Simplex Algorithm

$\max \; cx$

$\text{sub.to} \; Ax \le b$

- We have already seen how to find feasible solutions, via Fourier-Motzkin elimination.

- By LP-Duality, we can use this method to find optimal solutions by finding a feasible solution to

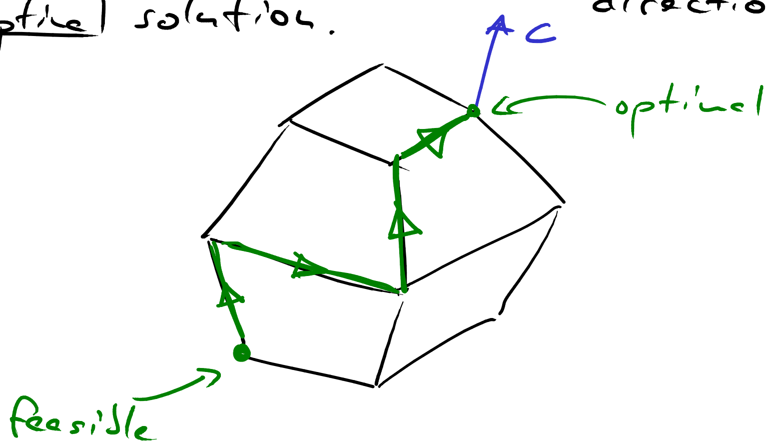
$$cx = yb$$
$$Ax \le b$$
$$yA = c$$
$$y \ge 0$$

- But: Fourier-Motzkin elimination

- The Simplex Algorithm is another method for find (optimal) solutions.
    - efficient in practice
    - in wide use today
    - NOT a polynomial time algorithm.

Simplex Algorithm

	GOAL	IDEA
PHASE I	Find some feasible solution, more precisely, a vertex of P.	Write down another LP, with a trivial feasible solution, whose optimal solution is a vertex of P.
PHASE II	Improve this solution until you find an optimal solution.	Walk along the edges of P in an improving direction.

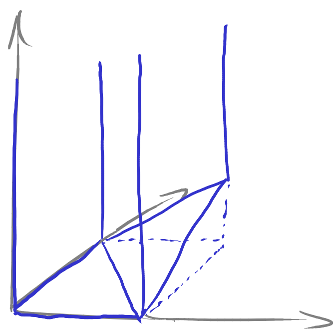


Phase I

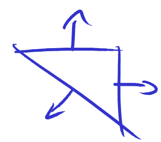
- Assume the LP is of the form $\max\ cx$, $Ax \le b$, $x \ge 0$ (1)

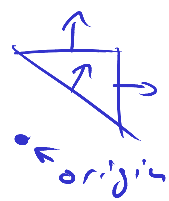
- Split $Ax \le b$ into $\begin{array}{l} A_1 x \le b_1 \\ A_2 x \ge b_2 \end{array}$ for $b_1 \ge 0$, $b_2 > 0$

$$\begin{array}{ll} \max & \mathbb{1}(A_2 x - z) \\ \text{subto} & A_1 x \le b_1 \\ & A_2 x - z \le b_2 \\ & x \ge 0 \\ & z \ge 0 \end{array} \qquad (2)$$



Example





$$\begin{array}{l} x_1 \le 1 \\ x_2 \le 1 \\ -x_1 - x_2 \le -1 \end{array}$$

$\swarrow$ $\searrow$

$$\begin{array}{l} x_1 \le 1 \\ x_2 \le 1 \end{array} \qquad x_1 + x_2 \ge 1$$

$\searrow$ $\swarrow$

$$\begin{array}{l} x_1 \le 1 \\ x_2 \le 1 \\ x_1 + x_2 - z \le 1 \\ x_1, x_2, z \ge 0 \end{array}$$

$$\max\ cx \quad Ax \le b \quad x \ge 0 \qquad (1)$$

$$\max\ \mathbb{1}(A_2 x - z) \quad \text{s.t.} \quad A_1 x \le b_1,\ A_2 x - z \le b_2,\ x \ge 0,\ z \ge 0 \qquad (2)$$

$A_1 x \le b_1$
$A_2 x \ge b_2$ for $b_1 \ge 0$, $b_2 > 0$.

▷ $x = 0$, $z = 0$ is feasible!

▷ If x' is a feasible solution for (1), then $x = x'$, $z = A_2 x' - b_2$ is a feasible solution for (2) with $\mathbb{1}(A_2 x - z) = \mathbb{1} b_2$ ✓

$$A_1 x \le b_1 \ ✓ \qquad x \ge 0 \ ✓$$

$$A_2 x - z = b_2 \le b_2 \ ✓ \qquad z = A_2 x' - b_2 \ge 0 \ ✓$$

▷ Any feasible solution of (2) has $\mathbb{1}(A_2 x - z) \le \mathbb{1} b_2$.

▷ If (1) has a solution, then (2) has an optimal solution x with $\mathbb{1}(A_2 x - z) = \mathbb{1} b_2$. Claim: x is a vertex of P.

$$A_2 x - z \le b_2 \ \wedge\ \mathbb{1}(A_2 x - z) = \mathbb{1} b_2 \Rightarrow A_2 x - z = b_2$$

$$\Rightarrow A_2 x = b_2 + z \ \wedge\ z \ge 0 \Rightarrow A_2 x \ge b_2 \ ✓$$

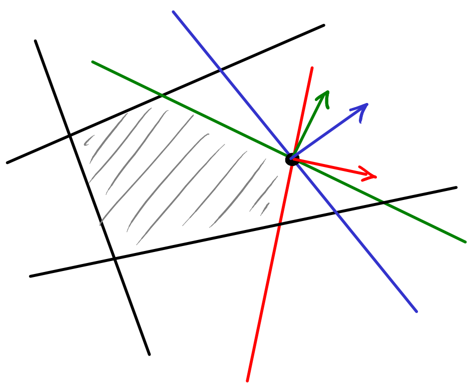
$$A_1 x \le b_1 \ ✓$$

TODO: Track vertices! □

Phase II Given $\max cx$ and vertex x_0 of $P = \{x \mid Ax \le b\}$.
$$Ax \le b$$

- Choose set of rows B such that $A_B x_0 = b_B$ and A is square and non-singular.
- Choose u such that $uA = c$. $\left[u_B = cA_B^{-1}\right.$ and $u_i = \begin{cases} u_{B_i} & i \in B \\ 0 & i \notin B \end{cases}$

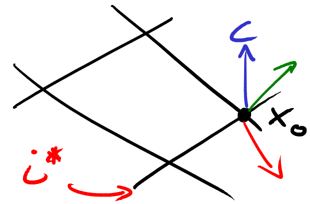
Case 1: $u \ge 0$. By the Duality Theorem, x is optimal and u an optimal solution of the dual problem!



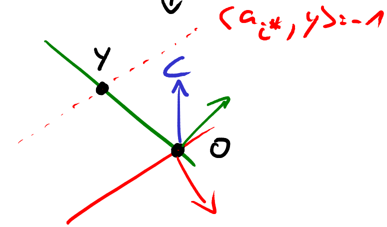
Case 2: $u \not\geq 0$, i.e., there exists a component $u_i < 0$.

$$A_B x_0 = b_B$$
$$uA = u_B A_B = c$$

- Let i^* be the smallest index such that $u_{i^*} < 0$.



translate A_B to origin!



- Choose y such that

$$A_B y = -e_{i^*} \iff$$

$$y = -A_B^{-1} e_{i^*} = i^*\text{-th column of } -A_B^{-1}$$

$$cy = uAy = u_B A_B y = -u_B e_{i^*} = -u_{i^*} > 0$$

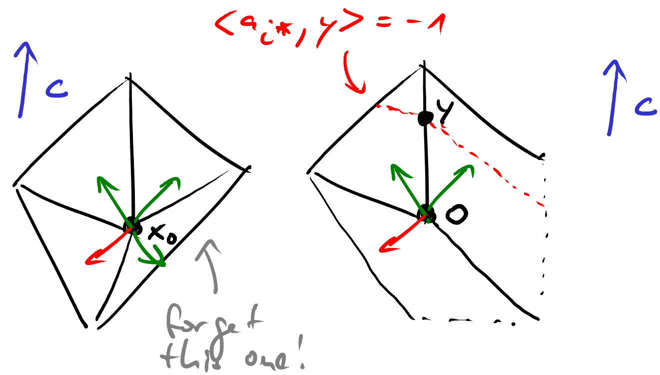
- Consider the ray

$$R = \{x_0 + \lambda y \mid \lambda \geq 0\}$$

Three possibilities

Case 2a • $R \subset P$

Case 2b $\begin{cases} \bullet\ R \text{ contains an edge of } P \\ \bullet\ R \cap P = \{x_0\} \end{cases}$



<u>Case 2a:</u> $Ay \leq 0$. $\Rightarrow$ $x_0 + \lambda y \in P$ for all $\lambda \geq 0$.

Then the maximum cx, $x \in P$ is unbounded.

<u>Case 2b:</u> There exists a row j such that $\langle a_j, y \rangle > 0$.

$x_0 + \lambda y \in R \cap H_{a_j, b_j} \Leftrightarrow a_j(x_0 + \lambda y) = b_j \Leftrightarrow \lambda a_j y = b_j - a_j x_0 \Leftrightarrow \lambda = \frac{b_j - a_j x_0}{a_j y}$

The "last" point in R that is still in P is given by

$$\lambda_0 = \min_j \{\lambda \mid x_0 + \lambda y \in R \cap H_{a_j, b_j},\ a_j y > 0\} = \min_j \left\{ \frac{b_j - a_j x_0}{a_j y} \;\middle|\; a_j y > 0 \right\}$$

Let j^* be the <u>smallest</u> index where this minimum is attained.

Restart Phase 2 with

vertex $x_0 + \lambda_0 y$ and "basis" $B \cup \{j^*\} \setminus \{i^*\}$.

$\longrightarrow$ Sequence $x_0, B_0;$ $x_1, B_1;$ $x_2, B_2;$ $\ldots$

Theorem The Simplex Algorithm terminates.

Proof: Let x_k, B_k, u_k, y_k be the parameters in the k-th iteration.

▷ $cx_k \leq cx_{k+1}$ with equality iff $x_k = x_{k+1}$.

Suppose S. A. does not terminate

$\Rightarrow$ There exist k, l with $B_k = B_l \Rightarrow x_k = x_l$

$\Rightarrow x_k = x_{k+1} = \dots = x_l$ Cycling!

Let r denote the highest index such that

$$\exists p \text{ with } k \leq p < l : \; r \in B_p \text{ but } r \notin B_{p+1}$$

$\Rightarrow \exists q \text{ with } p < q < l : \; r \notin B_q \text{ but } r \in B_{q+1}$

$\Rightarrow$ for all $j > r$: $\; j \in B_p \Leftrightarrow j \in B_q$

▷ r is the smallest index j with $u_{pj} < 0$.

▷ r is the smallest index j with $a_j x_q = b_j$ and $a_j y_q > 0$.

▷ $u_p A y_q = c y_q > 0 \Rightarrow \exists j: u_{pj}(a_j y_q) > 0$

▷ r is the smallest index j with $u_{pj} < 0$.

▷ r is the smallest index j with $a_j y_q > 0$ s.t. $\frac{b_j - a_j x_q}{a_j y_q}$ minimal

1) If $j \notin B_p$: $u_{pj} = 0$. ↯

2) If $j \in B_p$ and $j < r$: $u_{pj} \geq 0$ and $a_j y_q \leq 0$.

⌜Suppose $a_j y_q > 0$. Then

$x_p = x_q$ lies on all hyperplanes $j \in \bigcup_{k \leq n \leq \ell} B_n$!

$$0 = \frac{b_r - a_r x_q}{a_r y_q} < \frac{b_j - a_j x_q}{a_j y_q} = 0 \quad . \ ↯ \ ⌟$$

3) If $j \in B_p$ and $j = r$: $u_{pj} < 0$ and $a_j y_q > 0$. ↯

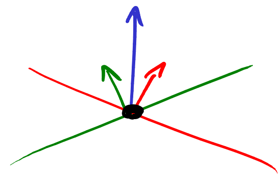
4) If $j \in B_p$ and $j > r$, then $j \in B_q \Rightarrow a_j y_q = 0$ ↯

□

Summary of Phase II

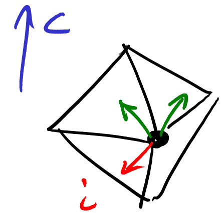
Given x s.t. $Ax \le b$ and B s.t. $A_B x = b_B$.

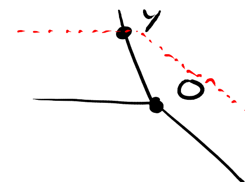
Compute u s.t. $uA = c$ and $\mathrm{supp}(u) \subset B$.



Case 1: $u \ge 0$.

optimal solution found!

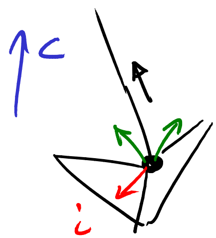




Case 2: $u_i < 0$.

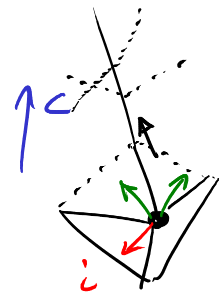
Compute y s.t. $A_B y = -e_i$.

$$R := \{x + \lambda y \mid \lambda \ge 0\}$$



Case 2a: $Ay \le 0$

unbounded optimal value!

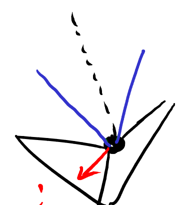


Case 2b: $a_j y > 0$

$$\lambda = \min_j \left\{ \frac{b_j - a_j x}{a_j y} \;\middle|\; a_j y > 0 \right\}$$

REPEAT: $x \to x + \lambda y$,

$B \to B \cup \{j\} \setminus \{i\}$



beware cycles when $x = x + \lambda y$!

- ▷ There are explicit and efficient update rules for B, x, y, u.

- ▷ There are entire courses on how to implement the Simplex Algorithm. (efficiency + stability)

- ▷ There are examples that force SA to run through all vertices, even though there <u>is</u> a short path to the optimum.
  $\rightarrow$ worst-case complexity <u>not</u> polynomial time.

- ▷ Are there polytopes that do not have short paths to the optimum?

- ▷ Hirsch Conjecture: diameter $\leq$ # facet − dimension
  $\rightarrow$ Counterexample by Paco Santos 2010.

- ▷ Polynomial Hirsch Conjecture: diameter $\leq$ poly(# facets, dimension)

▷ SA is very successful in practice - often more successful than methods that are polynomial time algorithms.

▷ For combinatorial optimization, it is often a good idea to run SA on the <u>dual</u> LP.

(Don't start with a working factory and make it more efficient. Start with a factory that does nothing and make it work!)