A polynomial time algorithm for a class of Quantified Integer Programs

K. Subramani
Department of Computer Science and Electrical Engineering,
West Virginia University,
Morgantown, WV
{ksmani@csee.wvu.edu}

Abstract
It is well known that the Quantified Satisfiability problem (QSAT) is PSPACE-complete. It follows that the problem of deciding the language of 0/1 Quantified Integer Programs (QIPs) i.e., testing whether a linear system of inequalities has a quantified algebraic point is PSPACE-complete. One aspect of research is to focus on designing polynomial time procedures for interesting special cases. In this paper, we show that if the constraint matrix defining a 0/1 QIP is totally unimodular (TUM), then the QIP can be decided in polynomial time.

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 2- person game. Consider a board game comprised of an initial configuration and two players A and B each having a finite set of moves. A can win the game if the decision problem: Given the initial configuration, does A have a first move (policy), such that for all possible first moves of B (imposed stimulus), A has a second move, such that for all possible second moves of B, ... , A eventually wins? can be answered affirmatively. The board configuration can be represented as a boolean expression or a constraint matrix; the effort involved in representing the board configuration typically determines the tractability of the decision problem.

Definition 1.1 Let \( \{x_1, x_2, \ldots, x_n\} \) be a set of \( n \) boolean variables. A disjunction of literals (a literal is either \( x_i \) or its complement \( \overline{x}_i \)) is called a clause, represented by \( C_i \). A satisfiability problem of the form:

\[
Q_1 x_1 Q_2 x_2 \ldots Q_n x_n C
\]

where each \( Q_i \) is either a \( \exists \) or \( \forall \) and \( C = C_1 \land C_2 \ldots \land C_m \), is called a Quantified Satisfiability (QSAT) problem.

QSAT has been shown to be PSPACE-complete, even when there are at most 3 literals per clause (Q3SAT) [Pap94], although polynomial time algorithms exist for the case in which there are at most two literals per clause [APT79, Gav93].

Definition 1.2 Let \( x_1, x_2, \ldots, x_n \) be a set of \( n \) 0/1 variables. An integer program of the form

\[
Q_1 x_1 \in \{0,1\} Q_2 x_2 \in \{0,1\} \ldots Q_n x_n \in \{0,1\} A[x] \leq \vec{b}
\]

where each \( Q_i \) is either \( \exists \) or \( \forall \) is called a 0/1 Quantified Integer Program (QIP).

The PSPACE-completeness of QIPs follows directly from the PSPACE-completeness of QSAT; in fact the reduction from QSAT to QIP is identical to the one from SAT to 0/1 Integer Programming. The matrix \( A \) is called the constraint matrix of the QIP. Without loss of generality, we assume that the quantifiers are strictly alternating, \( Q_i = \exists \); further we denote the existentially quantified variables using \( x_i, i = 1, 2, \ldots, n \) and the universally quantified variables using \( y_i, i = 1, 2, \ldots, n \). Thus we can write an arbitrary 0/1 QIP as:

\[
\exists x_1 \in \{0,1\} \forall y_1 \in \{0,1\} \exists x_2 \in \{0,1\} \forall y_2 \in \{0,1\} \ldots \exists x_n \in \{0,1\} \forall y_n \in \{0,1\} A[x,y]^T \leq \vec{b}
\]

for suitably chosen \( \vec{x}, \vec{y}, A, \vec{b}, n \)
Definition: 1.3 A TQIP is a QIP in which the constraint matrix is totally unimodular.

Definition: 1.4 A linear program of the form
\[ \exists x \in [0,1] \forall y \in [0,1] \exists x_1 \in [0,1] \forall y_1 \in [0,1] \exists x_2 \in [0,1] \forall y_2 \in [0,1] \ldots \exists x_n \in [0,1] \forall y_n \in [0,1] A.[x \ y]^T \leq \bar{b} \]
is called a 0/1 Quantified Linear Program (QLP).

Definition: 1.5 A TQLP is a QLP in which the constraint matrix is totally unimodular.

The complexity of QLPs (0/1 or otherwise) is not known [Joh], although the class of TQLPs can be decided in polynomial time [Sub01a] (See §A).

2 Algorithms and Complexity

Lemma: 2.1
\[ L \iff R \exists x \in [0,1] \exists y \in [0,1] \exists x \in [0,1] A.[x \ y]^T \leq \bar{b} \]
\[ \Leftrightarrow \ R : \exists x \in [0,1] \exists y \in [0,1] \exists x \in [0,1] A.[x \ y]^T \leq \bar{b} \]

Proof: R \Rightarrow L is trivial. We focus on L \Rightarrow R. Pick some vector \( \bar{y} \in \{0,1\}^n \); let \( \bar{x} = [x_1, x_2, \ldots, x_n]^T = [c_0, f_1(y_1), f_2(y_2), \ldots, f_n(y_1, y_2, \ldots, y_n)] \) be such that A.[\bar{x} \ \bar{y}]^T \leq \bar{b} (where the \( f_i \) are the Skolem functions capturing the dependence of \( x_i \) on \( y_1, y_2, \ldots, y_n \) and \( c_0 \) is a constant in [0,1]). Likewise, pick a second vector \( \bar{y}' \in \{0,1\}^n \) and let \( \bar{x}' = [x_1', x_2', \ldots, x_n']^T = f_n(y_1, y_2, \ldots, y_n) \), such that A.[\bar{x}' \ \bar{y}']^T \leq \bar{b} . Now consider the parametric point 
\( y^\lambda = \lambda \bar{y} + (1 - \lambda) \bar{y}' \), 0 \leq \lambda \leq 1. We shall show that the parametric point \( x^\lambda = \lambda \bar{x} + (1 - \lambda) \bar{x}' \), 0 \leq \lambda \leq 1 is such that A.[x^\lambda \ y^\lambda]^T \leq \bar{b} . Observe that A.[x^\lambda \ y^\lambda]^T = A.[\lambda \bar{x} + (1 - \lambda) \bar{x}' \ \lambda \bar{y} + (1 - \lambda) \bar{y}']^T = A.[\lambda \bar{x} \ \lambda \bar{y}]^T + A.[(1 - \lambda) \bar{x}' \ (1 - \lambda) \bar{y}']^T = \lambda A.[\bar{x} \ \bar{y}]^T + (1 - \lambda) A.[\bar{x}' \ \bar{y}']^T \leq \lambda \bar{b} + (1 - \lambda) \bar{b} \leq \bar{b} , since 0 \leq \lambda \leq 1. Thus the feasible solution space of a Quantified Linear Program is convex and the lemma is proven. \qed

Lemma: 2.2
\[ L \iff R \exists x \in [0,1] \exists y \in [0,1] \exists x \in [0,1] A.[x \ y]^T \leq \bar{b} \]
\[ \Leftrightarrow \ R : \exists x \in [0,1] \exists y \in [0,1] \exists x \in [0,1] A.[x \ y]^T \leq \bar{b} \]

Proof: Consider any vector \( \bar{y} = \{0,1\}^n \). Substituting this vector in System (3) results in a standard integer program of the form \( \exists x = \{0,1\}^n G x \leq d \), where \( G \) is totally unimodular. Consequently, this system has a solution if and only if the system \( \exists x = \{0,1\}^n G x \leq d \) is feasible and Lemma (2.2) follows. \qed

Theorem: 2.1 TQIPs can be relaxed to TQLPs, while preserving the integrality of the solution space and hence can be decided in polynomial time.

Proof: Use Lemma (2.1) to relax the universally quantified variables and Lemma (2.2) to relax the existentially quantified variables to get a TQLP; then use Algorithm (A.1) in Appendix §A to decide the TQLP in polynomial time. \qed

3 Conclusion

The technique used in this paper is different from the one used in [Sub01b] to provide a polyhedral projection procedure to decide Quantified 2-SAT problems.
A Deciding Quantified Linear Programs

In this section, we outline the strategy used in [Sub01a] to solve QLPs. The principal idea underlying Algorithm (A.1) is the elimination of the quantified variables while preserving the solution space. Elimination of a universally quantified variable leaves the number of constraints unchanged, whereas the elimination of an existentially quantified variable using a strategy such as Fourier-Motzkin elimination could lead to a quadratic increase in the number of constraints (see [Sch87]); consequently Algorithm (A.1) could take exponential time in the worst case. In the case of TQLPs though, it runs in time $O(n^5 \log n)$, where $n$ represents the number of variables in the QLP.

Fast convergence in TQLPs is guaranteed by the following lemma

**Lemma A.1** Given a totally unimodular matrix $A$ of dimensions $m \times n$, for a fixed $n$, $m = O(n^2)$, if each row is unique.

**Proof:** The above lemma was proved for a superset of totally unimodular matrices viz. totally balanced matrices in [Ans80, AF84]. It therefore follows that Lemma (A.1) is true. $\Box$

The proof of Lemma (A.1) is that a totally unimodular constraint matrix cannot have more than $O(n^2)$ non-redundant constraints. The elimination of an existentially quantified variable through Fourier-Motzkin elimination could potentially result in $O(n^4)$ constraints. Eliminating the redundant constraints is a sort operation, that can be implemented in time $O(n^5 \log n)$ time.\footnote{\(O(n^4)\) row vectors can be sorted in time $n^4 \log n^4$; each comparison takes $O(n)$ time.}

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

1. (The array $Q$ stores the quantifiers i.e. $Q[i] = Q_i$)
2. for ( $i = n$ down to 1 ) do
3. if ( $Q[i] = \exists$ ) then
4. Elim-Univ-Variable($y_i$)
5. if (CHECK-INCONSISTENCY()) then
6. return (false)
7. end if
8. Prune-Constraints()
9. else
10. Elim-Exist-Variable($x_i$)
11. if (CHECK-INCONSISTENCY()) then
12. return (false)
13. end if
14. end if
15. end for
16. System is feasible
17. return

**Algorithm A.1:** A Quantifier Elimination Algorithm for deciding Query $E$

**Function** Elim-Univ-Variable ($A$, $\vec{b}$, $i$)

1. Substitute $x_i = 0$ in each constraint that can be written in the form $x_i \geq ()$
2. Substitute $x_i = 1$ in each constraint that can be written in the form $x_i \leq ()$

**Algorithm A.2:** Eliminating Universally Quantified variable $x_i \in [0, 1]$
References


