Rational and Integer Matrix Games - A BRICS Mini Course

K. Subramani *
LCSEE,
West Virginia University,
Morgantown, WV
{ksmani@csee.wvu.edu}

1 Introduction

In this tutorial, we shall discuss the problems of Linear Programming and Integer Programming, within the framework of Matrix Games.

We shall discuss 2 algorithms for Linear Programming, viz., the Fourier-Motzkin (FM) elimination procedure and the Simplex method.

The FM procedure which was developed by Fourier [Fou24] and rediscovered by Motzkin [DE73] is a incremental, elegant procedure that works well when either the constraint system has a small dimension (2 or 3) or it has at most 2 non-zero variables per row [HN94]. However, in general, it is not a practical algorithm for moderate sized problems, which lack the above properties.

The Simplex method was developed by George B. Dantzig in 1947 and to date, is one of the most popular schemes for implementing linear programming models. Although, based on a number of simple observations, understanding the details of the algorithm requires a certain amount of mathematical maturity.

The pedagogical material for this tutorial has been taken in part from [IC93], [BJ77], [Sch87], [GLS88], [Sub03] and [Sub].

This tutorial has been organized into 4 sessions. The first 2 sessions will be held on August 12, while the remaining 2 sesssions will be held on August 13.

The topics covered in Session I are as follows:

1. Introduction to Linear Programming and Integer Programming.
2. The Fourier Motzkin elimination Procedure.
3. The Simplex Algorithm.

In Session II, the following topics will be covered:

1. The Simplex Algorithm (contd.)
2. Introduction to Quantified Linear Programming.
3. A decision procedure for deciding Quantified Linear Programs.
4. Proof of Correctness.

Session III will be concerned with the following topics:

1. Proof of Correctness (Contd.)
2. Analysis.

*This tutorial was funded in part by Aalborg Universitet, where the instructor was supported by a CISS Faculty Fellowship.

3. Oblivious and Clairvoyant games.

In Session IV, we shall cover the following topics:

1. Introduction to Quantified Integer Programming.
2. Total Unimodularity of the Constraint matrix.

2 Introduction to Linear Programming and Integer Programming

1. Definition.
2. Decision Variables, Constraints, feasibility region.
3. The Diet problem.
4. The Min-Cost Flow problem.
5. The Matchbox problem (Exercise) - A company manufactures 2 types of matchboxes, say T_1 and T_2. Selling a unit of T_1 nets a profit of \$10, while a unit of T_2 nets a profit of \$15. Each matchbox goes through a cutting process and a labeling process. A unit of T_1 requires 2 hours of cutting and an hour of labeling, whereas a unit of T_2 requires 1 hours of cutting and 2 hours of labeling. The company can support at most 10 hours of labeling and at most 10 hours of cutting hours, per week. Due to contractual obligations, the company must make at least 3 units of T_1 per week. What is the optimal product mix for the company?
   Solution: $z = 83\frac{1}{3}$, $3\frac{1}{3}$ units of T_1 and $3\frac{1}{3}$ units of T_2.
6. The Clausal Satisfiability problem.

2.1 The Graphical Method

1.
$$\begin{aligned} \min\ z &= 2x_1 + 5x_2 \\ s.t. & \\ x_1 + x_2 &\geq 6 \\ -x_1 - 2x_2 &\geq -18 \\ x_1, x_2 &\geq 0 \end{aligned}$$
   Solution: $x_1 = 6$, $x_2 = 0$, $z = 12$.

2.
$$\begin{aligned} \max\ z &= x_1 + 3x_2 \\ s.t. & \\ x_1 + x_2 &\leq 6 \\ -x_1 + 2x_2 &\leq 8 \\ x_1, x_2 &\geq 0. \end{aligned}$$
   Solution: $x_1 = \frac{4}{3}$, $x_2 = \frac{14}{3}$, $z = 15\frac{1}{3}$.

This method can be used for the case only when the number of variables is 2 or 3.
Possible events in any linear program:

1. Unique optimum.
2. Alternative optima.
3. Unbounded optimum.
4. Empty feasible region.

3 Fourier-Motzkin Elimination

The Fourier-Motzkin elimination procedure is an elegant, syntactic, variable elimination scheme to solve constraint systems that are comprised of linear inequalities. It was discovered initially by Fourier [Fou24] and later by Motzkin [DE73], who used it to solve general purpose Linear programs.

The key idea in the elimination procedure is that a constraint system in n variables (i.e. $\Re^n$), can be projected onto a space of $n-1$ variables (i.e. $\Re^{n-1}$), without altering the solution space. In other words, polyhedral projection of a constraint set is solution preserving. This idea is applied recursively, till we are left with a single variable (say x_1). If we have $a \leq x_1 \leq b, a \leq b$, then the system is consistent, for any value of x_1 in the interval $[a, b]$. Working backwards, we can deduce the values of all the variables $x_2, \ldots, x_n$. If $a > b$, we conclude that the system is infeasible.

Algorithm (3.1) is a formal description of the above procedure.

Function FOURIER-MOTZKIN ELIMINATION $(\mathbf{A}, \vec{\mathbf{b}})$

1: **for** ($i = n$ **down to** 2) **do**
2: Let $\mathbf{I}^+ = \{$ set of constraints that can be written in the form $x_i \geq ()\}$
3: Let $\mathbf{I}^- = \{$ set of constraints that can be written in the form $x_i \leq ()\}$
4: **for** (each constraint $k \in \mathbf{I}^+$) **do**
5: **for** (each constraint $l \in \mathbf{I}^-$) **do**
6: Add $k \leq l$ to the original constraints
7: **end for**
8: **end for**
9: Delete all constraints containing x_i
10: **end for**
11: **if** ($a \leq x_1 \leq b, \quad a, b \geq 0$) **then**
12: `Linear program is consistent`
13: **return**
14: **else**
15: `Linear program is inconsistent`
16: **return**
17: **end if**

Algorithm 3.1: The Fourier-Motzkin Elimination Procedure

Though elegant, this syntactic procedure suffers from an exponential growth in the constraint set, as it progresses. This growth has been observed both in theory [Sch87] and in practice [HLL90, LM91]. By appropriately choosing the constraint matrix $\mathbf{A}$, it can be shown that eliminating k variables causes the size of the constraint set to increase from m to $O(m^{2^k})$ [Sch87]. Algorithm (3.1) remains useful though as a tool for proving theorems on polyhedral spaces [VR99]. [Sch87] gives a detailed exposition of this procedure.

3.1 One Person Rational Matrix games

Both Linear Programming and Integer Programming can be thought as of guessing a vector that satisfies certain conditions. If the guessed vector meets the conditions, the game is won by the player; otherwise the game is lost.

4 Basic Feasible solutions

1. Extreme points and Optimality.
2. Basic Feasible Solutions (BFS).
3. Degeneracy.
4. Correspondence between Basic Feasible Solutions and Extreme Points.

5 The Simplex Method

1. Improving a BFS.
2. Picking a variable to enter the current Basis.
3. Picking a variable to leave the current Basis.
4. Termination and Unboundedness.

Algorithm (5.1) summarizes the above discussion.

Function SIMPLEX ($\mathbf{A}$, $\vec{\mathbf{b}}$, $\vec{\mathbf{c}}$)

1: We are solving the linear program

$$\begin{aligned} \min\ & \vec{\mathbf{c}} \cdot \vec{\mathbf{x}} \\ \mathbf{A} \cdot \vec{\mathbf{x}} &= \vec{\mathbf{b}} \\ \vec{\mathbf{x}} &\geq \vec{\mathbf{0}}. \end{aligned}$$

2: Assume that you are given an initial feasible basis $\mathbf{B}$.
3: Partition $\mathbf{A}$ into $(\mathbf{B}\ :\ \mathbf{N})$, $\vec{\mathbf{x}}$ into $(\vec{\mathbf{x_B}},\ \vec{\mathbf{x_N}})$ and $\vec{\mathbf{c}}$ into $(\vec{\mathbf{c_B}},\ \vec{\mathbf{c_N}})$.
4: Let J denote the index set of the nonbasic variables.
5: Set $\vec{\mathbf{x_B}} = \mathbf{B^{-1}}\vec{\mathbf{b}}$, $\vec{\mathbf{x_N}} = \vec{\mathbf{0}}$ and $z = \vec{\mathbf{c_B}} \cdot \vec{\mathbf{x_B}}$.
6: Set $\vec{\mathbf{w}} = \vec{\mathbf{c_B}}\mathbf{B^{-1}}$.
7: Calculate $z_j - c_j$ for all the nonbasic variables. Let

$$z_k - c_k = \max_{j \in J} z_j - c_j$$

8: **if** $(z_k - c_k) \leq 0$ **then**
9: **return**(The current basis is optimal.)
10: **end if**
11: Solve the system $\mathbf{B} \cdot \vec{\mathbf{y_k}} = \vec{\mathbf{a_k}}$.
12: **if** $\vec{\mathbf{y_k}} \leq \vec{\mathbf{0}}$ **then**
13: **return**(The optimal solution is unbounded.)
14: **end if**
15: x_k (actually a_k) enters the basis and x_{B_r} leaves the basis, where the index r is determined by the Minimum Ratio Test:

$$\frac{\overline{b_r}}{y_{rk}} = \min_{1 \leq i \leq m} \frac{\overline{b_i}}{y_{ik}} : y_{ik} > 0$$

16: Update the basis $\mathbf{B}$, the index set J and Goto Step 3 :.

Algorithm 5.1: The Simplex Algorithm

6 Quantified Linear Programming

Quantified Linear Programming is the problem of checking whether a polyhedron specified by a linear system of inequalities is non-empty, with respect to a specified quantifier string. Quantified Linear Programming subsumes traditional Linear Programming, since in traditional Linear Programming, all the program variables are

existentially quantified (implicitly), whereas, in Quantified Linear Programming, a program variable may be existentially quantified or universally quantified over a continuous range. On account of the alternation of quantifiers in the specification of a Quantified Linear Program (QLP), this problem is non-trivial. QLPs represent a class of Declarative Constraint Logic Programs (CLPs) that are extremely rich in their expressive power.

We are interested in deciding the following query:

$$\mathbf{G} : \exists x_1 \in [a_1, b_1]\ \forall y_1 \in [l_1, u_1]\ \exists x_2 \in [a_2, b_2]\ \forall y_2 \in [l_2, u_2]\ \ldots \exists x_n \in [a_n, b_n]\ \forall y_n \in [l_n, u_n]$$
$$\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}\ ,\ \vec{\mathbf{x}} \geq \vec{\mathbf{0}}? \qquad (1)$$

where

- $\mathbf{A}$ is an $m \times 2 \cdot n$ matrix called the constraint matrix,
- $\vec{\mathbf{x}}$ is a $n-$vector, representing the control variables (these are existentially quantified)
- $\vec{\mathbf{y}}$ is a $n-$vector, representing the variables that can assume values within a pre-specified range; i.e., component y_i has a lower bound of l_i and an upper bound of u_i (these are universally quantified);
- $\vec{\mathbf{b}}$ is an $m-$vector

The pair $(\mathbf{A}, \vec{\mathbf{b}})$ is called the *Constraint System*. Without loss of generality, we assume that the quantifiers are strictly alternating, since we can always add dummy variables (and constraints, if necessary) without affecting the correctness or complexity of the problem.

6.1 Constraint Satisfaction and Model Verification

Definition 6.1 *Let* $\mathbf{G} =< \mathbf{Q}(\vec{\mathbf{x}}, \vec{\mathbf{y}}), (\mathbf{A},\ \vec{\mathbf{b}}) >$ *represent an arbitrary Parametric Polytope (QLP) in the form specified by System (1). We say that* $\mathbf{G}$ *is true or non-empty if there exists an* $x_1 \in \Re$*, such that for all* $y_1 \in [l_1, u_1]$ *(which could depend upon* x_1*), there exists an* $x_2 \in \Re$ *(which could depend upon* y_1*), there exists a* $y_2 \in [l_2, u_2]$ *(which could depend upon* x_1 *and* x_2*) and so on such that* $\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$*, where* $\vec{\mathbf{x}} = [x_1, x_2, \ldots, x_n]^T$ *and* $\vec{\mathbf{y}} = [y_1, y_2, \ldots, y_n]^T$*.*

In order to better understand Definition (6.1), we resort to the following 2-person game. Let $\mathbf{X}$ denote the existential player and $\mathbf{Y}$ denote the Universal player. The game is played in a sequence of $2 \cdot n$ rounds, with $\mathbf{X}$ making his i^{th} move, x_i, in round $2 \cdot i - 1$ and $\mathbf{Y}$ making his i^{th} move, y_i, in round $2 \cdot i$. The initial constraint system $\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$ is referred to as the initial board configuration. The following conventions are followed in the game:

1. $x_i, y_i \in \Re,\ y_i \in [l_i, u_i]\ i = 1, 2, \ldots, n,$
2. The moves are strictly alternating, i.e., $\mathbf{X}$ makes his i^{th} move before $\mathbf{Y}$ makes his i^{th} move, before $\mathbf{X}$ makes his $(i+1)^{th}$ move and so on,
3. When either player makes a move, the configuration of the board changes; for instance, suppose that $\mathbf{X}$ makes the first move as 5. The current configuration is then transformed from $\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$ to $\mathbf{A'} \cdot [\vec{\mathbf{x'}}\ \ \vec{\mathbf{y'}}]^{\mathbf{T}} \leq \vec{\mathbf{b'}}$, where $\mathbf{A'}$ is obtained from $\mathbf{A}$, by dropping the first column, $\vec{\mathbf{x'}} = [x_2, x_3, \ldots, x_n]^T$ and $\vec{\mathbf{b'}} = \vec{\mathbf{b}} - 5 \cdot \vec{\mathbf{a_1}}$, with $\vec{\mathbf{a_1}}$ denoting the first column of $\mathbf{A}$.
4. The i^{th} move made by $\mathbf{X}$, viz., x_i may depend upon the current board configuration as well as the first $(i-1)$ moves made by $\mathbf{Y}$; likewise, y_i may depend upon the current board configuration and the first i moves made by $\mathbf{X}$.
5. Let $\vec{\mathbf{x_1}}$ denote the *numerical* vector of the n moves made by $\mathbf{X}$; $\vec{\mathbf{y_1}}$ is defined similarly. If $\mathbf{A} \cdot [\vec{\mathbf{x_1}}\ \ \vec{\mathbf{y_1}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$, then $\mathbf{X}$ is said to have won the game; otherwise, the game is a win for $\mathbf{Y}$. It is important to note that the game as described above is non-deterministic in nature, in that we have not specified *how* $\mathbf{X}$ and $\mathbf{Y}$ make their moves. Further, if it is possible for $\mathbf{X}$ to win the game, then he will make the correct sequence of

moves; likewise, if **X** cannot win the game, then corresponding to every sequence of moves that he makes, **Y** has a corresponding sequence of moves to ensure that at least one constraint in the constraint system is violated. (See [Pap94, HO02].)

6. From the above discussion, it is clear that the moves made by **X** will have the following form:

$$\vec{\mathbf{x}} = [c_1, f_1(y_1), f_2(y_1, y_2), \ldots, f_{n-1}(y_1, y_2, \ldots, y_{n-1})]^T \tag{2}$$

where c_1 is a constant and $x_i = f_{i-1}(y_1, y_2, \ldots, y_{i-1})$ captures the dependence of x_i on the first $(i-1)$ moves of **Y**.

Likewise, the moves made by **Y** have the following form:

$$\vec{\mathbf{y}} = [g_1(x_1), g_2(x_1, x_2), \ldots, g_n(x_1, x_2, \ldots, x_n)]^T \tag{3}$$

The $f_i()$ and $g_i()$ are Skolem functions.

The following phrases are equivalent and will be used interchangeably for the rest of this paper:

(a) $\vec{\mathbf{x}}$ is a solution vector of **G**,
(b) $\vec{\mathbf{x}}$ is a model for **G**,
(c) $\vec{\mathbf{x}}$ satisfies **G**,
(d) $\vec{\mathbf{x}}$ is appropriate for **G**,
(e) $\vec{\mathbf{x}} \in \mathbf{G}$,
(f) $\vec{\mathbf{x}}$ is a winning strategy for **G**.

Note that corresponding to any game (QLP) **G**, either the Existential player (**X**) has a winning strategy against all strategies employed by the Universal player (**Y**) or (mutually exclusively) the Universal player (**Y**) has a winnging strategy, against all strategies employed by the Existential player. However a specific strategy $\vec{\mathbf{x}}$, for **X** may or may not be winning; if it is not a winning strategy for **X**, then **Y** has a winning strategy $\vec{\mathbf{y}}(\vec{\mathbf{x}})$, *corresponding to* $\vec{\mathbf{x}}$.

Suppose that a solution vector $\vec{\mathbf{x}}$ in the form described by Equation (2) is given; then the above model verification algorithm requires an **infinite precision** Alternating Turing Machine, since there is no guarantee that the guessed values have polynomial size or even finite size!

7 A Decision Procedure for Quantified Linear Programs

7.1 Analysis

Lemma 7.1 *Given an $m \times n$ totally unimodular constraint matrix, Algorithm (7.1) runs in polynomial time.*

Proof: We have observed previously that eliminating a universally quantified variable does not increase the number of constraints and hence can be implemented in time $O(m \cdot n)$, through variable substitution. Further, since it corresponds to simple deletion of a column, the matrix stays totally unimodular. From [Sub03], we know that there are at most $O(n^2)$ non-redundant constraints. Eliminating an existentially quantified variable, could create at most as $O(n^4)$ constraints, since $m = O(n^2)$ [VR99]. A routine to eliminate the redundant constraints can be implemented in time $n \times O(n^4 \cdot \log n^4) = O(n^5 \cdot \log n)$ through a sort procedure. (There are $O(n^4)$ row vectors in all; comparing two row vectors takes time $O(n)$.) The procedures PRUNE-CONSTRAINTS() and CHECK-INCONSISTENCY() work only with single variable constraints and hence both can be implemented in time $O(m \cdot n)$. Thus a single iteration of the i loop in Algorithm (7.1) takes time at most $O(n^5 \cdot \log n)$ and hence the total time taken by Algorithm (7.1) to decide a given QLP is at most $O(n^6 \cdot \log n)$. □

Remark 7.1 *We could use a variation of* RADIX-SORT(), *to achieve sorting in time $O(n \times n^4)$, to give a total running time of $O(n^6)$.*

Function QLP-DECIDE $(\mathbf{A}, \vec{\mathbf{b}})$

$\mathbf{A'_{n+1}} = \mathbf{A}; \vec{\mathbf{b}}'_{\mathbf{n+1}} = \vec{\mathbf{b}}$
for $(i = n$ **down to** $2)$ **do**
  $(\mathbf{A'_i}, \vec{\mathbf{b}}'_{\mathbf{i}}) =$ ELIM-UNIV-VARIABLE $(\mathbf{A'_{i+1}}, \vec{\mathbf{b}}'_{\mathbf{i+1}}, y_i, l_i, u_i)$
  $(\mathbf{A'_i}, \vec{\mathbf{b}}'_{\mathbf{i}}) =$ ELIM-EXIST-VARIABLE $(\mathbf{A'_i}, \vec{\mathbf{b}}'_{\mathbf{i}}, x_i)$
  if (CHECK-INCONSISTENCY()) **then**
    return (false)
  end if
  PRUNE-CONSTRAINTS()
end for
$(\mathbf{A'_1}, \vec{\mathbf{b}}'_{\mathbf{1}}) =$ELIM-UNIV-VARIABLE $(\mathbf{A'_2}, \vec{\mathbf{b}}'_{\mathbf{2}}, y_1, l_1, u_1)$
{After the elimination of y_1, the original system is reduced to a one-variable system, i.e., a series of intervals on the x_1-axis. We can therefore check whether this system provides an interval or declares an inconsistency. An interval results if after the elimination of redundant constraints, we are left with $x_1 \geq a, x_1 \leq b, a \leq b$; an inconsistency results if we are left with $x_1 \geq a, x_1 \leq b, b < a$.}
if $(a \leq x_1 \leq b, \quad a, b \geq 0, \quad a \leq b)$ **then**
  `System is feasible`
  return
else
  `System is infeasible`
  return
end if

Algorithm 7.1: A Quantifier Elimination Algorithm for deciding Query **G**

Function ELIM-UNIV-VARIABLE $(\mathbf{A}, \vec{\mathbf{b}}, y_i, l_i, u_i)$

{Every constraint involving the variable y_i can be re-written in the form $y_i \leq ()$ or (exclusively) $y_i \geq ()$, i.e., in a way that the coefficient of y_i is $+1$.}
Substitute $y_i = l_i$ in each constraint that can be written in the form $y_i \geq ()$
Substitute $y_i = u_i$ in each constraint that can be written in the form $y_i \leq ()$
Create the new coefficient matrix $\mathbf{A'}$ and the new vector $\vec{\mathbf{b}}'$ after the requisite manipulations
return$(\mathbf{A'}, \vec{\mathbf{b}}')$

Algorithm 7.2: Eliminating Universally Quantified variable $y_i \in [l_i, u_i]$

Function ELIM-EXIST-VARIABLE $(\mathbf{A}, \vec{\mathbf{b}}, x_i)$

Form the set $L_{\leq}$ of every constraint that can be written in the form $x_i \leq ()$. If $x_i \leq m_j$ is a constraint in $(\mathbf{A}, \vec{\mathbf{b}})$, m_j is added to $L_{\leq}$.
Form the set $L_{\geq}$ of every constraint that can be written in the form $x_i \geq ()$. Corresponding to the constraint $x_i \geq n_k$ of $(\mathbf{A}, \vec{\mathbf{b}})$, n_k is added to $L_{\geq}$.
Form the set $L_{=}$ of every constraint that does not contain x_i
$\mathcal{L} = \phi$.
for each constraint $m_j \in L_{\leq}$ **do**
  for each constraint $n_k \in L_{\geq}$ **do**
    Create the new constraint $l_{kj} : n_k \leq m_j$; $\mathcal{L} = \mathcal{L} \cup l_{kj}$.
  end for
end for
Create the new coefficient matrix $\mathbf{A'}$ and the new vector $\vec{\mathbf{b}}'$, to include all the constraints in $\mathcal{L} = \mathcal{L} \cup L_{=}$, after the requisite manipulations.
return$(\mathbf{A'}, \vec{\mathbf{b}}')$

Algorithm 7.3: Eliminating Existentially Quantified variable x_i

8 Restricted Games

Definition 8.1 *An* **E***-QLP is a QLP in which all the existential quantifiers precede the universal quantifiers, i.e., a QLP of the form:*

$$\exists x_1 \exists x_2 \ldots \exists x_n \; \forall y_1 \in [l_1, u_1] \forall y_2 \in [l_2, u_2], \ldots \forall y_n \in [l_n, u_n] \;\; \mathbf{A} \cdot [\vec{\mathbf{x}} \;\; \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$$

Definition 8.2 *An* **F***-QLP is a QLP in which all the universal quantifiers precede the existential quantifiers, i.e., a QLP of the form:*

$$\forall y_1 \in [l_1, u_1] \; \forall y_2 \in [l_2, u_2], \ldots \forall y_n \in [l_n, u_n] \; \exists x_1 \; \exists x_2 \ldots \exists x_n \;\; \mathbf{A} \cdot [\vec{\mathbf{x}} \;\; \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}$$

9 Quantified Integer Programming

Definition 9.1 *Let $x_1, x_2, \ldots x_n$ be a set of n variables with integral ranges. A mathematical program of the form*

$$\begin{aligned} Q_1 \; x_1 \in \{a^1 - b^1\} \; Q_2 \; x_2 \in \{a^2 - b^2\}, \ldots \\ Q_n \; x_n \in \{a^n - b^n\} \;\; \mathbf{A} \cdot \vec{\mathbf{x}} \leq \vec{\mathbf{b}} \end{aligned} \tag{4}$$

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

Definition 9.2 *A TQLP is a QLP in which the constraint matrix is totally unimodular.*

Definition 9.3 *A Quantified Integer Program in which some variables have discrete ranges, while the rest have continuous ranges is called a Mixed QIP (MQIP).*

Theorem (9.1) 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 (9.2).

Theorem 9.1

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

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

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

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

$\mathbf{R} \Rightarrow \mathbf{L}$ is straightforward. Suppose that $\mathbf{X_R}$ has a strategy that is winning when the universal player $\mathbf{Y_R}$ can choose his i^{th} 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 i^{th} 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 $\mathbf{L} \Rightarrow \mathbf{R}$. Our proof uses induction on the length of the quantifier string and therefore on the dimension of $\mathbf{A}$. Note that as described in System (1), 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:

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

Let us say that $\mathbf{L}$ 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 $\mathbf{A} \cdot [x_1\ \ y_1]^T \leq \vec{\mathbf{b}}$ as: $x_1 \cdot \vec{\mathbf{g_1}} + y_1 \cdot \vec{\mathbf{h_1}} \leq \vec{\mathbf{b}}$. 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:

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

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

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

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 (9.1) 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:

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

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 §6.1. We consider 2 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 2 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 $\mathbf{G} \cdot \vec{\mathbf{x}} + \mathbf{H'} \cdot \vec{\mathbf{y'}} + y_{m+1} \cdot \vec{\mathbf{h_{m+1}}} \leq \vec{\mathbf{b}}$, 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:

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

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

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

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 Quantified Linear Program [Sub03]; 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 (9.1) is proven. □

Corollary 9.1 *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 9.2 *Let*

$$\begin{aligned}
\mathbf{R}: &\exists x_1 \in [a^1, b^1]\ \forall y_1 \in [c^1, d^1] \\
&\exists x_2 \in [a^2, b^2]\ \forall y_2 \in [c^2, d^2] \\
&\ldots \exists x_n \in [a^n, b^n]\ \forall y_n \in [c^n, d^n] \\
&\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}
\end{aligned} \tag{11}$$

have a model, where $\mathbf{A}$ *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 $\mathbf{R}$ is a QLP, hence we can use the algorithm developed in [Sub03] 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 $\mathbf{A}$ 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 [NW99]). Since $\mathbf{R}$ has a model, the algorithm in [Sub03] determines a range for x_1 of the form $a \leq x_1 \leq b$. Since $\mathbf{A}$ 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 9.2 *Let*

$$\begin{aligned}\mathbf{R}: &\exists x_1 \in [a^1, b^1]\ \forall y_1 \in \{c^1 - d^1\} \\ &\exists x_2 \in [a^2, b^2]\ \forall y_2 \in \{c^2 - d^2\} \\ &\ldots \exists x_n \in [a^n, b^n]\ \forall y_n \in \{c^n - d^n\} \\ &\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}\end{aligned} \tag{12}$$

be true, where $\mathbf{A}$ *is TUM. Then* x_1 *can always be chosen integral.*

Proof: Follows from Theorem (9.1) and Theorem (9.2). □

Theorem 9.3

$$\begin{aligned}\mathbf{L}: &\exists x_1 \in \{a^1 - b^1\}\ \forall y_1 \in \{c^1 - d^1\} \\ &\exists x_2 \in \{a^2 - b^2\}\ \forall y_2 \in \{c^2 - d^2\} \\ &\ldots \exists x_n \in \{a^n - b^n\}\ \forall y_n \in \{c^n - d^n\} \\ &\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}} \\ &\Leftrightarrow \\ \mathbf{R}: &\exists x_1 \in [a^1, b^1]\ \forall y_1 \in \{c^1 - d^1\} \\ &\exists x_2 \in [a^2, b^2]\ \forall y_2 \in \{c^2 - d^2\} \\ &\ldots \exists x_n \in [a^n, b^n]\ \forall y_n \in \{c^n - d^n\} \\ &\mathbf{A} \cdot [\vec{\mathbf{x}}\ \ \vec{\mathbf{y}}]^{\mathbf{T}} \leq \vec{\mathbf{b}}\end{aligned} \tag{13}$$

where $\mathbf{A}$ *is TUM.*

Proof: Let $\mathbf{X_L}$ and $\mathbf{Y_L}$ denote the existential and universal players of the game $\mathbf{L}$ respectively. Likewise, let $\mathbf{X_R}$ and $\mathbf{Y_R}$ denote the existential and universal players of the 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 i^{th} 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 i^{th} 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 against $\mathbf{Y_R}$ in the 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 $y_2 \in \{c_2 - d_2\}$, ..., 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 (9.2), 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 2 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 9.4 *TQIPs can be decided in polynomial time.*

Proof: Use Theorem (9.3) to relax the ranges of the existentially quantified variables and Theorem (9.1) to relax the ranges of the universally quantified variables to get a TQLP; then use the algorithm in [Sub03] to decide the TQLP in polynomial time. □

Remark 9.1 *We have thus shown that when the constraint matrix representing a system of linear inequalities is totally unimodular, a Quantified Integer Program can be relaxed to a Quantified Linear Program, in exactly the same way that a traditional Integer Program can be relaxed to a traditional Linear Program.*

References

[BJ77] Mokhtar S. Bazaara and John J. Jarvis. *Linear Programming and Network Flows.* John Wiley & Sons, 1977.

[DE73] G. B. Dantzig and B. C. Eaves. Fourier-Motzkin Elimination and its Dual. *Journal of Combinatorial Theory (A)*, 14:288–297, 1973.

[Fou24] J. B. J. Fourier. *Reported in: Analyse de travaux de l'Academie Royale des Sciences, pendant l'annee 1824, Partie Mathematique, Historyde l'Academie Royale de Sciences de l'Institue de France 7 (1827) xlvii-lv. (Partial English translation in: D.A. Kohler, Translation of a Report by Fourier on his work on Linear Inequalities. Opsearch 10 (1973) 38-42.).* Academic Press, 1824.

[GLS88] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer, Berlin, 1988.

[HLL90] Tien Huynh, Catherine Lassez, and Jean-Louis Lassez. Fourier Algorithm Revisited. In Hélène Kirchner and W. Wechler, editors, *Proceedings Second International Conference on Algebraic and Logic Programming*, volume 463 of *Lecture Notes in Computer Science*, pages 117–131, Nancy, France, October 1990. Springer-Verlag.

[HN94] Dorit S. Hochbaum and Joseph (Seffi) 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.

[HO02] Lane A. Hemaspaandra and Mitsunori Ogihara. *The Complexity Theory Companion.* Springer-Verlag, New York, 2002.

[IC93] James P. Ignizio and Tom P. Cavalier. *Linear Programming.* Prentice Hall, 1993.

[Joh] D.S. Johnson. Personal Communication.

[LM91] Jean-Louis Lassez and Michael Maher. On fourier's algorithm for linear constraints. *Journal of Automated Reasoning, to appear*, 1991.

[NW99] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, New York, 1999.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, New York, 1994.

[Sch87] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley and Sons, New York, 1987.

[Sub] K. Subramani. An analyis of selected quantified integer programs. Submitted to Theory of Computing Systems.

[Sub03] K. Subramani. An analysis of quantified linear programs. In et. al. C.S. Calude, editor, *Proceedings of the* 4^{th} *International Conference on Discrete Mathematics and Theoretical Computer Science (DMTCS)*, volume 2731 of *Lecture Notes in Computer Science*, pages 265–277. Springer-Verlag, July 2003.

[VR99] V.Chandru and M.R. Rao. Linear programming. In *Algorithms and Theory of Computation Handbook, CRC Press, 1999*. CRC Press, 1999.