposed to the boolean values that are assumed by the variables of an instance of QSAT. QIPs naturally represent two-person integer matrix games. The Quantified Integer Programming problem is `PSPACE-hard` in general, since the QSAT problem is `PSPACE-complete`. Quantified Integer Programming can be thought of as a restriction of Presburger Arithmetic, in that we allow only conjunctions of linear inequalities. We focus on analyzing various special cases of the general problem, with a view to discovering subclasses that are tractable. Subclasses of the general QIP problem are obtained by restricting either the constraint matrix or quantifier specification or both. We show that if the constraint matrix is totally unimodular, the problem of deciding a QIP can be solved in polynomial time. We also establish the computational complexities of Oblivious strategy games and Clairvoyant strategy games.

1. Introduction

Quantified decision problems are useful in modeling situations, wherein a policy (action) can depend upon the effect of imposed stimuli. A typical such situation is a two-person game. Consider a board game comprised of an initial configuration and two players A and B, each having a finite set of moves. We focus on the following decision problem: *Given the initial configuration*, *does Player A have a first move* (*policy*), *such that for all possible first moves of Player B* (*imposed stimulus*), *Player A has a second move*, *such that for all possible second moves of Player B*,..., *Player A eventually wins*? If this question can be answered affirmatively, then Player A has a winning strategy; otherwise Player B has one. The board configuration can be represented as a boolean expression or as a constraint matrix; the expressiveness of the board configuration typically determines the complexity of the decision problem. The use of quantified boolean expressions to capture problem domains has been widespread within the AI and Planning communities [CGS]. Our work in this paper is concerned with a problem closely related to Quantified Satisfiability called Quantified Integer Programming. Quantified Integer Programming generalizes the QSAT problem, in that program variables can assume arbitrary integral values. It follows that the Quantified Integer Programming problem is at least as hard as QSAT. Quantified Integer Programs (QIPs) naturally represent two-person integer matrix games. An interesting line of research is to focus on restrictions of the general problem, in order to discover useful subclasses that are tractable. In this paper we study a number of special subclasses of QIPS that are obtained by restricting either the constraint matrix or the quantifier specification.

The primary contributions of this paper are as follows:

1. A polynomial time algorithm for deciding QIPs, when the constraint matrix is totally unimodular.
2. A polynomial time algorithm to decide some Planar QIPs, i.e., QIPs in two dimensions.
3. A polynomial time algorithm to decide QIPs with only universal quantifiers.
4. Complexity results for Oblivious strategy and Clairvoyant strategy games.

The rest of this paper is organized as follows: Section 2 describes the QIP problem and some of the special subclasses that we study in this paper. In Section 3 we motivate the study of our special cases, while in Section 4 we discuss work in the literature that is related to our current efforts. We commence our analysis in Section 5 by deriving properties of arbitrary QIPs; these properties are then used in the succeeding sections. Section 6 demonstrates the existence of a polynomial time procedure for deciding totally unimodular QIPs. In Section 7 a polynomial time procedure for deciding some Planar QIPs is discussed, while in Section 8 we show that the class of Box QIPs can be solved in polynomial time. Section 9 analyzes and establishes the complexities of Oblivious strategy and Clairvoyant strategy games. We conclude in Section 10 by summarizing our results and outlining problems for future research.

2. Statement of Problems

Definition 2.1. Let $\{x_1, x_2, \ldots, x_n\}$ be a set of n boolean variables. A literal is either the variable x_i or its complement $\bar{x_i}$. A disjunction of literals is called a *clause*, represented by C_i. A boolean expression of the form

$$Q_1 x_1 \in \{\mathbf{true}, \mathbf{false}\}\ Q_2 x_2 \in \{\mathbf{true}, \mathbf{false}\} \cdots Q_n x_n \in \{\mathbf{true}, \mathbf{false}\}\ \ C,$$

where each Q_i is either $\exists$ or $\forall$ and $C = C_1 \wedge C_2 \cdots \wedge C_m$, is called a Quantified CNF formula (QCNF) and the problem of deciding whether it is true is called the Quantified Satisfiability problem (QSAT).

QSAT has been shown to be `PSPACE-complete`, even when there are at most three literals per clause (Q3SAT) [Pa]. Schaefer [Sc1] argued the existence of polynomial time algorithms for Q2SAT, although no constructive procedure was given. Aspvall et al. [APT] provided the first polynomial time algorithm for the Q2SAT problem. In [Ga] it was shown that the Q2SAT problem is in the parallel complexity class $\mathtt{NC}_2$. Experimental work on algorithms for QSAT problems has been the thrust of [CGS].

Definition 2.2. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of n variables with integral ranges. A mathematical program of the form

$$Q_1\ x_1 \in \{a^1 - b^1\}\ Q_2 x_2 \in \{a^2 - b^2\} \cdots Q_n x_n \in \{a^n - b^n\}$$
$$\mathbf{A} \cdot \vec{\mathbf{x}} \le \vec{\mathbf{b}}, \tag{1}$$

where each Q_i is either $\exists$ or $\forall$, is called a Quantified Integer Program (QIP).

The matrix **A** is called the constraint matrix of the QIP.

The `PSPACE-hardness` of QIPs follows immediately from the `PSPACE-completeness` of QSAT; in fact the reduction from QSAT to QIP is identical to that from SAT to 0/1 Integer Programming.

Without loss of generality, we assume that the quantifiers are strictly alternating and that $Q_1 = \exists$ (by using dummy variables, if necessary); further we denote the existentially quantified variables using $x_i \in \{a^i - b^i\}$, a^i, b^i integral, $i = 1, 2, \ldots, n$, and the universally quantified variables using $y_i \in \{c^i - d^i\}$, c^i, d^i integral, $i = 1, 2, \ldots, n$. Thus we can write an arbitrary QIP as

$$\begin{aligned}\mathbf{QIPG}: \quad & \exists x_1 \in \{a^1 - b^1\}\ \forall y_1 \in \{c^1 - d^1\}\\ & \exists x_2 \in \{a^2 - b^2\}\ \forall y_2 \in \{c^2 - d^2\}\\ & \cdots \exists x_n \in \{a^n - b^n\}\ \forall y_n \in \{c^n - d^n\}\\ & \quad \mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \le \vec{\mathbf{b}}\end{aligned} \tag{2}$$

for suitably chosen $\vec{\mathbf{x}}, \vec{\mathbf{y}}, \mathbf{A}, \vec{\mathbf{b}}, n$.

The specification $\exists x_1 \in \{a^1 - b^1\}\ \forall y_1 \in \{c^1 - d^1\}\ \exists x_2 \in \{a^2 - b^2\}\ \forall y_2 \in \{c^2 - d^2\} \cdots \exists x_n \in \{a^n - b^n\}\ \forall y_n \in \{c^n - d^n\}$ is called the quantifier specification or quantifier string of the QIP and is denoted by $\mathbf{Q}(\mathbf{x}, \mathbf{y})$. We note that $\mathbf{Q}(\mathbf{x}, \mathbf{y})$ imposes a linear ordering on the program variables of the QIP. System (2) is referred to as a *QIP in general form or a general QIP*, on account of the unbounded alternation in the quantifier specification $\mathbf{Q}(\mathbf{x}, \mathbf{y})$.

The following assertions hold for the rest of this paper:

1. **A** and $\vec{\mathbf{b}}$ are integral.
2. The intervals of the variables are defined by integers, i.e., a^i, b^i, c^i, d^i are integers, for all i.
3. We say that $x_i \in \{a^i - b^i\}$, $a^i \le b^i$, to mean that the valid values for x_i are the integers in the set $\{a^i, a^i + 1, \ldots, b^i\}$.
4. The range constraints on the existentially quantified variables can be made part of the constraint system $\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \le \vec{\mathbf{b}}$.

We now describe the various restrictions to System (2) that we analyze in this paper.

Definition 2.3. A matrix **A** is said to be totally unimodular (TUM), if every square submatrix of **A** has determinant 0, 1, or -1.

Remark 2.1. Definition 2.3 forces every entry in **A** to belong to $\{0, 1, -1\}$.

TUM matrices arise in network flow problems [AMO], scheduling problems [Pi], and a whole host of situations in which only strict difference constraints are permitted between program variables [Sc2].

Definition 2.4. A totally unimodular QIP (TQIP) is a QIP in which the constraint matrix (**A**) is totally unimodular.

The first problem that we consider is:

$\mathbf{P_1}$. *Is there a polynomial time procedure to decide an arbitrary TQIP as described in Definition* 2.4?

Definition 2.5. A Planar QIP (PQIP) is a QIP in which all constraints exist between at most two variables, i.e., every constraint is a half-plane in the x_1–y_1 plane.

Note that all constraints are between the same two variables (x_1 *and* y_1). Planar QIPs are also known as two-dimensional QIPs and should not be confused with Quantified Integer Programming with at most two nonzero variables per constraint, i.e., QIP(2). In the former case the dimension of the constraint matrix is 2, whereas in the latter case the dimension of the constraint matrix is $2 \cdot n$. The second problem that we consider is:

$\mathbf{P_2}$. *Is there a polynomial time procedure to decide an arbitrary PQIP as described in Definition* 2.5?

Definition 2.6. A Box QIP (BQIP) is a QIP, in which every quantifier is universal, i.e., a QIP of the form

$$\begin{aligned}&\forall y_1 \in \{c^1 - d^1\}\ \forall y_2 \in \{c^2 - d^2\} \cdots \forall y_n \in \{c^n - d^n\}\\ &\qquad \mathbf{A} \cdot \vec{\mathbf{y}} \leq \vec{\mathbf{b}}.\end{aligned} \tag{3}$$

The third problem that we consider is:

$\mathbf{P_3}$. *Is there a polynomial time procedure to decide an arbitrary BQIP as described in Definition* 2.6?

2.1. *Model Verification*

In this section we formally specify what it means for a vector $\vec{\mathbf{x}}$ to be a solution or a "model" of a QIP, such as **QIPG**. The specification involves the notion of a two-person integer matrix game.

Let **X** denote the existential player and let **Y** denote the universal player. The game consists of n rounds; in round i, **X** guesses a value for x_i which may depend upon $\{y_1, y_2, \ldots, y_{i-1}\}$, while **Y** guesses a value for y_i which may depend upon $\{x_1, x_2, \ldots, x_i\}$. At the end of n rounds, we construct the vectors $\vec{\mathbf{x}} = [x_1, x_2, \ldots, x_n]^T$ and $\vec{\mathbf{y}} = [y_1, y_2, \ldots, y_n]^T$. If $\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \le \vec{\mathbf{b}}$, the game is said to be a win for **X**; otherwise it is said to be a win for **Y**. The guesses made by both players are nondeterministic in that if **X** can win the game, his guesses will lead to a win; likewise if **Y** can thwart **X**, then the guesses made by **Y** will lead to **X** losing. If **X** wins the game, **QIPG** is said to have a model or be *true*.

Observe that a solution to a QIP, such as **QIPG**, is in general, a strategy for the existential player **X** and not a numeric vector. Thus a solution vector will have the form $\vec{\mathbf{x}}_{\mathbf{s}} = [x_1, x_2, \ldots, x_n]^T = [c_0, f_1(y_1), f_2(y_1, y_2), \ldots, f_{n-1}(y_1, y_2, \ldots, y_{n-1})]^T$, where the f_i are the Skolem functions capturing the dependence of x_i on $y_1, y_2, \ldots, y_{i-1}$ and c_0 is a constant in $\{a^1 - b^1\}$. Likewise, the universal player **Y** also makes its moves according to some strategy. Given a strategy $\vec{\mathbf{x}}_{\mathbf{s}}$ for **X** and a strategy $\vec{\mathbf{y}}_{\mathbf{s}}$ for **Y**, we say that $\vec{\mathbf{x}}_{\mathbf{s}}$ is winning against $\vec{\mathbf{y}}_{\mathbf{s}}$ if the consequence of **X** playing according to $\vec{\mathbf{x}}_{\mathbf{s}}$ and **Y** playing according to $\vec{\mathbf{y}}_{\mathbf{s}}$ is that **X** wins the game. A strategy $\vec{\mathbf{x}}_{\mathbf{s}}$ for **X** is said to be *always-winning*, or a model for System (2) (**QIPG**), if the consequence of **X** playing according to $\vec{\mathbf{x}}_{\mathbf{s}}$ is that **X** wins regardless of the strategy employed by **Y**. Thus the Quantified Integer Programming problem can be described as the problem of checking whether the existential player in a two-person integer matrix game has an always-winning strategy. For the rest of the paper we use the phrase "a winning strategy for the existential player" to mean an always-winning strategy for the existential player, when there is no confusion.

3. Motivation

One of the principal areas in which QIPs find applications is the modeling of uncertainty [Su2]. In most application models there is the inherent assumption of constancy in data, which is neither realistic nor accurate. For instance, in scheduling problems [Pi] it is standard to assume that the execution time of a job is fixed and known in advance. While this simplifying assumption leads to elegant models, the fact remains that in real-time systems such an assumption would lead to catastrophic consequences [SSRB]. Of late, there has been some interest in problems such as parametric flow [McC] and Selective Assembly [IMM], in which the capacities are assumed to be variable. Such flow problems are easily and naturally expressed as TQIPs.

Whereas Integer Programming is `NP-complete` [GJ], there are some interesting restrictions to the constraint matrix (other than total unimodularity) that have polynomial time decision procedures. Foremost among them is planar IP or Integer Programming in the plane. Planar IPs have been used to model and solve knapsack problems that arise in certain applications [Ka], [HW]. Problem $\mathbf{P_2}$ generalizes the planar IP problem to the quantified case; once again the applications are motivated by the uncertainty in knapsack parameters.

BQIPs are useful models to check the validity of Clausal Systems and Integer Programs, since the only way in which a BQIP is false is if there is a witness attesting to this fact. One of the more important application areas of BQIPs is the field of Constraint Databases [Re1], [Re2]. Queries of the form: *Enumerate all people between ages* 20 *and* 30 *who earn between* $20K$ *and* $30K$ *annually*, are naturally expressible as Box QIPs over the appropriate domain. Problem $\mathbf{P_3}$ generalizes the work in [Re3] in that the solutions that we seek are quantified lattice points and not rationals.

Additional applications of QIPs can be found in the areas of logical inference [CH], Computer Vision [vH], and Compiler Construction [Pu1], [Pu2].

4. Related Work

In a landmark thesis, Tarski analyzed a number of properties of real fields that make them amenable to the existence of decision procedures [Ta]. He showed that the language of reals with universal quantifiers is decidable. It is well known that a fragment of integer arithmetic called Presburger Arithmetic, in which multiplication and exponentiation are not permitted, is decidable [Pa]. It was shown in [FR] that the problem of deciding Presburger Arithmetic has super-exponential complexity. Quantified Integer Programming can be thought of as a restriction to Presburger Arithmetic in that, we only permit conjunctions of linear inequalities.

In [CD] the expressiveness of "full" first-order constraints is analyzed by considering equality as a unique relational symbol. Benhamou and Goulard [BG] handle universally quantified interval constraints by identifying their relationship with the computation of inner-approximations of real relations and thereby avoid the use of Cylindrical Algebraic Decomposition (CAD).

Lassez and Maher [LM] discuss empirical observations in using the Fourier–Motzkin elimination technique for linear constraints over both rational domains and lattice domains; practical issues in polyhedral projection are further addressed in [LL], where certain clausal systems are resolved by constructing the *approximate* convex hull of the feasible space.

In [Su3] we introduced Quantified Linear Programming, which is similar to Quantified Integer Programming; the principal difference between these two programming paradigms, is that in a Quantified Linear Program, the moves made by the two players can be arbitrary rationals which may or may not be integral. We also showed that Quantified Linear Programming can be decided in polynomial time, when the constraint matrix is totally unimodular. In [Su2] Quantified Linear Programs were used to model and evaluate a number of scheduling problems involving varying degrees of clairvoyance with respect to the execution times of jobs.

Our focus here is on polynomial time procedures to decide various restrictions to Quantified Integer Programming problems.

5. Properties of General QIPs

In this section we derive properties that are true of any general QIP. These properties when combined with constraint specific properties aid us in the design of polynomial time algorithms for specialized classes of QIPs.

Definition 5.1. A mathematical program of the form

$$\begin{aligned}&\exists x_1 \in [a^1, b^1]\ \forall y_1 \in [c^1, d^1]\\ &\exists x_2 \in [a^2, b^2] \forall y_2 \in [c^2, d^2]\\ &\cdots \exists x_n \in [a^n, b^n]\ \forall y_n \in [c^n, d^n]\\ &\qquad \mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}\end{aligned} \tag{4}$$

is called a Quantified Linear Program (QLP).

In System (4), the range of each program variable is a continuous real interval and not a discrete interval as is the case with QIPs. However, the entities defining the ranges, namely, $\{a^i, b^i, c^i, d^i\},\ i = 1, 2, \ldots, n$, are integral.

Definition 5.2. A TQLP is a QLP in which the constraint matrix is totally unimodular.

The complexity of deciding QLPs is not known [Jo], although the class of TQLPs can be decided in polynomial time (see [Su3]).

Definition 5.3. A QIP in which some variables have discrete ranges, while the rest have continuous ranges, is called a Mixed QIP (MQIP).

Theorem 5.1 argues the equivalence of certain MQIPs and QLPs. The consequences of this equivalence, when the constraint matrix is totally unimodular, are pointed out in Corollary 6.1.

Theorem 5.1.

$$\begin{aligned}\mathbf{L}\colon\ &\exists x_1 \in [a^1, b^1]\ \forall y_1 \in \{c^1 - d^1\}\\ &\exists x_2 \in [a^2, b^2]\ \forall y_2 \in \{c^2 - d^2\}\\ &\cdots \exists x_n \in [a^n, b^n]\ \forall y_n \in \{c^n - d^n\}\\ &\quad \mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}\\ &\Leftrightarrow\\ \mathbf{R}\colon\ &\exists x_1 \in [a^1, b^1]\ \forall y_1 \in [c^1, d^1]\\ &\exists x_2 \in [a^2, b^2]\ \forall y_2 \in [c^2, d^2]\\ &\cdots \exists x_n \in [a^n, b^n]\ \forall y_n \in [c^n, d^n]\\ &\quad \mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}.\end{aligned} \tag{5}$$

In other words, the existential player of game **L** *has a winning strategy if and only if the existential player of game* **R** *has a winning strategy.*

Proof. Let $\mathbf{X_L}$ and $\mathbf{Y_L}$ denote the existential and universal players of game **L**, respectively. Likewise, let $\mathbf{X_R}$ and $\mathbf{Y_R}$ denote the existential and universal players of game **R**.

Observe that **R** is a QLP, while **L** is an MQIP.

R $\Rightarrow$ **L** is straightforward. Suppose that $\mathbf{X_R}$ has a strategy that is winning when the universal player $\mathbf{Y_R}$ can choose his ith move from the continuous interval $[c^i, d^i]$; then clearly the strategy will also be winning, when the universal player has to choose his ith move from the discrete interval $\{c^i - d^i\}$. Thus $\mathbf{X_L}$ can adopt the same strategy as $\mathbf{X_R}$ and win against any strategy employed by $\mathbf{Y_L}$.

We now focus on proving **L** $\Rightarrow$ **R**. Our proof uses induction on the length of the quantifier string and therefore on the dimension of **A**. Note that as described in system (2), the quantifier string is always of even length, with the existentially quantified variables and the universally quantified variables strictly alternating. Further, the first variable is always existentially quantified and the last variable is always universally quantified.

In the base case of the induction, the length of the quantifier string is 2; accordingly, we have to show that

$$\begin{aligned}&\mathbf{L}\colon \exists x_1 \in [a^1, b^1]\ \forall y_1 \in \{c^1 - d^1\}\ \mathbf{A} \cdot [x_1\ \ y_1]^T \le \vec{\mathbf{b}}\\ &\qquad \Rightarrow \quad \mathbf{R}\colon \exists x_1 \in [a^1, b^1]\ \forall y_1 \in [c^1, d^1]\ \mathbf{A} \cdot [x_1\ \ y_1]^T \le \vec{\mathbf{b}}.\end{aligned}$$

Let us say that **L** is true and let $x_1 = c_0$ be a solution, where $c_0 \in [a^1, b^1]$. We can write the constraint system $\mathbf{A} \cdot [x_1\ \ y_1]^T \le \vec{\mathbf{b}}$ as $x_1 \cdot \vec{\mathbf{g_1}} + y_1 \cdot \vec{\mathbf{h_1}} \le \vec{\mathbf{b}}$. Note that c_0 is a fixed constant, independent of y_1 and holds for all integral values of y_1 in $\{c^1 - d^1\}$. Accordingly, we have

$$\begin{aligned}&c_0 \cdot \vec{\mathbf{g_1}} \le \vec{\mathbf{b}} - c^1 \cdot \vec{\mathbf{h_1}},\\ &c_0 \cdot \vec{\mathbf{g_1}} \le \vec{\mathbf{b}} - d^1 \cdot \vec{\mathbf{h_1}}.\end{aligned} \tag{6}$$

Now consider the (real) parametric point $y_1 = \lambda \cdot c^1 + (1 - \lambda) \cdot d^1,\ 0 \le \lambda \le 1$. Observe that

$$\begin{aligned}&\vec{\mathbf{b}} - (\lambda \cdot c^1 + (1-\lambda) \cdot d^1) \cdot \vec{\mathbf{h_1}}\\ &\qquad = \lambda \cdot \vec{\mathbf{b}} + (1-\lambda) \cdot \vec{\mathbf{b}} - \lambda \cdot c^1 \cdot \vec{\mathbf{h_1}} - (1-\lambda) \cdot d^1 \cdot \vec{\mathbf{h_1}}\\ &\qquad = \lambda \cdot (\vec{\mathbf{b}} - c^1 \cdot \vec{\mathbf{h_1}}) + (1-\lambda) \cdot (\vec{\mathbf{b}} - d^1 \cdot \vec{\mathbf{h_1}})\\ &\qquad \ge \lambda \cdot (c_0 \cdot \vec{\mathbf{g_1}}) + (1-\lambda) \cdot (c_0 \cdot \vec{\mathbf{g_1}})\\ &\qquad = c_0 \cdot \vec{\mathbf{g_1}}.\end{aligned}$$

In other words, $x_1 = c_0$ holds for all values of y_1 in the continuous range $[c^1, d^1]$, thereby proving the base case.

Assume that Theorem 5.1 always holds when the quantifier string has length $2 \cdot m$; we need to show that it holds when the quantifier string has length $2 \cdot m + 2$, i.e., we need to show that:

$$\begin{aligned}\mathbf{L}\colon\ &\exists x_1 \in [a^1, b^1]\ \forall y_1 \in \{c^1 - d^1\}\\ &\exists x_2 \in [a^2, b^2]\ \forall y_2 \in \{c^2 - d^2\}\\ &\cdots \exists x_{m+1} \in [a^{m+1}, b^{m+1}]\ \forall y_{m+1} \in \{c^{m+1} - d^{m+1}\}\\ &\quad \mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \le \vec{\mathbf{b}}\end{aligned}$$

$$\Rightarrow$$
$$\begin{aligned}\mathbf{R}:\ & \exists x_1 \in [a^1, b^1]\ \forall y_1 \in [c^1, d^1] \\ & \exists x_2 \in [a^2, b^2]\ \forall y_2 \in [c^2, d^2] \\ & \cdots \exists x_{m+1} \in [a^{m+1}, b^{m+1}]\ \forall y_{m+1} \in [c^{m+1}, d^{m+1}] \\ & \quad \mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}. \end{aligned} \tag{7}$$

Let $\vec{\mathbf{x_s}} = [x_1, x_2, \ldots, x_{m+1}]^T$ be a model for $\mathbf{L}$, in the manner described in Section 2.1. We consider two distinct games $\mathbf{L_1}$ and $\mathbf{L_2}$ to decide $\mathbf{L}$; one in which $\mathbf{Y_L}$ is forced to choose c^{m+1} for y_{m+1} and another in which $\mathbf{Y_L}$ is forced to pick $y_{m+1} = d^{m+1}$.

Consider the complete set of moves made in $(m+1)$ rounds by $\mathbf{X_L}$ and $\mathbf{Y_L}$ to decide $\mathbf{L}$ in both games; let $\vec{\mathbf{x_L}}$ denote the numeric vector guessed by $\mathbf{X_L}$, while $\vec{\mathbf{y_{L1}}}$ denotes the vector guessed by $\mathbf{Y_L}$ for game $\mathbf{L_1}$ and $\vec{\mathbf{y_{L2}}}$ denotes the vector guessed by $\mathbf{Y_L}$ for game $\mathbf{L_2}$. Note that the moves made by $\mathbf{X_L}$ cannot depend on y_{m+1} and hence the vector guessed by $\mathbf{X_L}$ is the same for both games; further $\vec{\mathbf{y_{L1}}}$ and $\vec{\mathbf{y_{L2}}}$ differ only in their $(m+1)$th component. We denote the m-vector (numeric) corresponding to the first m components of the two vectors $\vec{\mathbf{y_{L1}}}$ and $\vec{\mathbf{y_{L2}}}$ by $\vec{\mathbf{y_1}}$. We rewrite the constraint system $\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$ as $\mathbf{G} \cdot \vec{\mathbf{x}} + \mathbf{H}' \cdot \vec{\mathbf{y}}' + y_{m+1} \cdot \mathbf{h_{m+1}} \leq \vec{\mathbf{b}}$, where $\vec{\mathbf{x}} = [x_1, x_2, \ldots, x_{m+1}]^T$ and $\vec{\mathbf{y}}' = [y_1, y_2, \ldots, y_m]^T$.

Since $\vec{\mathbf{x_s}}$ is a model for $\mathbf{L}$, we must have

$$\mathbf{G} \cdot \vec{\mathbf{x_L}} + \mathbf{H}' \cdot \vec{\mathbf{y_1}} + c^{m+1} \cdot \vec{\mathbf{h_{m+1}}} \leq \vec{\mathbf{b}} \tag{8}$$

and

$$\mathbf{G} \cdot \vec{\mathbf{x_L}} + \mathbf{H}' \cdot \vec{\mathbf{y_1}} + d^{m+1} \cdot \vec{\mathbf{h_{m+1}}} \leq \vec{\mathbf{b}}. \tag{9}$$

Now consider the (real) parametric point

$$y_{m+1} = \lambda \cdot c^{m+1} + (1-\lambda) \cdot d^{m+1}.$$

Observe that

$$\begin{aligned} & \mathbf{G} \cdot \vec{\mathbf{x_L}} + \mathbf{H}' \cdot \vec{\mathbf{y_1}} + y_{m+1} \cdot \vec{\mathbf{h_{m+1}}} \\ & \quad = \mathbf{G} \cdot \vec{\mathbf{x_L}} + \mathbf{H}' \cdot \vec{\mathbf{y_1}} + (\lambda \cdot c^{m+1} + (1-\lambda) \cdot d^{m+1}) \cdot \vec{\mathbf{h_{m+1}}} \\ & \quad = \lambda \cdot (\mathbf{G} \cdot \vec{\mathbf{x_L}} + \mathbf{H}' \cdot \vec{\mathbf{y_1}} + c^{m+1} \cdot \vec{\mathbf{h_{m+1}}}) \\ & \qquad + (1-\lambda) \cdot (\mathbf{G} \cdot \vec{\mathbf{x_L}} + \mathbf{H}' \cdot \vec{\mathbf{y_1}} + d^{m+1} \cdot \vec{\mathbf{h_{m+1}}}) \\ & \quad \leq \lambda \cdot \vec{\mathbf{b}} + (1-\lambda) \cdot \vec{\mathbf{b}} \\ & \quad = \vec{\mathbf{b}}. \end{aligned}$$

In other words, $\vec{\mathbf{x_s}}$ serves as a winning strategy for $\mathbf{X_L}$ for all values of y_{m+1} in the continuous range $[c^{m+1}, d^{m+1}]$. Accordingly, we are required to show that

$$\begin{aligned}\mathbf{L}:\ & \exists x_1 \in [a^1, b^1]\ \forall y_1 \in \{c^1 - d^1\} \\ & \exists x_2 \in [a^2, b^2]\ \forall y_2 \in \{c^2 - d^2\} \\ & \cdots \exists x_{m+1} \in [a^{m+1}, b^{m+1}]\ \forall y_{m+1} \in [c^{m+1}, d^{m+1}] \\ & \quad \mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}} \end{aligned}$$

$$\begin{aligned}
&\Rightarrow\\
\mathbf{R}:\quad &\exists x_1 \in [a^1, b^1]\ \forall y_1 \in [c^1, d^1]\\
&\exists x_2 \in [a^2, b^2]\ \forall y_2 \in [c^2, d^2]\\
&\cdots \exists x_{m+1} \in [a^{m+1}, b^{m+1}]\ \forall y_{m+1} \in [c^{m+1}, d^{m+1}]\\
&\quad \mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}.
\end{aligned}\tag{10}$$

Observe that both y_{m+1} and x_{m+1} are continuous in $\mathbf{L}$ and $\mathbf{R}$ and hence can be eliminated using the quantifier elimination techniques used to eliminate the variables of a QLP [Su3]; y_{m+1} is eliminated using variable substitution, while x_{m+1} is eliminated using the Fourier–Motzkin elimination technique. Accordingly, the constraint system $\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$ is transformed into the system $\mathbf{A_1} \cdot [\vec{\mathbf{x_1}}\ \ \vec{\mathbf{y_1}}]^{\mathbf{T}} \leq \vec{\mathbf{b_1}}$, where $\vec{\mathbf{x_1}} = [x_1, x_2, \ldots, x_m]^T$ and $\vec{\mathbf{y_1}} = [y_1, y_2, \ldots, y_m]^T$. Since the quantifier string is now of length $2 \cdot m$, we can use the inductive hypothesis to conclude that $\mathbf{L} \Rightarrow \mathbf{R}$.

It follows that Theorem 5.1 is proven. $\square$

Corollary 5.1. *If all the existentially quantified variables of a QIP have continuous ranges, then the discrete ranges of the universally quantified variables can be relaxed into continuous ranges.*

Theorem 5.2. *Let*

$$\begin{aligned}
\mathbf{L}:\quad &\exists x_1 \in \{a^1 - b^1\}\ \forall y_1 \in \{c^1 - d^1\}\\
&\exists x_2 \in \{a^2 - b^2\}\ \forall y_2 \in \{c^2 - d^2\}\\
&\cdots \exists x_n \in \{a^n - b^n\}\ \forall y_n \in \{c^n - d^n\}\\
&\quad \mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}
\end{aligned}$$

and

$$\begin{aligned}
\mathbf{R}:\quad &\exists x_1 \in \{a^1 - b^1\}\ \forall y_1 \in \{c^1 - d^1\}\\
&\exists x_2 \in \{a^2 - b^2\}\ \forall y_2 \in \{c^2 - d^2\}\\
&\cdots \exists x_n \in \{a^n - b^n\}\ \forall y_n \in [c^n, d^n]\\
&\quad \mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}.
\end{aligned}$$

Then $\mathbf{L} \Leftrightarrow \mathbf{R}$.

Proof. As in Theorem 5.1, let $\mathbf{X_L}$ and $\mathbf{Y_L}$ denote the existential and universal players of game $\mathbf{L}$, respectively. Likewise, let $\mathbf{X_R}$ and $\mathbf{Y_R}$ denote the existential and universal players of game $\mathbf{R}$.

Once again note that $\mathbf{R} \Rightarrow \mathbf{L}$ is straightforward, since the strategy that enabled $\mathbf{X_R}$ to win when $\mathbf{Y_R}$ is allowed to choose his nth move from the continuous interval $[c^n, d^n]$ will also be winning when $\mathbf{Y_R}$ is allowed to choose his nth move from the restricted (discrete) interval $\{c^n - d^n\}$. In other words, a strategy that is winning for $\mathbf{X_R}$ is also winning for $\mathbf{X_L}$.

We focus on proving $\mathbf{L} \Rightarrow \mathbf{R}$.

Let $\vec{\mathbf{x_s}} = [x_1, x_2, \ldots, x_n]^T$ be a model for $\mathbf{L}$, in the manner described in Section 2.1. We consider two distinct games $\mathbf{L_1}$ and $\mathbf{L_2}$ to decide $\mathbf{L}$; one in which $\mathbf{Y_L}$ is forced to choose c^n for y_n and another in which $\mathbf{Y_L}$ is forced to pick $y_n = d^n$.

Consider the complete set of moves made in n rounds by $\mathbf{X_L}$ and $\mathbf{Y_L}$ to decide $\mathbf{L}$ in both games; let $\vec{\mathbf{x_L}}$ denote the numeric vector guessed by $\mathbf{X_L}$, while $\vec{\mathbf{y_{L1}}}$ denotes the vector guessed by $\mathbf{Y_L}$ for game $\mathbf{L_1}$ and $\vec{\mathbf{y_{L2}}}$ denotes the vector guessed by $\mathbf{Y_L}$ for game $\mathbf{L_2}$. Note that the moves made by $\mathbf{X_L}$ cannot depend on y_n and hence the vector guessed by $\mathbf{X_L}$ is the same for both games; further $\vec{\mathbf{y_{L1}}}$ and $\vec{\mathbf{y_{L2}}}$ differ only in their nth component. We denote the $(n-1)$-vector (numeric) corresponding to the first $(n-1)$ components of the two vectors $\vec{\mathbf{y_{L1}}}$ and $\vec{\mathbf{y_{L2}}}$ by $\vec{\mathbf{y_1}}$. We rewrite the constraint system $\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$ as $\mathbf{G} \cdot \vec{\mathbf{x}} + \mathbf{H}' \cdot \vec{\mathbf{y}}' + y_n \cdot \mathbf{h_{m+1}} \leq \vec{\mathbf{b}}$, where $\vec{\mathbf{x}} = [x_1, x_2, \ldots, x_n]^T$ and $\vec{\mathbf{y}}' = [y_1, y_2, \ldots, y_{n-1}]^T$.

Since $\vec{\mathbf{x_s}}$ is a model for $\mathbf{L}$, we must have

$$\mathbf{G} \cdot \vec{\mathbf{x_L}} + \mathbf{H}' \cdot \vec{\mathbf{y_1}} + c^n \cdot \vec{\mathbf{h_n}} \leq \vec{\mathbf{b}} \tag{11}$$

and

$$\mathbf{G} \cdot \vec{\mathbf{x_L}} + \mathbf{H}' \cdot \vec{\mathbf{y_1}} + d^n \cdot \vec{\mathbf{h_n}} \leq \vec{\mathbf{b}}. \tag{12}$$

Now consider the (real) parametric point

$$y_n = \lambda \cdot c^n + (1 - \lambda) \cdot d^n. \tag{13}$$

Observe that

$$\begin{aligned}
&\mathbf{G} \cdot \vec{\mathbf{x_L}} + \mathbf{H}' \cdot \vec{\mathbf{y_1}} + y_n \cdot \vec{\mathbf{h_n}} \\
&\quad = \mathbf{G} \cdot \vec{\mathbf{x_L}} + \mathbf{H}' \cdot \vec{\mathbf{y_1}} + (\lambda \cdot c^n + (1 - \lambda) \cdot d^n) \cdot \vec{\mathbf{h_n}} \\
&\quad = \lambda \cdot (\mathbf{G} \cdot \vec{\mathbf{x_L}} + \mathbf{H}' \cdot \vec{\mathbf{y_1}} + c^n \cdot \vec{\mathbf{h_n}}) + (1 - \lambda) \cdot (\mathbf{G} \cdot \vec{\mathbf{x_L}} + \mathbf{H}' \cdot \vec{\mathbf{y_1}} + d^n \cdot \vec{\mathbf{h_n}}) \\
&\quad \leq \lambda \cdot \vec{\mathbf{b}} + (1 - \lambda) \cdot \vec{\mathbf{b}} \\
&\quad = \vec{\mathbf{b}}.
\end{aligned}$$

In other words, $\vec{\mathbf{x_s}}$ serves as a winning strategy for $\mathbf{X_L}$ for all values of y_n in the continuous range $[c^n, d^n]$, i.e., the strategy that is winning for $\mathbf{X_L}$ is also winning for $\mathbf{X_R}$, thereby proving that $\mathbf{L} \Rightarrow \mathbf{R}$. □

6. Totally Unimodular Quantified Integer Programs

In this section we handle the first restriction to QIPs, namely, total unimodularity of the constraint matrix. Our techniques are based on showing that TQIPs can be relaxed to polynomial time solvable TQLPs, while preserving the solution space. Relaxing a QIP to a QLP involves replacing the discrete intervals $\{a^i - b^i\}$ with continuous intervals $[a'^i, b'^i]$, for suitably chosen a'^i, b'^i, in the case of both the universally and existentially quantified variables of the QIP. In the rest of this section we argue that such a solution-preserving relaxation exists, when the constraint matrix $\mathbf{A}$ of the QIP is TUM.

Theorem 6.1. *Let*

$$\begin{aligned}\mathbf{R}\colon\ & \exists x_1 \in [a^1, b^1]\ \forall y_1 \in [c^1, d^1]\\ & \exists x_2 \in [a^2, b^2]\ \forall y_2 \in [c^2, d^2]\\ & \cdots \exists x_n \in [a^n, b^n]\ \forall y_n \in [c^n, d^n]\\ & \quad \mathbf{A}\cdot[\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \le \vec{\mathbf{b}}\end{aligned} \tag{14}$$

have a model, where **A** *is totally unimodular. Then* x_1 *can always be chosen integral, by the existential player (say* $\mathbf{X_R}$*), without affecting the outcome of the game.*

Proof. Recall that $a_i, b_i, c_i, d_i,\ i = 1, 2, \ldots n$, are integral and $\vec{\mathbf{b}}$ is integral. Observe that **R** is a QLP, hence we can use the algorithm developed in [Su3] to decide it. The algorithm therein eliminates a universally quantified variable as follows: The vector $\vec{\mathbf{b}}$ is replaced with a new integral vector, say $\vec{\mathbf{b}}'$, obtained by subtracting an appropriate integral vector from $\vec{\mathbf{b}}$. The only change to the **A** matrix is that a column is eliminated and hence it stays totally unimodular. Existentially quantified variables are eliminated using Fourier–Motzkin elimination, which is a variation of pivoting and hence their elimination also preserves total unimodularity (see [NW]). Since **R** has a model, the algorithm in [Su3] determines a range for x_1 of the form $a \le x_1 \le b$. Since **A** is totally unimodular and stays so under the elimination operations, and $\vec{\mathbf{b}}$ is integral and stays so under the elimination operations, there is at least one integer in this range. □

Corollary 6.1. *Let*

$$\begin{aligned}\mathbf{R}\colon\ & \exists x_1 \in [a^1, b^1]\ \forall y_1 \in \{c^1 - d^1\}\\ & \exists x_2 \in [a^2, b^2]\ \forall y_2 \in \{c^2 - d^2\}\\ & \cdots \exists x_n \in [a^n, b^n]\ \forall y_n \in \{c^n - d^n\}\\ & \quad \mathbf{A}\cdot[\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \le \vec{\mathbf{b}}\end{aligned} \tag{15}$$

be true, where **A** *is TUM. Then* x_1 *can always be chosen integral.*

Proof. Follows from Theorems 5.1 and 6.1. □

Theorem 6.2.

$$\begin{aligned}\mathbf{L}\colon\ & \exists x_1 \in \{a^1 - b^1\}\ \forall y_1 \in \{c^1 - d^1\}\\ & \exists x_2 \in \{a^2 - b^2\}\ \forall y_2 \in \{c^2 - d^2\}\\ & \cdots \exists x_n \in \{a^n - b^n\}\ \forall y_n \in \{c^n - d^n\}\\ & \quad \mathbf{A}\cdot[\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \le \vec{\mathbf{b}}\\ & \Leftrightarrow\\ \mathbf{R}\colon\ & \exists x_1 \in [a^1, b^1]\ \forall y_1 \in \{c^1 - d^1\}\\ & \exists x_2 \in [a^2, b^2]\ \forall y_2 \in \{c^2 - d^2\}\\ & \cdots \exists x_n \in [a^n, b^n]\ \forall y_n \in \{c^n - d^n\}\\ & \quad \mathbf{A}\cdot[\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \le \vec{\mathbf{b}},\end{aligned} \tag{16}$$

where **A** *is TUM.*

Proof. Let $\mathbf{X_L}$ and $\mathbf{Y_L}$ denote the existential and universal players of game $\mathbf{L}$, respectively. Likewise, let $\mathbf{X_R}$ and $\mathbf{Y_R}$ denote the existential and universal players of game $\mathbf{R}$.

$\mathbf{L} \Rightarrow \mathbf{R}$ is straightforward. If there exists a winning strategy for $\mathbf{X_L}$ against $\mathbf{Y_L}$, when $\mathbf{X_L}$ is forced to choose his ith move from the discrete interval $\{a^i - b^i\}$, then the same strategy will also be winning for $\mathbf{X_L}$, when he is allowed to choose his ith move from the continuous interval $[a^i - b^i]$. Thus $\mathbf{X_R}$ can adopt the same strategy against $\mathbf{Y_R}$ and win.

We focus on proving $\mathbf{R} \Rightarrow \mathbf{L}$. Let $\mathbf{X_R}$ have a winning strategy against $\mathbf{Y_R}$ in game $\mathbf{R}$.

From the hypothesis, we know that there exists a rational $x_1 \in [a_1, b_1]$, for all integral values of $y_1 \in \{c_1 - d_1\}$, there exists a rational $x_2 \in [a_2, b_2]$ (which could depend on y_1), for all integral values of $y_2 \in \{c_2 - d_2\}, \ldots,$ there exists a rational $x_n \in [a_n, b_n]$ (which could depend upon $y_1, y_2, \ldots, y_{n-1}$), for all integral values of $y_n \in \{c_n - d_n\}$ such that $\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$. From Corollary 6.1, we know that x_1 can always be guessed integral (say p_1), since $\mathbf{A}$ is TUM. Since y_1 must be guessed integral (say q_1), at the end of round 1, we have guessed two integers and the constraint system $\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$ is transformed into $\mathbf{A}' \cdot [\vec{\mathbf{x}}'\ \ \vec{\mathbf{y}}']^{\mathbf{T}} \leq \vec{\mathbf{b}}'$, where $\mathbf{A}'$ is obtained by deleting the first column ($\vec{\mathbf{a_1}}$) and the $(n+1)$th column ($\vec{\mathbf{a_{n+1}}}$) of $\mathbf{A}$, $\vec{\mathbf{x}}' = [x_2, x_3, \ldots, x_n]^T$, $\vec{\mathbf{y}}' = [y_2, y_3, \ldots, y_n]^T$, and $\vec{\mathbf{b}}' = \vec{\mathbf{b}} - \vec{\mathbf{p_1}} - \vec{\mathbf{q_1}}$, where $\vec{\mathbf{p_1}}$ is the m-vector $(p_1 \cdot \vec{\mathbf{a_1}})$ and $\vec{\mathbf{q_1}}$ is the m-vector $(q_1 \cdot \vec{\mathbf{a_{n+1}}})$. Note that once again $\mathbf{A}'$ is TUM and $\vec{\mathbf{b}}'$ is integral. So x_2 can be guessed integral and y_2 must be integral. Thus the game can be played with the $\mathbf{X}$ constantly guessing integral values and $\mathbf{Y}$ being forced to make integral moves. From the hypothesis $\mathbf{R}$ is true; it follows that $\mathbf{L}$ is true. □

Theorem 6.3. *TQIPs can be decided in polynomial time.*

Proof. Use Theorem 6.2 to relax the ranges of the existentially quantified variables and Theorem 5.1 to relax the ranges of the universally quantified variables to get a TQLP; then use the algorithm in [Su3] to decide the TQLP in polynomial time. □

Remark 6.1. We have thus shown that when the constraint matrix representing a system of linear inequalities is totally unimodular, a QIP can be relaxed to a QLP in exactly the same way that a traditional Integer Program can be relaxed to a traditional Linear Program.

7. Planar Quantified Integer Programs

In this section we consider problem $\mathbf{P_2}$, i.e., Quantified Integer Programming in the x_1-y_1 plane. The constraints of the PQIP will be of the form

$$\begin{array}{lll} a_1 \cdot x_1 & +b_1 \cdot y_1 & \leq c_1, \\ a_2 \cdot x_1 & +b_2 \cdot y_1 & \leq c_2, \\ \vdots & \vdots & \vdots \\ a_m \cdot x_1 & +b_m \cdot y_1 & \leq c_m. \end{array}$$

Note that there is no restriction on the coefficients $\{a_i, b_i\}$, other than integrality. We once again point out that the class of PQIPs is not QIP (2) (*see Section* 10).

We use a case-by-case analysis to show that the language of PQIPs can be decided in polynomial time (except for one case, in which the running time is pseudo-polynomial). There are precisely two quantifiers Q_1 and Q_2 and hence the quantifier string has to take one of the following four forms:

1. $Q_1 = \exists,\ Q_2 = \exists$. This case is Integer Programming in the plane; consequently, we can use the algorithm outlined in [Le] that guarantees polynomial time convergence.
2. $Q_1 = \exists,\ Q_2 = \forall$. Our QIP is

$$\exists x_1 \in \{a^1 - b^1\}\ \forall y_1 \in \{c^1 - d^1\} \quad \mathbf{A}\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \le \vec{\mathbf{b}}, \tag{17}$$

where a^1, b^1, c^1, d^1 are all integers.

Using Theorem 5.2, we know that System (17) is equivalent to

$$\exists x_1 \in \{a^1 - b^1\}\ \forall y_1 \in [c^1, d^1] \quad \mathbf{A}\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \le \vec{\mathbf{b}}. \tag{18}$$

Since y_1 is a continuous valued variable, we can eliminate it using the following procedure: substitute $y_1 = d^1$ in every constraint that can be written in the form $y_1 \le ()$ and $y_1 = c^1$ in every constraint that can be written in the form $y_1 \ge ()$. From [Su3], we know that the elimination is solution preserving. The result of eliminating y_1 is a linear system in one variable, namely, x_1; consequently it is trivial to check for the existence of a lattice point in the intersection of $\{a^1 - b^1\}$ with the resultant interval.
3. $Q_1 = \forall,\ Q_2 = \exists$. The QIP has the form

$$\forall y_1 \in \{c^1 - d^1\}\ \exists x_1 \in \{a^1 - b^1\} \quad \mathbf{A}\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \le \vec{\mathbf{b}}. \tag{19}$$

Observe that the constraint system $\mathbf{A} \cdot [x_i \quad y_1]^T \le \vec{\mathbf{b}}$ is a polyhedron in two-dimensional space, as shown in Figure 1.

This polygon has at most $O(m^2)$ vertices, where m denotes the number of constraints in the constraint system, i.e., the number of rows in $\mathbf{A}$, since in the worst case, every pair of constraints could determine a vertex of this polygon.

Assume that the requirement $\exists x_1 \in \{a^1 - b^1\}$ is part of the constraint matrix $\mathbf{A}$, so that Specification (19) can be interpreted as follows: *Does the line $y_1 = z$ intersect at least one lattice point in the polyhedron $\mathbf{A} \cdot [x_1 \quad y_1]^T \le \vec{\mathbf{b}}$ for each $z \in \{c^1 - d^1\}$*. Answering the above question through enumeration leads to a pseudo-polynomial time algorithm, since the running time of the enumeration algorithm depends on $|d^1 - c^1|$.

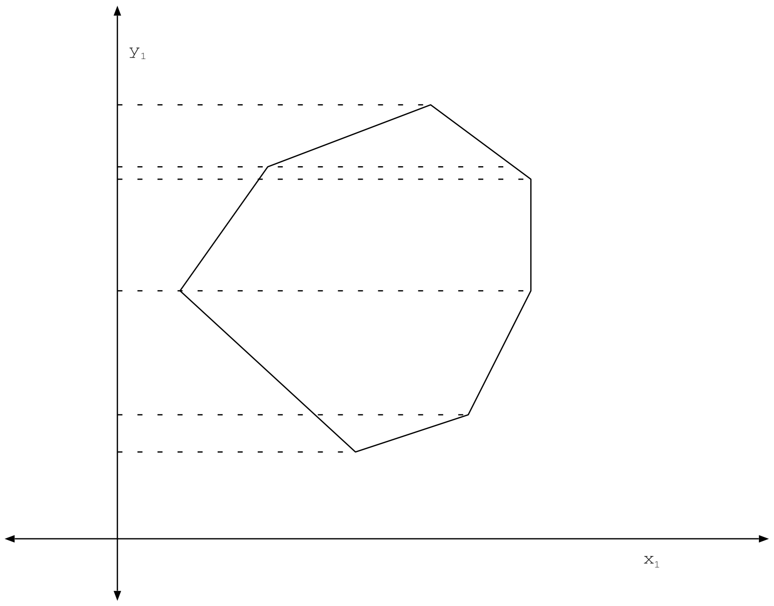

Fig. 1. Checking the truth of ∀∃ in PQIPs.

```
Function PQLP-DECIDE($\mathbf{A}$, $\vec{\mathbf{b}}$, $c^1$, $d^1$)
1: for ($i = c^1$ to $d^1$) do
2:     Substitute $y_1 = i$ in System (19)
3:     Let $L_i$ denote the resultant interval describing $x_1$
4:     if ( $L_i \cap \{a^1 - b^1\}$ does not contain a lattice point) then
5:         return( false )
6:     end if
7: end for
8: return( true )
```

Algorithm 7.1. Handling the ∀∃ case in PQIPs.

Note that the running time of Algorithm 7.1 is pseudo-polynomial in the size of the input and not polynomial, in that the running time is $O(|d^1 - c^1| \cdot m)$.

Observe that System (19) can also be written in the form

$$\forall y_1 \in \{c^1 - d^1\} \; \exists x_1 \in \{a^1 - b^1\} \quad x_1 \cdot \vec{\mathbf{g}} + y_1 \cdot \vec{\mathbf{h}} \leq \vec{\mathbf{b}}. \tag{20}$$

When all the entries of $\vec{\mathbf{h}}$ have the same sign, the solution space for x_1 is convex and a polynomial time algorithm exists to decide System (20).

We now discuss a procedure which is still pseudo-polynomial, but extremely practical for empirical cases. We make the following observations:

(a) Every planar polyhedron, i.e., polygon, can be partitioned into slices, such that each slice is composed of precisely two constraints, namely, a constraint which can be written in the form $x_1 \geq g_1 \cdot y_1 + f_1$ and a constraint which can be written in the form $x_1 \leq g_2 \cdot y_1 + f_2$ [GO] (see Figure 1).

(b) There are at most $O(m^2)$ such slices. Each slice corresponds to two intercepts on the y_1-axis.

(c) Consider the slice l_1 represented by the constraints $x_1 \geq g_1 \cdot y_1 + f_1$ and $x_1 \leq g_2 \cdot y_1 + f_2$. Let u_{l1} and v_{l1} denote the y_1-intercepts of l_1, where $u_{l1} \leq v_{l1}$. Clearly, we must have $c^1 \leq u_{l1}$ and $v_{l1} \leq d^1$. Now consider the following QIP:

$$\begin{aligned} &\forall y_1 \in [u_{l1} - v_{l1}]\ \exists x_1\ \ x_1 \geq g_1 \cdot y_1 + f_1, \\ &x_1 \leq g_2 \cdot y_1 + f_2, \\ &x_1 \text{ integer.} \end{aligned} \tag{21}$$

Without loss of generality, we can assume that u_{l1} and v_{l1} are integers; if not, set $u_{l1} = \lceil u_{l1} \rceil$ and $v_{l1} = \lfloor v_{l1} \rfloor$.

It is clear that the proposition represented by System (19) is true if and only if the proposition represented by System (21) is true, for every one of the $O(m^2)$ slices of the polyhedron.

Thus our procedure to test Specification (19) consists of the following steps:

(a) Compute the slices of the polygon represented by the constraint system in polynomial time.

(b) Check if any of the slices bears witness to the falsehood of Specification (19), by using the technique outlined above, i.e., enumeration.

(c) If any slice (and there are at most $O(m^2)$ of them) is bereft of integer solutions, we conclude that Specification (19) is false; otherwise, we conclude that Specification (19) is true.

Remark 7.1. Determining the exact complexity of this case is an interesting open problem.

4. $Q_1 = \forall,\ Q_2 = \forall.$ In this case the QIP is of the form

$$\begin{aligned} &\forall y_1 \in \{c^1 - d^1\}\ \forall y_2 \in \{c^2 - d^2\} \\ &\quad \mathbf{A} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \leq \vec{\mathbf{b}}, \end{aligned} \tag{22}$$

which is a Box QIP in two variables and can be solved using the techniques in Section 8.

8. Box Quantified Integer Programs

In this section we address the issue of designing an efficient algorithm for BQIPs. At first glance, the BQIP problem appears to be `coNP-complete`, but as we shall see there exists a strongly polynomial time algorithm for the same.

Theorem 8.1. *Let*

$$\begin{aligned}\mathbf{L}:\quad & \forall y_1 \in \{c^1 - d^1\}\ \forall y_2 \in \{c^2 - d^2\} \cdots \forall y_n \in \{c^n, d^n\}\\ & \mathbf{A}\cdot\vec{\mathbf{y}} \le \vec{\mathbf{b}}\end{aligned}$$

and

$$\begin{aligned}\mathbf{R}:\quad & \forall y_1 \in [c^1, d^1]\ \forall y_2 \in [c^2, d^2] \cdots \forall y_n \in [c^n, d^n]\\ & \mathbf{A}\cdot\vec{\mathbf{y}} \le \vec{\mathbf{b}}.\end{aligned}$$

Then $\mathbf{L} \Leftrightarrow \mathbf{R}$.

Proof. Observe that $\mathbf{R} \Rightarrow \mathbf{L}$ is obvious. We focus on proving $\mathbf{L} \Rightarrow \mathbf{R}$. Assume that $\mathbf{L}$ is true but $\mathbf{R}$ is false. Let $\vec{\mathbf{y}}' \in \Pi_{i=1}^n [c^i, d^i]$ be a point such that $\mathbf{A}\cdot\vec{\mathbf{y}}' \not\le \vec{\mathbf{b}}$. Since $\vec{\mathbf{y}}'$ lies within the axis-parallel hyper-rectangle (`aph`) $\mathbf{E} = \Pi_{i=1}^n [c^i, d^i]$, it can be expressed as a convex combination of the extreme points of $\mathbf{E}$. Accordingly, we can write

$$\begin{aligned}&\vec{\mathbf{y}}' = \sum_{i=1}^{l} \alpha_i \vec{\mathbf{a}}_{\mathbf{i}},\\ &\sum_{i=1}^{l} \alpha_i = 1,\\ &\alpha_i \ge 0, \qquad i = 1, 2, \ldots, l,\end{aligned}$$

where $\vec{\mathbf{a}}_{\mathbf{i}}$, $i = 1, 2, \ldots, l$, denote the extreme points of $\mathbf{E}$. Since $\mathbf{L}$ is true, we know that $\mathbf{A}\cdot\vec{\mathbf{a}}_{\mathbf{i}} \le \vec{\mathbf{b}}$, $i = 1, 2, \ldots, l$.

It therefore follows that

$$\begin{aligned}\mathbf{A}\cdot\vec{\mathbf{y}}' &= \mathbf{A}\cdot\sum_{i=1}^{l}\alpha_i\vec{\mathbf{a}}_{\mathbf{i}},\ \alpha_i \ge 0,\ \sum_{i=1}^{l}\alpha_i = 1\\ &= \sum_{i=1}^{l}\mathbf{A}\cdot\alpha_i\vec{\mathbf{a}}_{\mathbf{i}},\ \alpha_i \ge 0,\ \sum_{i=1}^{l}\alpha_i = 1\\ &= \sum_{i=1}^{l}\alpha_i\cdot\mathbf{A}\cdot\vec{\mathbf{a}}_{\mathbf{i}},\ \alpha_i \ge 0,\ \sum_{i=1}^{l}\alpha_i = 1\\ &\le \sum_{i=1}^{l}\alpha_i\cdot\vec{\mathbf{b}},\ \alpha_i \ge 0,\ \sum_{i=1}^{l}\alpha_i = 1\\ &= \vec{\mathbf{b}},\end{aligned}$$

which contradicts the hypothesis that $\mathbf{A}\cdot\vec{\mathbf{y}}' \not\le \vec{\mathbf{b}}$. Therefore, we must have $\mathbf{L} \Rightarrow \mathbf{R}$ and the theorem is proven. □

Accordingly, query (3) can be reduced to the following query:

$$\begin{aligned}&\forall y_1 \in [c^1, d^1]\ \forall y_2 \in [c^2, d^2] \cdots \forall y_n \in [c^n, d^n]\\ &\qquad \mathbf{A}\cdot\vec{\mathbf{y}} \le \vec{\mathbf{b}}.\end{aligned} \tag{23}$$

Note that System (23) is a QLP and hence we can use the quantifier elimination algorithm described in [Su3] to decide it. This algorithm eliminates the universal quantifiers one after another, starting from the innermost quantifier $\forall y_n \in [l_n, u_n]$. Quantifier y_i is eliminated as follows:

1. Substitute $y_i = c^i$ in each constraint of $\mathbf{A} \cdot \vec{\mathbf{y}} \le \vec{\mathbf{b}}$ that can be written in the form $y_i \ge ()$.
2. Substitute $y_i = d^i$ in each constraint of $\mathbf{A} \cdot \vec{\mathbf{y}} \le \vec{\mathbf{b}}$ that can be written in the form $y_i \le ()$.

It is clear that all the universally quantified variables can be eliminated simultaneously, i.e., for a given constraint we can perform all the substitutions in one sweep. Since there are n variables and m constraints, the total time taken to decide a BQIP is $O(m \cdot n)$.

9. Complexity of Restricted Games

In this section we introduce two classes of integer matrix games that are characterized by the nature of the strategies that are available to the existential player, namely, *Oblivious strategy games* and *Clairvoyant strategy games*. We then analyze the computational complexities of deciding whether the existential player can win in these games.

Definition 9.1. A two-person integer matrix game is said to be an Oblivious strategy game if the existential player has to make all his moves without any knowledge of the universal player's moves.

Note that Oblivious strategy integer matrix games are directly modeled as **E-QIP**s.

Definition 9.2. An **E-QIP** is defined as a QIP in which all the existential quantifiers precede the universal quantifiers.

We can write an arbitrary **E-QIP** as

$$\begin{aligned}&\exists x_1 \in \{a^1 - b^1\}\ \exists x_2 \in \{a^2 - b^2\} \cdots \exists x_n \in \{a^n - b^n\}\\ &\forall y_1 \in \{c^1 - d^1\}\ \forall y_2 \in \{c^2 - d^2\} \cdots \forall y_n \in \{c^n - d^n\}\\ &\qquad \mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \le \vec{\mathbf{b}},\end{aligned} \tag{24}$$

Determining whether the existential player has an oblivious strategy for a given constraint system is helpful in situations where online computation is not feasible.

Theorem 9.1. *The problem of deciding whether an arbitrary constraint system* $\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}] \le \vec{\mathbf{b}}$ *has an Oblivious strategy for the existential player is* `NP-complete`.

Proof. Notice that the problem of deciding whether the existential player has an Oblivious strategy is clearly in `NP`; an NDTM can guess a vector $\vec{\mathbf{x}}$ and we can check whether the resultant BQIP is true in polynomial time. We now establish the `NP-hardness` of the problem. Given an arbitrary integer program $\mathbf{G} \cdot \vec{\mathbf{x}} \leq \vec{\mathbf{b}},\ x_i \in \{a^i - b^i\},\ i = 1, 2, \ldots, n$, we construct the **E-QIP**

$$\begin{aligned}&\exists x_1 \in \{a^1 - b^1\}\ \exists x_2 \in \{a^2 - b^2\} \cdots \exists x_n \in \{a^n - b^n\}\\ &\forall y_1 \in \{c^1 - d^1\}\ \forall y_2 \in \{c^2 - d^2\} \cdots \forall y_n \in \{c^n - d^n\}\\ &\qquad \mathbf{G} \cdot \vec{\mathbf{x}} + \mathbf{O} \cdot \vec{\mathbf{y}} \leq \vec{\mathbf{b}},\end{aligned} \tag{25}$$

where $\mathbf{O}$ is the matrix of all zeros and $c^i = d^i = 0,\ \forall i = 1, 2, \ldots, n$. It is clear that the Integer Program has a solution if and only if the **E-QIP** has one, i.e., if the existential player has an Oblivious strategy. □

Definition 9.3. A two-person integer matrix game is said to be a Clairvoyant strategy game if the universal player has to make all his moves without any knowledge of the existential player's moves.

Note that Clairvoyant strategy games are directly modeled as **F-QIP**s.

Definition 9.4. An **F-QIP** is defined as a QIP in which all the universal quantifiers precede the existential quantifiers.

We can write an arbitrary **F-QIP** as

$$\begin{aligned}&\forall y_1 \in \{c^1 - d^1\}\ \forall y_2 \in \{c^2 - d^2\} \cdots \forall y_n \in \{c^n - d^n\}\\ &\exists x_1 \in \{a^1 - b^1\}\ \exists x_2 \in \{a^2 - b^2\} \cdots \exists x_n \in \{a^n - b^n\}\\ &\qquad \mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}.\end{aligned} \tag{26}$$

The existence of Clairvoyant strategies for constraint systems is an important concern in certain real-time scheduling problems [Su2].

Theorem 9.2. *The problem of deciding whether an arbitrary constraint system* $\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}] \leq \vec{\mathbf{b}}$ *has a Clairvoyant strategy for the existential player is* Π_2 **P**-`complete`, *where* $\Pi_2\ \mathbf{P} = \text{coNP}^{\text{NP}}$.

Proof. We first show that the problem of deciding whether an arbitrary **F-QIP** has a solution belongs to the class Π_2 **P**. Note that given an arbitrary **F-QIP**, we can use an Alternating Turing Machine that starts in state $\forall$, guesses a value for $\vec{\mathbf{y}}$, then switches to state $\exists$ and guesses a value for $\vec{\mathbf{x}}$. If $\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$, then the Turing Machine accepts, otherwise it rejects. To prove Π_2 **P**-hardness, observe that the natural complete problem for the class Π_2 **P** is the QBF

$$\forall y_1, y_2, \ldots, y_n\ \exists x_1, x_2, \ldots, x_n, \varphi, \tag{27}$$

where φ is a boolean formula in CNF [Pa]. Given such a QBF, we transform it into an **F-QIP** as follows:

1. Replace φ by a system of linear inequalities in precisely the same way that a Boolean satisfiability is reduced to Integer Programming.
2. Set $a^i = c^i = 0,\ \forall i = 1, 2, \ldots, n$, and $b^i = d^i = 1,\ \forall i = 1, 2, \ldots, n$. □

10. Conclusion

In this paper we studied a number of interesting classes of the Quantified Integer Programming problem. Our principal contribution has been to demonstrate that QIPs can be relaxed to QLPs, when the constraint matrix is totally unimodular. As discussed in Section 3, Quantified Integer Programming is a versatile tool that is used to model a number of real-world problems. The restrictions to QIPs discussed in this paper seem to capture fairly large subclasses of problems that occur in practical situations. Some of the important open problems are:

1. Are there other restrictions to the quantifier specification or the constraint matrix that can be solved in polynomial time? For instance, in [Su1], we showed that the Q2SAT problem could be modeled as a polynomial-time solvable QIP; our result there is more general than the result in [APT] and uses only polyhedral elimination techniques.
2. What is the simplest restriction to the constraint matrix for which the language of QIPs becomes hard, assuming an arbitrary quantifier string (i.e., unbounded alternation)? An extension to the planar case studied in this paper is QIP(2), which is defined as the class of QIPs in which every constraint has at most two existentially quantified variables. IP(2), i.e., Integer Programming with at most two variables per constraint, is known to be NP-complete [HN], [La]; we believe that QIP(2) is PSPACE-complete.
3. A problem that we are currently working on regards characterizing the structure of *Parametric Lattices*, i.e., lattices whose lattice points are integer functions and not integers. Understanding the structure of these lattices will be crucial towards designing algorithms for general classes of QIPs.

Acknowledgments

We are indebted to a number of colleagues who provided feedback on an earlier draft of this paper.

References

[AMO] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1993.

[APT] B. Aspvall, M. F. Plass, and R. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.

[BG] F. Benhamou and F. Goulard. Universally quantified interval constraints. In *Proceedings of Constraint Programming*, September 2000.

[CD] A. Colmerauer and T. Dao. Expressiveness of full first order constraints in the algebra of finite or infinite trees. In *Proceedings of Constraint Programming*, September 2000.

[CGS] M. Cadoli, A. Giovanardi, and M. Schaerf. An algorithm to evaluate quantified boolean formulae. In *AAAI*-98, July 1998.

[CH] V. Chandru and J. N. Hooker. *Optimization Methods for Logical Inference*. Series in Discrete Mathematics and Optimization. Wiley, New York, 1999.

[FR] M. J. Fischer and M. O. Rabin. Super-Exponential Complexity of Presburger Arithmetic. Project MAC Technical Memorandum 43, MIT, Cambridge, MA, 1974.

[Ga] F. Gavril. An efficiently solvable graph partition problem to which many problems are reducible. *Information Processing Letters*, 45(6):285–290, 1993.

[GJ] M. R. Garey and D. S. Johnson. *Computers and Intractability*: *A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, 1979.

[GO] J. E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry*. CRC Press, Boca Raton, FL, 1997.

[HN] D. S. Hochbaum and J. Naor. Simple and fast algorithms for linear and integer programs with two variables per inequality. *SIAM Journal on Computing*, 23(6):1179–1192, December 1994.

[HW] D. S. Hirschberg and C. K. Wong. A polynomial algorithm for the knapsack problem in 2 variables. *Journal of the ACM*, 23(1):147–154, 1976.

[IMM] S. Iwata, T. Matsui, and S. T. McCormick. A fast bipartite network flow algorithm for selective assembly. *Operations Research Letters*, 22:137–143, 1998.

[Jo] D. S. Johnson. Personal communication.

[Ka] R. Kannan. A polynomial algorithm for the two-variable integer programming problem. *Journal of the ACM*, 27(1):118–122, 1980.

[La] J. C. Lagarias. The computational complexity of simultaneous Diophantine approximation problems. *SIAM Journal on Computing*, 14(1):196–209, 1985.

[Le] H. W. Lenstra. Integer Programming with a Fixed Number of Variables. Technical Report 81-03, University of Amsterdam, Amsterdam, 1981.

[LL] C. Lassez and J. L.-Lassez. *Quantifier Elimination for Conjunctions of Linear Constraints via a Convex Hull Algorithm*. Academic Press, New York, 1993.

[LM] J.-L. Lassez and M. Maher. On Fourier's algorithm for linear constraints. *Journal of Automated Reasoning*, to appear.

[McC] S. T. McCormick. Fast algorithms for parametric scheduling come from extensions to parametric maximum flow. *Operations Research*, 47:744–756, 2000.

[NW] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, New York, 1999.

[Pa] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, New York, 1994.

[Pi] M. Pinedo. *Scheduling*: *Theory*, *Algorithms*, *and Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[Pu1] W. Pugh. The Definition of Dependence Distance. Technical Report CS-TR-2292, Department of Computer Science, University of Maryland, College Park, MD, November 1992.

[Pu2] W. Pugh. The omega test: a fast and practical integer programming algorithm for dependence analysis. *Communications of the ACM*, 35(8):102–114, August 1992.

[Re1] P. Revesz. Safe query languages for constraint databases. *ACM Transactions on Database Systems*, 23(1):58–99, 1998.

[Re2] P. Revesz. The evaluation and the computational complexity of datalog queries of boolean constraints. *International Journal of Algebra and Computation*, 8(5):553–574, 1998.

[Re3] P. Revesz. Safe datalog queries with linear constraints. In *Proceedings of the Fourth International Conference on the Principles and Practice of Constraint Programming*, pages 355–369, Pisa, Italy, October 1998. Volume 1520 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, 1998.

[Sc1] T. J. Schaefer. The complexity of satisfiability problems. In Alfred Aho, editor, *Proceedings of the* 10*th Annual ACM Symposium on Theory of Computing*, pages 216–226, New York City, 1978. ACM Press, New York, 1978.

[Sc2] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, New York, 1987.

[SSRB] J. A. Stankovic, M. Spuri, K. Ramamritham, and G. C. Buttazzo, editors. *Deadline Scheduling for Real-Time Systems*. Kluwer Academic, Dordrecht, 1998.

[Su1] K. Subramani. On identifying simple and quantified lattice points in the 2SAT polytope. In J. Calmet et al., editors, *Proceedings of the* 5^{th} *International Conference on Artificial Intelligence and Symbolic Computation* (*AISC*), pages 217–230. Volume 2385 of Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, July 2002.

[Su2] K. Subramani. A specification framework for real-time scheduling. In W. I. Grosky and F. Plasil, editors, *Proceedings of the* 29^{th} *Annual Conference on Current Trends in Theory and Practice of Informatics* (*SOFSEM*), pages 195–207. Volume 2540 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, November 2002.

[Su3] K. Subramani. An analysis of quantified linear programs. In C. S. Calude et al., editors, *Proceedings of the* 4th *International Conference on Discrete Mathematics and Theoretical Computer Science* (*DMTCS*), pages 265–277. Volume 2731 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, July 2003.

[Ta] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, Berkeley, CA, 1951.

[vH] P. van Hentenryck. Personal communication.

Received January 23, 2003, *and in final form January* 17, 2004. *Online publication December* 20, 2004.