Proving Bounds for Real Linear Programs in Isabelle/HOL

TYPES 2004, Jouy-en-Josas

Steven Obua

Technische Universität München

obua@in.tum.de

The Flyspeck Project

Initiated by Tom Hales in 2003.
Goal: *Complete* formalization and proof of the Kepler conjecture within a *mechanical theorem prover*.

The Flyspeck Project

Initiated by Tom Hales in 2003.
Goal: *Complete* formalization and proof of the Kepler conjecture within a *mechanical theorem prover*.

A subproblem: Prove upper bounds for about **100,000** real linear programs, each involving about **200** variables and **2000** constraints.

The General Linear Program (LP)

Given:

The General Linear Program (LP)

Given:

- a matrix $A \in \mathbb{R}^{m \times n}$

The General Linear Program (LP)

Given:

- a matrix $A \in \mathbb{R}^{m \times n}$
- a column vector $b \in \mathbb{R}^{m \times 1}$

The General Linear Program (LP)

Given:

- a matrix $A \in \mathbb{R}^{m \times n}$
- a column vector $b \in \mathbb{R}^{m \times 1}$
- a row vector $c \in \mathbb{R}^{1 \times n}$

The General Linear Program (LP)

Given:

- a matrix $A \in \mathbb{R}^{m \times n}$
- a column vector $b \in \mathbb{R}^{m \times 1}$
- a row vector $c \in \mathbb{R}^{1 \times n}$

Goal:

maximize $c * x$,

where the column vector $x \in \mathbb{R}^{n \times 1}$ is constrained by

$$A * x \leq b.$$

Our Approach

Usual method for solving LP's: *Simplex method*.

Our Approach

Usual method for solving LP's: *Simplex method*.

But our goal is:

- we don't want only an approximate maximum: we also want it to be an ***upper bound***
- we need a ***proof within Isabelle/HOL*** that we actually computed an upper bound

Our Approach

Usual method for solving LP's: *Simplex method*.

But our goal is:

- we don't want only an approximate maximum: we also want it to be an ***upper bound***
- we need a ***proof within Isabelle/HOL*** that we actually computed an upper bound

We want a theorem like

$$A * x \leq b \Longrightarrow c * x \leq \text{bound}$$

Our Approach

Usual method for solving LP's: *Simplex method*.

But our goal is:

- we don't want only an approximate maximum: we also want it to be an ***upper bound***
- we need a ***proof within Isabelle/HOL*** that we actually computed an upper bound

We want a theorem like

$$A * x \leq b \Longrightarrow c * x \leq \text{bound}$$

Proposal by Tom Hales:

Use the solution y of the dual linear program as a *certificate*.

The Dual Linear Program

Given:

- a matrix $A \in \mathbb{R}^{m \times n}$
- a column vector $b \in \mathbb{R}^{m \times 1}$
- a row vector $c \in \mathbb{R}^{1 \times n}$

The Dual Linear Program

Given:

- a matrix $A \in \mathbb{R}^{m \times n}$
- a column vector $b \in \mathbb{R}^{m \times 1}$
- a row vector $c \in \mathbb{R}^{1 \times n}$

Goal:

$$\text{minimize } y * b,$$

where the row vector $y \in \mathbb{R}^{1 \times m}$ is constrained by

$$y * A = c \wedge y \geq 0.$$

Duality Theorem

Given an LP:

- a *feasible solution* is a vector that fulfills the constraints of the LP
- the *feasible region* is the set of all feasible solutions of the LP

Duality Theorem

Given an LP:

- a *feasible solution* is a vector that fulfills the constraints of the LP
- the *feasible region* is the set of all feasible solutions of the LP

Observation: $y * b \geq y * (A * x) = (y * A) * x = c * x$
for all feasible solutions y of the dual LP and all feasible solutions x of the general LP

Duality Theorem

Given an LP:

- a *feasible solution* is a vector that fulfills the constraints of the LP
- the *feasible region* is the set of all feasible solutions of the LP

Observation: $y * b \geq y * (A * x) = (y * A) * x = c * x$
for all feasible solutions y of the dual LP and all feasible solutions x of the general LP

Duality Theorem: If both general and dual LP have feasible solutions, then

$$\min\{y * b \mid \ldots\} = \max\{c * x \mid \ldots\}$$

Steps in Proving a Bound for an LP

Prove a **generic theorem** based on duality that gives an upper bound for any LP that fulfills certain assumptions.

Steps in Proving a Bound for an LP

Prove a **generic theorem** based on duality that gives an upper bound for any LP that fulfills certain assumptions. Then for a given LP:

1. get the LP data
   - right now, the data is loaded from a file
   - in the complete proof of the Kepler Conjecture, the data will arise from previous steps

Steps in Proving a Bound for an LP

Prove a **generic theorem** based on duality that gives an upper bound for any LP that fulfills certain assumptions. Then for a given LP:

1. get the LP data
   - right now, the data is loaded from a file
   - in the complete proof of the Kepler Conjecture, the data will arise from previous steps
2. form the dual LP and calculate the certificate y
   - calculation can be done by any external solver
   - interface to GLPK, CPLEX implemented

Steps in Proving a Bound for an LP

Prove a **generic theorem** based on duality that gives an upper bound for any LP that fulfills certain assumptions. Then for a given LP:

1. get the LP data
   - right now, the data is loaded from a file
   - in the complete proof of the Kepler Conjecture, the data will arise from previous steps
2. form the dual LP and calculate the certificate y
   - calculation can be done by any external solver
   - interface to GLPK, CPLEX implemented
3. instantiate the generic theorem with the LP data and the calculated certificate, then simplify

Generic Theorem for Upper Bound

Data A and c of general LP is known only ***approximately***:

$$A_1 \leq A \leq A_2,\ c_1 \leq c \leq c_2$$

Generic Theorem for Upper Bound

Data A and c of general LP is known only ***approximately***:

$$A_1 \leq A \leq A_2,\ c_1 \leq c \leq c_2$$

We can deal only with ***a priori bounded feasible regions***:

$$r_1 \leq x \leq r_2$$

Generic Theorem for Upper Bound

Data A and c of general LP is known only ***approximately***:

$$A_1 \leq A \leq A_2,\ c_1 \leq c \leq c_2$$

We can deal only with ***a priori bounded feasible regions***:

$$r_1 \leq x \leq r_2$$

Variables that will be instantiated are shown in red:

$$[\![A * x \leq b; A_1 \leq A; A \leq A_2; c_1 \leq c; c \leq c_2; r_1 \leq x; x \leq r_2; 0 \leq y]\!]$$
$$\Longrightarrow c * x$$
$$\leq y * b +$$
$$(\mathit{let}\ s_1 = c_1 - y * A_2; s_2 = c_2 - y * A_1$$
$$\mathit{in}\ {s_2}^+ * {r_2}^+ + {s_1}^+ * {r_2}^- + {s_2}^- * {r_1}^+ + {s_1}^- * {r_1}^-)$$

Before / After Simplification

Before: Variables that will be instantiated are shown in red:

$$[\![A * x \leq b; A_1 \leq A; A \leq A_2; c_1 \leq c; c \leq c_2; r_1 \leq x; x \leq r_2; 0 \leq y]\!]$$
$$\Longrightarrow c * x$$
$$\leq y * b +$$
$$(\mathit{let}\ s_1 = c_1 - y * A_2; s_2 = c_2 - y * A_1$$
$$\mathit{in}\ {s_2}^+ * {r_2}^+ + {s_1}^+ * {r_2}^- + {s_2}^- * {r_1}^+ + {s_1}^- * {r_1}^-)$$

After: Simplified terms without variables are shown in blue:

$$[\![A * x \leq b; A_1 \leq A; A \leq A_2; c_1 \leq c; c \leq c_2; r_1 \leq x; x \leq r_2]\!]$$
$$\Longrightarrow c * x \leq \mathit{bound}$$

Further Details

- Finite Matrices
- Lattice-ordered Rings
- Axiomatic Type Classes
- Sparse Matrix Representation
- Representation of Real Numbers

Finite Matrices

type $'a$ infmatrix $=$ nat $\Rightarrow$ nat $\Rightarrow$ $'a$

Finite Matrices

type 'a infmatrix = nat ⇒ nat ⇒ 'a

typedef 'a matrix =

{A::'a infmatrix | finite {(r, c) | A r c ≠ 0}}

Finite Matrices

type 'a infmatrix = nat ⇒ nat ⇒ 'a

typedef 'a matrix =

{A::'a infmatrix | finite {(r, c) | A r c ≠ 0}}

Operations on Finite Matrices

We can now define the usual operations on matrices:

- Matrix multiplication $A * B$
- Addition $A + B$
- Negation $-A$
- Subtraction $A - B$

Operations on Finite Matrices

We can now define the usual operations on matrices:

- Matrix multiplication $A * B$
- Addition $A + B$
- Negation $-A$
- Subtraction $A - B$

We also need:

- Componentwise comparison $A \leq B$
- Positive part A^+ , negative part A^-

Example: Addition of Finite Matrices

Example: Addition of Finite Matrices

$$\begin{pmatrix} 1 & 7 & 3 \\ 2 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 7 & 3 \\ 2 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Example: Addition of Finite Matrices

$$\begin{pmatrix} 1 & 7 & 3 \\ 2 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 7 & 3 \\ 2 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 \\ 2 \\ 10 \\ 5 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 2 & 0 & 0 \\ 10 & 0 & 0 \\ 5 & 0 & 0 \end{pmatrix}$$

Example: Addition of Finite Matrices

$$\begin{pmatrix} 1 & 7 & 3 \\ 2 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 7 & 3 \\ 2 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 \\ 2 \\ 10 \\ 5 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 2 & 0 & 0 \\ 10 & 0 & 0 \\ 5 & 0 & 0 \end{pmatrix}$$

$$\Rightarrow$$

$$\begin{pmatrix} 1 & 7 & 3 \\ 2 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 2 \\ 10 \\ 5 \end{pmatrix} = \begin{pmatrix} 1 & 7 & 3 \\ 4 & 1 & 0 \\ 10 & 0 & 0 \\ 5 & 0 & 0 \end{pmatrix}$$

Lattice-ordered Rings

In *Lattice theory* (G. Birkhoff, 1967):
the ring of $n \times n$-matrices is a lattice-ordered ring
$\Rightarrow$ this can be adapted to our finite matrices

Lattice-ordered Rings

In *Lattice theory* (G. Birkhoff, 1967):
the ring of $n \times n$-matrices is a lattice-ordered ring
$\Rightarrow$ this can be adapted to our finite matrices

Definition: A *lattice-ordered* ring (l-ring) is a structure, s.t.

Lattice-ordered Rings

In *Lattice theory* (G. Birkhoff, 1967):
the ring of $n \times n$-matrices is a lattice-ordered ring
$\Rightarrow$ this can be adapted to our finite matrices

Definition: A *lattice-ordered* ring (l-ring) is a structure, s.t.

- the structure is a ring

Lattice-ordered Rings

In *Lattice theory* (G. Birkhoff, 1967):
the ring of $n \times n$-matrices is a lattice-ordered ring
$\Rightarrow$ this can be adapted to our finite matrices

Definition: A *lattice-ordered* ring (l-ring) is a structure, s.t.

- the structure is a ring
- the structure is a lattice

Lattice-ordered Rings

In *Lattice theory* (G. Birkhoff, 1967):
the ring of $n \times n$-matrices is a lattice-ordered ring
$\Rightarrow$ this can be adapted to our finite matrices

Definition: A *lattice-ordered* ring (l-ring) is a structure, s.t.

- the structure is a ring
- the structure is a lattice
- the partial order is compatible with addition and multiplication:
  - $a \leq b \ \rightarrow \ c + a \leq c + b$
  - $a \leq b \ \wedge \ 0 \leq c \ \rightarrow \ c * a \leq c * b$
  - $a \leq b \ \wedge \ 0 \leq c \ \rightarrow a * c \leq b * c$

Utilizing Axiomatic Type Classes

Axiomatic type classes are Isabelle's built-in mechanism for abstracting the properties of a type

Utilizing Axiomatic Type Classes

Axiomatic type classes are Isabelle's built-in mechanism for abstracting the properties of a type

- a special axiomatic type class models the property of a type to form a lattice-ordered ring

Utilizing Axiomatic Type Classes

Axiomatic type classes are Isabelle's built-in mechanism for abstracting the properties of a type

- a special axiomatic type class models the property of a type to form a lattice-ordered ring
- lattice-ordered rings have been integrated with Isabelle's numeric type classes:
  types `int` and `real` are lattice-ordered rings

Utilizing Axiomatic Type Classes

Axiomatic type classes are Isabelle's built-in mechanism for abstracting the properties of a type

- a special axiomatic type class models the property of a type to form a lattice-ordered ring
- lattice-ordered rings have been integrated with Isabelle's numeric type classes:
  types `int` and `real` are lattice-ordered rings
- if α is a lattice-ordered ring, then so is α `matrix`

Utilizing Axiomatic Type Classes

Axiomatic type classes are Isabelle's built-in mechanism for abstracting the properties of a type

- a special axiomatic type class models the property of a type to form a lattice-ordered ring
- lattice-ordered rings have been integrated with Isabelle's numeric type classes:
  types `int` and `real` are lattice-ordered rings
- if α is a lattice-ordered ring, then so is α `matrix`

$\Rightarrow$ many already existing theorems and tactics are reused

Bounds for Products in L-Rings

Given elements a and b of an l-ring:

$$\begin{array}{ccccc} a_1 & \leq & a & \leq & a_2 \\ b_1 & \leq & b & \leq & b_2 \end{array}$$

Bounds for $a * b$?

Bounds for Products in L-Rings

Given elements a and b of an l-ring:

$$\begin{array}{ccccc} a_1 & \leq & a & \leq & a_2 \\ b_1 & \leq & b & \leq & b_2 \end{array}$$

Bounds for $a * b$?

Use positive and negative part:

$$\begin{array}{rccccc} & x^+ & = & \text{join } x\ 0 & \geq & 0 \\ & x^- & = & \text{meet } x\ 0 & \leq & 0 \\ \Rightarrow & x & = & x^+ + x^- & & \end{array}$$

Bounds for Products in L-Rings

Given elements a and b of an l-ring:

$$\begin{array}{ccccc} a_1 & \leq & a & \leq & a_2 \\ b_1 & \leq & b & \leq & b_2 \end{array}$$

Bounds for $a * b$?

Use positive and negative part:

$$\begin{array}{rccccc} & x^+ & = & \text{join } x\ 0 & \geq & 0 \\ & x^- & = & \text{meet } x\ 0 & \leq & 0 \\ \Rightarrow & x & = & x^+ + x^- & & \end{array}$$

Upper Bound:

$$a * b$$

Bounds for Products in L-Rings

Given elements a and b of an l-ring:

$$\begin{array}{ccccc} a_1 & \leq & a & \leq & a_2 \\ b_1 & \leq & b & \leq & b_2 \end{array}$$

Bounds for $a * b$?

Use positive and negative part:

$$\begin{array}{rccccc} & x^+ & = & \text{join } x\ 0 & \geq & 0 \\ & x^- & = & \text{meet } x\ 0 & \leq & 0 \\ \Rightarrow & x & = & x^+ + x^- & & \end{array}$$

Upper Bound:

$$a * b \quad = \quad (a^+ + a^-) * (b^+ + b^-)$$

Bounds for Products in L-Rings

Given elements a and b of an l-ring:

$$\begin{array}{ccccc} a_1 & \leq & a & \leq & a_2 \\ b_1 & \leq & b & \leq & b_2 \end{array}$$

Bounds for $a * b$?

Use positive and negative part:

$$\begin{array}{rccccc} & x^+ & = & \text{join } x\ 0 & \geq & 0 \\ & x^- & = & \text{meet } x\ 0 & \leq & 0 \\ \Rightarrow & x & = & x^+ + x^- & & \end{array}$$

Upper Bound:

$$\begin{array}{rcl} a * b & = & (a^+ + a^-) * (b^+ + b^-) \\ & = & a^+ * b^+ + a^+ * b^- + a^- * b^+ + a^- * b^- \end{array}$$

Bounds for Products in L-Rings

Given elements a and b of an l-ring:

$$\begin{array}{ccccc} a_1 & \leq & a & \leq & a_2 \\ b_1 & \leq & b & \leq & b_2 \end{array}$$

Bounds for $a * b$?

Use positive and negative part:

$$\begin{array}{rccccc} & x^+ & = & \text{join } x\ 0 & \geq & 0 \\ & x^- & = & \text{meet } x\ 0 & \leq & 0 \\ \Rightarrow & x & = & x^+ + x^- & & \end{array}$$

Upper Bound:

$$\begin{array}{rcl} a * b & = & (a^+ + a^-) * (b^+ + b^-) \\ & = & a^+ * b^+ + a^+ * b^- + a^- * b^+ + a^- * b^- \\ & \leq & a_2^+ * b_2^+ + a_1^+ * b_2^- + a_2^- * b_1^+ + a_1^- * b_1^- \end{array}$$

Sparse Matrix Representation

We need to actually calculate with matrices
$\Rightarrow$ *sparse* representation of matrices as list of lists:

type 'a spvec = (nat * 'a) list

type 'a spmat = (nat * 'a spvec) list

Sparse Matrix Representation

We need to actually calculate with matrices
$\Rightarrow$ *sparse* representation of matrices as list of lists:

type 'a spvec = (nat * 'a) list

type 'a spmat = (nat * 'a spvec) list

Example:

$$[(1, [(1, a), (3, b)]), (2, [(0, c), (1, d)])]$$

Sparse Matrix Representation

We need to actually calculate with matrices
$\Rightarrow$ *sparse* representation of matrices as list of lists:

type 'a spvec = (nat * 'a) list

type 'a spmat = (nat * 'a spvec) list

Example:

$$[(1, [(1, a), (3, b)]), (2, [(0, c), (1, d)])]$$

$$\xrightarrow{\text{sparse-row-matrix}}$$

Sparse Matrix Representation

We need to actually calculate with matrices
$\Rightarrow$ *sparse* representation of matrices as list of lists:

type 'a spvec = (nat * 'a) list

type 'a spmat = (nat * 'a spvec) list

Example:

$$[(1, [(1, a), (3, b)]), (2, [(0, c), (1, d)])]$$

$$\xrightarrow{\text{sparse-row-matrix}}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & a & 0 & b \\ c & d & 0 & 0 \end{pmatrix}$$

Representation of Real Numbers

constdefs

pow2 :: int $\Rightarrow$ real

pow2 z $\equiv$ if z ≥ 0 then $2^{\text{nat z}}$ else inverse $2^{\text{nat } (-\text{z})}$

float :: int$*$int $\Rightarrow$ real

float (m, e) $\equiv$ (real m) $*$ (pow2 e)

Representation of Real Numbers

constdefs

pow2 :: int $\Rightarrow$ real

pow2 z $\equiv$ if z ≥ 0 then $2^{\text{nat z}}$ else inverse $2^{\text{nat } (-\text{z})}$

float :: int$*$int $\Rightarrow$ real

float (m, e) $\equiv$ (real m) $*$ (pow2 e)

Advantages of this representation:

- simple rules for calculation with floats

Representation of Real Numbers

constdefs

pow2 :: int $\Rightarrow$ real

pow2 z $\equiv$ if z ≥ 0 then $2^{\text{nat z}}$ else inverse $2^{\text{nat } (-\text{z})}$

float :: int$*$int $\Rightarrow$ real

float (m, e) $\equiv$ (real m) $*$ (pow2 e)

Advantages of this representation:

- simple rules for calculation with floats
- given $r \in \mathbb{R}$, and $\delta > 0$, there are m_1, m_2, e_1, e_2 such that

$$r - \delta \leq \text{float}(m_1, e_1) \leq r \leq \text{float}(m_2, e_2) \leq r + \delta$$

Performance Issues

Three options for simplifying:

- Isabelle's simplifier
- Theorem generating code generator (speedup of factor 3.5)
- Reflection via rewriting oracle (further speedup of factor ≈ 100)

Performance Issues

Three options for simplifying:

- Isabelle's simplifier
- Theorem generating code generator (speedup of factor 3.5)
- Reflection via rewriting oracle (further speedup of factor ≈ 100)

Using reflection, a bound for a typical Flyspeck-LP is obtained on a 3Ghz Pentium IV in about

8 seconds

$\Longrightarrow$ the bounds for all LP's should be provable in about

10 days

Future Work / Next Step

Bring together LP's with the work (in progress) of

- Gertrud Bauer: Graph Generation
- Roland Zumkeller: Nonlinear Inequalities

Thanks for listening!!