



# Linear Programming

Leo Liberti

LIX, École Polytechnique

`liberti@lix.polytechnique.fr`



# Contents

- LP formulations and examples
- The simplex method
- Optimality conditions
- Duality



# Lecture material

- Lecture notes:

[http://www.lix.polytechnique.fr/~liberti/isic/isc612-07/linear\\_programming.pdf](http://www.lix.polytechnique.fr/~liberti/isic/isc612-07/linear_programming.pdf)

- J.-B. Hiriart-Urruty, *Optimisation et analyse convexe*, PUF, Paris 1998 (Ch. 5)

- C. Papadimitriou, K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Dover, New York, 1998

# Definitions

- Mathematical programming formulation:

$$\left. \begin{array}{l} \min_x f(x) \\ \text{s.t. } g(x) \leq 0 \end{array} \right\} [P] \quad (1)$$

- A point  $x^*$  is *feasible* in  $P$  if  $g(x^*) \leq 0$ ;  
 $F(P)$  = set of feasible points of  $P$
- A feasible  $x^*$  is a *local minimum* if  $\exists B(x^*, \varepsilon)$  s.t.  
 $\forall x \in F(P) \cap B(x^*, \varepsilon)$  we have  $f(x^*) \leq f(x)$
- A feasible  $x^*$  is a *global minimum* if  $\forall x \in F(P)$  we have  
 $f(x^*) \leq f(x)$
- Thm.: if  $f$  and  $F(P)$  convex, any local min. is also global
- If  $g_i(x^*) = 0$  for some  $i$ ,  $g_i$  is *active* at  $x^*$

# Canonical form

- $P$  is a *linear programming problem* (LP) if  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$  are **linear forms**
- LP in *canonical form*:

$$\left. \begin{array}{l} \min_x \quad c^\top x \\ \text{s.t.} \quad Ax \leq b \\ \quad \quad x \geq 0 \end{array} \right\} [C] \quad (2)$$

- Can reformulate inequalities to equations by adding a non-negative *slack variable*  $x_{n+1} \geq 0$ :

$$\sum_{j=1}^n a_j x_j \leq b \quad \Rightarrow \quad \sum_{j=1}^n a_j x_j + x_{n+1} = b \quad \wedge \quad x_{n+1} \geq 0$$

# Standard form

- LP in *standard form*: all inequalities transformed to equations

$$\left. \begin{array}{l} \min_x \quad (c')^T x \\ \text{s.t.} \quad A'x = b \\ \quad \quad x \geq 0 \end{array} \right\} [S] \quad (3)$$

- where  $x = (x_1, \dots, x_n, x_{n+1}, \dots, x_{n+m})$ ,  
 $A' = (A, I_m)$ ,  $c' = (c, \underbrace{0, \dots, 0}_m)$

- Standard form useful because linear systems of equations are computationally easier to deal with than systems of inequalities
- Used in simplex algorithm

# Diet problem I

- Consider set  $M$  of  $m$  nutrients (e.g. sugars, fats, carbohydrates, proteins, vitamins, ...)
- Consider set  $N$  of  $n$  types of food (e.g. pasta, steak, potatoes, salad, ham, fruit, ...)
- A diet is healthy if it has at least  $b_i$  units of nutrient  $i \in M$
- Food  $j \in N$  contains  $a_{ij}$  units of nutrient  $i \in M$
- A unit of food  $j \in N$  costs  $c_j$
- Find a healthy diet of minimum cost

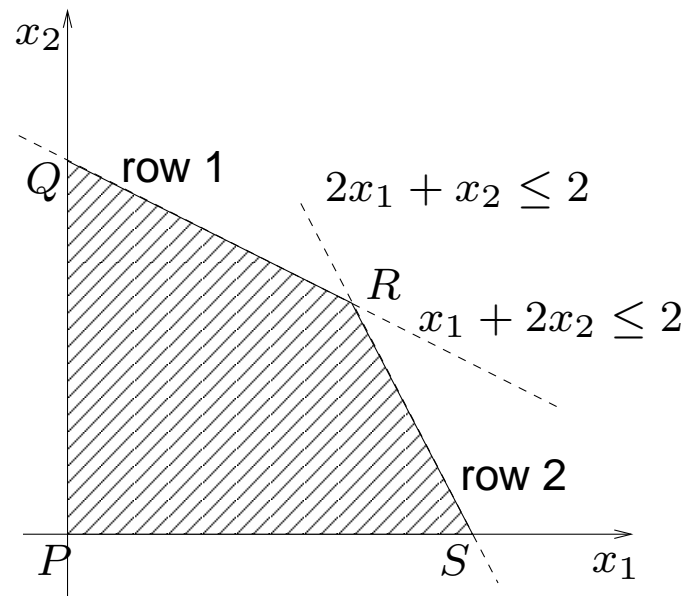
# Diet problem II

- Parameters:  $m \times n$  matrix  $A = (a_{ij})$ ,  $b = (b_1, \dots, b_m)$ ,  
 $c = (c_1, \dots, c_n)$
- Decision variables:  $x_j =$  quantity of food  $j$  in the diet
- Objective function:  $\min_x \sum_{j=1}^n c_j x_j$
- Constraints:  $\forall i \in M \sum_{j=1}^n a_{ij} x_j \geq b_i$
- Limits on variables:  $\forall j \in N x_j \geq 0$
- Canonical form:  $\min\{c^T x \mid -Ax \leq -b\}$
- Standard form: add slack variables  $y_i =$  surplus  
quantity of  $i$ -th nutrient, get  $\min\{c^T x \mid -Ax + I_m y = -b\}$



# Geometry of LP

- A *polyhedron* is the intersection of a finite number of closed halfspaces. A bounded, non-empty polyhedron is a *polytope*



Canonical feas. polyhedron:  $F(C) = \{x \in \mathbb{R}^n \mid Ax \leq b \wedge x \geq 0\}$

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}, b^T = (2, 2)$$

Standard feas. polyhedron:  $F(S) = \{(x, y) \in \mathbb{R}^{n+m} \mid Ax + I_m y = b \wedge (x, y) \geq 0\}$

- $P = (0, 0, 2, 2), Q = (0, 1, 0, 1), R = (\frac{2}{3}, \frac{2}{3}, 0, 0), S = (1, 0, 1, 0)$
- Each vertex corresponds to an intersection of at least  $n$  hyperplanes  $\Rightarrow \geq n$  coordinates are zero

# Basic feasible solutions

- Consider polyhedron in “equation form”  
 $K = \{x \in \mathbb{R}^n \mid Ax = b \wedge x \geq 0\}$ .  $A$  is  $m \times n$  of rank  $m$   
**(N.B.  $n$  here is like  $n + m$  in last slide!)**
- A subset of  $m$  linearly independent columns of  $A$  is a *basis* of  $A$
- If  $\beta$  is the set of column indices of a basis of  $A$ , variables  $x_i$  are *basic* for  $i \in \beta$  and *nonbasic* for  $i \notin \beta$
- Partition  $A$  in a square  $m \times m$  nonsingular matrix  $B$  (columns indexed by  $\beta$ ) and an  $(n - m) \times m$  matrix  $N$
- Write  $A = (B|N)$ ,  $x_B \in \mathbb{R}^m$  basics,  $x_N \in \mathbb{R}^{n-m}$  nonbasics
- Given a basis  $(B|N)$  of  $A$ , the vector  $x = (x_B, x_N)$  is a *basic feasible solution* (bfs) of  $K$  with respect to the given basis if  $Ax = b$ ,  $x_B \geq 0$  and  $x_N = 0$

# Fundamental Theorem of LP

- Given a non-empty polyhedron  $K$  in “equation form”, there is a surjective mapping between bfs and vertices of  $K$
- For any  $c \in \mathbb{R}^n$ , either there is at least one bfs that solves the LP  $\min\{c^T x \mid x \in K\}$ , or the problem is unbounded
- Proofs not difficult but long (see lecture notes or Papadimitriou and Steiglitz)
- Important correspondence between bfs’s and vertices suggests geometric solution method based on exploring vertices of  $K$

# Simplex Algorithm: Summary

- Solves LPs in form  $\min_{x \in K} c^T x$  where  $K = \{Ax = b \wedge x \geq 0\}$
- Starts from any vertex  $x$
- Moves to an adjacent improving vertex  $x'$   
(i.e.  $x'$  is s.t.  $\exists$  edge  $\{x, x'\}$  in  $K$  and  $c^T x' \leq c^T x$ )
- Two bfs's with basic vars indexed by sets  $\beta, \beta'$   
correspond to adjacent vertices if  $|\beta \cap \beta'| = m - 1$
- Stops when no such  $x'$  exists
- Detects unboundedness and prevents cycling  $\Rightarrow$   
convergence
- $K$  convex  $\Rightarrow$  global optimality follows from local  
optimality at termination

# Simplex Algorithm I

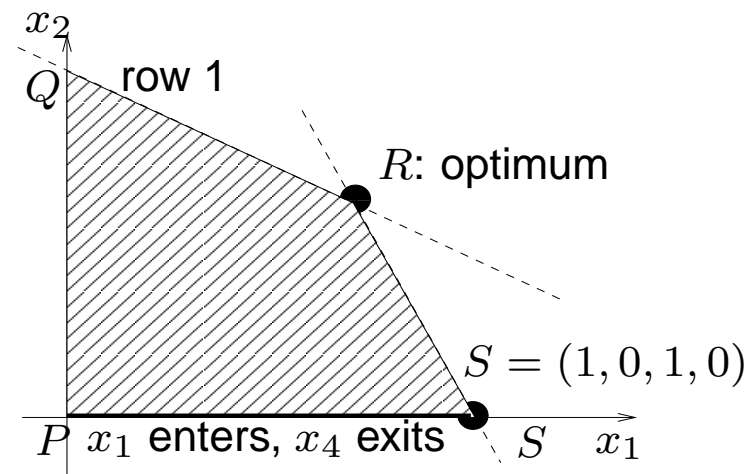
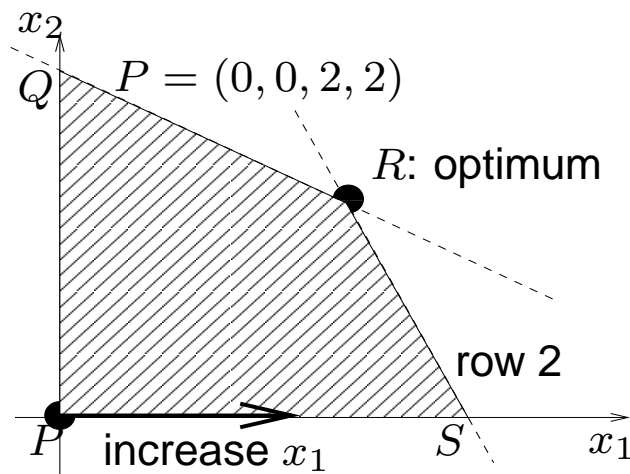


- Let  $x = (x_1, \dots, x_n)$  be the current bfs, write  $Ax = b$  as  $Bx_B + Nx_N = b$
- Express basics in terms of nonbasics:  
 $x_B = B^{-1}b - B^{-1}Nx_N$  (this system is a *dictionary*)
- Express objective function in terms of nonbasics:  
 $c^T x = c_B^T x_B + c_N^T x_N = c_B^T (B^{-1}b - B^{-1}Nx_N) + c_N^T x_N \Rightarrow$   
 $\Rightarrow c^T x = c_B^T B^{-1}b + \bar{c}_N^T x_N$   
( $\bar{c}_N^T = c_N^T - c_B^T B^{-1}N$  are the *reduced costs*)
- Select an improving direction: choose a nonbasic variable  $x_h$  with negative reduced cost; increasing its value will decrease the objective function value
- If no such  $h$  exists, no improving direction, local minimum  $\Rightarrow$  global minimum  $\Rightarrow$  termination

# Simplex Algorithm II



- Iteration start:  $x_h$  is out of basis  $\Rightarrow$  its value is zero
- We want to increase its value to strictly positive to decrease objective function value
- ... corresponds to “moving along an edge”
- We stop when we reach another (improving) vertex
- ... corresponds to setting a basic variable  $x_l$  to zero



- $x_h$  enters the basis,  $x_l$  exits the basis

# Simplex Algorithm III

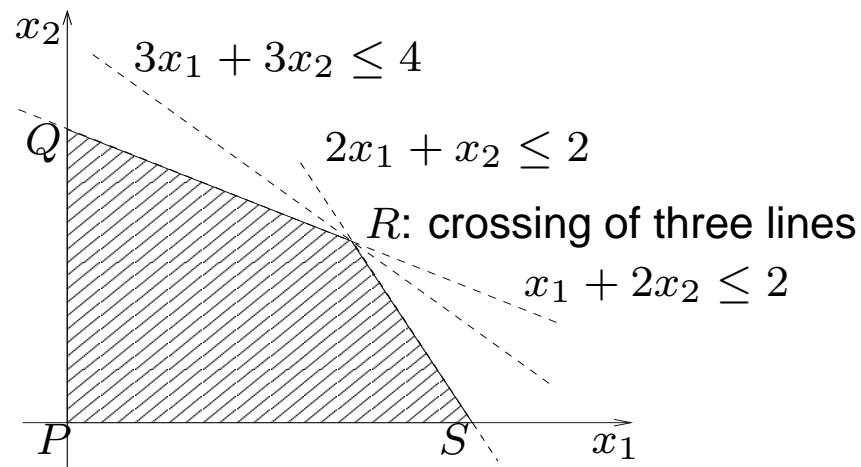


- How do we determine  $l$  and new positive value for  $x_h$ ?
- Recall dictionary  $x_B = B^{-1}b - B^{-1}Nx_N$ , write  $\bar{b} = B^{-1}b$  and  $\bar{A} = (\bar{a}_{ij}) = B^{-1}N$
- For  $i \in \beta$  (basics),  $x_i = \bar{b}_i - \sum_{j \notin \beta} \bar{a}_{ij}x_j$
- Consider nonbasic index  $h$  of variable entering basis (all the other nonbasics stay at 0), get  $x_i = \bar{b}_i - \bar{a}_{ih}x_h, \forall i \in \beta$
- Increasing  $x_h$  may make  $x_i < 0$  (infeasible), to prevent this enforce  $\forall i \in \beta (\bar{b}_i - \bar{a}_{ih}x_h \geq 0)$
- Require  $x_h \leq \frac{\bar{b}_i}{\bar{a}_{ih}}$  for  $i \in \beta$  and  $\bar{a}_{ih} > 0$ :  
$$l = \operatorname{argmin} \left\{ \frac{\bar{b}_i}{\bar{a}_{ih}} \mid i \in \beta \wedge \bar{a}_{ih} > 0 \right\}, \quad x_h = \frac{\bar{b}_l}{\bar{a}_{lh}}$$
- If all  $\bar{a}_{ih} \leq 0$ ,  $x_h$  can increase without limits: problem unbounded

# Simplex Algorithm IV



- Suppose  $> n$  hyperplanes cross at vtx  $R$  (*degenerate*)
- May get improving direction s.t. adjacent vertex is still  $R$
- Objective function value does not change
- Seq. of improving dirs. may fail to move away from  $R$
- $\Rightarrow$  simplex algorithm cycles indefinitely
- Use Bland's rule: among candidate entering / exiting variables, choose that with *least index*







# Example: Formulation

● Consider problem:

$$\left. \begin{array}{l} \max_{x_1, x_2} \quad x_1 + x_2 \\ \text{s.t.} \quad x_1 + 2x_2 \leq 2 \\ \quad \quad 2x_1 + x_2 \leq 2 \\ \quad \quad x \geq 0 \end{array} \right\}$$

● Standard form:

$$\left. \begin{array}{l} - \min_x \quad -x_1 - x_2 \\ \text{s.t.} \quad x_1 + 2x_2 + x_3 = 2 \\ \quad \quad 2x_1 + x_2 + x_4 = 2 \\ \quad \quad x \geq 0 \end{array} \right\}$$

● Obj. fun.:  $\max f = - \min -f$ , simply solve for  $\min f$



# Example, itn 1: start

- Objective function vector  $c^T = (-1, -1, 0, 0)$
- Constraints in matrix form:

$$\begin{pmatrix} 1 & 2 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

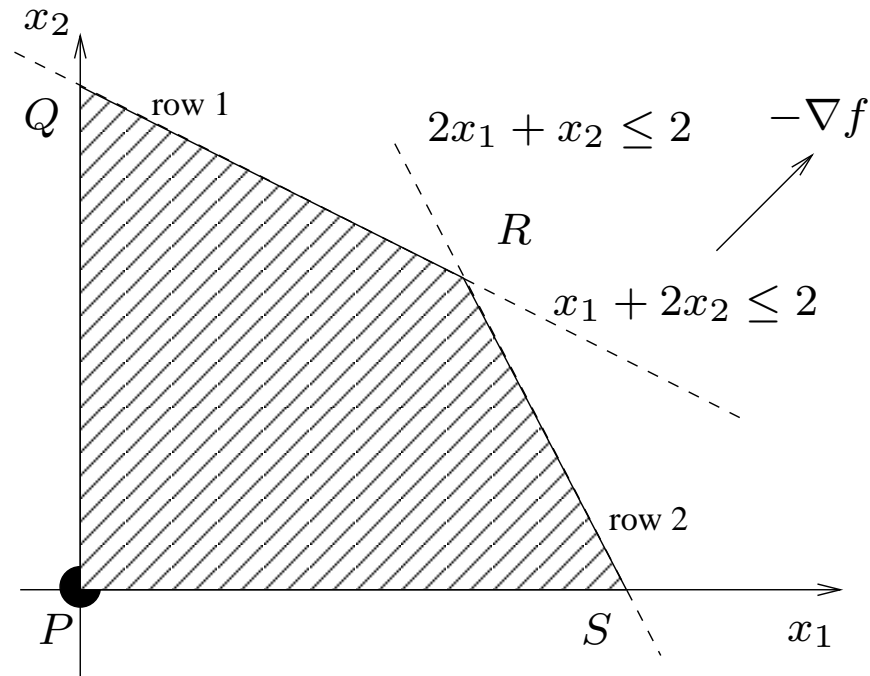
- Choose obvious starting basis with

$$B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, N = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}, \beta = \{3, 4\}$$

- Corresponds to point  $P = (0, 0, 2, 2)$

# Example, itn 1: dictionary

- Start the simplex algorithm with basis in  $P$



- Compute dictionary  $x_B = B^{-1}b - B^{-1}Nx_N = \bar{b} - \bar{A}x_N$ , where

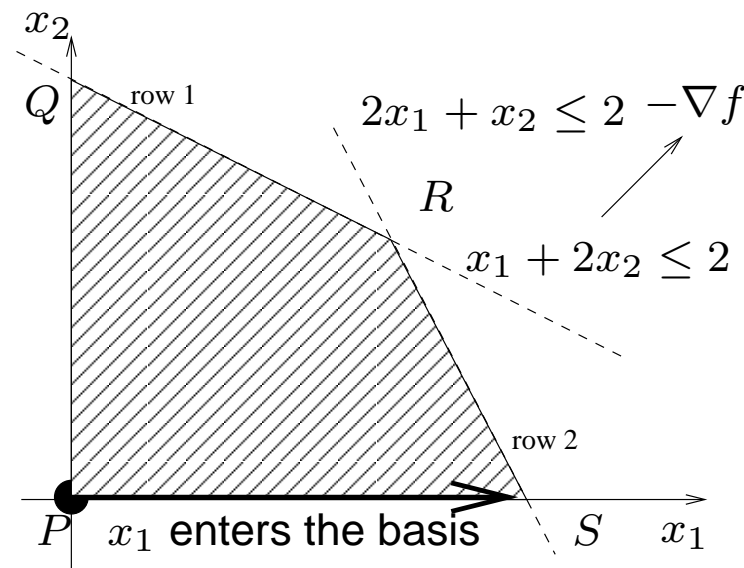
$$\bar{b} = \begin{pmatrix} 2 \\ 2 \end{pmatrix} ; \quad \bar{A} = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$$

# Example, itn 1: entering var

- Compute reduced costs  $\bar{c}_N = c_N^T - c_B^T \bar{A}$ :

$$(\bar{c}_1, \bar{c}_2) = (-1, -1) - (0, 0)\bar{A} = (-1, -1)$$

- All nonbasic variables  $\{x_1, x_2\}$  have negative reduced cost, can choose whichever to enter the basis
- Bland's rule: choose entering nonbasic with least index in  $\{x_1, x_2\}$ , i.e. pick  $h = 1$  (move along edge  $\overline{PS}$ )





# Example, itn 1: exiting var

- Select exiting basic index  $l$

$$\begin{aligned} l &= \operatorname{argmin}\left\{\frac{\bar{b}_i}{\bar{a}_{ih}} \mid i \in \beta \wedge \bar{a}_{ih} > 0\right\} = \operatorname{argmin}\left\{\frac{\bar{b}_1}{\bar{a}_{11}}, \frac{\bar{b}_2}{\bar{a}_{21}}\right\} \\ &= \operatorname{argmin}\left\{\frac{2}{1}, \frac{2}{2}\right\} = \operatorname{argmin}\{2, 1\} = 2 \end{aligned}$$

- Means: “select second basic variable to exit the basis”, i.e.  $x_4$
- Select new value  $\frac{\bar{b}_l}{\bar{a}_{lh}}$  for  $x_h$  (recall  $h = 1$  corresponds to  $x_1$ ):

$$\frac{\bar{b}_l}{\bar{a}_{lh}} = \frac{\bar{b}_2}{\bar{a}_{21}} = \frac{2}{2} = 1$$

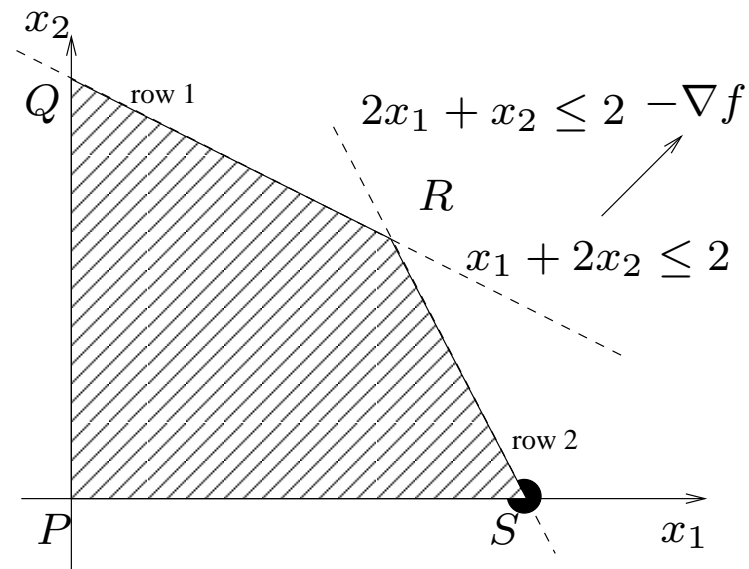
- $x_1$  enters,  $x_4$  exits (apply swap  $(1, 4)$  to  $\beta$ )

# Example, itn 2: start

- Start of new iteration: basis is  $\beta = \{1, 3\}$

$$B = \begin{pmatrix} 1 & 1 \\ 2 & 0 \end{pmatrix} ; \quad B^{-1} = \begin{pmatrix} 0 & \frac{1}{2} \\ 1 & -\frac{1}{2} \end{pmatrix}$$

- $x_B = (x_1, x_3) = B^{-1}b = (1, 1)$ , thus current bfs is  $(1, 0, 1, 0) = S$



# Example, itn 2: entering var

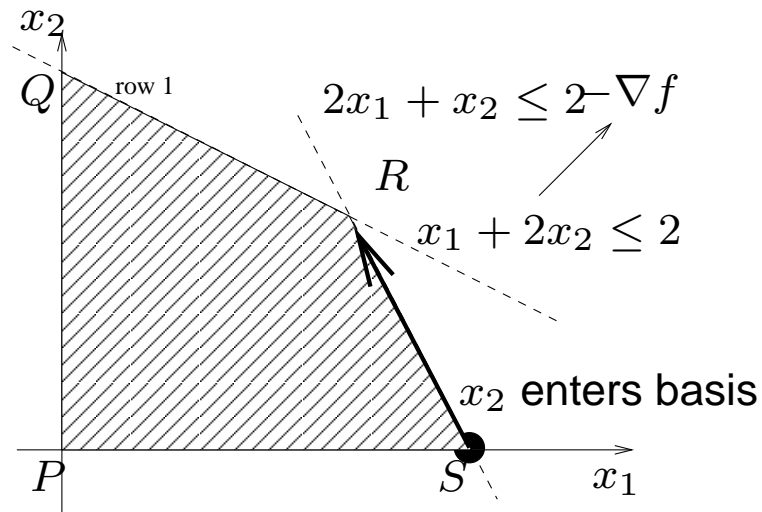
- Compute dictionary:  $\bar{b} = B^{-1}b = (1, 1)^\top$ ,

$$\bar{A} = B^{-1}N = \begin{pmatrix} 0 & \frac{1}{2} \\ 1 & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} 2 & 0 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{3}{2} & -\frac{1}{2} \end{pmatrix}$$

- Compute reduced costs:

$$(\bar{c}_2, \bar{c}_4) = (-1, 0) - (-1, 0)\bar{A} = (-1/2, 1/2)$$

- Pick  $h = 1$  (corresponds to  $x_2$  entering the basis)



# Example, itn 2: exiting var

- Compute  $l$  and new value for  $x_2$ :

$$\begin{aligned} l &= \operatorname{argmin}\left\{\frac{\bar{b}_1}{\bar{a}_{11}}, \frac{\bar{b}_2}{\bar{a}_{21}}\right\} = \operatorname{argmin}\left\{\frac{1}{1/2}, \frac{1}{3/2}\right\} = \\ &= \operatorname{argmin}\{2, 2/3\} = 2 \end{aligned}$$

- $l = 2$  corresponds to second basic variable  $x_3$
- New value for  $x_2$  entering basis:  $\frac{2}{3}$
- $x_2$  enters,  $x_3$  exits (apply swap  $(2, 3)$  to  $\beta$ )

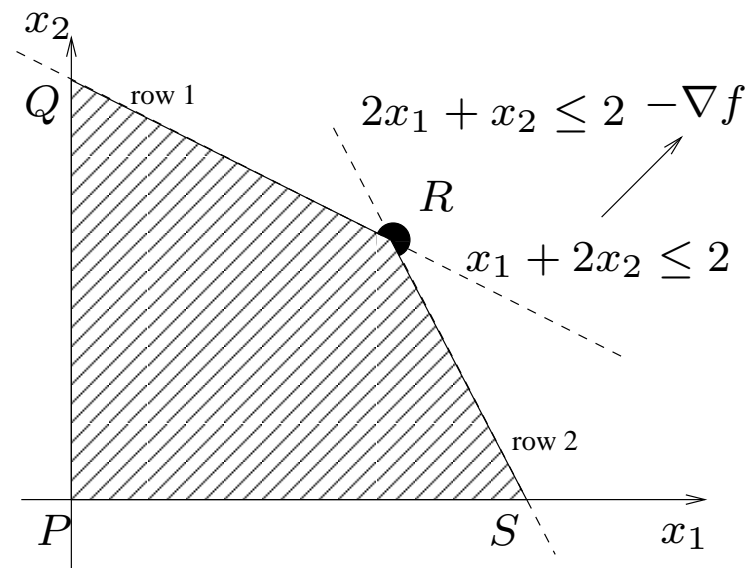


# Example, itn 3: start

- Start of new iteration: basis is  $\beta = \{1, 2\}$

$$B = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} ; \quad B^{-1} = \begin{pmatrix} -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{pmatrix}$$

- $x_B = (x_1, x_2) = B^{-1}b = (\frac{2}{3}, \frac{2}{3})$ , thus current bfs is  $(\frac{2}{3}, \frac{2}{3}, 0, 0) = R$



# Example, itn 3: termination

- Compute dictionary:  $\bar{b} = B^{-1}b = (2/3, 2/3)^T$ ,

$$\bar{A} = B^{-1}N = \begin{pmatrix} -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{pmatrix}$$

- Compute reduced costs:

$$(\bar{c}_3, \bar{c}_4) = (0, 0) - (-1, -1)\bar{A} = (1/3, 1/3)$$

- No negative reduced cost: algorithm terminates

- Optimal basis:  $\{1, 2\}$

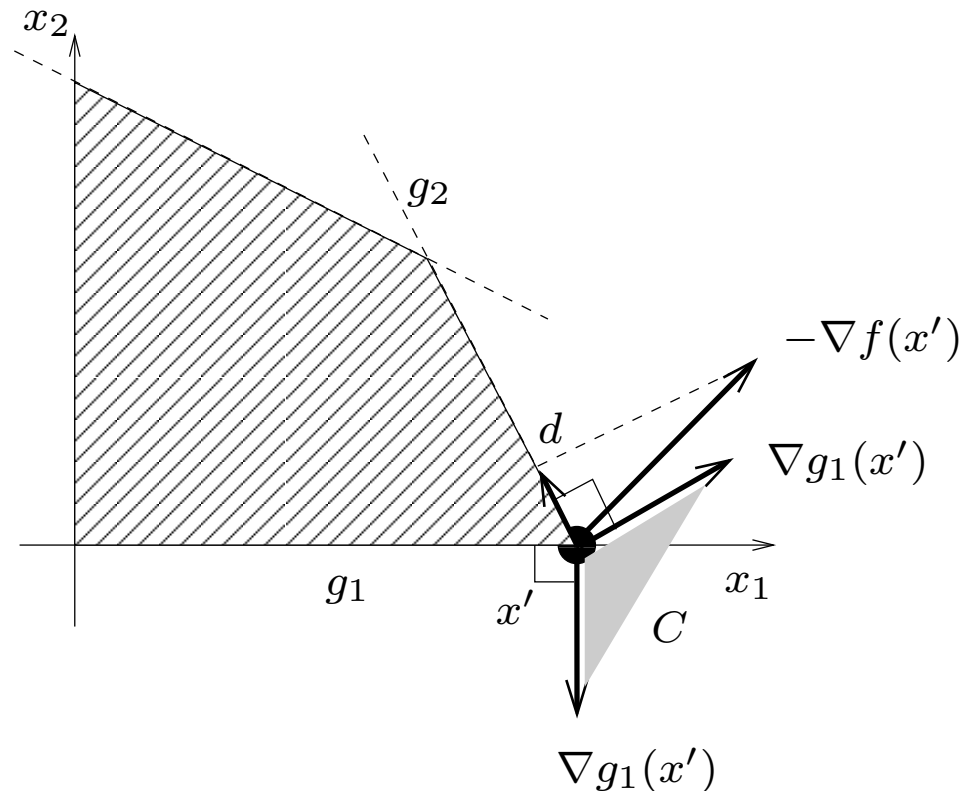
- Optimal solution:  $R = (\frac{2}{3}, \frac{2}{3})$

- Optimal objective function value  $f(R) = -\frac{4}{3}$

- Permutation to apply to initial basis  $\{3, 4\}$ :  $(1, 4)(2, 3)$

# Optimality Conditions I

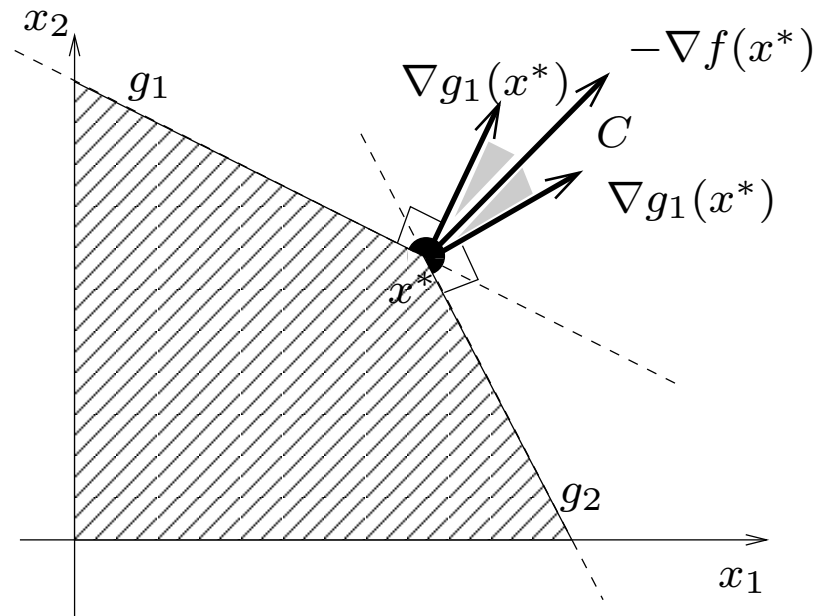
- If we can project improving direction  $-\nabla f(x')$  on an active constraint  $g_2$  and obtain a feasible direction  $d$ , point  $x'$  is not optimal



- Implies  $-\nabla f(x') \notin C$  (cone generated by active constraint gradients)

# Optimality Conditions II

- Geometric intuition: situation as above does not happen when  $-\nabla f(x^*) \in C$ ,  $x^*$  optimum



- Projection of  $-\nabla f(x^*)$  on active constraints is never a feasible direction

# Optimality Conditions III



If:

1.  $x^*$  is a local minimum of problem

$$P \equiv \min\{f(x) \mid g(x) \leq 0\},$$

2.  $I$  is the index set of the active constraints at  $x^*$ ,

$$\text{i.e. } \forall i \in I (g_i(x^*) = 0)$$

3.  $\nabla g_I(x^*) = \{\nabla g_i(x^*) \mid i \in I\}$  is a linearly independent set of vectors



then  $-\nabla f(x^*)$  is a conic combination of  $\nabla g_I(x^*)$ ,

i.e.  $\exists \lambda \in \mathbb{R}^{|I|}$  such that

$$\nabla f(x^*) + \sum_{i \in I} \lambda_i \nabla g_i(x^*) = 0$$

$$\forall i \in I \lambda_i \geq 0$$



# Karush-Kuhn-Tucker Conditions

- Define

$$L(x, \lambda) = f(x) + \sum_{i=1}^m \lambda_i g_i(x)$$

as the *Lagrangian* of problem  $P$

- KKT: If  $x^*$  is a local minimum of problem  $P$  and  $\nabla g(x^*)$  is a linearly independent set of vectors,  $\exists \lambda \in \mathbb{R}^m$  s.t.

$$\begin{aligned} \nabla_{x^*} L(x, \lambda) &= 0 \\ \forall i \leq m \quad (\lambda_i g_i(x^*)) &= 0 \\ \forall i \leq m \quad (\lambda_i &\geq 0) \end{aligned}$$

# Weak duality



Thm.

Let  $\bar{L}(\lambda) = \min_{x \in F(P)} L(x, \lambda)$  and  $x^*$  be the global optimum of  $P$ . Then  $\forall \lambda \geq 0 \quad \bar{L}(\lambda) \leq f(x^*)$ .

Proof

Since  $\lambda \geq 0$ , if  $x \in F(P)$  then  $\lambda_i g_i(x) \leq 0$ , hence  $L(x, \lambda) \leq f(x)$ ; result follows as we are taking the minimum over all  $x \in F(P)$ .

- Important point:  $\bar{L}(\lambda)$  is a lower bound for  $P$  for all  $\lambda \geq 0$
- The problem of finding the tightest Lagrangian lower bound

$$\max_{\lambda \geq 0} \min_{x \in F(P)} L(x, \lambda)$$

is the *Lagrangian dual* of problem  $P$

# Dual of an LP I

- Consider LP  $P$  in form:  $\min\{c^T x \mid Ax \geq b \wedge x \geq 0\}$
- $L(x, s, y) = c^T x - s^T x + y^T(b - Ax)$  where  $s \in \mathbb{R}^n$ ,  $y \in \mathbb{R}^m$
- Lagrangian dual:

$$\max_{s, y \geq 0} \min_{x \in F(P)} (yb + (c^T - s - yA)x)$$

- KKT: for a point  $x$  to be optimal,

$$c^T - s - yA = 0 \text{ (KKT1)}$$

$$\forall j \leq n (s_j x_j = 0), \forall i \leq m (y_i (b_i - A_i x) = 0) \text{ (KKT2)}$$

$$s, y \geq 0 \text{ (KKT3)}$$

- Consider Lagrangian dual s.t. (KKT1), (KKT3):





# Dual of an LP II

● Obtain:

$$\left. \begin{array}{ll} \max_{s,y} & yb \\ \text{s.t.} & yA + s = c^T \\ & s, y \geq 0 \end{array} \right\}$$

● Interpret  $s$  as slack variables, get *dual of LP*:

$$\left. \begin{array}{ll} \min_x & cx \\ \text{s.t.} & Ax \geq b \\ & x \geq 0 \end{array} \right\} [P] \longrightarrow \left. \begin{array}{ll} \max_y & yb \\ \text{s.t.} & yA \leq c^T \\ & y \geq 0 \end{array} \right\} [D]$$

# Strong Duality

Thm.

If  $x$  is optimum of a linear problem and  $y$  is the optimum of its dual, primal and dual objective functions attain the same values at  $x$  and respectively  $y$ .

Proof

- Assume  $x$  optimum, KKT conditions hold
- Recall (KKT2)  $\forall j \leq n (s_j x_j = 0)$ ,  
 $\forall i \leq m (y_i (b_i - A_i x) = 0)$
- Get  $y(b - Ax) = sx \Rightarrow yb = (yA + s)x$
- By (KKT1)  $yA + s = c^T$
- Obtain  $yb = c^T x$

# The dual of the Diet Problem

- Recall diet problem: select minimum-cost diet of  $n$  foods providing  $m$  nutrients
- Suppose firm wishes to set the prices  $y \geq 0$  for  $m$  nutrient pills
- To be competitive with normal foods, the equivalent in pills of a food  $j \leq n$  must cost less than the cost of the food  $c_j$
- Objective:  $\max \sum_{i \leq m} b_i y_i$
- Constraints:  $\forall j \leq n \quad \sum_{i \leq m} a_{ij} y_i \leq c_j$
- Economic interpretation:  
at optimum, cost of pills = cost of diet



# Example: Dual formulation

- Primal problem  $P$  and canonical form:

$$\left. \begin{array}{l} \max_{x_1, x_2} \quad x_1 + x_2 \\ \text{s.t.} \quad x_1 + 2x_2 \leq 2 \\ \quad \quad 2x_1 + x_2 \leq 2 \\ \quad \quad x \geq 0 \end{array} \right\} \Rightarrow \left. \begin{array}{l} - \min_{x_1, x_2} \quad -x_1 - x_2 \\ \text{s.t.} \quad -x_1 - 2x_2 \geq -2 \\ \quad \quad -2x_1 - x_2 \geq -2 \\ \quad \quad x \geq 0 \end{array} \right\}$$

- Dual problem  $D$  and reformulation:

$$\left. \begin{array}{l} - \max_{y_1, y_2} \quad -2y_1 - 2y_2 \\ \text{s.t.} \quad -y_1 - 2y_2 \leq -1 \\ \quad \quad -2y_1 - y_2 \leq -1 \\ \quad \quad y \geq 0 \end{array} \right\} \Rightarrow \left. \begin{array}{l} \min_{y_1, y_2} \quad 2y_1 + 2y_2 \\ \text{s.t.} \quad y_1 + 2y_2 \geq 1 \\ \quad \quad 2y_1 + y_2 \geq 1 \\ \quad \quad y \geq 0 \end{array} \right\}$$

# Primal and Dual

Primal	Dual
min	max
<b>variables</b> $x$	<b>constraints</b>
<b>constraints</b>	<b>variables</b> $y$
objective coefficients $c$	constraint right hand sides $c$
constraint right hand sides $b$	objective coefficients $b$
$A_i x \geq b_i$	$y_i \geq 0$
$A_i x \leq b_i$	$y_i \leq 0$
$A_i x = b_i$	$y_i$ unconstrained
$x_j \geq 0$	$y A^j \leq c_j$
$x_j \leq 0$	$y A^j \geq c_j$
$x_j$ unconstrained	$y A^j = c_j$

$A_i$ :  $i$ -th row of  $A$

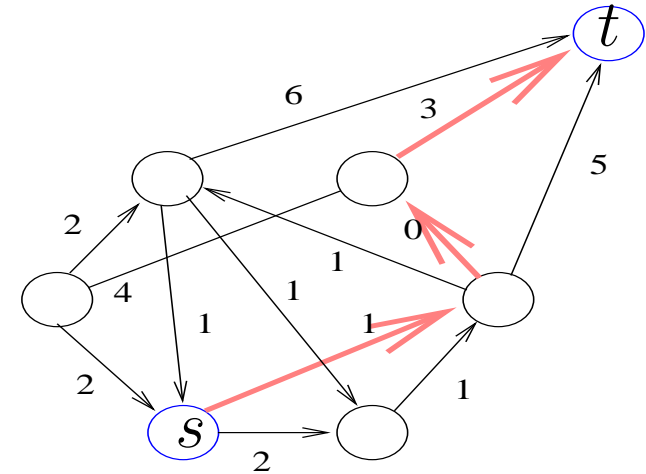
$A^j$ :  $j$ -th column of  $A$

# Example: Shortest Path

SHORTEST PATH PROBLEM.

*Input:* weighted digraph  $G = (V, A, c)$ , and  $s, t \in V$ .

*Output:* a minimum-weight path in  $G$  from  $s$  to  $t$ .

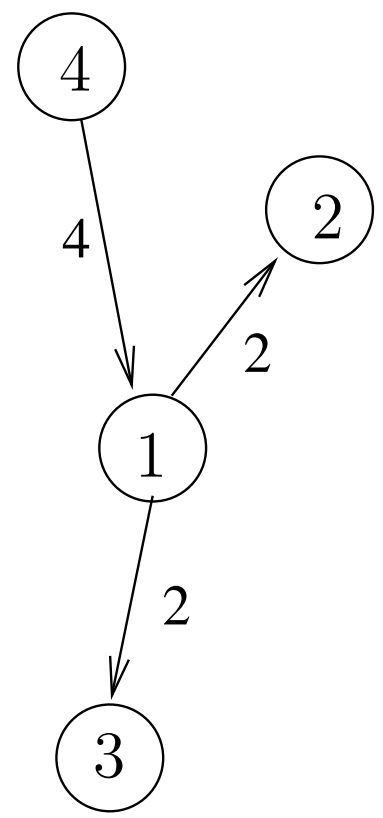


$$\left. \begin{array}{l} \min_{x \geq 0} \sum_{(u,v) \in A} c_{uv} x_{uv} \\ \forall v \in V \quad \sum_{(u,v) \in A} x_{uv} - \sum_{(v,u) \in A} x_{vu} = \begin{cases} 1 & v = s \\ -1 & v = t \\ 0 & \text{othw.} \end{cases} \end{array} \right\} [P]$$

$$\left. \begin{array}{l} \max_y y_s - y_t \\ \forall (u, v) \in A \quad y_v - y_u \leq c_{uv} \end{array} \right\} [D]$$



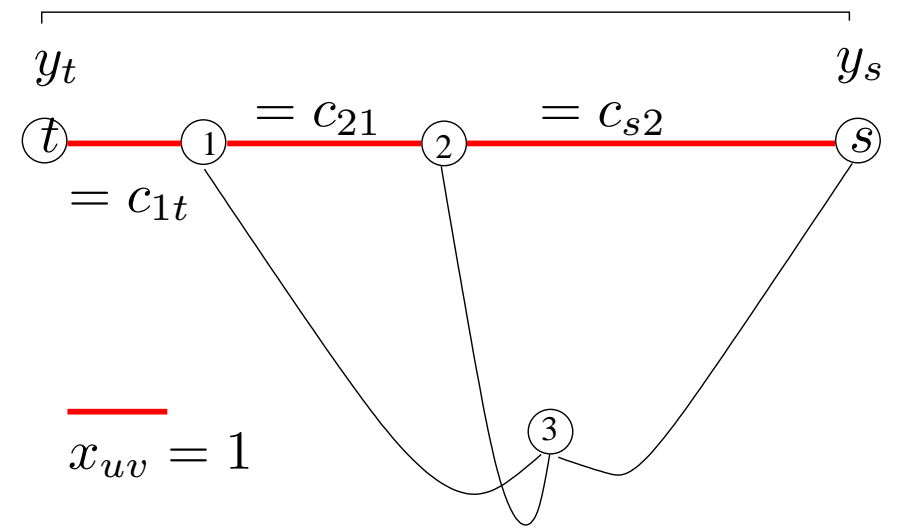
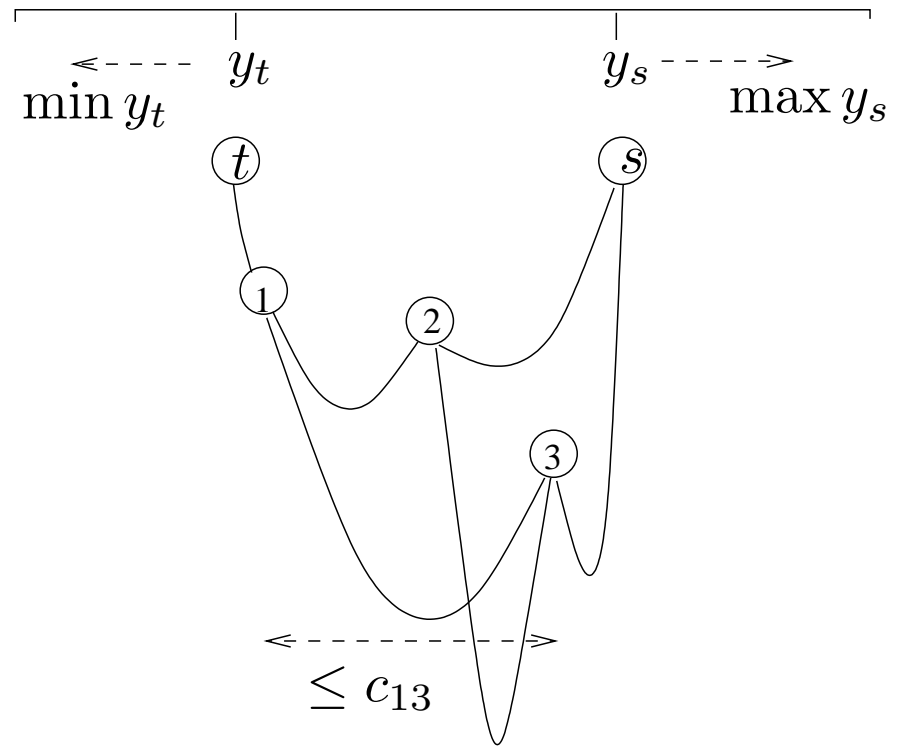
# Shortest Path Dual



cols	(1,2)	(1,3)	...	(4,1)	...		
rows	$c_{12}$	$c_{13}$	...	$c_{41}$	...	$b$	
1	1	1	...	-1	...	0	$y_1$
2	-1	0	...	0	...	0	$y_2$
3	0	-1	...	0	...	0	$y_3$
4	0	0	...	1	...	0	$y_4$
⋮	⋮	⋮		⋮		⋮	⋮
s	0	0	...	0	...	1	$y_s$
⋮	⋮	⋮		⋮		⋮	⋮
t	0	0	...	0	...	-1	$y_t$
⋮	⋮	⋮		⋮		⋮	⋮
	$x_{12}$	$x_{13}$	...	$x_{41}$	...		



# SP Mechanical Algorithm



$$\text{KKT2 on [D]} \Rightarrow \forall (u, v) \in A (x_{uv}(y_v - y_u - c_{uv}) = 0) \Rightarrow$$

$$\forall (u, v) \in A (x_{uv} = 1 \rightarrow y_v - y_u = c_{uv})$$

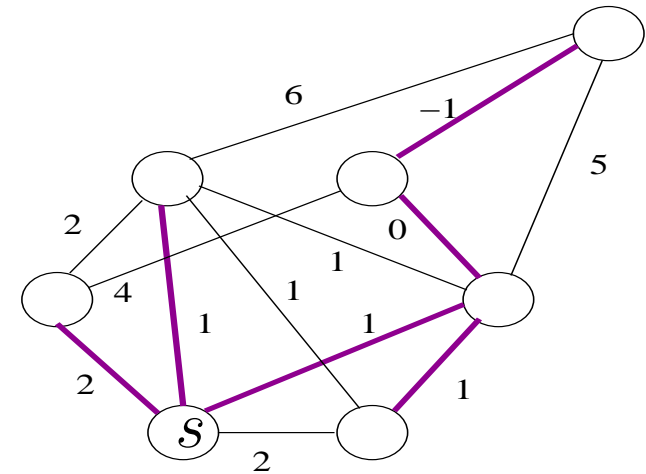


# Single-source SP

SINGLE-SOURCE SP.

*Input:* weighted digraph  $G = (V, A, c)$ , and  $s \in V$ .

*Output:* a shortest path tree in  $G$  rooted in  $s$ .



$$\left. \begin{array}{l} \min_{x \geq 0} \sum_{(u,v) \in A} c_{uv} x_{uv} \\ \forall v \in V \quad \sum_{(u,v) \in A} x_{uv} - \sum_{(v,u) \in A} x_{vu} = \begin{cases} 1 - n & v = s \\ 1 & \text{othw.} \end{cases} \end{array} \right\} [P]$$

$$\left. \begin{array}{l} \max_y (1 - n)y_s + \sum_{u \neq s} y_u \\ \forall (u, v) \in A \quad y_v - y_u \leq c_{uv} \end{array} \right\} [D]$$

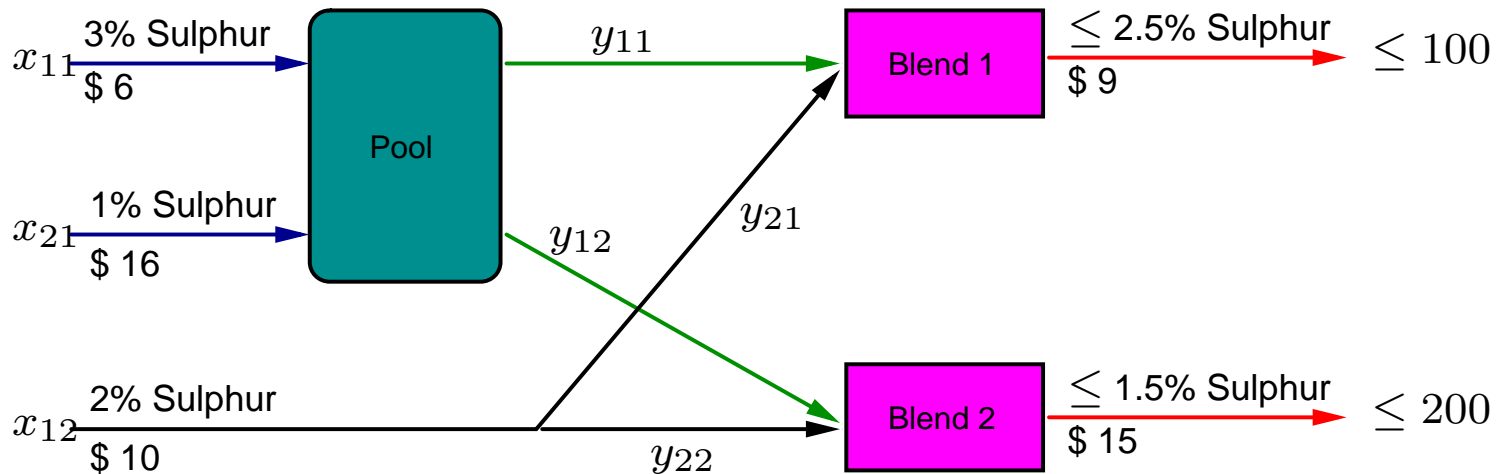


# Haverly's Recursion Algorithm

- Heuristic for solving bilinear programming problems
- Formulation includes bilinear terms  $x_i y_j$  where  $i \in I, j \in J$
- Problem is nonconvex  $\Rightarrow$  many local optima
- Fact: fix  $x_i, i \in I$ , get LP<sub>1</sub>; fix  $y_j, j \in J$ , get LP<sub>2</sub>
- Algorithm: solve LP<sub>1</sub>, get values for  $y$ , update and solve LP<sub>2</sub>, get values for  $x$ , update and solve LP<sub>1</sub>, and so on
- Iterate until no more improvement
- **Warning:** no convergence may be attained, and no guarantee to obtain global optimum



# Haverly's pooling problem



$$\begin{aligned}
 & \min_{x,y,p} \quad 6x_{11} + 16x_{21} + 10x_{12} - \\
 & \quad \quad \quad -9(y_{11} + y_{21}) - 15(y_{12} + y_{22}) \\
 & \text{s.t.} \quad x_{11} + x_{21} - y_{11} - y_{12} = 0 \text{ linear} \\
 & \quad \quad x_{12} - y_{21} - y_{22} = 0 \text{ linear} \\
 & \quad \quad y_{11} + y_{21} \leq 100 \text{ linear} \\
 & \quad \quad y_{12} + y_{22} \leq 200 \text{ linear} \\
 & \quad \quad 3x_{11} + x_{21} - p(y_{11} + y_{12}) = 0 \\
 & \quad \quad py_{11} + 2y_{21} \leq 2.5(y_{11} + y_{21}) \\
 & \quad \quad py_{12} + 2y_{22} \leq 1.5(y_{12} + y_{22})
 \end{aligned}$$



# HRA applied to HPP

Problem LP<sub>1</sub>: fixing  $p$

$$\left\{ \begin{array}{l} \min_{x,y} \quad 6x_{11} + 16x_{21} + 10x_{12} - \\ \quad \quad \quad -9y_{11} - 9y_{21} - 15y_{12} - 15y_{22} \\ \text{s.t.} \quad x_{11} + x_{21} - y_{11} - y_{12} = 0 \\ \quad \quad x_{12} - y_{21} - y_{22} = 0 \\ \quad \quad y_{11} + y_{21} \leq 100 \\ \quad \quad y_{12} + y_{22} \leq 200 \\ \quad \quad 3x_{11} + x_{21} - \mathbf{p}y_{11} - \mathbf{p}y_{12} = 0 \\ \quad \quad (\mathbf{p} - 2.5)y_{11} - 0.5y_{21} \leq 0 \\ \quad \quad (\mathbf{p} - 1.5)y_{12} + 0.5y_{22} \leq 0 \end{array} \right.$$

Problem LP<sub>2</sub>: fixing  $y_{11}, y_{12}$

$$\left\{ \begin{array}{l} \min_{x,y_{21},y_{22},p} \quad 6x_{11} + 16x_{21} + 10x_{12} - \\ \quad \quad \quad -(9(\mathbf{y}_{11} + \mathbf{y}_{21}) + 15(\mathbf{y}_{12} + \mathbf{y}_{22})) \\ \text{s.t.} \quad x_{11} + x_{21} = \mathbf{y}_{11} + \mathbf{y}_{12} \\ \quad \quad x_{12} - y_{21} - y_{22} = 0 \\ \quad \quad y_{21} \leq 100 - \mathbf{y}_{11} \\ \quad \quad y_{22} \leq 200 - \mathbf{y}_{12} \\ \quad \quad 3x_{11} + x_{21} - (\mathbf{y}_{11} + \mathbf{y}_{12})p = 0 \\ \quad \quad \mathbf{y}_{11}p - 0.5y_{21} \leq 2.5\mathbf{y}_{11} \\ \quad \quad \mathbf{y}_{12}p + 0.5y_{22} \leq 1.5\mathbf{y}_{12} \end{array} \right.$$

HRA Algorithm:

1. Solve LP<sub>1</sub>, find value for  $y_{11}, y_{12}$ , update LP<sub>2</sub>
2. Solve LP<sub>2</sub>, find value for  $p$ , update LP<sub>1</sub>
3. Repeat until solution does not change / iteration limit exceeded



# History of LP I

- 1788: Optimality conditions for equality-constrained programs (Lagrange)
- 1826: Solution of a system of linear equations (Gauss)
- 1873: Solution of a system of linear equations with nonnegative variables (Gordan)
- 1896: Representation of convex polyhedra (Minkowski)
- 1936: Solution of a system of linear inequalities (Motzkin)
- 1939: Optimality conditions (Karush, Kuhn & Tucker)
- 1939-45: Blackett's Circus, UK Naval Op. Res. , US Navy Antisubmarine Warfare Op. Res. Group, USAF Op. Res., Project RAND
- 1945: The diet problem (Stigler)

# History of LP II



- 1947: The simplex method (Dantzig)
- 1953: The revised simplex method (Dantzig)
- 1954: Cutting planes applied to TSP (Dantzig, Fulkerson, Johnson)
- 1954: Max flow / min cut theorem (Ford & Fulkerson), declassified 1999
- 1954: Dual simplex method (Lemke)
- 1954: Branch and Bound applied to TSP (Eastman)
- 1955: Stochastic programming (Dantzig & Beale)
- 1956: Dijkstra's algorithm (Dijkstra)
- 1958: Cutting planes for integer programming (Gomory)
- 1958: Dantzig-Wolfe decomposition (Dantzig & Wolfe)

# History of LP III



- 1962: Benders' decomposition (Benders)
- 1963: *Linear programming and extensions* (Dantzig)
- 1970: Lagrangian relaxation for integer programming (Held & Karp)
- 1970: Ellipsoid method (Khachian)
- 1971: NP-completeness (Cook, Karp)
- 1972: Simplex method is not polynomial (Klee & Minty)
- 1977: Bland's rule for simplex method (Bland)
- 1982: Average running time of simplex method (Borgwardt)
- 1984: Interior point method for LP (Karmarkar)
- 1985: Branch-and-cut on TSP (Padberg & Grötschel)