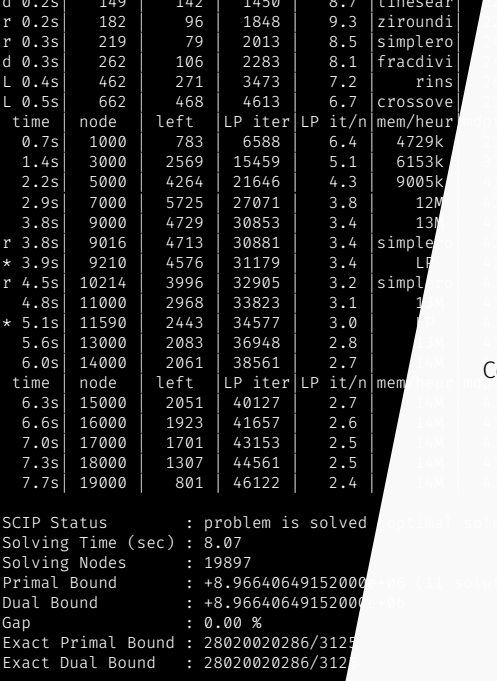


Computational Linear Programming

Ambros Gleixner

HTW Berlin and Zuse Institute Berlin

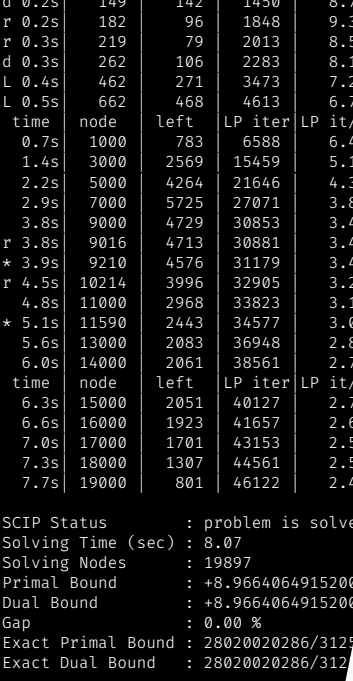
gleixner@htw-berlin.de

Computational Integer Optimization · TU Berlin · August 24, 2026

Hochschule für Technik und Wirtschaft Berlin

University of Applied Sciences

ZIB ZUSE INSTITUTE BERLIN



1. LPs in general form

row representation and basic solutions

2. Dual solutions

certificates of optimality and infeasibility

3. Computational form

basic solutions in column representation

4. Primal and dual simplex

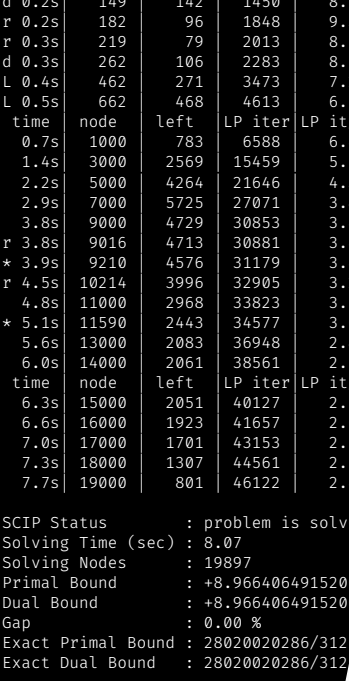
variants and tricks

5. Warmstarting and hotstarting

the simplex engine: LU factorization and updates

6. Beyond simplex

interior point and first-order methods



1. LPs in general form

row representation and basic solutions

2. Dual solutions

certificates of optimality and infeasibility

3. Computational form

basic solutions in column representation

4. Primal and dual simplex

variants and tricks

5. Warmstarting and hotstarting

the simplex engine: LU factorization and updates

6. Beyond simplex

interior point and first-order methods

LPs in General Form

$$
\begin{array}{lll}
\min & c_1 x_1 + \ldots + c_n x_n & \\
\text{s.t.} & L_i \leq a_{i,1} x_1 + \ldots + a_{i,n} x_n \leq U_i & \text{for } i = 1, \ldots, m \\
& \ell_k \leq x_k \leq u_k & \text{for } k = 1, \ldots, n
\end{array}
\quad \Leftrightarrow \quad
\begin{array}{ll}
\min & c^T x \\
\text{s.t.} & L \leq Ax \leq U \\
& \ell \leq x \leq u
\end{array}
$$

where

- lower bounds L_i, ℓ_k may be $-\infty$ and upper bounds U_i, u_k may be $+\infty$,
- so constraints may be inequalities, ranged rows ($-\infty < L_i < U_i < +\infty$) or equations,
- and variables may be free ($-\infty = \ell_k \wedge u_k = +\infty$).
- The term row can refer to a constraint or a row vector $a_{i,*}$,
- the term column can refer to a variable or a column vector $a_{i,*}$ of the constraint matrix A.

Often, A is highly sparse, i.e., most $a_{i,k} = 0$. The other coefficients are called nonzeros.

Vertices = primal feasible basic solutions

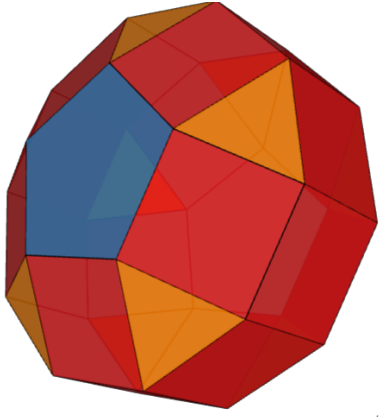
This inequality-based form aligns well with the intuitive 3D visualization of the feasible region as a full-dimensional polytope (sometimes wrong, but nevertheless helpful).

Its vertices have a discrete representation as a set of n linearly independent tight inequalities.

In a simplex solver, this information is encoded for each row and column via the

$$\text{basis status} \in \begin{cases} \text{basic (not tight)} \\ \text{tight on lower bound} \\ \text{tight on upper bound} \\ \text{fixed at zero (also called free/superbasic)} \end{cases}$$

All in all, 4 possibilities = 2 bits per row and column needs $2(m+n)$ bits (less than the numerical solution).



`https://polyhedra.tessera.li/`

Basic solutions in general

In general, any set of n linearly independent tight inequalities defines a unique solution, which can be computed by a **linear system of equations** $Bx = b$ with

- B an $n \times n$ submatrix of $\begin{pmatrix} A \\ I \end{pmatrix}$ corresponding to non-basic (tight) rows and columns,
- b the vector of bounds L_i, U_i, ℓ_k, u_k or 0 according to basis status.

Basic solutions in general

In general, any set of n linearly independent tight inequalities defines a unique solution, which can be computed by a **linear system of equations** $Bx = b$ with

- B an $n \times n$ submatrix of $\begin{pmatrix} A \\ I \end{pmatrix}$ corresponding to non-basic (tight) rows and columns,
- b the vector of bounds L_i, U_i, ℓ_k, u_k or 0 according to basis status.

In general, such a basis can be

1. **PRIMAL feasible** if $x = B^{-1}b$ satisfies all constraints
2. **OPTIMAL** if primal feasible and $c^T x$ minimal

4. **REGULAR** if none of the above
5. **SINGULAR** if $rank(B) < n$ (error, possibly due to numerics)

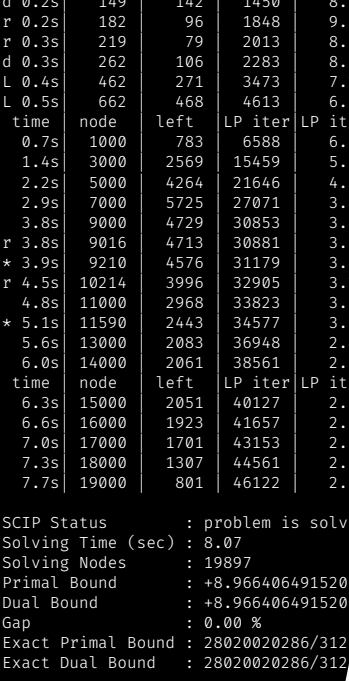
Basic solutions in general

In general, any set of n linearly independent tight inequalities defines a unique solution, which can be computed by a **linear system of equations** $Bx = b$ with

- B an $n \times n$ submatrix of $\begin{pmatrix} A \\ I \end{pmatrix}$ corresponding to non-basic (tight) rows and columns,
- b the vector of bounds L_i, U_i, ℓ_k, u_k or 0 according to basis status.

In general, such a basis can be

1. **PRIMAL feasible** if $x = B^{-1}b$ satisfies all constraints
2. **OPTIMAL** if primal feasible and $c^T x$ minimal
3. **DUAL feasible** if ...?
4. **REGULAR** if none of the above
5. **SINGULAR** if $rank(B) < n$ (error, possibly due to numerics)



The dual LP: special case

First consider

$$\min c^T x \text{ s.t. } Ax \geq b,\ x \geq 0$$

Goal: get lower bound $c^T x \geq d$.

The dual LP: special case

First consider

$$\min c^T x \text{ s.t. } Ax \geq b,\ x \geq 0$$

Goal: get lower bound $c^T x \geq d$.

Idea: express objective direction c as conic combination of constraints

$$c = y^T A + r^T I \text{ with } y, r \geq 0$$

Why?

The dual LP: special case

First consider

$$\min c^T x \text{ s.t. } Ax \geq b,\ x \geq 0$$

Goal: get **lower bound** $c^T x \geq d$.

Idea: express objective direction c as **conic combination of constraints**

$$c = y^T A + r^T I \text{ with } y, r \geq 0$$

Why? This gives a valid lower bound a.k.a. the **weak duality theorem**:

$$c^T x = (y^T A + r^T I)x = y^T Ax + r^T x \geq y^T b + 0 := d$$

The **dual LP** computes the tightest bound:

$$\max b^T y \text{ s.t. } A^T y + r = c,\ y, r \geq 0.$$

The dual LP: general case

$$
\begin{array}{lll}
\min & c^T x & \\
\text{s.t.} & L \le Ax \le U & \\
 & \ell \le \ x \le u &
\end{array}
\quad \geq \quad
\begin{array}{lll}
\max & L^T y^L - U^T y^U + \ell^T r^\ell - u^T r^u & \\
\text{s.t.} & A^T(y^L - y^U) + (r^\ell - r^u) = c & \\
 & y^L, y^U, r^\ell, r^u \ge 0 &
\end{array}
$$

The dual LP: general case

$$\begin{array}{lll} \min & c^T x & \\ \text{s.t.} & L \le Ax \le U & \\ & \ell \le \ x \le u & \end{array} \quad \geq \quad \begin{array}{lll} \max & L^T y^L - U^T y^U + \ell^T r^\ell - u^T r^u & \\ \text{s.t.} & A^T(y^L - y^U) + (r^\ell - r^u) = c & \\ & y^L, y^U, r^\ell, r^u \ge 0 & \end{array}$$

Strong duality ("=") holds if x and y, r are complementary slack, i.e.,

$$\begin{aligned} y^L_i > 0 \Rightarrow y^U_i = 0, (Ax)_i = L_i \qquad & r^\ell_k > 0 \Rightarrow r^u_k = 0, x_k = \ell_k \\ y^U_i > 0 \Rightarrow y^L_i = 0, (Ax)_i = U_i \qquad & r^u_k > 0 \Rightarrow r^\ell_k = 0, x_k = u_k \end{aligned}$$

because then $c^T x = (y^L - y^U)^T Ax + (r^\ell - r^u)^T x = \ldots ?$.

The dual LP: general case

$$
\begin{array}{lll}
\min & c^T x & \\
\text{s.t.} & L \le Ax \le U & \\
 & \ell \le \ x \le u &
\end{array}
\quad \geq \quad
\begin{array}{lll}
\max & L^T y^L - U^T y^U + \ell^T r^\ell - u^T r^u & \\
\text{s.t.} & A^T(y^L - y^U) + (r^\ell - r^u) = c & \\
 & y^L, y^U, r^\ell, r^u \ge 0 &
\end{array}
$$

Strong duality ("=") holds if x and y, r are complementary slack, i.e.,

$$
\begin{aligned}
y^L_i > 0 \Rightarrow y^U_i = 0, (Ax)_i = L_i \qquad & r^\ell_k > 0 \Rightarrow r^u_k = 0, x_k = \ell_k \\
y^U_i > 0 \Rightarrow y^L_i = 0, (Ax)_i = U_i \qquad & r^u_k > 0 \Rightarrow r^\ell_k = 0, x_k = u_k
\end{aligned}
$$

because then $c^T x = (y^L - y^U)^T Ax + (r^\ell - r^u)^T x = L^T y^L - U^T y^U + \ell^T r^\ell - u^T r^u$.

Note: Solvers return only one dual multiplier $y_i = y^L_i - y^U_i$ and $r_k = r^\ell_k - r^u_k$ per row/column, because sign defines the meaning.

Basic solutions in general

In general, any set of n linearly independent tight inequalities defines a unique solution, which can be computed by a **linear system of equations** $Bx = b$ with

- B an $n \times n$ submatrix of $\begin{pmatrix} A \\ I \end{pmatrix}$ corresponding to non-basic (tight) rows and columns,
- b the vector of bounds L_i, U_i, ℓ_k, u_k or 0 according to basis status.

In general, such a basis can be

1. **PRIMAL feasible** if $x = B^{-1}b$ satisfies all constraints
2. **OPTIMAL** if primal feasible and $c^T x$ minimal
3. **DUAL feasible** if ...?
4. **REGULAR** if none of the above
5. **SINGULAR** if $rank(B) < n$ (error, possibly due to numerics)

Basic solutions in general

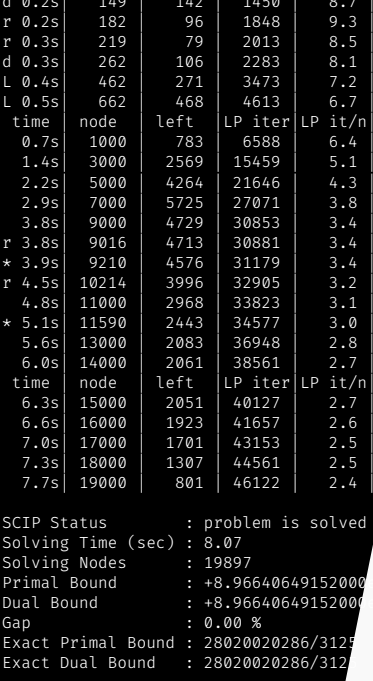
In general, any set of n linearly independent tight inequalities defines a unique solution, which can be computed by a **linear system of equations** $Bx = b$ with

- B an $n \times n$ submatrix of $\begin{pmatrix} A \\ I \end{pmatrix}$ corresponding to non-basic (tight) rows and columns,
- b the vector of bounds L_i, U_i, ℓ_k, u_k or 0 according to basis status.

In general, such a basis can be

1. **PRIMAL feasible** if $x = B^{-1}b$ satisfies all constraints
2. **OPTIMAL** if primal feasible and $c^T x$ minimal
3. **DUAL feasible** if $B^T\gamma = c$ defines a dual solution y, r **with correct sign**
   (patched by 0 for the basic (non-tight) rows/columns)
4. **REGULAR** if none of the above
5. **SINGULAR** if $rank(B) < n$ (error, possibly due to numerics)

Note: By this definition, basic solutions are complementary slack, so primal + dual = optimal!



1. LPs in general form

row representation and basic solutions

2. Dual solutions

certificates of optimality and infeasibility

3. Computational form

basic solutions in column representation

4. Primal and dual simplex

variants and tricks

5. Warmstarting and hotstarting

the simplex engine: LU factorization and updates

6. Beyond simplex

interior point and first-order methods

Computational form

Most simplex solvers use a different form for their computations, introducing slack variables s:

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & L \le Ax \le U \\ & \ell \le \ x \le u \end{array} \quad \Leftrightarrow \quad \begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax = s \\ & \ell \le \ x \le u \\ & L \le s \le U \end{array} \quad \Leftrightarrow \quad \begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax + s = b \\ & \ell \le \ x \le u \\ & b - U \le s \le b - L \end{array}$$

The feasible region is then embedded in an n-dimensional affine subspace of R^{n+m} and a basic solution is defined by n variables fixed at lower or upper bound (or zero).

Redefining $A = (A, I)$, $x = (x, s)$ and $\ell = (\ell, L)$, $u = (u, U)$ gives the LP

$$\min c^T x \text{ s.t. } Ax = b,\ \ell \le x \le u$$

sometimes called standard form or column representation (vs. previous row representation).

Computational form

Most simplex solvers use the form $\min c^T x$ s.t. $Ax = b$, $\ell \le x \le u$ for their computations. Let

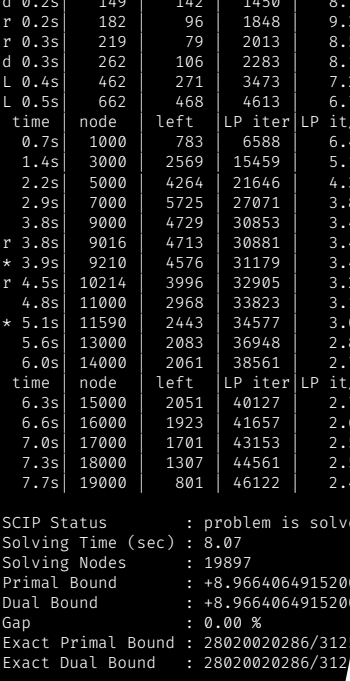
- $\mathcal{L}, \mathcal{U} \subseteq \{1, \ldots, n+m\}$ be the index sets of nonbasic variables at lower/upper bound,
- $\mathcal{B} \subseteq \{1, \ldots, n+m\}$ be the index set of (m) basic variables, and
- $B = A_{*,\mathcal{B}}$ be the $m \times m$ basis matrix,

then the values of the

- basic variables are given by $Bx_{\mathcal{B}} = b - A_{*,\mathcal{L}}\ell_{\mathcal{L}} - A_{*,\mathcal{U}}u_{\mathcal{U}}$,
- dual multipliers are given by $B^T y = c$.

Consequence:

- primal feasibility reduces to $\ell \le x_{\mathcal{B}} \le u$,
- dual feasibility reduces to $\bar{c}_k := c_k - y^T A_{*,k} \begin{cases} \le 0 & \text{if } k \notin \mathcal{L}, \\ \ge 0 & \text{if } k \notin \mathcal{U}, \end{cases}$ for all k.



1. LPs in general form

row representation and basic solutions

2. Dual solutions

certificates of optimality and infeasibility

3. Computational form

basic solutions in column representation

4. Primal and dual simplex

variants and tricks

5. Warmstarting and hotstarting

the simplex engine: LU factorization and updates

6. Beyond simplex

interior point and first-order methods

The primal simplex

Starting from a primal feasible basis $(\mathcal{L}, \mathcal{U}, \mathcal{B})$:

1. If dual feasible return optimal.
2. Price: choose violated non-basic column $k \notin \mathcal{B}$
3. Ratio test:
   - compute direction $d^k = B^{-1}A_{*,k}$ and step length $\hat{\delta} := \max\{\delta : \ell \leq x_{\mathcal{B}} - \delta d^k \leq u$
   - if $\hat{\delta} = \infty$ return unbounded, otherwise
   - choose basic variable $j \in \mathcal{B}$ for which $x_j - \delta d^k_j$ becomes tight
4. Pivoting: exchange k and i and update all data structures, continue.

The primal simplex

Starting from a primal feasible basis $(\mathcal{L}, \mathcal{U}, \mathcal{B})$:

1. If dual feasible return optimal.
2. Price: choose violated non-basic column $k \notin \mathcal{B}$
3. Ratio test:
   - compute direction $d^k = B^{-1}A_{*,k}$ and step length $\hat{\delta} := \max\{\delta : \ell \leq x_{\mathcal{B}} - \delta d^k \leq u$
   - if $\hat{\delta} = \infty$ return unbounded, otherwise
   - choose basic variable $j \in \mathcal{B}$ for which $x_j - \delta d^k_j$ becomes tight
4. Pivoting: exchange k and i and update all data structures, continue.

Obtain a starting basis by "Phase I": the auxiliary LP

$$\min \sum z_i \text{ s.t. } Ax \pm z = b,\ \ell \leq x \leq u,\ z \geq 0$$

has a trivially primal feasible basis: x_k nonbasic, slack variables z_i basic with $\pm$ according to $\text{sign}(\bar{b} := b - A_{*,\mathcal{L}}\ell_{\mathcal{L}} - A_{*,\mathcal{U}}u_{\mathcal{U}})$.

More efficient: use slack variables s_i from column representation directly.

The dual simplex

Essentially performs the primal simplex on the dual LP, but has several advantages:

- slack basis often dual feasible (if lower and upper bounds finite)
- some linear algebra operations more efficient
- bound flipping ratio test: remember dual feasibility condition

$$\bar{c}_k := c_k - y^T A_{*,k} \begin{cases} \leq 0 & \text{if } k \notin \mathcal{L}, \\ \geq 0 & \text{if } k \notin \mathcal{U}, \end{cases}$$

When $\bar{c}_k$ becomes zero, one can often flip basis status and continue the step.

For more details, see Bob Bixby's CO@Work lectures:
`https://www.youtube.com/playlist?list=PLYWmzh0Y6EOYbrsKy1lyf9CSswLKkywY2`

Pricing strategies

The textbook rule (sometimes called “Dantzig” pricing):
choose k with max. reduced cost $|\bar{c}_k|$ over all violated columns.

This is terrible, because it ignores the direction d^k of the pivot.

Improvements:

1. Exact steepest edge: choose max. $|c^T d^k|/\|d^k\|$ (Goldfarb, Reid 1977)
   - need to initialize norms, but fast update available
2. Devex: cheaper, approximate steepest edge (Harris 1973)
3. Quickstart steepest edge: initialize norms with 1, update converges to steepest edge

Other tricks: partial pricing for LPs with many columns.

And a great talk by Sophie Huiberts on the contributions of Paula Harris to computational LP:
`https://www.youtube.com/watch?v=tbOZvbpZp44`

Bound shifting and perturbation

A common problem:

- primal degeneracy: basic variables x_k at one of their bounds
- dual degeneracy: nonbasic reduced costs $\bar{c}_k = 0$

Both can lead to zero step length and cycling in primal/dual simplex.

Textbook solution (also known as Blands rule) terrible:
always price smallest violated index.

Bound shifting and perturbation

A common problem:

- primal degeneracy: basic variables x_k at one of their bounds
- dual degeneracy: nonbasic reduced costs $\bar{c}_k = 0$

Both can lead to zero step length and cycling in primal/dual simplex.

Textbook solution (also known as Blands rule) terrible:
always price smallest violated index.

Better: bound shifting/perturbation

- Add small random values to primal/dual bounds to break degeneracy
- the added violation is sometimes removed automatically later,
- or one can run phase 1.

Bound shifting and perturbation

A common problem:

- primal degeneracy: basic variables x_k at one of their bounds
- dual degeneracy: nonbasic reduced costs $\bar{c}_k = 0$

Both can lead to zero step length and cycling in primal/dual simplex.

Textbook solution (also known as Blands rule) terrible:
always price smallest violated index.

Better: bound shifting/perturbation

- Add small random values to primal/dual bounds to break degeneracy
- the added violation is sometimes removed automatically later,
- or one can run phase 1.

Also used in the Harris ratio test: allow longer steps if it gives for numerically better pivots.

Composite simplex: interleaving primal and dual

Bound shifting can also be used to make a basis artificially feasible:

- primal feasible: by relaxing ℓ, u to include $x_{\mathcal{B}}$
- dual feasible: by shifting c_k to make $\bar{c}_k = c_k - y^T A_{*,k}$ zero

This gives an alternative to phase 1:

- Choose a regular basis
- Shift objective to reach dual feasibilitiy
- Run dual simplex to optimality
- Remove objective shift: preserves primal feasibility
- Run primal simplex to optimality

This is, e.g., the default strategy of SoPlex.
Can be started with the primal simplex, and interleaved.

```
d 0.2s|   149 |   142 |  1450 |     8.7 | linesear
r 0.2s|   182 |    96 |  1848 |     9.3 | zirounding
r 0.3s|   219 |    79 |  2013 |     8.5 | simplerou
d 0.3s|   262 |   106 |  2283 |     8.1 | fracdivin
L 0.4s|   462 |   271 |  3473 |     7.2 |     rins
L 0.5s|   662 |   468 |  4613 |     6.7 | crossover
 time |  node |  left | LP iter | LP it/n | mem/heur
  0.7s|  1000 |   783 |  6588 |     6.4 |   4729k
  1.4s|  3000 |  2569 | 15459 |     5.1 |   6153k
  2.2s|  5000 |  4264 | 21646 |     4.3 |   9005k
  2.9s|  7000 |  5725 | 27071 |     3.8 |     12M
  3.8s|  9000 |  4729 | 30853 |     3.4 |     13M
r 3.8s|  9016 |  4713 | 30881 |     3.4 | simplero
* 3.9s|  9210 |  4576 | 31179 |     3.4 |     LP
r 4.5s| 10214 |  3996 | 32905 |     3.2 | simplero
  4.8s| 11000 |  2968 | 33823 |     3.1 |
* 5.1s| 11590 |  2443 | 34577 |     3.0 |
  5.6s| 13000 |  2083 | 36948 |     2.8 |
  6.0s| 14000 |  2061 | 38561 |     2.7 |
 time |  node |  left | LP iter | LP it/n | mem/heur
  6.3s| 15000 |  2051 | 40127 |     2.7 |
  6.6s| 16000 |  1923 | 41657 |     2.6 |
  7.0s| 17000 |  1701 | 43153 |     2.5 |
  7.3s| 18000 |  1307 | 44561 |     2.5 |
  7.7s| 19000 |   801 | 46122 |     2.4 |

SCIP Status        : problem is solved [optimal solution found]
Solving Time (sec) : 8.07
Solving Nodes      : 19897
Primal Bound       : +8.96640649152000e+06 (11 solutions)
Dual Bound         : +8.96640649152000e+06
Gap                : 0.00 %
Exact Primal Bound : 28020020286/3125
Exact Dual Bound   : 28020020286/312
```

1. *LPs in general form*
   row representation and basic solutions
2. *Dual solutions*
   certificates of optimality and infeasibility
3. *Computational form*
   basic solutions in column representation
4. *Primal and dual simplex*
   variants and tricks
5. *Warmstarting and hotstarting*
   the simplex engine: LU factorization and updates
6. *Beyond simplex*
   interior point and first-order methods

Important in MIP: warmstarting after small changes

Suppose you have reached an optimal basic solution and you ...

- change the objective: basis stays

Important in MIP: warmstarting after small changes

Suppose you have reached an optimal basic solution and you ...

- change the objective: basis stays primal feasible
  $\rightarrow$ warm-start primal simplex
- add a column: basis stays primal feasible if

Important in MIP: warmstarting after small changes

Suppose you have reached an optimal basic solution and you ...

- change the objective: basis stays primal feasible
  $\rightarrow$ warm-start primal simplex
- add a column: basis stays primal feasible if $\ell_k \leq 0 \leq u_k$ (add to nonbasic)
  $\rightarrow$ warm-start primal simplex
- add a row: basis stays

Important in MIP: warmstarting after small changes

Suppose you have reached an optimal basic solution and you ...

- change the objective: basis stays primal feasible
  $\rightarrow$ warm-start primal simplex
- add a column: basis stays primal feasible if $\ell_k \leq 0 \leq u_k$ (add to nonbasic)
  $\rightarrow$ warm-start primal simplex
- add a row: basis stays dual feasible (add slack to basis)
  $\rightarrow$ warm-start dual simplex
- change a right hand side / variable bound: basis stays

Important in MIP: warmstarting after small changes

Suppose you have reached an optimal basic solution and you ...

- change the objective: basis stays primal feasible
  $\rightarrow$ warm-start primal simplex
- add a column: basis stays primal feasible if $\ell_k \leq 0 \leq u_k$ (add to nonbasic)
  $\rightarrow$ warm-start primal simplex
- add a row: basis stays dual feasible (add slack to basis)
  $\rightarrow$ warm-start dual simplex
- change a right hand side / variable bound: basis stays dual feasible
  $\rightarrow$ warm-start dual simplex

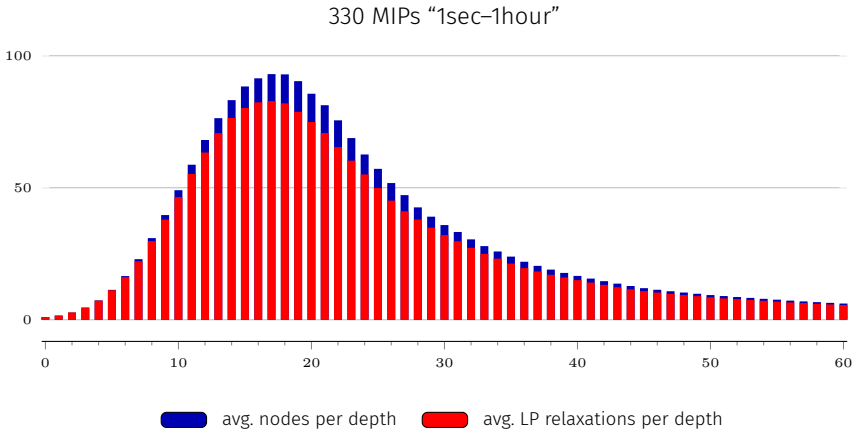
This makes the simplex a very efficient method in branch-and-cut and other algorithms where similar LPs need to be solved after small modifications.

In all cases the basis stays regular $\Rightarrow$ warm-starting always possible with phase 1/composite simplex, also if matrix coefficients change outside the basis matrix (but not always supported).

An experiment: warm-started dual simplex during tree search

depth	avg. iters
root	**383.7**
1	17.0
2	14.9
3	12.2
4	10.3
5	9.0
6	8.2
⋮	⋮
13	4.2
14	4.1
15	3.8
16	3.4
17	3.3
18	3.1
19	2.8
20	2.7
21	2.5
22	2.4
⋮	⋮

383.7/3.3 ≈ 116x speedup



Important in MIP: hotstarting after small changes

Suppose you have reached an optimal basic solution and you ...

- change the objective: basis stays primal feasible
  $\rightarrow$ hot-start primal simplex
- add a column: basis stays primal feasible if $\ell_k \leq 0 \leq u_k$ (add to nonbasic)
  $\rightarrow$ hot-start primal simplex
- add a row: basis stays dual feasible (add slack to basis)
  $\rightarrow$ warm-start dual simplex
- change a right hand side / variable bound: basis stays dual feasible
  $\rightarrow$ hot-start dual simplex

Hot-starting = reusing the (permuted) LU factorization $PBQ = LU$.

Row representation allows hot-starting after adding a row $\rightarrow$ favorable for branch-and-cut (and generally for LPs with many rows).

```
d 0.2s|   149 |   142 |  1450 |      |  8.7 | tinesear
r 0.2s|   182 |    96 |  1848 |      |  9.3 | zirounditi
r 0.3s|   219 |    79 |  2013 |      |  8.5 | simplero
d 0.3s|   262 |   106 |  2283 |      |  8.1 | fracdivi
L 0.4s|   462 |   271 |  3473 |      |  7.2 |     rins
L 0.5s|   662 |   468 |  4613 |      |  6.7 | crossove
 time |  node |  left | LP iter | LP it/n | mem/heur
  0.7s|  1000 |   783 |  6588 |      |  6.4 |   4729k
  1.4s|  3000 |  2569 | 15459 |      |  5.1 |   6153k
  2.2s|  5000 |  4264 | 21646 |      |  4.3 |   9005k
  2.9s|  7000 |  5725 | 27071 |      |  3.8 |     12M
  3.8s|  9000 |  4729 | 30853 |      |  3.4 |     13M
r 3.8s|  9016 |  4713 | 30881 |      |  3.4 | simple
* 3.9s|  9210 |  4576 | 31179 |      |  3.4 |     LP
r 4.5s| 10214 |  3996 | 32905 |      |  3.2 | simpl
  4.8s| 11000 |  2968 | 33823 |      |  3.1 |
* 5.1s| 11590 |  2443 | 34577 |      |  3.0 |
  5.6s| 13000 |  2083 | 36948 |      |  2.8 |
  6.0s| 14000 |  2061 | 38561 |      |  2.7 |
 time |  node |  left | LP iter | LP it/n | mem
  6.3s| 15000 |  2051 | 40127 |      |  2.7 |
  6.6s| 16000 |  1923 | 41657 |      |  2.6 |
  7.0s| 17000 |  1701 | 43153 |      |  2.5 |
  7.3s| 18000 |  1307 | 44561 |      |  2.5 |
  7.7s| 19000 |   801 | 46122 |      |  2.4 |

SCIP Status        : problem is solved [optimal solution found]
Solving Time (sec) : 8.07
Solving Nodes      : 19897
Primal Bound       : +8.96640649152000e+06 (11 solutions)
Dual Bound         : +8.96640649152000e+06
Gap                : 0.00 %
Exact Primal Bound : 28020020286/3125
Exact Dual Bound   : 28020020286/312
```

1. LPs in general form

row representation and basic solutions

2. Dual solutions

certificates of optimality and infeasibility

3. Computational form

basic solutions in column representation

4. Primal and dual simplex

variants and tricks

5. Warmstarting and hotstarting

the simplex engine: LU factorization and updates

6. Beyond simplex

interior point and first-order methods

Alternatives to the simplex method

Computational challenges

- The simplex method parallelizes horribly (for all LPs).
- For some MIPs, the simplex takes much more iterations to reoptimize.
- For some MIPs, the root LP is the bottleneck.

Alternatives to the simplex method

Computational challenges

- The simplex method parallelizes horribly (for all LPs).
- For some MIPs, the simplex takes much more iterations to reoptimize.
- For some MIPs, the root LP is the bottleneck.

Interior point methods (a.k.a. barrier)

- parallelize very well, but do not warm start well:
- often combined with crossover to basic solution.

Alternatives to the simplex method

Computational challenges

- The simplex method parallelizes horribly (for all LPs).
- For some MIPs, the simplex takes much more iterations to reoptimize.
- For some MIPs, the root LP is the bottleneck.

Interior point methods (a.k.a. barrier)

- parallelize very well, but do not warm start well:
- often combined with crossover to basic solution.

PDHG: Primal-dual hybrid gradient methods

- parallelize and warm-start beautifully (mostly matrix-vector multiply),
- but need many iterations (first-order method)
- and sometimes get stuck at lower precision.
- Growing amount of open-source GPU implementations available, e.g., cuPDLP-C

For more details, see Joachim Dahl's CO@Work lecture: `https://youtu.be/alJhQBgJusI`