

Solving Linear Programs: The Dual Simplex Algorithm

Outline

- ▶ LP basics
- ▶ Primal and dual simplex algorithms
- ▶ Implementing the dual simplex algorithm

Some Basic Theory

Linear Program – Definition

A linear program (LP) in standard form is an optimization problem of the form

$$\begin{array}{ll} \text{Minimize} & c^T x \\ \text{Subject to} & Ax = b \\ & x \geq 0 \end{array} \quad (\text{P})$$

Where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$, and x is a vector of n variables. $c^T x$ is known as the **objective function**, $Ax = b$ as the **constraints**, and $x \geq 0$ as the **nonnegativity conditions**. b is called the **right-hand side**.

Dual Linear Program – Definition

The dual (or adjoint) linear program corresponding to (P) is the optimization problem

$$\begin{array}{ll} \text{Maximize} & b^T \pi \\ \text{Subject to} & A^T \pi \leq c \\ & \pi \text{ free} \end{array} \quad (\text{D})$$

In this context, (P) is referred to as the **primal linear program**.

$$\left(\begin{array}{ll} \text{Primal} \\ \text{Minimize} & c^T x \\ \text{Subject to} & Ax = b \\ & x \geq 0 \end{array} \right)$$

Weak Duality Theorem

(von Neumann 1947)

Let x be feasible for (P) and π feasible for (D).
Then

$$\boxed{\text{Maximize}} \quad b^T \pi \leq c^T x \quad \boxed{\text{Minimize}}$$

If $b^T \pi = c^T x$, then x is optimal for (P) and π is optimal for (D); moreover, if either (P) or (D) is unbounded, then the other problem is infeasible.

Proof:

$$\pi^T b \quad = \quad \pi^T A x \quad \leq \quad c^T x \quad \blacksquare$$

$$\boxed{Ax = b} \quad \boxed{\pi^T A \leq c^T \ \& \ x \geq 0}$$

Solving Linear Programs

- ▶ Three types of algorithms are available
 - Primal simplex algorithms (Dantzig 1947)
 - Dual simplex algorithms (Lemke 1954)
 - Developed in context of game theory
 - Primal–dual log barrier algorithms
 - Interior–point algorithms (Karmarkar 1989)
 - Reference: Primal–Dual Interior Point Methods, S. Wright, 1997, SIAM

Primary focus: **Dual simplex algorithms**

Basic Solutions – Definition

Let B be an ordered set of m distinct indices (B_1, \dots, B_m) taken from $\{1, \dots, n\}$. B is called a **basis** for (P) if A_B is nonsingular. The variables x_B are known as the **basic variables** and the variables x_N as the **non-basic variables**, where $N = \{1, \dots, n\} \setminus B$. The corresponding **basic solution** $X \in \mathbb{R}^n$ is given by $x_N = 0$ and $x_B = A_B^{-1} b$. B is called **(primal) feasible** if $x_B \geq 0$.

$$\text{Note: } AX = b \Rightarrow A_B x_B + A_N x_N = b \Rightarrow A_B x_B = b \Rightarrow x_B = A_B^{-1} b$$

Primal Simplex Algorithm

(Dantzig, 1947)

Input: A feasible basis B and vectors

$$X_B = A_B^{-1}b \quad \text{and} \quad D_N = c_N - A_N^T A_B^{-T} c_B.$$

▶ Step 1: (Pricing) If $D_N \geq 0$, stop, B is optimal; else let
$$j = \operatorname{argmin}\{D_k : k \in N\}.$$

▶ Step 2: (FTRAN) Solve $A_B y = A_j$.

▶ Step 3: (Ratio test) If $y \leq 0$, stop, (P) is unbounded; else, let

$$i = \operatorname{argmin}\{X_{Bk}/y_k : y_k > 0\}.$$

▶ Step 4: (BTRAN) Solve $A_B^T z = e_i$.

▶ Step 5: (Update) Compute $\alpha_N = -A_N^T z$. Let $B_i = j$. Update X_B (using y) and D_N (using α_N)

Note: x_j is called the **entering** variable and x_{B_i} the **leaving** variable. The D_N values are known as **reduced costs** – like partial derivatives of the objective function relative to the nonbasic variables.

Primal Simplex Example

The Primal Simplex Algorithm

Consider the following simple LP:

$$\begin{array}{ll} \text{Maximize} & 3x_1 + 2x_2 + 2x_3 \\ \text{Subject to} & x_1 + \quad \quad x_3 \leq 8 \\ & x_1 + x_2 \leq 7 \\ & x_1 + 2x_2 \leq 12 \\ & x_1, x_2, x_3 \geq 0 \end{array}$$

The Primal Simplex Algorithm

Maximize $z = 3x_1 + 2x_2 + 2x_3$

Add slacks: Initial basis B = (4,5,6)

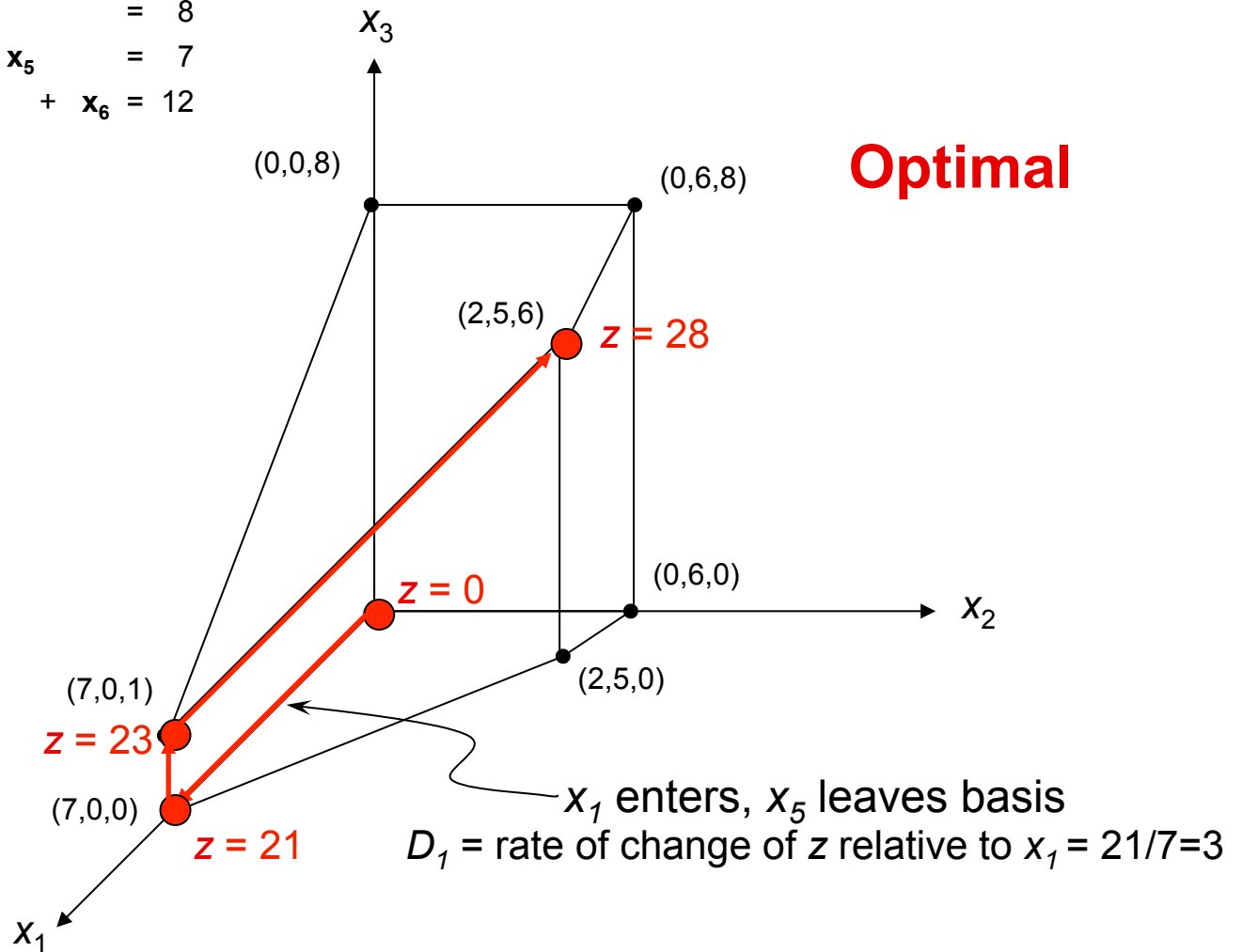
Maximize $3x_1 + 2x_2 + 2x_3 + 0x_4 + 0x_5 + 0x_6$

Subject to $x_1 + x_3 + x_4 = 8$

$x_1 + x_2 + x_5 = 7$

$x_1 + 2x_2 + x_6 = 12$

$x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$



Dual Simple Algorithm – Setup

Simplex algorithms apply to problems with constraints in equality form. We convert (D) to this form by adding the dual **slacks** d :

$$\begin{array}{l} \text{Maximize} \quad b^T \pi \\ \text{Subject to} \quad A^T \pi + d = c \\ \quad \quad \quad \pi \text{ free, } d \geq 0 \end{array} \quad \Leftrightarrow \quad A^T \pi \leq c$$

Dual Simple Algorithm – Setup

$$\begin{array}{l} \text{Maximize } b^T \pi \\ \text{Subject to } A^T \pi + d = c \\ \pi \text{ free, } d \geq 0 \end{array} \quad \leftarrow \quad \begin{bmatrix} A_B^T & I_B & 0 \\ A_N^T & 0 & I_N \end{bmatrix} \begin{bmatrix} \pi \\ d_B \\ d_N \end{bmatrix} = \begin{bmatrix} c_B \\ c_N \end{bmatrix}$$

Given a basis B , the corresponding **dual basic variables** are π and d_N . d_B are the **nonbasic variables**. The corresponding **dual basic solution** Π, D is determined as follows:

$$D_B = 0 \Rightarrow \Pi = A_B^{-T} c_B \Rightarrow D_N = c_N - A_N^T \Pi$$

B is **dual feasible** if $D_N \geq 0$.

Dual Simple Algorithm – Setup

$$\begin{array}{l} \text{Maximize } b^T \pi \\ \text{Subject to } A^T \pi + d = c \\ \pi \text{ free, } d \geq 0 \end{array} \quad \leftarrow \quad \begin{bmatrix} A_B^T & I_B & 0 \\ A_N^T & 0 & I_N \end{bmatrix} \begin{bmatrix} \pi \\ d_B \\ d_N \end{bmatrix} = \begin{bmatrix} c_B \\ c_N \end{bmatrix}$$

Observation: We may assume that every dual basis has the above form.

Proof: Assuming that the primal has a basis is equivalent to assuming that $\text{rank}(A)=m$ (# of rows), and this implies that all π variables can be assumed to be basic.

This observation establishes a 1-1 correspondence between primal and dual bases. ■

An Important Fact

If X and Π, D are corresponding primal and dual basic solutions determined by a basis B , then

$$\Pi^T b = c^T X.$$

Hence, by weak duality, if B is both primal and dual feasible, then X is optimal for (P) and Π is optimal for (D).

Proof:

$$\begin{aligned} c^T X &= c_B^T X_B && \text{(since } X_N = 0) \\ &= \Pi^T A_B X_B && \text{(since } \Pi = A_B^{-T} c_B) \\ &= \Pi^T b && \text{(since } A_B X_B = b) \quad \blacksquare \end{aligned}$$

Dual Simplex Algorithm

(Lemke, 1954)

Input: A dual feasible basis B and vectors

$$X_B = A_B^{-1}b \quad \text{and} \quad D_N = c_N - A_N^T B^{-T} c_B.$$

- ▶ Step 1: (Pricing) If $X_B \geq 0$, stop, B is optimal; else let
$$i = \operatorname{argmin}\{X_{Bk} : k \in \{1, \dots, m\}\}.$$
- ▶ Step 2: (BTRAN) Solve $A_B^T z = e_i$. Compute $\alpha_N = -A_N^T z$.
- ▶ Step 3: (Ratio test) If $\alpha_N \leq 0$, stop, (D) is unbounded; else, let

$$j = \operatorname{argmin}\{D_k / \alpha_k : \alpha_k > 0\}.$$

- ▶ Step 4: (FTRAN) Solve $A_B y = A_j$.
- ▶ Step 5: (Update) Set $B_i = j$. Update X_B (using y) and D_N (using α_N)

Note: d_{B_i} is the **entering** variable and d_j is the **leaving** variable.
(Expressed in terms of the primal: x_{B_i} is the leaving variable and x_j is the entering variable)

Simplex Algorithms

Input: A primal feasible basis B and vectors

$$X_B = A_B^{-1}b \quad \& \quad D_N = c_N - A_N^T A_B^{-T} c_B.$$

- ▶ Step 1: (Pricing) If $D_N \geq 0$, stop, B is optimal; else, let $j = \operatorname{argmin}\{D_k : k \in N\}$.
- ▶ Step 2: (FTRAN) Solve $A_B y = A_j$.
- ▶ Step 3: (Ratio test) If $y \leq 0$, stop, (P) is unbounded; else, let $i = \operatorname{argmin}\{X_{Bk}/y_k : y_k > 0\}$.
- ▶ Step 4: (BTRAN) Solve $A_B^T z = e_i$.
- ▶ Step 5: (Update) Compute $\alpha_N = -A_N^T z$. Let $B_i = j$. Update X_B (using y) and D_N (using α_N)

Input: A dual feasible basis B and vectors

$$X_B = A_B^{-1}b \quad \& \quad D_N = c_N - A_N^T A_B^{-T} c_B.$$

- ▶ Step 1: (Pricing) If $X_B \geq 0$, stop, B is optimal; else, let $i = \operatorname{argmin}\{X_{Bk} : k \in \{1, \dots, m\}\}$.
- ▶ Step 2: (BTRAN) Solve $A_B^T z = e_i$. Compute $\alpha_N = -A_N^T z$.
- ▶ Step 3: (Ratio test) If $\alpha_N \leq 0$, stop, (D) is unbounded; else, let $j = \operatorname{argmin}\{D_k/\alpha_k : \alpha_k > 0\}$.
- ▶ Step 4: (FTRAN) Solve $A_B y = A_j$.
- ▶ Step 5: (Update) Set $B_i = j$. Update X_B (using y) and D_N (using α_N)

Summary:

What we have done and what we have to do

- ▶ Done
 - Defined primal and dual linear programs
 - Proved the weak duality theorem
 - Introduced the concept of a basis
 - Stated primal and dual simplex algorithms
- ▶ To do (for dual simplex algorithm)
 - Show correctness
 - Describe key implementation ideas

Correctness: Dual Simplex Algorithm

- ▶ Termination criteria
 - Optimality (DONE – by “An Important Fact”)
 - Unboundedness
- ▶ Other issues
 - Finding starting dual feasible basis, or showing that no feasible solution exists
 - Input conditions are preserved (i.e., that B is still a feasible basis)
 - Finiteness

Dual Unboundedness

(\Rightarrow primal infeasible)

- ▶ We carry out a key calculation
- ▶ As noted earlier, in an iteration of the dual

$$\begin{array}{l} d_{Bi} \text{ enters basis} \\ d_j \text{ leaves basis} \end{array} \quad \text{in} \quad \begin{array}{l} \text{Maximize } b^T \pi \\ \text{Subject to } A^T \pi + d = c \\ \pi \text{ free, } d \geq 0 \end{array}$$

- ▶ The idea: Currently $d_{Bi} = 0$, and $X_{Bi} < 0$ has motivated us to increase d_{Bi} to $\theta > 0$, leaving the other components of d_B at 0 (the object being to increase the objective). Letting $\underline{d}, \underline{\pi}$ be the corresponding dual solution as a function of θ , we obtain

$$\underline{d}_B = \theta e_i \quad \underline{\pi} = \Pi - \theta z \quad \underline{d}_N = D_N - \theta \alpha_N$$

where α_N and z are as computed in the algorithm.

(Dual Unboundedness – cont.)

- ▶ Letting $\underline{d}, \underline{\pi}$ be the corresponding dual solution as a function of θ . Using α_N and z from dual algorithm,

$$\underline{d}_B = \theta e_i \quad \underline{d}_N = D_N - \theta \alpha_N \quad \underline{\pi} = \pi - \theta z.$$

- ▶ Using $\theta > 0$ and $X_{Bi} < 0$ yields

$$\begin{aligned} \text{new_objective} &= \underline{\pi}^T b = (\Pi - \theta z)^T b \\ &= \Pi^T b - \theta e_i^T A_B^{-1} b = \Pi^T b - \theta e_i^T X_B \\ &= \text{old_objective} - \theta X_{Bi} > \text{old_objective} \end{aligned}$$

- ▶ **Conclusion 1:** If $\alpha_N \leq 0$, then $\underline{d}_N \geq 0 \forall \theta > 0 \Rightarrow$ (D) is unbounded.

- ▶ **Conclusion 2:** If α_N not ≤ 0 , then

$$\begin{aligned} \underline{d}_N \geq 0 &\Rightarrow \theta \leq D_j / \alpha_j \quad \forall \alpha_j > 0 \\ &\Rightarrow \theta_{\max} = \min\{D_j / \alpha_j : \alpha_j > 0\} \end{aligned}$$

(Dual Unboundedness – cont.)

- ▶ Feasibility preserved: follows from the ratio test.
- ▶ Nonsingularity preserved: follows from (also yields update)
 - new $A_B = A_B (I + (y - e_j) e_j^T)$
 - new $A_B^{-1} = (I - (1/y_j) (y - e_j) e_j^T) A_B^{-1}$
- ▶ Finiteness: If $D_B > 0$ for all dual feasible bases B , then the dual simplex algorithm is finite: The dual objective strictly increases at each iteration \Rightarrow no basis repeats, and there are a finite number of bases.
- ▶ There are various approaches to guaranteeing finiteness in general:
 - Bland's Rules: Purely combinatorial, bad in practice.
 - Gurobi: A perturbation is added to “guarantee” $D_B > 0$.

Implementing the Dual Simplex Algorithm

Some Motivation

- ▶ Dual simplex vs. primal: Dual $>$ 2x faster
- ▶ Dual is the best algorithm for MIP
- ▶ There isn't much in books about implementing the dual.

Dual Simplex Algorithm

(Lemke, 1954: Commercial codes ~1990)

Input: A dual feasible basis B and vectors

$$X_B = A_B^{-1}b \quad \text{and} \quad D_N = c_N - A_N^T B^{-T} c_B.$$

- ▶ Step 1: (Pricing) If $X_B \geq 0$, stop, B is optimal; else let
$$i = \operatorname{argmin}\{X_{B_k} : k \in \{1, \dots, m\}\}.$$
- ▶ Step 2: (BTRAN) Solve $B^T z = e_i$. Compute $\alpha_N = -A_N^T z$.
- ▶ Step 3: (Ratio test) If $\alpha_N \leq 0$, stop, (D) is unbounded; else, let

$$j = \operatorname{argmin}\{D_k / \alpha_k : \alpha_k > 0\}.$$

- ▶ Step 4: (FTRAN) Solve $A_B y = A_j$.
- ▶ Step 5: (Update) Set $B_i = j$. Update X_B (using y) and D_N (using α_N)

Dual Simplex Algorithm

(Lemke, 1954: Commercial codes ~1990)

Input: A **dual feasible basis** B and vectors

$$X_B = A_B^{-1}b \quad \text{and} \quad D_N = c_N - A_N^T A_B^{-T} c_B.$$

- ▶ Step 1: (Pricing) If $X_B \geq 0$, stop, B is optimal; else let
 $i = \operatorname{argmin}\{X_{B_k} : k \in \{1, \dots, m\}\}.$
- ▶ Step 2: (BTRAN) **Solve** $B^T z = e_i$. Compute $\alpha_N = -A_N^T z$.
- ▶ Step 3: (Ratio test) If $\alpha_N \leq 0$, stop, (D) is unbounded; else, let
 $j = \operatorname{argmin}\{D_k / \alpha_k : \alpha_k > 0\}.$
- ▶ Step 4: (FTRAN) **Solve** $A_B y = A_j$.
- ▶ Step 5: (Update) Set $B_i = j$. Update X_B (using y) and D_N (using α_N)

Implementation Issues for Dual Simplex

1. Finding an initial feasible basis or concluding that there is none: Phase I of the simplex algorithm.
2. Pricing: dual steepest edge
3. Solving the linear systems
 - LU factorization and factorization update
 - BTRAN and FTRAN – exploiting sparsity
4. Numerically stable ratio test: Bound shifting and perturbation
5. Bound flipping: Exploiting “boxed” variables to combine many iterations into one.

Issue 0

Preparation: Bounds on Variables

In practice, simplex algorithms need to accept LPs in the following form:

$$\begin{array}{ll} \text{Minimize} & c^T x \\ \text{Subject to} & Ax = b \\ & l \leq x \leq u \end{array} \quad (P_{BD})$$

where l is an n -vector of **lower bounds** and u an n -vector of **upper bounds**. l is allowed to have $-\infty$ entries and u is allowed to have $+\infty$ entries. (Note that (P_{BD}) is in standard form if $l_j = 0, u_j = +\infty \forall j$.)

(Issue 0 – Bounds on variables) Basic Solution

A **basis** for (P_{BD}) is a triple (B, L, U) where B is an ordered m -element subset of $\{1, \dots, n\}$ (just as before), (B, L, U) is a partition of $\{1, \dots, n\}$, $l_j > -\infty \forall j \in L$, and $u_j < +\infty \forall j \in U$. $N = L \cup U$ is the set of **nonbasic** variables. The associated **(primal) basic solution** X is given by $X_L = l_L$, $X_U = u_U$ and

$$X_B = A_B^{-1}(b - A_L l_L - A_U u_U).$$

This solution is **feasible** if

$$l_B \leq X_B \leq u_B.$$

The associated **dual basic solution** is defined exactly as before: $D_B = 0$, $\Pi^T A_B = c_B^T$, $D_N = c_N - A_N^T \Pi$. It is **dual feasible** if

$$D_L \geq 0 \text{ and } D_U \leq 0.$$

(Issue 0 – Bounds on variables) The Full Story

- ▶ Modify simplex algorithm
 - Only the “Pricing” and “Ratio Test” steps must be changed substantially.
 - The complicated part is the ratio test
- ▶ Reference: See Chvátal for the primal

Issue 1

The Initial Feasible Basis – Phase I

- ▶ Two parts to the solution
 1. Finding some initial basis (probably not feasible)
 2. Modified simplex algorithm to find a feasible basis

(Issue 1 – Initial feasible basis) Initial Basis

- ▶ Primal and dual bases are the same. We begin in the context of the primal. Consider

$$\begin{array}{ll} \text{Minimize} & c^T x \\ \text{Subject to} & Ax = b \\ & l \leq x \leq u \end{array} \quad (\text{P}_{\text{BD}})$$

- ▶ Assumption: Every variable has some finite bound.
- ▶ Trick: Add artificial variables x_{n+1}, \dots, x_{n+m} :

$$Ax + I \begin{pmatrix} x_{n+1} \\ \vdots \\ x_{n+m} \end{pmatrix} = b$$

where $l_j = u_j = 0$ for $j = n+1, \dots, n+m$.

- ▶ Initial basis: $B = (n+1, \dots, n+m)$ and for each $j \notin B$, pick some finite bound and place j in L or U , as appropriate.
- ▶ Free-Variable Refinement: Make free variables non-basis at value 0. This leads to a notion of a *superbasis*, where non-basis variables can be between their bounds.

(Issue 1 – Initial feasible basis) Solving the Phase I

- ▶ If the initial basis is not dual feasible, we consider the problem:

$$\begin{aligned} & \text{Maximize } \sum (d_j : d_j < 0) \\ & \text{Subject to } A^T \pi + d = c \end{aligned}$$

- ▶ This problem is “locally linear”: Define $\kappa \in \mathbb{R}^n$ by $\kappa_j = 1$ if $D_j < 0$, and 0 otherwise. Let

$$K = \{j : D_j < 0\} \text{ and } \underline{K} = \{j : D_j \geq 0\}$$

Then our “local” problem becomes

$$\begin{aligned} & \text{Maximize } \kappa^T d \\ & \text{Subject to } A^T \pi + d = c \\ & \quad d_K \leq 0, d_{\underline{K}} \geq 0 \end{aligned}$$

- ▶ Apply dual simplex, and whenever d_j for $j \in K$ becomes 0 , move it to \underline{K} .

Solving Phase I: An Interesting Computation

- ▶ Suppose d_{B_i} is the entering variable. Then $x_{B_i} < 0$ where X_B is obtained using the following formula:

$$X_B = A_B^{-1} A_N \kappa$$

- ▶ Suppose now that d_j is determined to be the leaving variable. Then in terms of the phase I objective, this means κ_j is replaced by $\kappa_j + \varepsilon e_j$, where $\varepsilon \in \{0, +1, -1\}$. It can then be shown that

$$\underline{x}_{B_i} = X_{B_i} + \varepsilon \alpha_j$$

- ▶ Conclusion: If $x_{B_i} < 0$, then the current iteration can continue without the necessity of changing the basis.
- ▶ Advantages
 - Multiple iterations are combined into one.
 - x_{B_i} will tend not to change sign precisely when α_j is small. Thus this procedure tends to avoid unstable pivots.

Issue 2 Pricing

- ▶ The textbook rule is **TERRIBLE**: For a problem in standard form, select the entering variable using the formula

$$j = \operatorname{argmin}\{X_{B_i} : i = 1, \dots, m\}$$

- ▶ Geometry is wrong: Maximizes rate of change relative to axis; better to do relative to edge.
- ▶ Goldfard and Forrest 1992 suggested the following steepest-edge alternative

$$j = \operatorname{argmin}\{X_{B_i} / \eta_i : i = 1, \dots, m\}$$

where $\eta_i = \|e_i^T A_B^{-1}\|_2$, and gave an efficient update.

- ▶ Note that there are two ingredients in the success of Dual SE:
 - *Significantly reduced iteration counts*
 - *The fact that there is a very efficient update for η*

Example: Pricing

Model: dfl001

Pricing: Greatest infeasibility

Solved in 281829 iterations and 118.68 seconds
Optimal objective 1.126639304e+07

Pricing: Goldfarb-Forrest steepest-edge

Solved in 18412 iterations and 5.36 seconds
Optimal objective 1.126639304e+07

Issue 3

Solving FTRAN, BTRAN

- ▶ Computing LU factorization: See Suhl & Suhl (1990). “Computing sparse LU factorization for large-scale linear programming basis”, ORSA Journal on Computing 2, 325–335.
- ▶ Updating the Factorization: Forrest–Tomlin update is the method of choice. See Chvátal Chapter 24.
 - There are multiple, individually relatively minor tweaks that collectively have a significant effect on update efficiency.
- ▶ Further exploiting sparsity: This is the main recent development.ds

(Issue 3 – Solving FTRAN & BTRAN)

We must solve two linear systems per iteration:

$$\begin{array}{ll} \text{FTRAN} & \text{BTRAN} \\ A_B y = A_j & A_B^T z = e_i \end{array}$$

where

A_B = basis matrix (very sparse)

A_j = entering column (very sparse)

e_i = unit vector (very sparse)

$\Rightarrow y$ and z are typically very sparse

Example: Model pla85900 (from TSP)

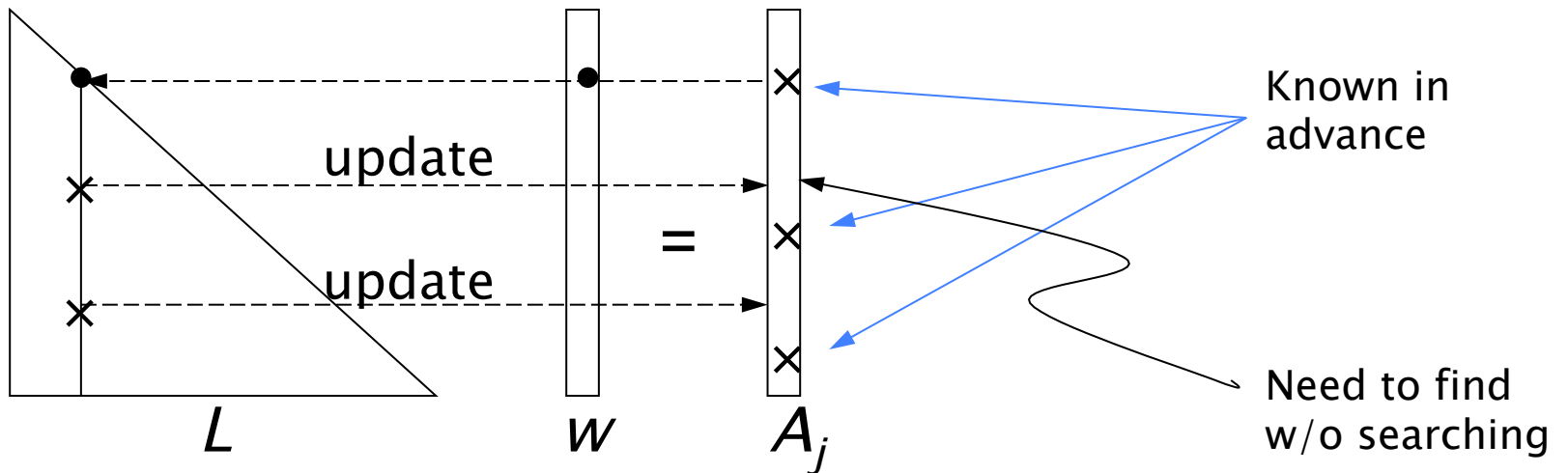
Constraints 85900

Variables 144185

Average $|y|$ 15.5

$$A_B = \begin{bmatrix} L & U \end{bmatrix}$$

Triangular solve: $LW = A_j$ ($A_B Y = L(UY) = A_j$)



Graph structure: Define an acyclic digraph $D = (\{1, \dots, m\}, E)$ where $(i, j) \in E \Leftrightarrow l_{ij} \neq 0$ and $i \neq j$.

Solving using D : Let $X = \{i \in V : L_{ij} \neq 0\}$. Compute $\underline{X} = \{i \in V : \exists \text{ a directed path from } i \text{ to } X\}$.

\underline{X} can be computed in time linear in $|E(\underline{X})| + |\underline{X}|$.

PDS Models (2002)

“Patient Distribution System”: Carolan, Hill, Kennington, Niemi, Wichmann, *An empirical evaluation of the KORBX algorithms for military airlift applications*, Operations Research 38 (1990), pp. 240–248

MODEL	ROWS	CPLEX1.0 1988	CPLEX5.0 1997	CPLEX8.0 2002	SPEEDUP 1.0 → 8.0
<i>pds02</i>	2953	0.4	0.1	0.1	4.0
<i>pds06</i>	9881	26.4	2.4	0.9	29.3
<i>pds10</i>	16558	208.9	13.0	2.6	80.3
<i>pds20</i>	33874	5268.8	232.6	20.9	247.3
<i>pds30</i>	49944	15891.9	1154.9	39.1	406.4
<i>pds40</i>	66844	58920.3	2816.8	79.3	743.0
<i>pds50</i>	83060	122195.9	8510.9	114.6	1066.3
<i>pds60</i>	99431	205798.3	7442.6	160.5	1282.2
<i>pds70</i>	114944	335292.1	21120.4	197.8	1695.1

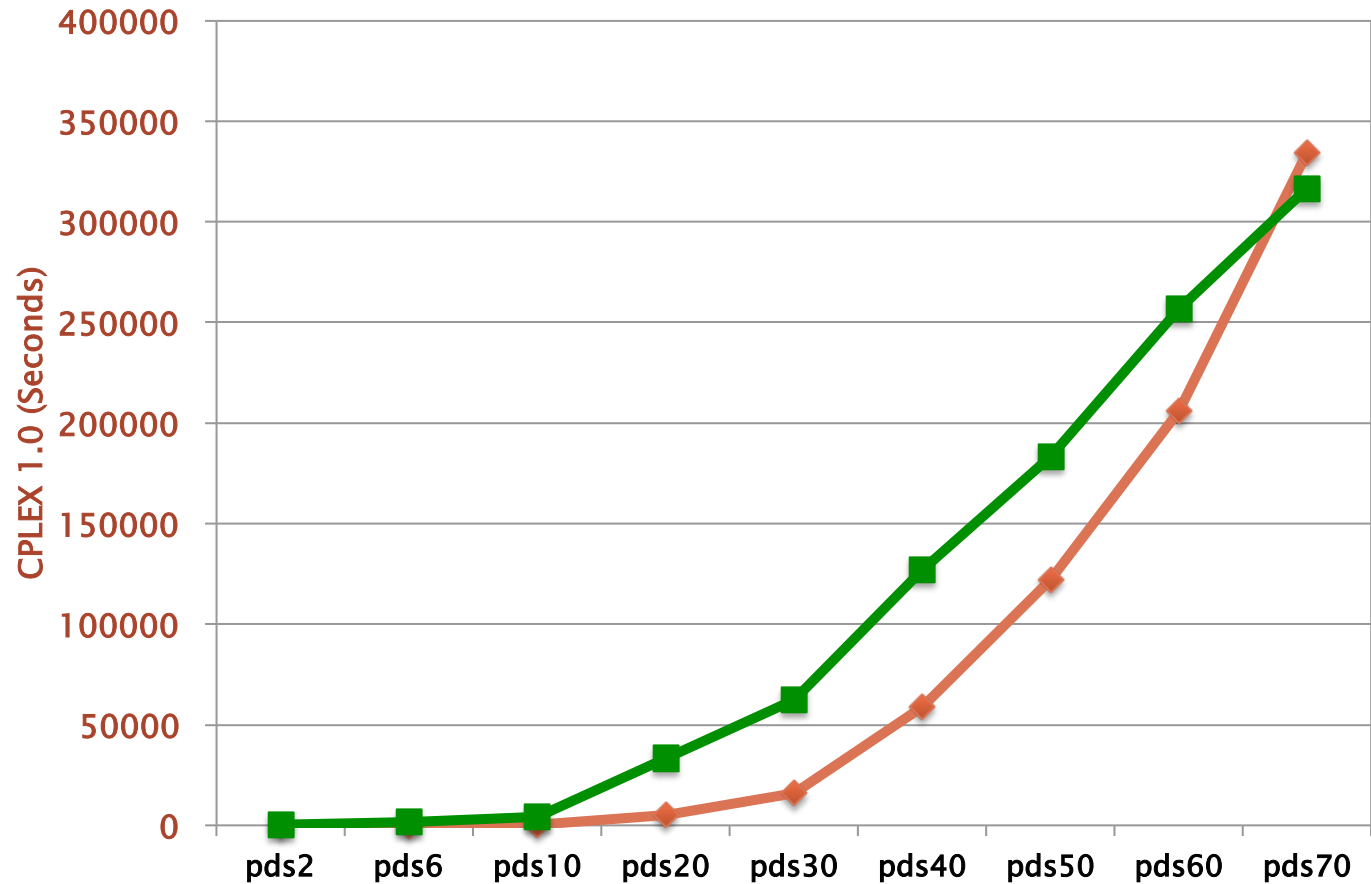
Primal
Simplex

Dual
Simplex

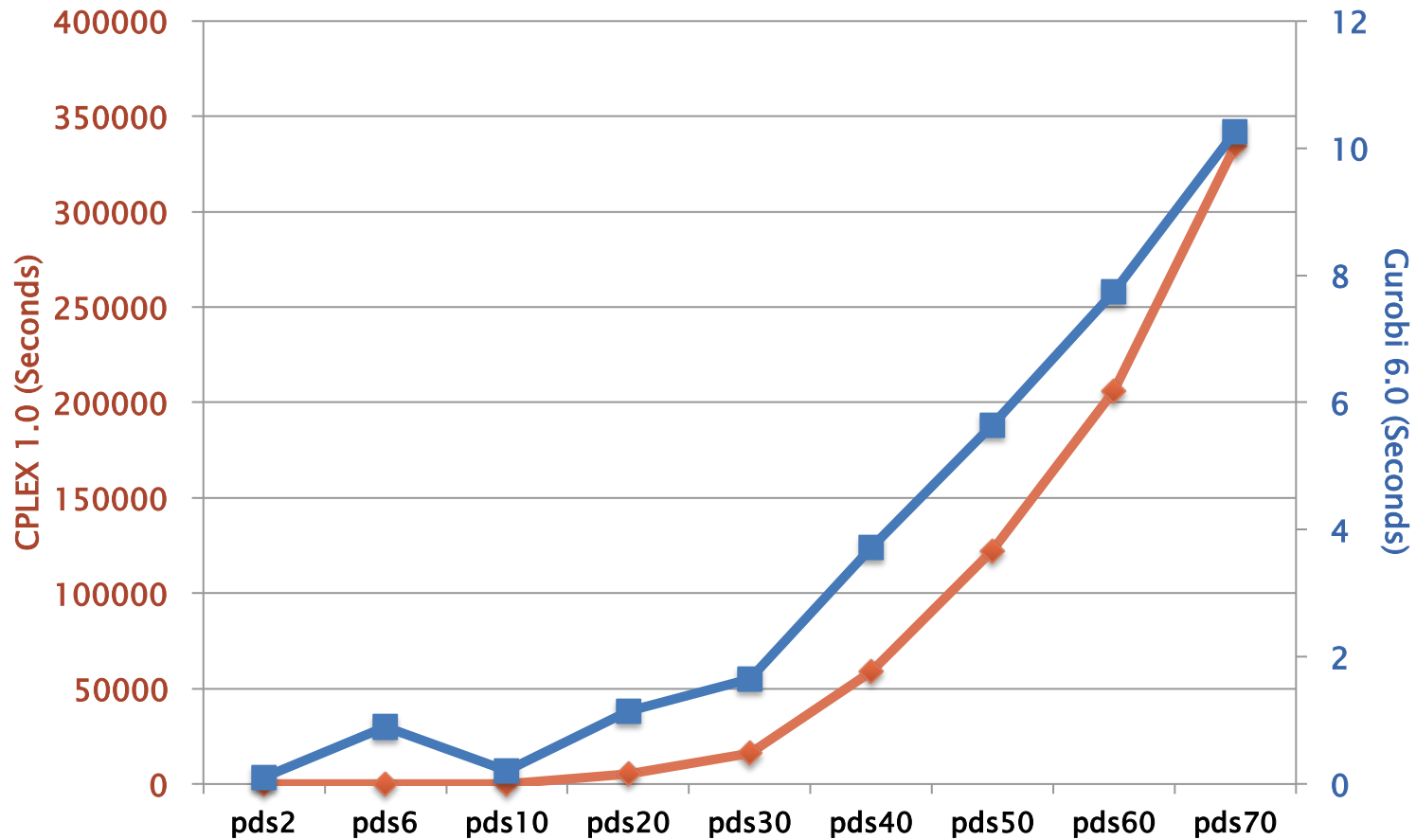
Dual
Simplex



Not just faster -- Growth with size: Quadratic *then* & Linear *now*!



Gurobi Headline goes here...



Issue 4

Ratio Test and Finiteness

The “standard form” dual problem is

$$\begin{array}{ll} \text{Maximize} & b^T \pi \\ \text{Subject to} & A^T \pi + d = c \\ & d \geq 0 \end{array}$$

Feasibility means

$$d \geq 0$$

However, in practice this condition is replaced by

$$d \geq -\varepsilon e$$

where $e^T = (1, \dots, 1)$ and $\varepsilon = 10^{-6}$, the feasibility tolerance.

Reason: **Degeneracy**. In 1972 Paula Harris suggested exploiting this fact to improve numerical stability.

(Issue 4 – Ratio test & finiteness)

$$\boxed{\text{STANDARD RATIO TEST}} \quad j_{\text{enter}} = \operatorname{argmin}\{D_j / \alpha_j : \alpha_j > 0\}$$

Motivation: Feasibility \Rightarrow step length θ satisfies

$$D_N - \theta \alpha_N \geq 0$$

Since the bigger the step length, the bigger the change in the objective, we choose

$$\theta_{\max} = \min\{D_j / \alpha_j : \alpha_j > 0\}$$

Using ε , we have

$$\theta^{\varepsilon}_{\max} = \min\{(D_j + \varepsilon) / \alpha_j : \alpha_j > 0\} > \theta_{\max}$$

$$\boxed{\text{HARRIS RATIO TEST}} \quad j_{\text{enter}} = \operatorname{argmax}\{\alpha_j : \theta_{\max} \leq D_j / \alpha_j \leq \theta^{\varepsilon}_{\max}, \alpha_j > 0\}$$

(Issue 4 – Ratio test & finiteness)

- ▶ Advantages
 - Numerical stability – $\alpha_{jenter} =$ “pivot element”
 - Degeneracy – Reduces # of 0-length steps
- ▶ Disadvantage
 - $D_{jenter} < 0 \Rightarrow$ objective goes in wrong direction
- ▶ Solution: BOUND SHIFTING
 - If $D_{jenter} < 0$, we replace the lower bound on d_{jenter} by something less than its current value.
 - Note that this shift changes the problem and must be removed: 5% of cases, this produces dual infeasibility \Rightarrow process is iterated.

Example: Bound-Shifting Removal

Read MPS format model from file gap12.mps.bz2

Optimize a model with 3192 rows, 8856 columns and 38304 nonzeros

Iteration	Objective	Primal Inf.	Dual Inf.	Time
0	0.0000000e+00	1.230000e+02	0.000000e+00	0s
101	0.0000000e+00	7.229833e+02	0.000000e+00	0s
173	3.3669520e+00	9.125960e+02	0.000000e+00	0s
....				
49843	5.2387894e+02	5.585623e+01	0.000000e+00	32s
50213	5.2388556e+02	7.361090e+00	0.000000e+00	32s
50584	5.2388824e+02	1.648797e+01	0.000000e+00	32s
50744	5.2388840e+02	0.000000e+00	0.000000e+00	33s
Switch to primal				
50934	5.2289692e+02	0.000000e+00	3.404469e+01	33s
51123	5.2289527e+02	0.000000e+00	1.021229e+00	33s
51312	5.2289450e+02	0.000000e+00	2.841123e-01	33s
51499	5.2289434e+02	0.000000e+00	1.686059e-01	33s
Perturbation ends				
51516	5.2289435e+02	0.000000e+00	0.000000e+00	33s

Shift removed

Shift removed

Solved in 51516 iterations and 33.15 seconds

Optimal objective 5.228943506e+02

(Issue 4 – Ratio test & finiteness)

Finiteness: Bound shifting is closely related to the “perturbation” method employed in Gurobi if no progress is being made in the objective.

If “insufficient” progress is being made, replace

$$d_j \geq -\varepsilon \quad j = 1, \dots, n$$

by

$$d_j \geq -\varepsilon - \varepsilon_j \quad j = 1, \dots, n,$$

where ε_j is pseudo-random uniform on $[0, \varepsilon]$. This makes the probability of a 0-length step very small, and in practice has been sufficient to guarantee finiteness.

Issue 5

Bound Flipping (Long-Step Dual)

- ▶ A basis is given by a triple (B,L,U)
 - L = non-basics at lower bound: Feasibility $D_L \geq 0$
 - U = non-basics at upper bound: Feasibility $D_U \leq 0$
- ▶ **Ratio test:** Suppose X_{B_i} is the leaving variable, and the step length is blocked by some variable $d_j, j \in L$, where d_j is about to become negative and $u_j < +\infty$:
 - **Flipping means:** Move j from L to U .
 - **Check:** Do an update to see if X_{B_i} is still favorable (just as we did in Phase II)
- ▶ Can combine many iterations into a single iteration.

Example: Bound Flipping

Read MPS format model from file fit2d.mps.bz2
Optimize a model with 25 rows, 10500 columns and 129018 nonzeros

Iteration	Objective	Primal Inf.	Dual Inf.	Time
0	-9.1662550e+04	9.553095e+03	0.000000e+00	0s
6023	-6.8464293e+04	0.000000e+00	0.000000e+00	1s

Solved in 6023 iterations and 0.82 seconds
Optimal objective -6.846429329e+04

w/o flipping

Read MPS format model from file fit2d.mps.bz2
Optimize a model with 25 rows, 10500 columns and 129018 nonzero

Iteration	Objective	Primal Inf.	Dual Inf.	Time
0	-9.1662550e+04	9.553095e+03	0.000000e+00	0s
255	-6.8464293e+04	0.000000e+00	0.000000e+00	0s

Solved in 255 iterations and 0.07 seconds
Optimal objective -6.846429329e+04

w/ flipping