# Linear and Integer Programming
# (ADM II)

Script

Rolf Möhring
WS 2010/11

Contents 1-1

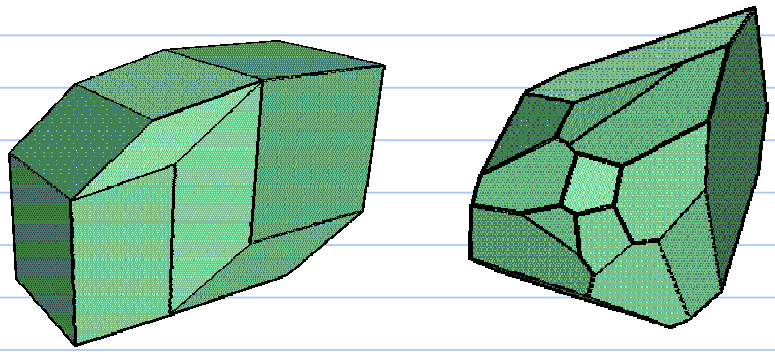
- **On the history of ADM**
  - Young area, has its roots in
    - algebra, graph theory, combinatorics
    - computer science (algorithm design and complexity theory)
    - optimization
  - Deals with optimization problems having a disrete structure
    - graphs, networks
    - finite solution space
  - Applications
    - telecommunication networks, traffic networks
    - logistics, production planning, location planning
    - ...
- ADM at TU Berlin



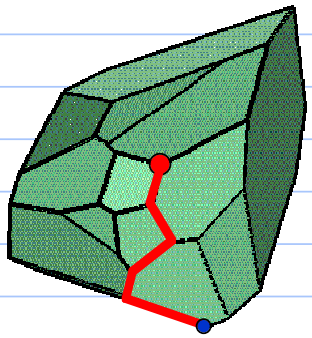
- Basic courses
  - Graph and network algorithms (ADM I)
  - Linear and integer optimization (ADM II)
- Special courses (ADM III)
  - Scheduling problems
  - Applied network optimization
  - Polyhedral theory
  - ...
- Seminar (partly parallel with ADM II or ADM III)
- Bachelor thesis or master thesis

- Linear optimization problems
  - Linear objective function, linear inequalities as side constraints
    - Linear optimization: min $c^T x$ subject to $Ax \leq b, x \geq 0$
  - Simplex algorithm
  - Duality
  - Geometry of linear optimization problems
    - $Ax \leq b, x \geq 0$ define a polyhedron



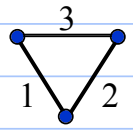


- Optimum is attained in a vertex (corner point)
- The simplex algorithm traverses vertices

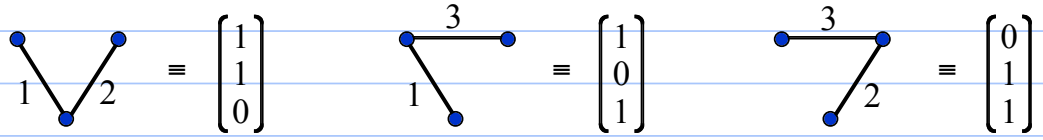


- Discrete problems as linear optimization problems
  - polyhedral theory
  - Discrete problems as geometric problems
    - Minimum spanning trees as vectors
      - Graph $G$

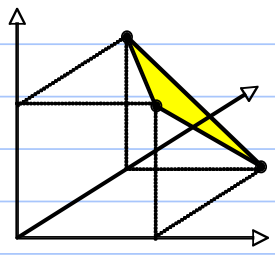





- Minimum spanning trees of $G$ as vectors (incidence vectors)




- Convex hull of incidence vectors = polytope (yellow set)



polytope = yellow set

- Computing a minimum spanning tree = linear optimization over this polytope

- Integer linear optimization
  - variables may only attain integer values
    - much more difficult problems
  - Solution methods
    - Lagrangian relaxation
    - cutting plane algorithms
    - LP-based approximation algorithms
    - ...
- Exercises with implementation assignments



- Torsten Gellert (Exercises)
- Christoph Hansknecht (Tutorials)
- Website
  - @ http://www.math.tu-berlin.de/coga/teaching/wt08/adm2/
  - @ http://www.math.tu-berlin.de/coga/teaching/wt10/adm2/
  - @ Notebook: http://www.math.tu-berlin.de/~moehring/adm2/
- Literature
  - C. H. Papadimitriou and K. Steiglitz

    Combinatorial Optimization: Algorithms and Complexity

    Prentice Hall, Englewood Cliffs, NJ, 1982

    Pocket book - 512 pages - Dover Publications

    First published: Juli 1998

    Auflage: Unabridged

    ISBN: 0486402584
  - B. Korte, J. Vygen:

    Combinatorial Optimization: Theory and Algorithms

Springer, 2000/2002/2006/2008

jetzt auch auf deutsch

- W. J. Cook, W. H. Cunningham, W. R. Pulleyblank and A. Schrijver
  Combinatorial Optimization
  Wiley 1998
- V. Chvátal
  Linear Programming
  Freeman, New York, 1983
- G. L. Nemhauser and L. A. Wolsey
  Integer and Combinatorial Optimization
  John Wiley & Sons, New York, 1988
- M. Grötschel, L. Lovász, and A. Schrijver
  Geometric Algorithms and Combinatorial Optimization
  Springer-Verlag, Berlin, 2nd ed., 1993
- D. S. Hochbaum, ed.
  Approximation Algorithms for NP-hard problems



PWS Publishing Company, Boston, MA, 1997

- H. M. Salkin and K. Mathur
  Foundations of Integer Programming
  North-Holland, Amsterdam, 1989.
- R. J. Vanderbei
  Linear Programming: Foundations and Extensions
  Kluwer Acad. Publ., Dordrecht, 2nd ed., 2001.
  http://www.princeton.edu/~rvdb/LPbook/index.html
- Encyclopedia
  A. Schrijver:
  Combinatorial Optimization: Polyhedra and Efficiency
  Springer, 2003
  3 volumes with 1881 Seiten, aso available as CD

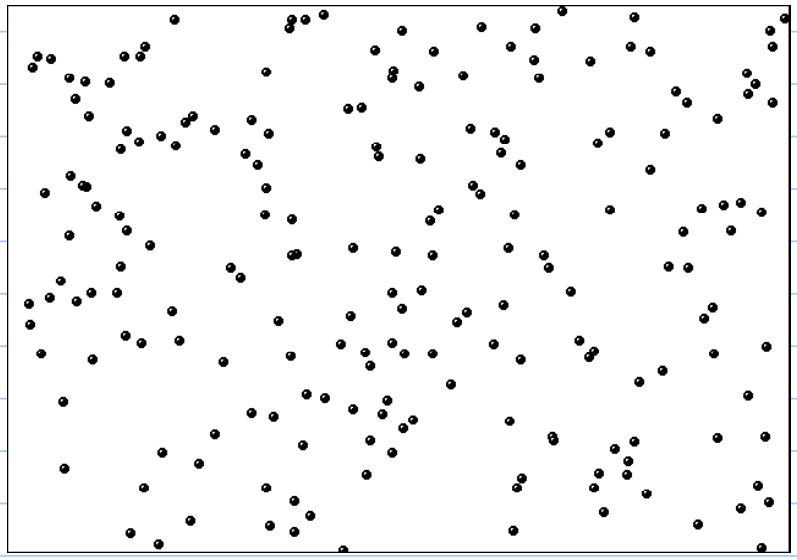
## 2.1 Examples

- An (NP-) optimization problem $P_0$ is defined as follows
  - Each instance $I \in P_0$ has a feasibility domain $S_I$. Its elements $y \in S_I$ are called solutions
    - Feasibility ($y \in S_I$) can in be tested in polynomial time
  - Task:
    - Given an instance $I$ and an objective function $c : S_I \to Q$ (rational numbers), find an optimal solution $y = OPT(I)$

      i.e., $y \in S_I$ with $OPT(I) = c(y) \leq c(x)$ for all $x \in S_I$
      - $OPT(I)$ denotes both, the optimal solution and the objective function value of the optimal solution
    - such a solution is called a global optimum (global minimum) or optimum (minimum)
  - An algorithm that does this is called exact

- 2.1 Example: Traveling Salesman problem (TSP)
  - Instance
    - Complete Graph $K_n$, $n \geq 3$
    - Rational edge weights $c(e) \geq 0$
  - Task

- Compute a Hamiltonian cycle $C$ with minimal length
  - $c(C) = \sum_{e \in E(C)} c(e)$
- A concrete instance:
  - $G = K_n$ in the plane with Euclidean distances



  - Two feasible solutions



- **Neighborhoods**
  - Neighborhoods are defined as ε-niggarded (w.r.t. some norm) for continuous problems. How for discrete problems?
  - A neighborhood for a problem class $P_0$ is given by a mapping

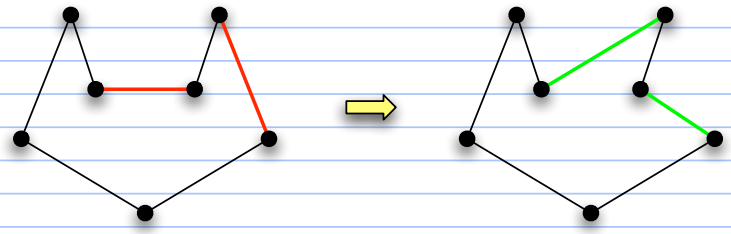
$$N_I : S_I \rightarrow 2^{S_I}$$

for each instance $I \in P_0$

$N_I(y)$ is called the neighborhood of $y \in S_I$. We write $N(y)$ of $I$ is clear from the context

- 2.4 Example: TSP
  - Define a neighborhood by a 2-exchange

$N_2(y) := \{ x \in S_I \mid x$ results from $y$ by exchanging $\leq 2$ edges from y by other edges $\}$



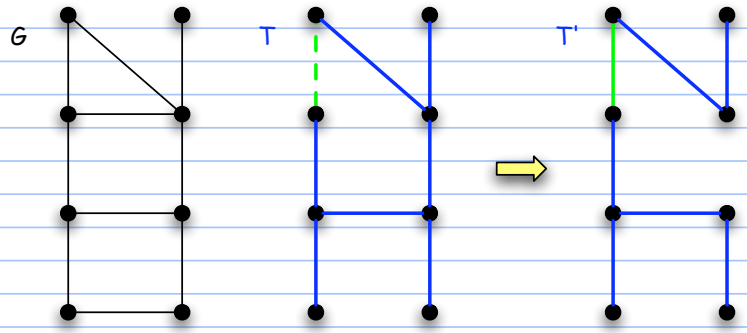
This generalizes to any $k \geq 2$ and yields the neighborhood $N_k(y)$

2.5 Example: MST

Define a neighborhood by exchanging an edge on a fundamental cycle

$N(y) := \{ x \in S_I \mid x$ results from $y$ by adding an edge to $y$ and deleting another edge on the resulting cycle$\}$

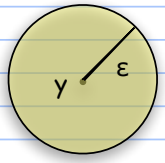





2.6 Example: LP

Define a neighborhood as ε-neighborhood

$N_\varepsilon(y) := \{ x \mid Ax = b, x \geq 0, \| y-x \| \leq \varepsilon \}$

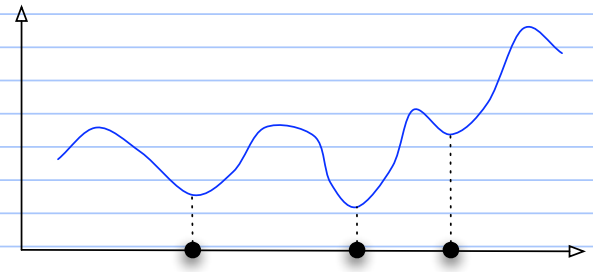


**Local and global optima**

Consider a problem class $P_0$ with neighborhood $N$ and let $I \in P_0$

$y \in S_I$ is called locally optimal w.r.t. $N$ if $c(y) \leq c(x)$ for all $x \in N_I(y)$

2.7 Example: local minima in calculus



2.8 Example: TSP

Locally optimal solutions w.r.t. $N_k$ are called k-optimal or k-opt for short

exact neighborhood

A neighborhood $N$ for a problem class $P_0$ is called exact



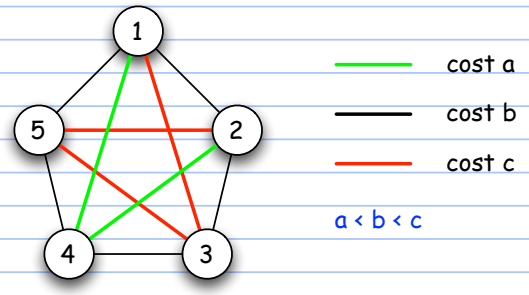
:<=> every local optimum w.r.t. $N$ is a global optimum

More precisely, for all $I \in P_0$, every locally optimal $y \in S_I$ w.r.t. $N_I$ is globally optimal

2.9 Example: TSP

$N_2$ is not exact

Counter example :



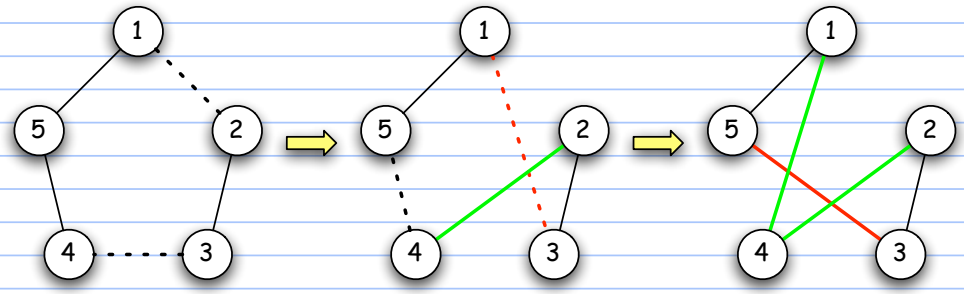

tour y = "outer edges" has cost 5b

Since both green edges are adjacent, every 2-exchange can at most add one green edge, thus at best one green and one red

The new tour is worse if $a+c > 2b$

=> y is locally optimal w.r.t. $N_2$

Two successive 2-exchanges add both green edges to the tour




This tour is better than y if $2a+c < 3b$

$a < b < c$ and $a+c > 2b$ and $2a+c < 3b$ are fulfilled by $a = 1, b = 4, c = 8$

$N_n$ is exact

clear since $N_n(y) = S_I$ ❑

2.10 Example: MST

The neighborhood of MST is exact



Proof :

Use a theorem from ADM I:

T is optimal <=> every non-tree-edge e is the most expensive edge in the cycle induced by e in T + e

❑

Neighborhoods motivate the principle of local search

Algorithm local search

Input

instance I of an optimization problem $P_0$ with neighborhood $N_I$

start solution $y \in S_I$

Output

local optimum w.r.t. $N_I$

Method

iterative improvement

while there is a better solution $x \in N_I(y)$ do

choose better solution $x \in N_I(y)$

y := x

return y

2.11 Theorem (local search for MST)

Local search w.r.t. the MST-neighborhood is a polynomial algorithm for computing a (globally) optimal MST if

(a) it always chooses a non-tree-edge $f$ that is cheaper than the most expensive edge of the cycle $K$ induced by $f$

(b) it always deletes a most expensive tree-edge $e$ from the cycle $K$ induced by $f$

Proof:

1. Since the neighborhood is exact, the algorithm has computed a globally optimal solution at termination

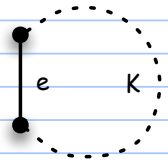
2. The algorithm terminates in polynomial time

Claim 1: A deleted edge never returns into the tree

Proof by contradiction

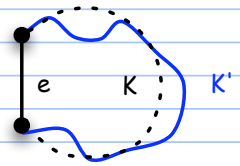
Let $K$ be the cycle when edge $e$ is removed






Consider the first later point in time t at which $e$ is chosen to enter the tree

=> $e$ is currently a non-tree-edge and induces a cycle $K'$




=> $K'$ results from $K$ by the local search steps until time t

In every of these steps, $e$ is a non-tree-edge and induces a cycle $K(e)$ in the current tree

Claim 2: In every of these steps, $c(e) \geq c(g)$ for all edges $g \in K(e)$

Proof by induction along the sequence of cycles $K = K_1, K_2, K_3, \ldots$

Base case: clear for $K = K_1$ by construction

Inductive step from $K_i$ to $K_{i+1}$

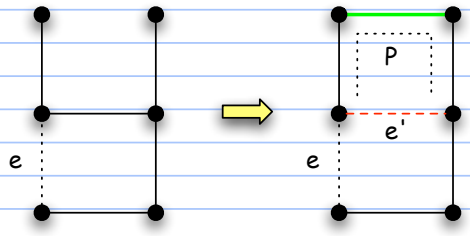
clear for $K_i = K_{i+1}$

let $K_i \neq K_{i+1}$

two cases are possible

$K_{i+1}$ makes $K_i$ larger

=> $K_{i+1} = K_i$ - (some edges including the currently deleted edge e') + P (P is part of the current cycle from which we delete e')



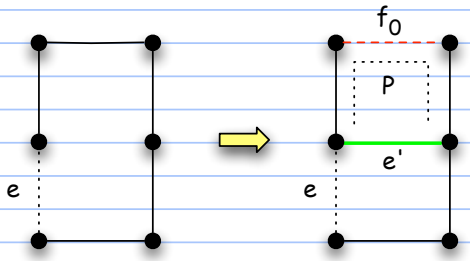

=> $c(e') \geq c(f)$ for all edges $f \in P$ because of (b)

$c(e) \geq c(e')$ by the inductive assumption, (b) => $c(e) \geq c(f)$ for all edges $f \in K_{i+1}$

$K_{i+1}$ makes $K_i$ smaller

=> $K_{i+1} = K + $ (some edges including the currently added edge e') - P (P is part of $K_i$)






=> $c(e') < c(f_0)$ for the removed edge $f_0$, $c(e) \geq c(f)$ for all edges $f \in P$ because of the inductive assumption

=> $c(e) \geq c(f)$ for all edges $f \in K_{i+1}$

Claim 2 contradicts the choice of e specified in (a)

So there are at most m-n+1 exchanges (m = # edges, n = # vertices). Every exchange step can be done in $O(n)$ time

determine the most expensive edge e in the cycle induced by the non-tree-edge f and compare c(e) and c(f)

[by breadth first search in $O$(# edges in tree) = $O(n)$]

exchange e and f if $c(e) < c(f)$

[$O(n)$ if the tree is maintained as array of adjacency lists]

Thus $O((m-n+1)n) = O(mn)$ altogether ❑

Remark: we have obtained better algorithms in ADM I: Kruskal $O(m \cdot \log n)$ and Prim $O(n^2)$

Exercises: Analyze local search for TSP with the k-opt neighborhood. Does it run in polynomial time?



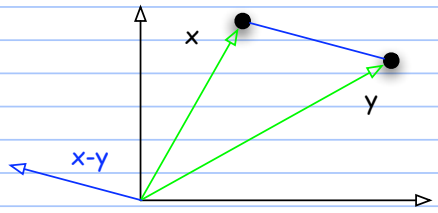
Convex combination of two vectors

Let $x, y \in R^n$. Then every point

$z = \lambda \cdot x + (1-\lambda) \cdot y$ with $0 \leq \lambda \leq 1$

is called a convex combination of $x$ and $y$ (a strictly convex combination if $0 < \lambda < 1$)

the convex combinations of $x$ and $y$ are exactly the points on the line segment from $x$ to $y$



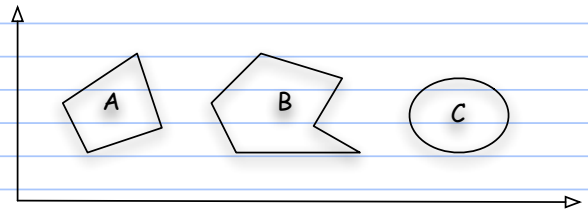

points on this line segment are vectors of the form $y + \lambda(x-y)$

Convex set

$S \subseteq R^n$ is called convex if $S$ contains all convex combinations of any two points $x, y \in S$

2.12 Example:

- $R^n$, $\emptyset$, $\{x\}$, $x \in R^n$ are convex
- The convex subsets of $R^1$ are precisely the intervals
- Convex sets in $R^2$ are those without "bays"




2.13 Lemma

The intersection of (any number of) convex sets is convex

Proof:

Let $S = \cap_{i \in I} S_i$, $S_i$ convex

Let $x, y \in S$, $0 \le \lambda \le 1$, $z = \lambda \cdot x + (1-\lambda) \cdot y$

Def. of $S$ => $x,y \in S_i$ for all $i$ => $z \in S_i$ for all $i$ => $z \in S$ ❑

Lemma 2.13 is the basis for the definition of the convex hull of a set



The convex hull conv(S) of a set $S$ is the smallest convex set containing $S$, i.e.,

$$\text{conv}(S) = \bigcap_{S \subseteq M,\ M \text{ konvex}} M$$

This intersection exists, since $R^n$ is one of the sets $M$

An equivalent description is (exercise)

$\text{conv}(S) = \{ \lambda_1 x^1 + \ldots + \lambda_k x^k \mid x^i \in S,\ \lambda_i \ge 0,\ \sum \lambda_i = 1,\ k \text{ finite} \}$

Theorem of Caratheodory: in $R^n$, $k \le n+1$ suffices

Convex function

Let $S \subseteq R^n$ be a convex set. A function $c : S \to R^1$ is called convex in $S$ if

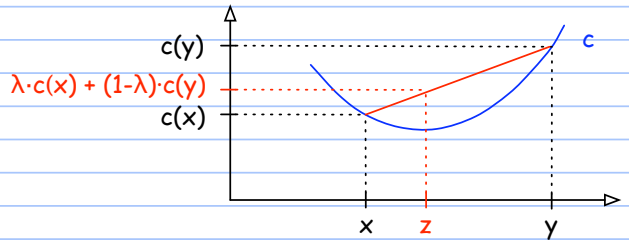
$$c(\lambda \cdot x + (1-\lambda) \cdot y) \le \lambda \cdot c(x) + (1-\lambda) \cdot c(y)$$

for all $x, y \in S$, all $0 \le \lambda \le 1$

2.14 Example

Every linear function is convex

Interpretation of convex functions $c : R^1 \to R^1$




$z := \lambda \cdot x + (1-\lambda) \cdot y \qquad c(z) \leq \lambda \cdot c(x) + (1-\lambda) \cdot c(y)$

**2.15 Lemma**

Let $c$ be convex in $S \subseteq R^n$. Then, for every real number $t$, the level set

$$S_t := \{ x \in S \mid c(x) \leq t \}$$

is convex

Proof:

Consider $z := \lambda \cdot x + (1-\lambda) \cdot y$ with $x, y \in S_t$, $0 \leq \lambda \leq 1$

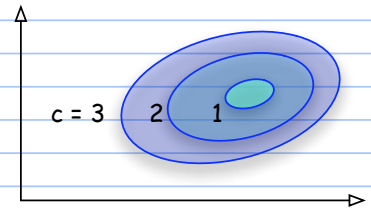
$\Rightarrow$ $c(z) \leq \lambda \cdot c(x) + (1-\lambda) \cdot c(y)$ since c is convex

$\leq \lambda \cdot t + (1-\lambda) \cdot t$ since $x, y \in S_t$

$\Rightarrow$ $z \in S_t$ ❑



Level sets of a convex function $c : R^2 \rightarrow R^1$




Concave function

$c$ in $S \subseteq R^n$ is called concave if $-c$ is convex

$\Leftrightarrow$ $c(\lambda \cdot x + (1-\lambda) \cdot y) \geq \lambda \cdot c(x) + (1-\lambda) \cdot c(y)$ for all $x, y \in S$, all $0 \leq \lambda \leq 1$

Convex optimization = minimizing a convex function on a convex set.

Important principle: local Optima are global Optima

2.16 Theorem (local - global)

Consider an instance $I$ of an optimization problem with $S_I \subseteq \mathbb{R}^n$ convex and $c$ convex in $S_I$.

=> The neighborhood

$$N_\varepsilon(y) := \{ x \in S_I : || y-x || \leq \varepsilon \}$$

defined by the Euclidean distance is exact for every $\varepsilon > 0$.

Proof:

Let $\varepsilon > 0$ and let $y$ be locally optimal w.r.t. $N_\varepsilon$.

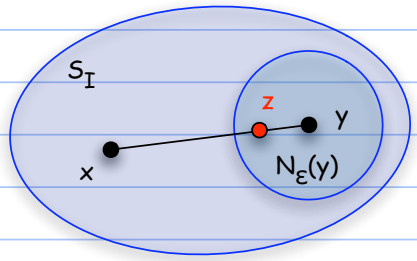
Consider $x \in S_I$. Show that $c(y) \leq c(x)$.

This is trivial if $x \in N_\varepsilon(y)$

So assume $x \notin N_\varepsilon(y)$

=> then there is some $\lambda$ such that $z := \lambda \cdot x + (1-\lambda) \cdot y \in N_\varepsilon(y)$ and $z \neq y$






c convex => $c(z) \leq \lambda \cdot c(x) + (1-\lambda) \cdot c(y)$, moreover $c(z) \geq c(y)$ since $y$ is locally optimal

$$\Rightarrow c(x) \geq \frac{c(z) - (1-\lambda)c(y)}{\lambda} \geq \frac{c(y) - (1-\lambda)c(y)}{\lambda} = c(y)$$

Hence $c(x) \geq c(y)$ ❑

Observe: this holds without any further assumptions on $c$; in particular, $c$ need not be differentiable.

Historical definition of convex optimization problems

An instance $I$ of an optimization problem is called a convex optimization problem if

- $S_I$ is specified as set of all $x \in R^n$ fulfilling side constraints of the form:
  - $g_i(x) \geq 0 \qquad i = 1,\ldots,m$
  - $g_i : R^n \to R^1$ concave, $\qquad i = 1,\ldots,m$
- c is convex in $S_I$

**2.17 Lemma**

The feasible set $S_I$ of an instance I of a historically defined convex optimization problem is convex

Proof:

$g_i$ concave $\Rightarrow$ $-g_i$ convex

$\Rightarrow$ $S_i := \{ x \mid -g_i(x) \leq 0 \} = \{ x \mid g_i(x) \geq 0 \}$ convex because of Lemma 2.15

$\Rightarrow$ $S_I = \cap_i S_i$ is convex because of Lemma 2.13 ❑

**2.18 Theorem**

In a convex optimization problem, every local optimum is a global optimum

**2.19 Remark**



- There can be many global optima
- Every instance of LP is a convex optimization problem

  $\Rightarrow$ every local minimum is a global minimum
- Calculus offers sufficient criteria for smooth functions to be convex:
  - $D \subseteq R^n$ open, $c : D \to R^1$ is twice continuously differentiable,

    Hessian matrix (= matrix of 2nd partial derivatives) of c is positive semidefinite

## 3.1 Forms of linear programs

(3.1) General Form

$\min c^T x \qquad x \in \mathbb{R}^n, c \in \mathbb{R}^n$

$$\begin{array}{lllll} \text{such that} & a_i^T x & = & b_i & i \in M \qquad a_i \in \mathbb{R}^n \\ & a_i^T x & \geq & b_i & i \in \overline{M} \\ & x_j & \geq & 0 & j \in N \\ & x_j & & \text{arbitrary} & j \in \overline{N} \end{array}$$

3.1 Example: Diet problem (historically the oldest LP)

n foods, j = 1,..., n

m nutrients (proteins, vitamins etc.) i = 1,..., m

$a_{ij}$ = amount of nutrient i per unit of food j,

$r_i$ = required amount of nutrient i per time period (week)

$x_j$ = amount of food j per time period (week)

$c_j$ = cost per per unit of food j

$x = (x_1, x_2, ..., x_n)^T$ models a weekly diet

feasible diet fulfills $Ax \geq r$ with $A = (a_{ij})$, $r = (r_1, r_2, ..., r_m)^T$

(3.2) Computing a "cheapest" feasible diet is the LP

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

- An LP of the form (3.2) is called in canonical form
- An LP of the form

(3.3) $\min\ c^T x$ s.t. $Ax = b$, $x \geq 0$

is called in standard form

- An LP of the form (3.1) is called in general form

3.2 Lemma (Equivalence of the three forms)

All 3 forms are equivalent

in the sense that an instance I of one form can be transformed into an instance I' of any other form by a simple transformation such that one can easily construct an optimal solution of I from an optimal solution of I'



Proof

it suffices to provide the following transformations:

general form -> canonical form

eliminate equality constraints and unrestricted variables

- $\sum_{j=1}^{n} a_{ij}x_j = b_i \quad \rightarrow \quad \sum_{j=1}^{n} a_{ij}x_j \geq b_i \text{ and } \sum_{j=1}^{n} a_{ij}x_j \leq b_i$
- $x_j$ unrestricted $\quad \rightarrow \quad x_j = x_j^+ - x_j^-, \quad x_j^+, x_j^- \geq 0$

general form -> standard form

eliminate "≥" by introducing surplus variables $s_i \geq 0$

$\sum_{j=1}^{n} a_{ij}x_j \geq b_i \quad \rightarrow \quad \sum_{j=1}^{n} a_{ij}x_j - s_i = b_i$

eliminate "≤" by introducing slack variables $s_i \geq 0$

$\sum_{j=1}^{n} a_{ij}x_j \leq b_i \quad \rightarrow \quad \sum_{j=1}^{n} a_{ij}x_j + s_i = b_i$ ❑

Goal: Develop an algorithm to solve LPs

Starting point: LP in standard form

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

with initial assumptions 3.1 - 3.3 (which we will get rid of later)

Assumption 3.1: $A$ is an $(m \times n)$-matrix with full row rank $m$

3.3 Example

$$\begin{array}{lrrrrrrrl}
\min & 2x_1 & & & +x_4 & & +5x_7 & & \\
\text{s.t.} & x_1 & +x_2 & +x_3 & +x_4 & & & = 4 & \\
& x_1 & & & & +x_5 & & = 2 & \\
& & & +x_3 & & & +x_6 & = 3 & \\
& & +3x_2 & +x_3 & & & +x_7 & = 6 & \\
& x_j & \geq 0 & \forall j & & & & &
\end{array}$$



$$(A|b) = \left(\begin{array}{cccccc|c}
1 & 1 & 1 & 1 & & & 4 \\
1 & & & & 1 & & 2 \\
& & 1 & & & 1 & 3 \\
& 3 & 1 & & & & 6
\end{array}\right)$$

$A$ has full row rank $m = 4$

**Recall from linear algebra**:

column rank = row rank = rank($A$) = $m$

A basis of $A$ is a set of $m$ linearly independent columns $B = \{A_j, A_k, \ldots, A_r\}$, and these columns are called basic columns. The other columns are called non-basic columns.

We denote the submatrix of these columns by $B$ (sometimes $A_B$) and the corresponding indices by $B(1),\ldots, B(m)$.

So $B = (A_{B(1)}, A_{B(2)}, \ldots, A_{B(m)})$. Sometimes we will identify $B$ with the set of indices, i.e., $B = \{ B(1), \ldots, B(m) \}$.

$A_N$ denotes the submatrix of non-basic columns.

A has at most $\binom{n}{m}$ bases

3.3 Example (continued)

Basis 1:

$$\left(\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 4 \\ 1 & & & 1 & 2 \\ & & 1 & & 3 \\ & 3 & 1 & & 6 \end{array}\right)$$

B(1) = 4, B(2) = 5, B(3) = 6, B(4) = 7

Basis 2

$$\left(\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 4 \\ 1 & & & 1 & 2 \\ & & 1 & & 3 \\ & 3 & 1 & & 6 \end{array}\right)$$

B(1) = 2, B(2) = 5, B(3) = 6, B(4) = 7

Basis 3



$$\left(\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 4 \\ 1 & & & 1 & 2 \\ & & 1 & & 3 \\ & 3 & 1 & & 6 \end{array}\right)$$

B(1) = 2, B(2) = 1, B(3) = 3, B(4) = 7

$$B = \begin{pmatrix} 1 & 1 & 1 & \\ & 1 & & \\ & & & 1 \\ 3 & & 1 & 1 \end{pmatrix}$$

Every basis matrix is invertible and can be transformed into the identity matrix by elementary row operations and column permutations (Gaussian elimination).

If we transform the whole extended matrix (A|b) with these operations, we obtain a solution of Ax = b by setting the basic variables to the (transformed) right hand side, and the non-basic variables to 0. This solution is called the basic solution for basis B.

The applet below can be used to carry out these operations

http://people.hofstra.edu/faculty/Stefan_Waner/RealWorld/tutorialsf1/scriptpivot2.html

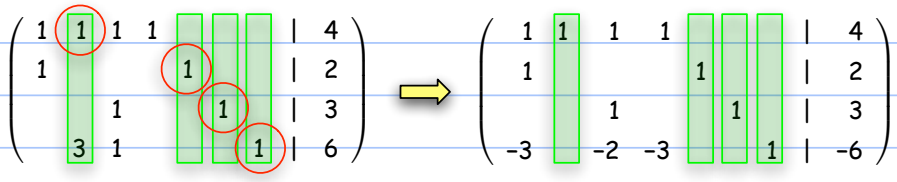
3.3 Example (continued)

Basis 1

no transformation needed since B = identity matrix

basic solution: $x_4 = 4, x_5 = 2, x_6 = 3, x_7 = 6, x_j = 0$ otherwise

Basis 2

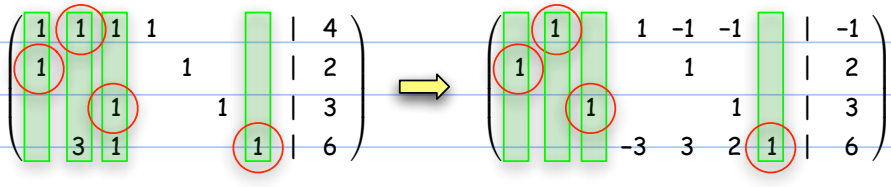


B(1) = 2, B(2) = 5, B(3) = 6, B(4) = 7

basic solution: $x_2 = 4, x_5 = 2, x_6 = 3, x_7 = -6, x_j = 0$ otherwise

Basis 3





B(1) = 2, B(2) = 1, B(3) = 3, B(4) = 7

basic solution: $x_2 = -1, x_1 = 2, x_3 = 3, x_7 = 6, x_j = 0$ otherwise

If we permute the columns of $A$ and $x$ such that $A = (A_B, A_N)$ and $x = (x_B, x_N)^T$, then the elementary transformations correspond to multiplying the linear system $(A_B, A_N)(x_B, x_N)^T = b$ from the left with the inverse $B^{-1}$ of the basis:

$$B^{-1}(A_B, A_N)(x_B, x_N)^T = B^{-1}b$$

$$\Leftrightarrow \quad B^{-1}A_B x_B + B^{-1}A_N x_N = B^{-1}b$$

$$\Leftrightarrow \quad x_B + B^{-1}A_N x_N = B^{-1}b$$

If we set $x_N = 0$ in the basic solution, we obtain $x_B = B^{-1}b$

So if $B$ is a basis of $A$, then we obtain the associated basic solution $x = (x_B, x_N)^T$ as

$x_B = B^{-1}b,\ x_N = 0$

3.3 Example (continued)

Basis 3

$$B = \begin{pmatrix} 1 & 1 & 1 & \\ & 1 & & \\ & & 1 & \\ 3 & & 1 & 1 \end{pmatrix} \Rightarrow B^{-1} = \begin{pmatrix} 1 & -1 & -1 & \\ & 1 & & \\ & & 1 & \\ -3 & 3 & 2 & 1 \end{pmatrix}$$

$$\Rightarrow x_B = \begin{pmatrix} x_2 \\ x_1 \\ x_3 \\ x_7 \end{pmatrix} = B^{-1}b = \begin{pmatrix} 1 & -1 & -1 & \\ & 1 & & \\ & & 1 & \\ -3 & 3 & 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 2 \\ 3 \\ 6 \end{pmatrix} = \begin{pmatrix} -1 \\ 2 \\ 3 \\ 6 \end{pmatrix}$$

A basic solution $x$ is called a basic feasible solution (bfs for short) if $x \geq 0$, i.e., $x$ is a feasible solution of the LP

3.3 Example (continued)

The basic solution of basis 1 is feasible, those of basis 2 and basis 3 are not.



**The role of basic solutions for the simplex algorithm**

From an algebraic view, the simplex algorithm will turn out to be a local search on the set of basic feasible solutions

To see this we need

- a neighborhood (two basic feasible solutions are neighbors if they differ in at most one column)
- an algorithmic analysis how to go from one basic feasible solution to a neighbor (pivot operation, pivot step, or simply pivot)
- we start with a few mathematical properties of (feasible) basic solutions

**Some mathematical properties of (feasible) basic solutions**

3.4 Lemma (The values of basic variables are bounded)

Let the entries of $A$ and $b$ be integer numbers and let $x$ be a basic solution of $Ax = b$.

Let $\alpha := \max_{ij} |a_{ij}|$ and $\beta := \max_i |b_i|$. Then

$$|x_j| \leq m!\, \alpha^{m-1}\, \beta \quad \text{for all } j$$

Proof:

- Let $x_B = B^{-1}b$ with $B = (A_{B(1)}, A_{B(2)}, \ldots, A_{B(m)})$.
  By Cramer's rule we obtain

$$x_{B(i)} = \frac{\det B^i}{\det B} \quad \text{mit } B^i = (A_{B(1)}, \ldots, A_{B(i-1)}, b, A_{B(i+1)}, \ldots, A_{B(m)})$$

- A, b integer => det B integer => $|\det B| \geq 1$
- Expanding $\det B^i$ along column b yields m summands of the form $b_i \cdot$ [(m-1)x(m-1) sub-determinant of A].
  Each such sub-determinant is the sum of (m-1)! products of (m-1) entries from A. Hence:
- $|x_{B(i)}| = \frac{|\det B^i|}{|\det B|} \leq |\det B^i| \leq m \cdot \beta \cdot (m-1)! \cdot \alpha^{m-1} = m!\alpha^{m-1}\beta$ ❑

3.5 Lemma (Every basic feasible solution can be optimal, we cannot do with a subset)

- Let x be a basic feasible solution of $Ax = b$, $x \geq 0$ with basis B.
  => there is a cost vector c such that x is the only optimal solution of

$$\begin{aligned} &\min\ c^T x \\ &\text{s.t. } Ax = b \\ &\qquad x \geq 0 \end{aligned}$$



Proof:

- Set $c_j := \begin{cases} 0 & \text{if } j \in B \\ 1 & \text{if } j \notin B \end{cases}$
- => $c^T x = 0$, as non-basic variables are 0
  => x is optimal, as $c^T y \geq 0$ for every feasible solution y
- let y be another optimal solution
  => $y_j = 0$ for all $j \notin B$
  => $Ay = b$ reduces to $By_B = b$ => $y_B = B^{-1}b = x_B$
  => x is the only optimal solution ❑

- We will see later that basic feasible solutions also suffice, i.e., they constitute the smallest set of feasible solutions on which the optimum is attained for all cost vectors c

**Existence of basic feasible solutions**

- Question: does every LP have a basic feasible solution?

Assumption 3.2: The feasible domain $S_I$ of an LP is non-empty

3.6 Theorem (Existence of basic feasible solutions)

With assumptions

3.1: rank(A) = m

3.2: $S_I \neq \emptyset$

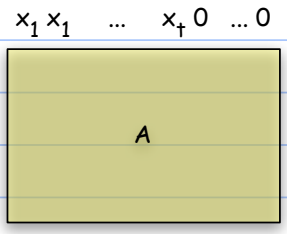
there exists at least one basic feasible solution

Proof:

$S_I \neq \emptyset$ => there are feasible solutions

let x be a solution with the most 0-entries and let w.o.l.g. $x_1, \ldots, x_t > 0$, $x_{t+1}, \ldots, x_n = 0$

=> $Ax = b$ reduces to $A_1x_1 + \ldots + A_tx_t = b$ (3.4)






let $A' := (A_1, \ldots, A_t)$ and $r := \text{rank}(A')$

=> $0 \leq r \leq \min\{t, m\}$

case distinction

r = 0

=> $A_j = 0$ for $j = 1, \ldots, t$ => (3.4) $Ax = 0$

=> (choice of x) $x = 0$

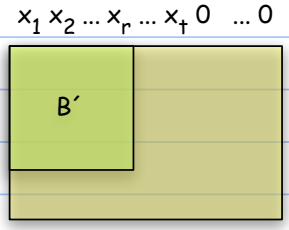
=> x is a basic feasible solution for any basis, a particular basis exists because of assumption 3.1

$0 < r < t$

we generate a contradiction

let B´ be a non-singular submatrix of A' with rank r, w.l.o.g.

$$B' = \begin{pmatrix} a_{11} & \ldots & a_{1r} \\ \vdots & \ddots & \vdots \\ a_{r1} & \ldots & a_{rr} \end{pmatrix}$$



let $B_j'$ the columns of the first $r$ entries of $A_j$ $(j = 1, ..., t)$ and let $b'$ be the vector of the first $r$ entries of $b$

$r < t$ => the last $m - r$ equations of (3.4) are redundant and we can write (3.4) as

$B_1' x_1 + ... + B_t' x_t = b'$

<=> $B_1' x_1 + ... + B_r' x_r = b' - (B_{r+1}' x_{r+1} + ... + B_t' x_t)$

$$\Leftrightarrow \quad (B_1', \dots, B_r') \begin{pmatrix} x_1 \\ \vdots \\ x_r \end{pmatrix} = b' - (B_{r+1}', \dots, B_t') \begin{pmatrix} x_{r+1} \\ \vdots \\ x_t \end{pmatrix}$$

Multiplying from the left with $(B')^{-1}$ yields

$$\begin{pmatrix} x_1 \\ \vdots \\ x_r \end{pmatrix} = (B')^{-1}b' - (B')^{-1} (B_{r+1}', \dots, B_t') \begin{pmatrix} x_{r+1} \\ \vdots \\ x_t \end{pmatrix}$$

$$\Leftrightarrow \begin{pmatrix} x_1 \\ \vdots \\ x_r \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_r \end{pmatrix} + \begin{pmatrix} \alpha_{1,r+1} & \dots & \alpha_{1t} \\ \vdots & \ddots & \vdots \\ \alpha_{r,r+1} & \dots & \alpha_{rt} \end{pmatrix} \begin{pmatrix} x_{r+1} \\ \vdots \\ x_t \end{pmatrix} \quad (*)$$

for some numbers $\alpha_{ij}$ and $\beta_i$,



i.e., $x_1, ..., x_r$ depend affinely linear on $x_{r+1}, ..., x_t$

Use (*) to construct a new feasible solution with more 0-entries

there is a row $i$ with $\alpha_{i,r+1} \neq 0$ (otherwise we may choose $x_{r+1} = 0$ which contradicts the choice of $x$)

vary $x_{r+1}$ such that one of $x_1, ..., x_r$ or $x_{r+1}$ becomes 0, but all stay $\geq 0$

$\alpha_{i,r+1} < 0$ => $x_i$ grows when $x_{r+1}$ gets smaller => $x_1, ..., x_t$ stay $\geq 0$

$\alpha_{i,r+1} > 0$ => decreasing $x_{r+1}$ by $\theta > 0$ yields the condition $x_i - \alpha_{i,r+1} \cdot \theta \geq 0$

$$\text{choose } \theta := \min \left\{ x_{r+1}, \frac{x_i}{\alpha_{i,r+1}} \mid \alpha_{i,r+1} > 0 \right\}$$

and set $y_{r+1} := x_{r+1} - \theta$ and $y_j := x_j$ for $j > r+1$

=> $y_1, ..., y_r$ are determined by (*) and are $\geq 0$

=> $y = (y_1, ..., y_n)^T$ is a feasible solution of the LP

2 cases:

$y_{r+1} = 0$ (occurs if $\theta = x_{r+1}$)

or some $y_1, ..., y_r$ become 0

=> contradiction to the choice of $x$

- $0 < r = t \le m$
  - A' has t columns, rank(A') = t => the t columns of A' are linearly independent
    => (rank(A) = m) they can be augmented to a basis B of A by adding columns of A
    => x is a basic feasible solution for basis B ❑

- **Boundedness of the feasible domain**
  - last assumption
  - Assumption 3.3: $\{ c^T x \mid Ax = b, x \ge 0 \}$ is bounded from below
    - this will show that we can restrict ourselves to bounded feasibility domains $S_I = \{ x \mid Ax = b, x \ge 0 \}$
  - 3.7 Theorem (The feasible domain can be assumed as being bounded)
    - Assume
      3.1: rank(A) = m
      3.2: $S_I \neq \emptyset$
      3.3: $\{ c^T x \mid Ax = b, x \ge 0 \}$ is bounded from below
    - Then the LP
      $\min \{ c^T x \mid Ax = b, x \ge 0 \}$ (LP)



is equivalent to the LP

$\min \{ c^T x \mid Ax = b, x \ge 0, x_j \le M \}$ (LP*)

with $M := (m+1)! \, \alpha^m \beta$

$\alpha := \max_{ij} \{ |a_{ij}|, |c_j| \}$

$\beta := \max_i \{ |b_i|, |z| \}$

$z := \inf \{ c^T x \mid Ax = b, x \ge 0 \}$

in the sense that the optimal values coincide and (LP) and (LP*) have a common optimal solution that is a basic solution of (LP).

- Proof:
  - let $G := \{ c^T x \mid Ax = b, x \ge 0 \} \subseteq R^1$ => $z := \inf G > -\infty$ because of assumption (3.3)
    G is closed (since defined by = and ≥ and the linear function $c^T x$) => $z \in G$
    => $\{ x \in R^n \mid c^T x = z, Ax = b, x \ge 0 \}$ is the set of optimal solutions of (LP) (3.5)
  - Two cases
    - $\text{rank} \{ c^T x = z, Ax = b \} = m+1$
      - Theorem 3.6 => (3.5) has a basic feasible solution x with basis B
        Lemma 3.4 => x fulfills the bounds $x_j \le M$

Moreover: $x$ is feasible for (LP)

(3.5) => $x$ is optimal for (LP) and this also for (LP*), as $S_{LP^*} \subseteq S_{LP}$

rank = m+1 => $B$ without the row for $c^T x = z$ contains $m$ linearly independent columns from $A$

=> $x$ is a basic solution of (LP)

rank $\{ c^T x = z, Ax = b \} = m$

=> $c$ is a linear combination of the rows $a_i$ of $A$, say $c = \sum d_i a_i$

=> $c^T x = (\sum d_i a_i)^T x = \sum d_i a_i^T x = \sum d_i b_i$ = constant, independent of $x$

=> every such solution of (LP) is optimal, in particular a feasible basic solution.

This fulfills the bounds because of Lemma 3.4 ❑



**Main statements of this chapter**

- The feasible domain of an LP in canonical form is a polyhedron
- The basic feasible solutions of the associated LP in standard form correspond to vertices of this polyhedron
- The optimum is attained in a vertex of the polyhedron / in a basic feasible solution of the associated LP in standard form

**Basic geometric facts**

- $\emptyset \neq S \subseteq R^d$ is a linear subspace of $R^d$
  - <=> $S$ is closed under vector addition and scalar multiplication
  - <=> $S$ is the set of solutions of a homogeneous linear system $Ax = 0$

    Then $\dim(S) + \text{rank}(A) = d$
- $T \subseteq R^d$ is an affine subspace of $R^d$
  - <=> $T$ is a linear subspace translated by some vector, i.e.,

    $T = \{ u + x \mid x \in S \}$, $S$ linear subspace, $u \in R^d$
  - <=> $T$ is set of solutions of an inhomogeneous linear system $Ax = b$
  - dimension $\dim(T) := \dim(S)$

- Dimension of a set $S \subseteq R^d$
  - = dimension of the smallest affine subspace containing $S$ (affine hull of $S$)
  - Examples:
    - a line has dimension 1
    - $|S| = k \leq d+1 \Rightarrow \dim(S) \leq k-1$
    - The set of solutions $S = \{ x \mid Ax = b, x \geq 0 \}$ of an LP in standard form fulfills $\dim(S) \leq n-m$
      - because:

        $\operatorname{rank}(A) = m \Rightarrow \{ x \mid Ax = b \}$ has dimension $n-m$ (as long as $Ax = b$ is solvable)

        the sign restrictions $x_j \geq 0$ can lower the dimension

        e.g., if $\{ x \mid Ax = b \} \cap \{ x_j \geq 0 \} = \{0\}$
- Hyperplane in $R^d$
  - = affine subspace of dimension $d-1$
  - = set of solutions of an equation $a_1x_1 + \ldots + a_dx_d = b$ (not all $a_j = 0$)
  - it defines two (closed) halfspaces

    $\{ x \mid a_1x_1 + \ldots + a_dx_d \geq b \}$ and $\{ x \mid a_1x_1 + \ldots + a_dx_d \leq b \}$
- Polyhedron in $R^d$

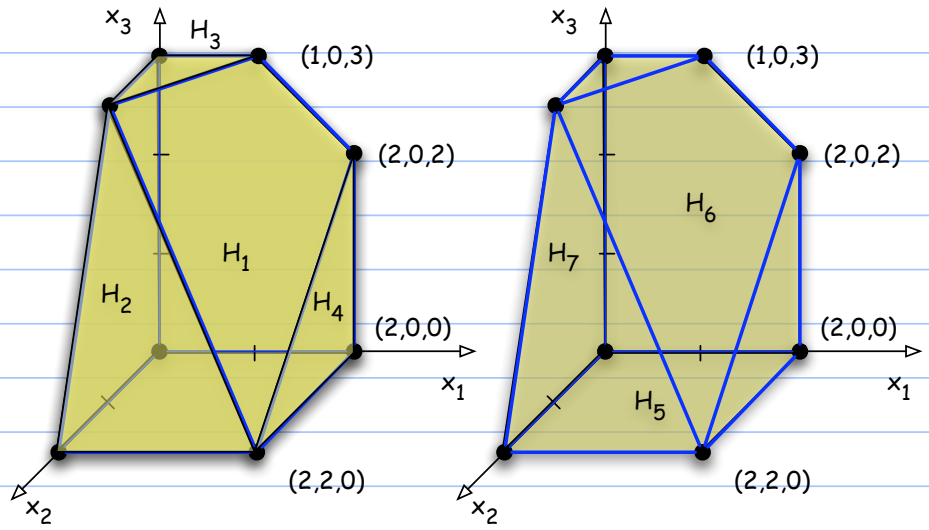


  - = non-empty intersection of finitely many halfspaces (generated by hyperplanes)
  - $\Rightarrow$ polyhedra are convex
- Polytope in $R^d$
  - = bounded polyhedron
  - Example: Platonic Solids
    - http://www.3quarks.com/GIF-Animations/PlatonicSolids/index-de.html

**Geometric aspects of polytopes**

Feasible domains $S$ of LPs in canonical form are polyhedra. They are polytopes if $S$ is bounded or can be assumed to be bounded.

- 3.8 Example




The polytope is the intersection of the following halfspaces

$x_1 + x_2 + x_3 \le 4$ hyperplane $H_1$

$x_1 \le 2$ hyperplane $H_4$

$x_3 \le 3$ hyperplane $H_3$

$3x_2 + x_3 \le 6$ hyperplane $H_2$



$x_j \ge 0$ hyperplanes $H_5$, $H_6$, $H_7$

A hyperplane $H$ supports polyhedron $P$

:<=> $H \cap P \neq \emptyset$ and $P$ is contained in one of the halfspaces defined by $H$

$f := H \cap P$ is then called a face of $P$, and $H$ is called a $P$ supporting hyperplane defining $f$

important: facet := face of dimension d-1

vertex or extreme point := face of dimension 0 (a point)

edge := face of dimension 1 (a line segment)

Some geometric facts (without proof)

(a) A facet defining hyperplane of $P$ belongs to one of the halfspaces that define $P$ (i.e., deleting $H$ changes $P$)

(b) Face defining hyperplanes do generally not fulfill (a).

The halfspace $x_2 \le 2$ in Example 3.8 defines a face ($\{ x_2 \le 2 \} \cap P$) that is an edge but not a facet).

This halfspace is redundant, its generating inequality is already implied by others:

$x_3 \ge 0$, $3x_2 + x_3 \le 6$ => $3x_2 \le 3x_2 + x_3 \le 6$ => $x_2 \le 2$

(c) An edge is a line segment that connects two vertices and lies on the boundary of the polyhedron (the converse is not true)

3.9 Theorem (Minkowski 1896)

(a) Every polytope is the convex hull of its vertices,

i.e., every point $x$ of a polytope $P$ can be represented as

$x = \lambda_1 x^1 + \ldots + \lambda_k x^k$, $x^i$ vertex of $P$, $\lambda_i \geq 0$, $\sum \lambda_i = 1$, $k$ finite

(b) if $V \subseteq R^d$ is finite, then conv(V) is a polytope $P$ and $\{$ vertices of $P \} \subseteq V$

Proof

partly an exercise

(a) by induction on dimension d

In example 3.8 we have

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 2 \\ 2 \\ 0 \end{pmatrix} + \frac{1}{3}\begin{pmatrix} 0 \\ 0 \\ 3 \end{pmatrix} + \frac{1}{6}\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

(b) is intuitively clear geometrically, but more complicated to prove algebraically. ❑



Consequence: There are two views on a polytope:

1. as convex hull of a finite set of points

(e.g., of the incidence vectors of solutions of a combinatorial optimization problem)

2. as the intersection of finitely many halfspaces (if this intersection is bounded)

(this is the natural view when the inequalities are explicitly given, e.g., as for LPs in canonical form)

A third, algebraic, will be derived in the sequel. It consists of a linear system $Ax = b$, $x \geq 0$ whose basic feasible solutions correspond to the vertices of $P$.

**Algebraic interpretation of polytopes**

From the polytope to the linear system

Let $P$ be a polytope in $R_+^{n-m}$ given by the $n$ inequalities (3.6)

$$x_i \geq 0 \qquad i = 1, \ldots, n-m$$

$$h_{i,1}x_1 + \ldots + h_{i,n-m}x_{n-m} + g_i \leq 0 \qquad i = n-m+1, \ldots, n$$

Let $H$ be the coefficient matrix of inequalities $i = n-m+1, \ldots, n$

Introduce slack variables $x_i$ for inequalities $i = n-m+1, \ldots, n$ (m many)

=> (3.6) becomes $Ax = b,\ x \geq 0,\ x \in R^n$ with $A = (H|I)$, $b = -(g_{n-m+1}, \ldots, g_n)^T$

i.e., the feasible domain $S$ of an LP in standard form

This induces the following mapping (transformation) $f : P \to S$ (3.7)

$$x' = \begin{pmatrix} x'_1 \\ \vdots \\ x'_{n-m} \end{pmatrix} \in P \quad \mapsto \quad \begin{pmatrix} x_1 \\ \vdots \\ x_{n-m} \\ \vdots \\ x_n \end{pmatrix} \in S$$

$$\text{with} \quad x_i = \begin{cases} x'_i & i = 1, \ldots, n-m \\ \underbrace{-g_i - \sum_{j=1}^{n-m} h_{ij} x'_j}_{\text{slack}} & i = n-m+1, \ldots, n \end{cases}$$

Observe: $x$ is uniquely determined by $x'$, i.e., $f$ is injective

From the linear system to the polytope

Let $Ax = b,\ x \geq 0,\ x \in R^n$ be the feasible domain $S$ of an LP in standard form (under assumptions 3.1 - 3.3). Then there exists a basis $B$, and we consider the partition $A = (A_B, A_N)$ and $x = (x_B, x_N)^T$ in basic and non-basic variables.



W.o.l.g. let $B = \{ n-m+1, \ldots, n \}$ (last m columns of $A$)

Multiplying $Ax = b$ from the left by $B^{-1}$ gives

$B^{-1}(A_B, A_N)(x_B, x_N)^T = x_B + B^{-1}A_N x_N = B^{-1}b =: b'$

With $B^{-1}A_j =: A'_j$ we obtain

$$x_{B(i)} = b'_i - \sum_{j=1}^{n-m} a'_{ij} x_j \quad i = 1, \ldots, m$$

Hence $Ax = b,\ x \geq 0$ is equivalent to (3.8)

$$\begin{aligned} b'_i - \textstyle\sum_{j=1}^{n-m} a'_{ij} x_j &\geq 0 \quad i = 1, \ldots, m \\ x_j &\geq 0 \quad j \notin B \end{aligned}$$

These are $n$ inequalities in the variables $x_1, \ldots, x_{n-m}$, which define a polytope $P$ in $R^{n-m}$ (since $S$ is bounded)

This construction defines the following mapping $g : S \to P$ ( for $B = \{ n-m+1, \ldots, n \}$ ) (3.9)

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_{n-m} \\ \vdots \\ x_n \end{pmatrix} \in S \quad \mapsto \quad x' = \begin{pmatrix} x'_1 \\ \vdots \\ x'_{n-m} \end{pmatrix} \in P$$

with $x_j' := x_j \quad j = 1, \ldots, n-m$ (we "forget" the slack of basic variables).

Obviously, $g$ is a linear function and injective, as $x_{n-m+1}, \ldots, x_n$ are uniquely determined by $x_1, \ldots, x_{n-m}$.

Moreover:

$x_j = 0$ for $j = 1, \ldots, n-m$, i.e. $j \notin B$, implies that $x'$ lies on a "coordinate hyperplane" in $P$

$x_j = 0$ for $j = n-m+1, \ldots, n$, i.e. $j \in B$, implies that a $P$-defining inequality holds with equality

(basic variables "correspond" to slack variables, where this role is defined somewhat arbitrarily by the choice of $B$)

3.8 Example (continued)

$$A = \left(\begin{array}{ccc:ccc} 1 & 1 & 1 & 1 & & \\ 1 & & & & 1 & \\ & 1 & & & & 1 \\ 3 & 1 & & & & 1 \end{array}\right) \qquad b = \begin{pmatrix} 4 \\ 2 \\ 3 \\ 6 \end{pmatrix}$$



$$x' = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \in P \quad \leftrightarrow \quad x = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 2 \\ 1 \\ 1 \\ 3 \end{pmatrix} \in S$$

values of the given variables

values of the slack variables

Observe: $f$ and $g$ are bijections between $P$ and $S$, where $g$ depends on the choice of the basis and determines $P$.

3.10 Theorem (Interpretation of vertices in the three views)

Let $P$ be a polytope and let $S = \{ x \mid Ax = b, x \geq 0 \}$ be the associated feasible set of an LP in standard form.

Let $y' = (y_1', \ldots, y_{n-m}')^T \in P$ and let $y = (y_1, \ldots, y_n)^T \in S$ be the associated vector according to (3.7)

Then the following statements are equivalent:

(1) $y'$ is a vertex of $P$

(2) $y'$ cannot be represented as a strict convex combination of other points from $P$

(3) $y$ is a basic feasible solution of $S$

Proof

(1) => (2)

Assume there are $z', z'' \in P$, $0 < \lambda < 1$ with $y' = \lambda z' + (1-\lambda)z''$

$y'$ is a vertex => there is a halfspace $HS = \{ z \mid h^T z \leq g \}$ with $HS \cap P = \{y'\}$

$z', z'' \notin HS$ => $h^T z' > g$ and $h^T z'' > g$ => $h^T(\lambda z' + (1-\lambda)z'') > g$

=> $y' \notin HS$, a contradiction

(2) => (3)

Let $y \in S$ be constructed from $y'$ according to (3.7) and let $B' := \{ j \mid y_j > 0 \}$

Claim: the columns $A_j$ with $j \in B'$ are linearly independent

if not, there exist numbers $d_j$ (not all 0) with $\sum_{j \in B'} d_j A_j = 0$ (3.10)

Definition of $B'$ => $\sum_{j \in B'} y_j A_j = b$ (3.11)

(3.11) + $\theta$·(3.10) and (3.11) - $\theta$·(3.10) yield

$\sum_{j \in B'} (y_j + \theta d_j) A_j = b$ and $\sum_{j \in B'} (y_j - \theta d_j) A_j = b$

$y_j > 0$ for $j \in B'$ => $\theta > 0$ can be chosen in such a way that $y_j + \theta d_j \geq 0$ and $y_j - \theta d_j \geq 0$.

To this end we must have:



$\theta \leq y_j / |d_j|$ for negative and positive $d_j$

=> this is possible for all $j \in B'$ simultaneously

define $x^1$ and $x^2$ from $S$ by

$$x_j^1 := \begin{cases} y_j - \theta d_j & j \in B' \\ 0 & \text{otherwise} \end{cases} \qquad x_j^2 := \begin{cases} y_j + \theta d_j & j \in B' \\ 0 & \text{otherwise} \end{cases}$$

=> $x^1, x^2 \in S$, different from $y$, and

$$y = \frac{1}{2}x^1 + \frac{1}{2}x^2$$

$g : S \to P$ defined by (3.9) is linear and $g(y) = y'$

$$\Rightarrow y' = g(y) = g\left(\frac{1}{2}x^1 + \frac{1}{2}x^2\right) = \frac{1}{2}g(x^1) + \frac{1}{2}g(x^2)$$

with $g(x^1), g(x^2) \in P$ => a contradiction to (2)

=> $|B'| \leq m$ => (Assumption 3.1) $B'$ can be augmented to a basis $B$

=> $y$ is a basic feasible solution for $B$

(3) => (1)

let $y$ be a basic feasible solution of $Ax = b, x \geq 0$ with basis $B$

Lemma 3.5 => there is a cost vector $c$ such that $y$ is the only optimal solution of the LP with cost vector $c$.

In other words,

$y$ is the only solution of (3.12)

$$c^T x \le c^T y =: b_0 ,$$

$$Ax = b,\ x \ge 0$$

- Transform (3.12) into standard form with a slack variable $x_{n+1}$ for $c^T x \le b_0$ :

$$\begin{pmatrix} c_1 & \dots & c_n & 1 \\ & & & 0 \\ A_1 & \dots & A_n & \vdots \\ & & & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_{n+1} \end{pmatrix} = \begin{pmatrix} b_0 \\ b \end{pmatrix}$$

=> $B \cup \{n+1\}$ is a basis of (3.12)

- Transform (3.12) according to transformation $g$ defined by (3.9).

This yields a system of linear inequalities whose $m$ last inequalities define the polytope $P$.

The first inequality is transformed into $c_1' x_1' + \dots + c_{n-m}' x_{n-m}' \le b_0'$ (3.13)

- The transformation is a bijection



=> $y \to y'$, and $y'$ is the only point in $P$ fulfilling (3.13) with equality

=> $c_1' x_1' + \dots + c_{n-m}' x_{n-m}' = b_0'$ is a supporting hyperplane $H$ of $P$ and $H \cap P = \{y'\}$

=> $y'$ is vertex of $P$ ❑

- There is a similar characterization for edges of the polytope (later)

- Corollary (Feasible solutions are convex combinations of basic feasible solutions)
  - Under assumptions 3.1 - 3.3, every feasible solution in S is a convex combination of basic feasible solutions.
  - Proof:
    - Because of Theorem 3.7, we may assume that the feasible domain $S$ is bounded

      => the associated polyhedron $P$ is a polytope
    - Let $x \in S$ and let $x'$ be the associated point in $P$
    - => (Minkowski's Theorem) $x'$ is a convex combination of vertices $x'^i$ of $P$

      => $x$ is a convex combination of the basic feasible solutions corresponding to these vertices
    - because:
      - let $x' = \sum \lambda_i x'^i$ => $z := \sum \lambda_i x^i \in S$, as $S$ is convex

- g linear => $g(z) = \sum \lambda_i g(x^i) = \sum \lambda_i x'^i = x'$
- g injective, $g(x) = x' = g(z)$ => $x = z$ ❑

A more precise analysis of the correspondence "basic feasible solution <-> vertex"

- vertices $x'$, $y'$ are different <=> associated basic feasible solutions $x$, $y$ are different
- But: in general not the associated bases!

  i.e., different bases $\not\Rightarrow$ different associated basic solutions

Example 3.8 (continued)

$$A = \begin{pmatrix} 1 & 1 & 1 & \vdots & 1 & & \\ 1 & & & \vdots & & 1 & \\ & & 1 & \vdots & & & 1 \\ & 3 & 1 & \vdots & & & \end{pmatrix} \qquad b = \begin{pmatrix} 4 \\ 2 \\ 3 \\ 6 \end{pmatrix}$$

$B = \{ 1, 2, 3, 6 \}$ $B' = \{ 1, 2, 4, 6 \}$

=> $B^{-1}b = (B')^{-1}b' = (2, 2, 0, 3)^T$. In both cases the associated basic solution is $(2, 2, 0, 0, 0, 3, 0)^T$

Geometric view:

Non-basic variables in $B$ are $x_4, x_5, x_7$ => $x_4 = x_5 = x_7 = 0$



=> $x' \in H_1 \cap H_2 \cap H_4$

Non-basic variables in $B'$ are $x_3, x_5, x_7$ non-basic variables => $x_3 = x_5 = x_7 = 0$

=> $x' \in H_5 \cap H_2 \cap H_4$

i.e., $x'$ lies on more than 3 facets

in both cases, $x'$ is the same vertex $(2,2,0)^T$ and $x = (2, 2, 0, 0, 0, 3, 0)^T$ is the same basic feasible solution

degenerate basic feasible solution, degenerate vertex

= basic feasible solution (corresponding to a vertex) with more than $n-m$ zero entries

3.11 Theorem (Characterizing degenerate basic feasible solutions and degenerate vertices)

- $x$ is a basic feasible solution for several bases => $x$ is degenerate
- $x'$ is a degenerate vertex <=> $x'$ lies in the intersection of more than $n-m$ facets

Proof

Let $B$ and $B'$ be different bases for $x$

=> $x_j = 0$ for the $n-m$ Indices $j \in B$ and for the $n-m$ Indices $j \in B'$

=> $x$ is degenerate

- The statement for vertices is left as an exercise ❑

**The Fundamental Theorem of linear optimization**

We are now prepared to show that the optimum of an LP is attained at a vertex / a basic feasible solution (assuming 3.1 - 3.3).

Note that there can also be optima attained at other points (e.g., if $c$ is constant on a face of the polytope)

3.12 Theorem (Fundamental Theorem of linear optimization)

Every instance of LP attains its optimum at a basic feasible solution.

Every convex combination of basic optimal feasible solutions is optimal.

Proof

First statement: geometrically with associated transformed objective function $c^Tx \to d^Tx'$

Transformation of the objective function into the geometric view

Let $S_I$ be given by $\{ Ax = b, x \geq 0 \}$ and let $B$ be a basis for a basic feasible solution.

Consider the polytope $P$ associated with basis $B$ according to transformation (3.9)

=> $P$ is defined via the non-basic variables $x_N$ and $x_B = B^{-1}b - B^{-1}A_N x_N$

The objective function $c^Tx$ is then transformed as follows:



$$\begin{aligned} c^T x &= c_B^T x_B + c_N^T x_N \\ &= c_B^T (B^{-1}b - B^{-1}A_N x_N) + c_N^T x_N \\ &= \underbrace{c_B^T B^{-1} b}_{\text{constant}} + \underbrace{(c_N^T - c_B^T B^{-1} A_N)}_{d^T} x_N \end{aligned}$$

=> min $c^Tx$ (algebraic) corresponds to min $d^Tx_N$ (geometric)

Proof in the geometric view

$P$ is closed and bounded, $y \to d^Ty$ is continuous on $P$

=> optimum is attained on $P$

let $y^0$ be optimal and let $y^k$, $k = 1, \ldots, r$ be the vertices of $P$

Minkowski's Theorem => $y^0 = \sum \lambda_k y^k$ with $\lambda_k \geq 0$, $\sum \lambda_k = 1$

let $j$ be an index for which $d^Ty^j$ is minimum among the vertices $y^k$, $k = 1, \ldots, r$

=> $d^Ty^0 = d^T \sum \lambda_k y^k = \sum \lambda_k d^Ty^k \geq \sum \lambda_k d^Ty^j = d^Ty^j \sum \lambda_k = d^Ty^j$

=> $y^j$ is optimal => optimum is attained at a vertex

correspondence vertices <-> basic feasible solutions => optimum is attained at a basic feasible solution

The second statement is easy to see ❑

- Corollary
  - Every optimal solution is a convex combination of basic optimal feasible solutions
- Proof:
  - Let $x$ be an optimal solution of $\min c^T x,\ Ax = b,\ x \geq 0$
  - Corollary of Theorem 3.10 => $x$ is a convex combination of basic feasible solutions

    say $x = \lambda_1 x^1 + \ldots + \lambda_k x^k$, with basic feasible solutions $x^i$, $\lambda_i \geq 0$, $\sum \lambda_i = 1$, $k$ finite
  - => $c^T x = \sum \lambda_i c^T x^i$ (*)
  - Assume that $x^r$ is not optimal => $c^T x < c^T x^r$

    (*), $\lambda_i \geq 0$ => there is some $x^s$ in the convex combination with $c^T x > c^T x^s$

    => this contradicts the fact that $x$ is optimal
  - => all basic solutions in the convex combination are optimal ❑

- Theorem 3.12 is the basis for the simplex algorithm
  - clever local search among the vertices (geometric view)
  - clever local search among the basic feasible solutions (algebraic view)



- **Main topics of this chapter**
  - We consider the pivot operation that defines the neighborhood of basic feasible solutions
  - We analyze the underlying algebraic calculus
  - We will introduce the use of tableaus

- **First thoughts on the pivot operation**
  - Let $y$ be a basic feasible solution of $Ax = 0,\ x \geq 0$ with basis $B$

    => can write $Ay = b$ as $\sum_{i \in B} A_{B(i)} y_{B(i)} = b$ (3.14)
  - $B$ basis => every column $A_j \notin B$ is a linear combination of basic columns

    => there are numbers $x_{ij}$ $(i = 1, \ldots, m)$ with $\sum_{i=1,\ldots,m} A_{B(i)} x_{ij} = A_j$ (3.15)
  - (3.14) - $\theta \cdot$(3.15) => $\sum_{i=1,\ldots,m} A_{B(i)} (y_{B(i)} - \theta x_{ij}) + \theta A_j = b$ (3.16)
  - We will now change $\theta$ in (3.16) to obtain another basic feasible solution.

    We consider 3 cases:
    - Case 1: $y$ is not degenerate and not all $x_{ij} \leq 0$ $(i = 1, \ldots, m)$
      - $y$ not degenerate => all $y_{B(i)} > 0$ $(i = 1, \ldots, m)$

        transition from $\theta = 0$ to $\theta > 0$ corresponds to the transition from $y$ to $x(\theta)$ with

$$x_\ell(\theta) = \begin{cases} y_{B(i)} - \theta x_{ij} & \ell = B(i),\ i = 1,\dots,m \\ \theta & \ell = j \\ 0 & \text{otherwise} \end{cases}$$

=> x(θ) has m+1 entrees > 0 (for small θ)

x(θ) stays feasible as long as x(θ) ≥ 0

since: $Ax(\theta) = \Sigma_{i=1,\dots,m} A_{B(i)}(y_{B(i)} - \theta x_{ij}) + \theta A_j = b$ because of (3.16)

x(θ) ≥ 0 holds as long as $0 \le \theta \le \theta_0$ with

$$\theta_0 = \min\left\{\frac{y_{B(i)}}{x_{ij}} \mid x_{ij} > 0,\ i = 1,\dots,m\right\} \quad (3.17)$$

If the minimum of (3.17) is attained at k, then $x(\theta_0)$ is a basic feasible solution with basis $B' = B - \{B(k)\} \cup \{j\}$

since

1. the columns $A_j$ with $j \in B'$ are linearly independent

Suppose not. Then there exist numbers $d_i$ (not all 0) with $\Sigma_{i=1,\dots,m,\, i \neq k} d_i A_{B(i)} + d_j A_j = 0$

(3.15) => $\Sigma_{i=1,\dots,m,\, i \neq k} d_i A_{B(i)} + d_j(\Sigma_{i=1,\dots,m} A_{B(i)} x_{ij}) = 0$



this is a linear combination of the columns of B that gives 0.

So all coefficients in the linear combination must be 0

=> $(d_i + d_j x_{ij}) = 0$ for $i = 1, \dots, m,\ i \neq k$ and $d_j x_{kj} = 0$

$x_{kj}$ determines $\theta_0$ => $x_{kj} > 0$ => $d_j = 0$ => $d_i = 0$ for all i, a contradiction

2. $x(\theta_0)_{N'} = 0$ by construction

3.3 Example (continued)

$$\left(\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 4 \\ 1 & & & 1 & 2 \\ & & 1 & & 3 \\ 3 & & 1 & & 6 \end{array}\right) \Rightarrow y_B = \begin{pmatrix} y_3 \\ y_1 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 1 \\ 4 \end{pmatrix}$$

$$A_5 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = A_1 \cdot x_{15} + A_3 \cdot x_{25} + A_6 \cdot x_{35} + A_7 \cdot x_{45}$$

$= 1 \cdot A_1 - 1 \cdot A_3 + 1 \cdot A_6 + 1 \cdot A_7$

=> (3.16) reads as $(2 - \theta) \cdot A_1 + (2 + \theta) \cdot A_3 + (1 - \theta) \cdot A_6 + (4 - \theta) \cdot A_7 + \theta \cdot A_5$

=> transition from y = x(0) to x(θ) reads as

$$y = \begin{pmatrix} 2 \\ 0 \\ 2 \\ 0 \\ 0 \\ 1 \\ 4 \end{pmatrix} \rightarrow x(\theta) = \begin{pmatrix} 2-\theta \\ 0 \\ 2+\theta \\ 0 \\ \theta \\ 1-\theta \\ 4-\theta \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \\ 3 \\ 0 \\ 1 \\ 0 \\ 3 \end{pmatrix} \quad \text{for } \theta = 1$$

x(1) is a basic feasible solution with basis B = { 3, 1, 5, 7 }

Case 2: y is degenerate

so there is an index i with $y_{B(i)} = 0$ and $x_{ij} > 0$

=> $\theta_0 = 0$

=> no movement in $R^n$ and thus also not in $R^{n-m}$

=> we stay in the same vertex / basic feasible solution, but obtain another basis, as $A_{B(i)}$ and $A_j$ are exchanged

Case 3: all $x_{ij} \leq 0$ (i = 1, ..., m)

=> θ can be made arbitrarily large and x(θ) stays feasible

=> $S_I$ is unbounded



=> objective function is not bounded from below because of Theorem 3.7

**Basis exchange**

3.13 Theorem (basis exchange)

When computing $\theta_0$, suppose that the minimum is attained at i = k, then $x(\theta_0)$ is a basic feasible solution with basis B´, and

$$B'(i) = \begin{cases} B(i) & i \neq k \\ j & i = k \end{cases}$$

$x(\theta_0)$ is degenerate if k is not unique

Proof: this follows from the previous arguments

In the example we obtain k = 3, B(3) = 6, B´(3) = 5 ❑

This step from one basic feasible solution to another one is called a pivot step (also pivot operation or simply pivot)

We say that: $A_{B(k)}$ leaves the basis and $A_j$ enters the basis

$x_{B(k)}$ leaves the basis and $x_j$ enters the basis

- $\theta_0$ is called the primal step length
- Two basic feasible solutions with different bases are called neighbors, if one can be obtained from the other by a pivot operation. So pivot operations define a neighborhood on the set of basic feasible solutions.



**Goal: make the pivot operation efficient**

=> must get hold of the $x_{ij}$

=> must express non-basic columns as a linear combination of basic columns

this can be done by transforming the basis to an identity matrix

**An example**

Let $Ax = b$ be given as

$$3x_1 + 2x_2 + x_3 = 1$$
$$5x_1 + x_2 + x_3 + x_4 = 3$$
$$2x_1 + 5x_2 + x_3 + x_5 = 4$$

We write this down in a tableau:

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| 1 | 3 | 2 | 1 | 0 | 0 |
| 3 | 5 | 1 | 1 | 1 | 0 |
| 4 | 2 | 5 | 1 | 0 | 1 |

with b as column 0

We transform the linear system / the tableau w.r.t. a basis $B$ is such a way that the basic columns form the identity matrix.

For $B = \{ 3, 4, 5 \}$ this gives the following tableau

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| 1 | 3 | 2 | 1 | 0 | 0 |
| 2 | 2 | -1 | 0 | 1 | 0 |
| 3 | -1 | 3 | 0 | 0 | 1 |

=> $x_{B(1)} = x_3 = 1$

$x_{B(2)} = x_4 = 2$

$x_{B(3)} = x_5 = 3$

Moreover $A_1 = 3A_{B(1)} + 2A_{B(2)} - 1A_{B(3)}$. The coefficients are obviously given by the column of $x_1$

=> the numbers $x_{ij}$ are in column $j$ of the transformed tableau (transformed such that $B$ is the identity matrix)

So if $A_1$ is to enter the basis, then

$$\theta_0 = \min \{ \frac{y_{B(i)}}{x_{i1}} \mid x_{i1} > 0,\ i = 1, \ldots, m \} = \min \{ \frac{1}{3}, \frac{2}{2} \} = \frac{1}{3} \text{ und } k = 1$$



The tableau for the new basis is then obtained by transforming it so that $A_1$ becomes the new unit vector in the basis

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| $\frac{1}{3}$ | 1 | $\frac{2}{3}$ | $\frac{1}{3}$ | 0 | 0 |
| $\frac{4}{3}$ | 0 | $-\frac{7}{3}$ | $-\frac{2}{3}$ | 1 | 0 |
| $\frac{10}{3}$ | 0 | $\frac{11}{3}$ | $\frac{1}{3}$ | 0 | 1 |

=> $x_{B'(1)} = x_1 = 1/3$

$x_{B'(2)} = x_4 = 4/3$

$x_{B'(3)} = x_5 = 10/3$

In this basis exchange, $a_{11}$ plays the role of the pivot element (as in Gaussian elimination)

**The pivoting rules**

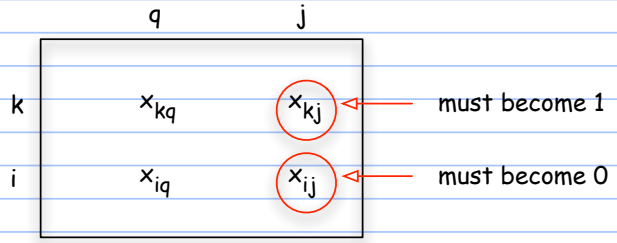
Let $(x_{ij})$ be the tableau for basis $B$ and let $(x'_{ij})$ be the tableau for the new basis $B'$, both with the right-hand-side as column 0, i.e. $(x_{i0})$ and. $(x'_{i0})$. Let $x_{kj}$ be the pivot element. Then the entries of the new tableau

(3.18)

$$\begin{cases} x'_{kq} &= \frac{x_{kq}}{x_{kj}} & q = 0,\dots,n \\ x'_{iq} &= x_{iq} - x'_{kq}x_{ij} & i = 1,\dots,m,\ i \neq k;\ q = 0,\dots,n \\ B'(i) &= B(i) & i = 1,\dots,m,\ i \neq k \\ B'(i) &= j & i = k \end{cases}$$

are obtained by

Mnemonic:






Goal: Understand the change of the objective function for a basis exchange

**First thoughts on the change of the objective function**

Let y be a basic feasible solution with basis B

=> y has the objective function value (cost) $z_y = \Sigma_{i=1,\dots,m}\ c_{B(i)}y_{B(i)}$

If $A_j$ enters the basis, then $Ax(\theta) = \Sigma_{i=1,\dots,m}\ A_{B(i)}(y_{B(i)} - \theta x_{ij}) + \theta A_j$ ( = b )

$\theta$ = 1 => for every "unit" of variable $x_j$ entering the basis, $x_{ij}$ units of $y_{B(i)}$ leave basis

=> the cost changes by $1 \cdot c_j - \Sigma_{i=1,\dots,m}\ x_{ij}c_{B(i)}$

Set

$$z_j := \sum_{i=1}^{m} x_{ij}c_{B(i)}$$

$$\bar{c}_j := c_j - z_j$$

$\bar{c}_j$ is called the reduced cost coefficient (or simply reduced cost) of $x_j$

it describes the change in cost when $x_j$ is entering the basis with value $x_j = 1$

$z_j$ describes the change in cost contributed by the the basic variables of basis B when $x_j$ enters the basis with value $x_j = 1$

We will show:

- $\bar{c}_j < 0$

=> we can improve the cost by letting $x_j$ enter the basis

$\bar{c}_j \geq 0$ for all non-basic variables $x_j$

=> current feasible basic solution is locally optimal w.r.t. the basis exchange neighborhood

=> we have a globally optimal feasible basic solution, i.e., the neighborhood w.r.t. basis exchange is exact

**Notation**

X = current tableau for basis B

A = initial matrix, b initial right hand side

Then

$X = B^{-1}(b|A)$

i.e., current tableau is obtained by multiplying $(b|A)$ with $B^{-1}$ from the left

$z^T = c_B^T X = c_B^T B^{-1} A$

i.e., this equation describes the change in cost $z_j$ for the basic variables when $x_j = 1$ enters the basis

because:



$$z_j = \sum_{i=1}^{m} x_{ij} c_{B(i)} = (c_{B(1)}, \ldots, c_{B(m)}) \begin{pmatrix} x_{1j} \\ \vdots \\ x_{mj} \end{pmatrix}$$

$\Rightarrow z^T = c_B^T X = c_B^T B^{-1} A$

**Optimality criterion**

3.14 Theorem (Optimality Criterion)

Let $x$ be a basic feasible solution with basis $B$. Then

(1) A pivot step at which $x_j$ enters the basis with value $x_j = \theta_0$ changes the cost by

$$\theta_0 \bar{c}_j = \theta_0 (c_j - z_j) \quad (3.19)$$

(2) If

$$\bar{c} = c - z \geq 0 \quad (3.20)$$

i.e., all reduced costs are non-negative, then $x$ is optimal.

Proof

Proof of (1) (intuitively clear from our first thoughts above, here is the precise derivation)

When $x_j$ enters the basis, the pivoting rules (3.18) yield the following new right hand side (= column 0 of the tableaus)

$$x'_{i0} = \begin{cases} x_{i0} - \theta_0 x_{ij} & i \neq k \\ \theta_0 & i = k \end{cases} \qquad \text{with } \theta_0 = \frac{x_{k0}}{x_{kj}}$$

Let z be the old objective function value function value and let z' be the new one

=> new objective function value is

$$z' = \sum_{i=1,\, i\neq k}^{m} (x_{i0} - \theta_0 x_{ij}) c_{B(i)} + \theta_0 c_j$$

$$= \underbrace{\sum_{i=1}^{m} x_{i0} c_{B(i)}}_{=z} - \theta_0 \sum_{i=1}^{m} x_{ij} c_{B(i)} - (x_{k0} - \theta_0 x_{kj}) c_{B(k)} + \theta_0 c_j$$

$$= z + \theta_0 \Bigg( x_{kj} c_{B(k)} + \underbrace{c_j - \sum_{i=1}^{m} x_{ij} c_{B(i)}}_{=c_j - z_j} \Bigg) - x_{k0} c_{B(k)}$$



$$= z + \theta_0 (c_j - z_j) + \underbrace{\theta_0 x_{kj} c_{B(k)}}_{= x_{k0} c_{B(k)} \text{ as } \theta_0 = x_{k0}/x_{kj}} - x_{k0} c_{B(k)}$$

$= z + \theta_0 (c_j - z_j)$

=> (3.19)

Proof of (2)

Let y be a feasible solution of the LP, i.e., Ay = b, y ≥ 0

$\bar{c} = c - z \geq 0 \Rightarrow c \geq z$. Together with $y \geq 0$ we obtain

$$c^T y \geq z^T y = c_B^T B^{-1} A y = c_B^T B^{-1} b = c_B^T x_B$$

So x is globally optimal (and in particular also a best basic feasible solution) ❑

**Tableau with reduced costs**

The optimality criterion suggests to make the reduced costs available in the tableau.

=> add them as row 0

Question: How to obtain the initial reduced costs from the given objective c?

Write the cost as

$0 = -z_0 + c_1x_1 + \ldots + c_nx_n$

This can be seen as enlarging $Ax = b$ to

$$\begin{pmatrix} 1 & \vdots & c^T \\ \cdots & \cdots & \cdots \\ 0 & \vdots & A \end{pmatrix} \begin{pmatrix} -z_0 \\ \cdots \\ x \end{pmatrix} = \begin{pmatrix} 0 \\ \cdots \\ b \end{pmatrix}$$

For a basic variable $x_j$ we obtain

$$\bar{c}_j = c_j - z_j = c_j - \sum_{i=0}^{m} x_{ij}c_{B(i)} = 0$$

since $(x_{ij})$ is a unit vector with 1 at entry $i_0$ with $B(i_0) = j$

$$\Rightarrow c_j - \sum_{i=0}^{m} x_{ij}c_{B(i)} = c_j - c_{B(i_0)} = c_j - c_j = 0$$

So $\bar{c}_j = 0$ for basic variables

for non-basic variables $x_j$ we obtain: (3.21)

$$\bar{c}_j = c_j - z_j = c_j - \sum_{i=1}^{m} x_{ij}c_{B(i)} = c_j - c_B^T B^{-1}A_j = c_j - c_B^T X_j$$



i.e., $\bar{c}_j$ is obtained from $c$ and $X$ for non-basic variables

for $-z_0$ we obtain: (3.22)

$$-z_0 = -\sum_{i=1}^{m} c_{B(i)}x_{B(i)} = -c_B^T X_0$$

i.e., $-z_0$ is also obtained from $c$ and $X$

New enlarged tableau (in the following denoted as tableau)

| | $-z_0$ | $\bar{c}_1$ | $\ldots$ | $\bar{c}_j$ | $\ldots$ | $\bar{c}_n$ |
|---|---|---|---|---|---|---|
| | $x_{10}$ | $x_{11}$ | $\ldots$ | $x_{1j}$ | $\ldots$ | $x_{1n}$ |
| $x_B$ | $\vdots$ | $\vdots$ | | $\vdots$ | | $\vdots$ |
| | $x_{m0}$ | $x_{m1}$ | $\ldots$ | $x_{mj}$ | $\ldots$ | $x_{mn}$ |
| | $B^{-1}b$ | | | $B^{-1}A_j$ | | |

3.15 Theorem (Updating the reduced cost)

At a basis exchange, the reduced costs are updated by the same rules (3.18) as all other rows of the tableau (3.18).

In other words: for column $j$ entering the basis, row 0 is transformed so that $x_{0j} = \bar{c}_j$ becomes 0. The

resulting changes in row 0 give the new reduced costs and the new $-z_0$

Proof: checking ❑

3.16 Corollary (Test for an unbounded objective function)

$\bar{c}_j < 0$, all $x_{ij} \leq 0$ $(i = 1,...m)$ => $c^T x$ is not bounded from below

Proof

$\theta$ and thus $x(\theta)$ can then become arbitrarily large

=> the objective function decreases by $\theta \cdot \bar{c}_j \to \infty$ ❑

**The basic version of the simplex algorithm**

Algorithm (Generic simplex algorithm)

Input

Tableau $X$ of an LP in standard form with full row rank and with identity matrix as feasible basis

Output

At termination: an basic optimal feasible solution or the message that the objective function is unbounded

Termination is not guaranteed

Method (iterative pivoting in the direction of decreasing cost)



while there is a column $j$ with $\bar{c}_j < 0$ do

choose column $j$ with $\bar{c}_j < 0$

if $x_{ij} \leq 0$ for all $i$ then return "objective function is unbounded"

determine $\theta_0$ and index $k$, for which the minimum in (3.17) is attained

pivot with pivot entry $x_{kj}$ according to (3.18) (also for row 0)

return $x_B$

Example

min $x_1 + x_2 + x_3 + x_4 + x_5$

s.t. $3x_1 + 2x_2 + x_3 = 1$

$5x_1 + x_2 + x_3 + x_4 = 3$

$2x_1 + 5x_2 + x_3 + x_5 = 4$

$x_j \geq 0$

Initial tableau (not yet with reduced costs and identity matrix as basis)

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 3 | 2 | 1 | 0 | 0 |
| 3 | 5 | 1 | 1 | 1 | 0 |
| 4 | 2 | 5 | 1 | 0 | 1 |

- Transform it w.r.t. basis B = { 3, 4, 5 }

  i.e., columns 3, 4, 5 change into unit vectors (including row 0 for the cost coefficients)

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| -z | -6 | -3 | -3 | 0 | 0 | 0 |
| $x_3$ | 1 | 3 | 2 | 1 | 0 | 0 |
| $x_4$ | 2 | 2 | -1 | 0 | 1 | 0 |
| $x_5$ | 3 | -1 | 3 | 0 | 0 | 1 |

- A specific transformation:
  1. subtracting row 1 from row 2 and from row 3 yields the identity matrix in rows 1-3;
  2. subtracting rows 1-3 from row 0 correspond to (3.21) and (3.22)
- Pivot operation:



Columns 1 and 2 have reduced costs < 0, we choose $x_2$ to enter the basis.

Computing $\theta_0$ yields $\theta_0 = \min\{ 1/2, 3/3 \} = 1/2$ with $k = 1$. Hence $x_{B(1)} = x_3$ leaves the basis.

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| -z | -9/2 | 3/2 | 0 | 3/2 | 0 | 0 |
| $x_2$ | 1/2 | 3/2 | 1 | 1/2 | 0 | 0 |
| $x_4$ | 5/2 | 7/2 | 0 | 1/2 | 1 | 0 |
| $x_5$ | 3/2 | -11/2 | 0 | -3/2 | 0 | 1 |

All reduced costs ≥ 0 => optimal solution reached

x = (0, 1/2, 0, 5/2, 3/2) is as basic optimal solution with cost z = 9/2

- **Applets for pivoting**
  - Advanced Simplex Pivoting Tool
    - Advanced Simplex Pivot Tool
  - Matrix Row Operation Tool
    - http://people.hofstra.edu/faculty/Stefan_Waner/RealWorld/tutorialsf1/scriptpivot2.html

**Main topics of this chapter**

- Strategies for the choice of a column (pivoting rules)
- Strategies for the choice of a pivot element in a column if there are more than one
- In particular: Ensure termination of the simplex algorithm

**Strategies for the choice of a column (Pricing)**

- We have a choice if several $\bar{c}_j < 0$, this choice influences the number of pivot operations.
- There is no "best" choice. Commercial solves apply a wide selection of heuristics. Here are some of them:
  - (1) Steepest relative descent among the non-basic variables
    - Among all columns $j$ with $\bar{c}_j < 0$, choose $j$ with maximum $|\bar{c}_j|$
    - This column has the largest relative descent of cost in the direction of a non-basic variable. However, since $\theta_0$ is not known yet, this choice does not necessarily lead to the largest descent.
  - (2) Steepest absolute descent among the non-basic variables
    - Among all columns $j$ with $\bar{c}_j < 0$, choose $j$ with maximum $|\theta_0 \bar{c}_j|$
    - This column has the largest absolute descent of cost in the direction of a non-basic variable. It generally reduces the number of pivots, but at the price of an increase of computational effort by a factor of $m$ per



pivot compared with (1)

- (3) Largest relative descent in the whole feasible domain
  - Among all columns $j$ with $\bar{c}_j < 0$, choose $j$ with maximum
    $$\frac{\bar{c}_j}{\sqrt{1+\sum_{i=1}^{m} x_{ij}^2}} = \frac{\bar{c}_j}{\|x(\theta)-x\|} \text{ with } \theta = 1$$
  - This column has the largest relative descent of cost in the direction of $x(\theta) - x$, but is computationally very expensive

**Strategies for the choice of the pivot element**

- This choice is not so important for termination of the simplex algorithm if $\theta_0 > 0$, because then the pivot operation leads to a new vertex / a new basic feasible solution with a better objective function value.
- It is however important if $\theta_0 = 0$. In this case, the current basic solution is degenerate and the pivot operation changes only the basis, but not the basic solution / vertex, see Theorem 3.13
- This may lead to cycling, i.e., starting from tableau $X$ we arrive again at tableau $X$ after several pivots
  => we have traversed a "cycle" of different bases with the same basic solution
  => the simplex algorithm does not terminate

3.15 Example (An example for cycling , Gass 1964)

min $-3/4\ x_1 + 150\ x_2 - 1/50\ x_3 + 6\ x_4$

s.t. $1/4\ x_1 - 60\ x_2 - 1/25\ x_3 + 9\ x_4 \le 0$

$1/2\ x_1 - 90\ x_2 - 1/50\ x_3 + 3\ x_4 \le 0$

$x_3 \le 1$

$x_j \ge 0$

Initial tableau with a basis of slack variables

| | | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
|---|---|---|---|---|---|---|---|---|
| $-z$ | 0 | -3/4 | 150 | -1/50 | 6 | 0 | 0 | 0 |
| $x_5$ | 0 | 1/4 | -60 | -1/25 | 9 | 1 | 0 | 0 |
| $x_6$ | 0 | 1/2 | -90 | -1/50 | 3 | 0 | 1 | 0 |
| $x_7$ | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

Choosing the sequence

$x_{11}\ x_{22}\ x_{13}\ x_{24}\ x_{15}\ x_{26}$

of pivot element, we arrive again at the initial tableau



Proof: check, e.g. with the Matrix Row Operation Tool

@ http://people.hofstra.edu/faculty/Stefan_Waner/RealWorld/tutorialsf1/scriptpivot2.html

Anti-cycling rules

The lexicographic anti-cycling rule

Divide every row i with $x_{ij} > 0$ by $x_{ij}$ and choose the lexicographically smallest of them as pivot row

Lexicographic rule:

Choose k such that

$$\frac{1}{x_{kj}} \cdot X_{k,\cdot} = \text{lex-min}\left\{ \frac{1}{x_{ij}} \cdot X_{i,\cdot} \mid i = 1, \ldots, m,\ x_{ij} > 0 \right\}$$

where $X_{i,\cdot}$ = i-th row of tableau X and lex-min = lexicographic minimum

$\theta_0 := \frac{x_{k0}}{x_{kj}}$

3.16 Theorem (The lexicographic anti-cycling rule avoids cycling )

(a) The rows i = 1, ..., m of the current tableau can be made lexicographically positive.

(b) If the rows i = 1, ..., m of the current tableau are lexicographically positive, then the simplex algorithm with lexicographic anti-cycling rule terminates after finitely many steps.

Proof

(1) The rows of the current tableau can be made lexicographically positive.

e.g. by permuting the basic columns to positions 1,2,...,m

=> every row either starts with $x_{i0} > 0$ or with (0,...,0,1,...), where $x_{i,B(i)} = 1$

=> every row is lexicographically positive, i.e., $>_{lex}$ (0,0,...,0)

(2) The lex-min is unique

Suppose not => there are rows i, r with

$$\frac{1}{x_{ij}}(x_{i0}, x_{i1}, \ldots, x_{in}) = \frac{1}{x_{rj}}(x_{r0}, x_{r1}, \ldots, x_{rn})$$

=> rows i and r are linearly dependent

=> this contradicts assumption 3.1 that rank(A) = m

(3) All rows of the tableaus stay lexicographically positive after a pivot step

if $x_{kj}$ is the pivot element, then the new rows after pivoting are

for i = k:



$$\frac{1}{x_{kj}}(x_{k0}, x_{k1}, \ldots, x_{kn})$$

$x_{kj} > 0$ => we stay lexicographically positive

for i ≠ k:

$$(x_{i0}, x_{i1}, \ldots, x_{in}) - x_{ij}\frac{1}{x_{kj}}(x_{k0}, x_{k1}, \ldots, x_{kn})$$

This is $>_{lex}$ (0,0,...,0)

$$\Leftrightarrow \frac{1}{x_{kj}}(x_{k0}, x_{k1}, \ldots, x_{kn}) <_{lex} \frac{1}{x_{ij}}(x_{i0}, x_{i1}, \ldots, x_{in})$$

This is the case since row k is the lex-min, and the lex-min is unique (equality cannot happen)

(4) Row 0 (the cost row) strictly increases lexicographically after every pivot step

if $x_{kj}$ is the pivot element, then the new row 0 after the pivot operation is obtained as

$$(-z, \bar{c}_1, \ldots, \bar{c}_n) - \frac{1}{x_{kj}} \bar{c}_j \, (x_{k0}, x_{k1}, \ldots, x_{kn})$$

lex positive

$> 0$ as pivot element

$< 0$ as reduced cost in the pivot column

=> the new row is lexicographically larger

(5) Termination

Every basis uniquely determines row 0 because of (3.21) and (3.22).

Row 0 grows lexicographically => corresponding basic solutions must be different

There are only finitely many basic solutions => termination ❑

3.15 Example (continued)

The initial tableau is already lexicographically positive



| | | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
|---|---|---|---|---|---|---|---|---|
| $-z$ | 0 | -3/4 | 150 | -1/50 | 6 | 0 | 0 | 0 |
| $x_5$ | 0 | 1/4 | -60 | -1/25 | 9 | 1 | 0 | 0 |
| $x_6$ | 0 | 1/2 | -90 | -1/50 | 3 | 0 | 1 | 0 |
| $x_7$ | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

The sequence

$x_{11}$ $x_{22}$ $x_{23}$ $x_{34}$ $x_{35}$

of pivot elements fulfills the lexicographic anti-cycling rule and terminates with an optimal solution. The initial degenerate basic solution is left after pivoting with $x_{34}$.

Proof: check, e.g. with the Matrix Row Operation Tool

@ http://people.hofstra.edu/faculty/Stefan_Waner/RealWorld/tutorialsf1/scriptpivot2.html

Bland's anti-cycling rule

Among all columns $j$ with $\bar{c}_j < 0$ choose the one with smallest column index $j$

Among all rows $i$ with $\theta_0 = x_{i0}/x_{ij}$ choose the one with smallest column index $B(i)$

Proof: Exercise ❑

- Empirical tests show that Bland's rule needs many pivots

- Anti-cycling in practice
  - Experience shows that cycling is mostly already resolved by the rounding caused by floating point arithmetic. Commercial solvers control the progress of the objective function and change their pivot rule if there is evidence of cycling until they reach a different basic feasible solution.



- Goal: Provide an initial tableau for the simplex algorithm, i.e., a tableau that is already transformed w.r.t. to the basis of a basic feasible solution
  - This is easy if the LP is given in the form $Ax \le b,\ x \ge 0$ with $b \ge 0$. Then we just introduce slack variables $s_1, \ldots, s_m$ and the associated columns form a basis

Initial tableau

| | | $x_1$ | … | $x_n$ | $s_1$ | $s_2$ | … | $s_m$ |
|---|---|---|---|---|---|---|---|---|
| $-z$ | 0 | $c_1$ | … | $c_n$ | 0 | 0 | … | 0 |
| $s_1$ | $b_1$ | $a_{11}$ | … | $a_{1n}$ | 1 | 0 | … | 0 |
| ⋮ | ⋮ | ⋮ | ⋱ | ⋮ | ⋮ | ⋮ | ⋱ | 0 |
| $s_m$ | $b_m$ | $a_{m1}$ | … | $a_{mn}$ | 0 | 0 | … | 1 |

- For general LP in standard form, this is done with the two-phase-method
  - Given: LP in standard form

$$\begin{aligned} \min\ & c^T x \\ \text{s.t. } & Ax = b \\ & x \ge 0 \end{aligned}$$

and w.l.o.g. $b \ge 0$ (otherwise multiply rows by -1)

- Phase I
  - Introduce artificial variables $x^a = (x_1^a, \ldots, x_m^a)^T$ ( $^a$ for artificial ) and solve the LP

$$\begin{aligned} \min \quad \xi &= x_1^a + \ldots + x_m^a \\ \text{s.t.} \quad x^a + Ax &= b \qquad (3.23) \\ x^a, x &\geq 0 \end{aligned}$$

  with the simplex algorithm (the artificial variables form a basis and the cost coefficients w.r.t. $\xi$ are in reduced form (3.21))
  - There are 3 possible outcomes
    - Case 1: The minimum of $\xi$ is 0 and all $x_i^a$ are non-basic variables
      - => all $x_i^a = 0$ and we have a basic feasible solution for the initial problem
    - Case 2: The minimum of $\xi$ is 0 but some $x_i^a$ are basic variables
      - Let $x_{B(i)}$ be an artificial variable.

        $\xi = 0$ => $x_{B(i)} = 0$

        Try to eliminate $x_{B(i)}$ from the basis.

        To that end we need in tableau $X$ a non-artificial non-basic column $j$ with $x_{ij} \neq 0$

        (since $x_{B(i)} = 0$, we can pivot with $x_{ij}$ (first multiply row $i$ by -1 if $x_{ij} < 0$)



      - Case 2a: There is such a non-artificial non-basic column $j$ with $x_{ij} \neq 0$
        - Pivot with $x_{ij}$

          => we have a basis with fewer artificial variables
      - Case 2b: There is no non-artificial non-basic column $j$ with $x_{ij} \neq 0$
        - => $x_{ij} = 0$ for $j = 1, \ldots, n$, i.e. for all non-artificial variables
          - clear from the above for non-basic variables $x_j$
          - basic variables $x_j$ have a 0 in position $i$ in their column because only $x_{B(i)}$ has a 1 there
        - => rank($A$) < $m$

          => assumption 3.1 violated

          this implies that row $i$ of $A$ is a linear combination of the other rows of $A$ and can be deleted together with artificial variable $x_{B(i)}$
        - We can therefore drop assumption 3.1, Phase I provides a test for rank($A$) = $m$
      - Repetition of these steps results in a basic feasible solution of the initial LP (possibly after deleting some linearly dependent rows from $A$. The remaining rows have full rank.)
    - Case 3: The minimum of $\xi$ is > 0
      - => the initial LP has no feasible solution, i.e., $S = \emptyset$

- We can therefore drop assumption 3.2, Phase I provides a test for $S \neq \emptyset$

- Phase II
  - Continue with the basic feasible solution from Phase I (if $S \neq \emptyset$)
  - To this end, we must compute the reduced costs $\bar{c}_j$ and the objective function value $-z_0$ of the original objective $c$ for the current basis from Phase I
    - either with (3.21) and (3.22)
    - or by applying all transformations of Phase I also to the cost coefficients $c_j$

      Theorem 3.15 => these transformations give the reduced costs $\bar{c}_j$ at the start of Phase II

- Algorithm (Two-Phase-Method)
  - Input
    - LP in standard form
  - Output
    - At termination : basic optimal solution or the message that the objective function is unbounded or that $S = \emptyset$
    - Termination can be guaranteed by anti-cycling rules



- Method
  - Phase I
    - add artificial variables $x_1^a, \ldots, x_m^a$
    - call simplex algorithm with objective $\xi = \sum x_i^a$
    - if $\xi_{opt} > 0$ then return "there is no feasible solution"
    - if basis contains artificial variables
      - then
        - if one of these artificial variables cannot be removed from the basis
          - then
            - delete the associated row
            - call Phase I for the resulting LP
      - else // $\xi_{opt} = 0$ and the basis contains no artificial variables
        - call Phase II
  - Phase II
    - call simplex algorithm with the original objective w.r.t. the basis from Phase I

3.17 Theorem (Two-Phase-Method)

The two-phase-method solves every LP in standard form. Assumption (3.1)-(3.3) can be dropped. They are checked in the Two-Phase-Method.

Proof

clear from the above ❑

Example

min $x_1 + x_2 + x_3 + x_4 + x_5$

s.t. $3x_1 + 2x_2 + x_3 = 1$

$5x_1 + x_2 + x_3 + x_4 = 3$

$2x_1 + 5x_2 + x_3 + x_5 = 4$

$x_j \geq 0$

Initial tableau for Phase I, we also keep track of the given objective function



| | $x_1^a$ | $x_2^a$ | $x_3^a$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|---|---|---|
| $-z$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| $-\xi$ | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| $x_1^a$ | 1 | 1 | 0 | 0 | 3 | 2 | 1 | 0 | 0 |
| $x_2^a$ | 3 | 0 | 1 | 0 | 5 | 1 | 1 | 1 | 0 |
| $x_3^a$ | 4 | 0 | 0 | 1 | 2 | 5 | 1 | 0 | 1 |

To obtain the reduced costs w.r.t. $\xi$ for Phase I, we must transform the costs of the artificial variables to 0 (reduced costs are 0 for basic variables).

We achieve this by subtracting each row 1,...,m from the $\xi$-row (the z-row has already reduced costs 0 in the basic columns)

Start tableau for Phase I

| | $x_1^a$ | $x_2^a$ | $x_3^a$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|---|---|---|
| $-z$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| $-\xi$ | -8 | 0 | 0 | 0 | -10 | -8 | -3 | -1 | -1 |
| $x_1^a$ | 1 | 1 | 0 | 0 | 3 | 2 | 1 | 0 | 0 |
| $x_2^a$ | 3 | 0 | 1 | 0 | 5 | 1 | 1 | 1 | 0 |
| $x_3^a$ | 4 | 0 | 0 | 1 | 2 | 5 | 1 | 0 | 1 |

Pivoting with the pivot elements marked by a red circle yields

| | $x_1^a$ | $x_2^a$ | $x_3^a$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|---|---|---|
| $-z$ | -1/3 | -1/3 | 0 | 0 | 0 | 1/3 | 2/3 | 1 | 1 |
| $-\xi$ | -14/3 | -10/3 | 0 | 0 | 0 | -4/3 | 1/3 | -1 | -1 |
| $x_1$ | 1/3 | 1/3 | 0 | 0 | 1 | 2/3 | 1/3 | 0 | 0 |
| $x_2^a$ | 4/3 | -5/3 | 1 | 0 | 0 | -7/3 | -2/3 | 1 | 0 |
| $x_3^a$ | 10/3 | -2/3 | 0 | 1 | 0 | 11/3 | 1/3 | 0 | 1 |



| | $x_1^a$ | $x_2^a$ | $x_3^a$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|---|---|---|
| $-z$ | -1/2 | -1/2 | 0 | 0 | -1/2 | 0 | 1/2 | 1 | 1 |
| $-\xi$ | -4 | 4 | 0 | 0 | 2 | 0 | 1 | -1 | -1 |
| $x_2$ | 1/2 | 1/2 | 0 | 0 | 3/2 | 1 | 1/2 | 0 | 0 |
| $x_2^a$ | 5/2 | -1/2 | 1 | 0 | 7/2 | 0 | 1/2 | 1 | 0 |
| $x_3^a$ | 3/2 | -15/6 | 0 | 1 | -11/2 | 0 | -3/2 | 0 | 1 |

| | $x_1^a$ | $x_2^a$ | $x_3^a$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|---|---|---|
| $-z$ | -3 | 0 | -1 | 0 | -4 | 0 | 0 | 0 | 1 |
| $-\xi$ | -3/2 | 7/2 | 1 | 0 | 11/2 | 0 | 3/2 | 0 | -1 |
| $x_2$ | 1/2 | 1/2 | 0 | 0 | 3/2 | 1 | 1/2 | 0 | 0 |
| $x_4$ | 5/2 | -1/2 | 1 | 0 | 7/2 | 0 | 1/2 | 1 | 0 |
| $x_3^a$ | 3/2 | -5/2 | 0 | 1 | -11/2 | 0 | -3/2 | 0 | 1 |

| | $x_1^a$ | $x_2^a$ | $x_3^a$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|---|---|---|
| -z | -9/2 | 5/2 | -1 | -1 | 3/2 | 0 | 3/2 | 0 | 0 |
| -ξ | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| $x_2$ | 1/2 | 1/2 | 0 | 0 | 3/2 | 1 | 1/2 | 0 | 0 |
| $x_4$ | 5/2 | -1/2 | 1 | 0 | 7/2 | 0 | 1/2 | 1 | 0 |
| $x_5$ | 3/2 | -5/2 | 0 | 1 | -11/2 | 0 | -3/2 | 0 | 1 |

We are lucky since the final tableau of Phase I is already optimal for Phase II (all reduced costs w.r.t. c are ≥ 0)



- Goal: Interpret the simplex algorithm geometrically
  - It walks from vertex to vertex along edges of the polyhedron with occasional extra pivots in degenerate vertices

- 3.8 Example (continued)
  - max $x_1 + 14x_2 + 6x_3$ <=> min $-x_1 - 14x_2 - 6x_3$

    s.t. $x_1 + x_2 + x_3 \le 4$

    $x_1 \le 2$

    $x_3 \le 3$

    $3x_2 + x_3 \le 6$

    $x_j \ge 0$
  - After adding slack variables (= first basis) , the simplex algorithm yields the following tableaus:

1

| | | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
|---|---|---|---|---|---|---|---|---|
| $-z$ | 0 | -1 | -14 | -6 | 0 | 0 | 0 | 0 |
| $x_4$ | 4 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| $x_5$ | 2 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| $x_6$ | 3 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| $x_7$ | 6 | 0 | 3 | 1 | 0 | 0 | 0 | 1 |

2

| | | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
|---|---|---|---|---|---|---|---|---|
| $-z$ | 2 | 0 | -14 | -6 | 0 | 1 | 0 | 0 |
| $x_4$ | 2 | 0 | 1 | 1 | 1 | -1 | 0 | 0 |
| $x_1$ | 2 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| $x_6$ | 3 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| $x_7$ | 6 | 0 | 3 | 1 | 0 | 0 | 0 | 1 |

3

| | | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
|---|---|---|---|---|---|---|---|---|
| $-z$ | 14 | 0 | -8 | 0 | 6 | -5 | 0 | 0 |
| $x_3$ | 2 | 0 | 1 | 1 | 1 | -1 | 0 | 0 |
| $x_1$ | 2 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| $x_6$ | 1 | 0 | -1 | 0 | -1 | 1 | 1 | 0 |
| $x_7$ | 4 | 0 | 2 | 0 | -1 | 1 | 0 | 1 |

4

| | | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
|---|---|---|---|---|---|---|---|---|
| $-z$ | 30 | 0 | 0 | 8 | 14 | -13 | 0 | 0 |
| $x_2$ | 2 | 0 | 1 | 1 | 1 | -1 | 0 | 0 |
| $x_1$ | 2 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| $x_6$ | 3 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| $x_7$ | 0 | 0 | 0 | -2 | -3 | 3 | 0 | 1 |



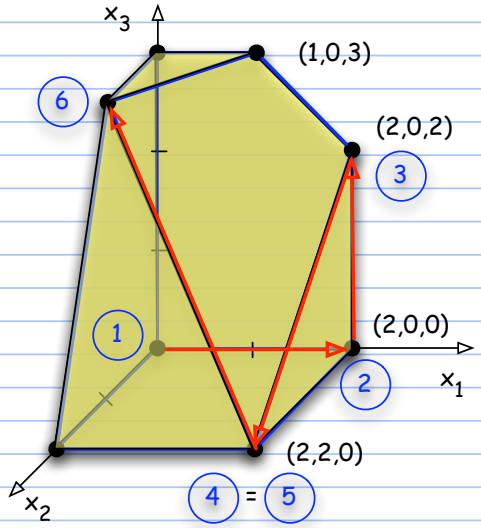
5

| | | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
|---|---|---|---|---|---|---|---|---|
| $-z$ | 30 | 0 | 0 | -2/3 | 1 | 0 | 0 | 13/3 |
| $x_2$ | 2 | 0 | 1 | 1/3 | 0 | 0 | 0 | 1/3 |
| $x_1$ | 2 | 1 | 0 | 2/3 | 1 | 0 | 0 | -1/3 |
| $x_6$ | 3 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| $x_5$ | 0 | 0 | 0 | -2/3 | -1 | 1 | 0 | 1/3 |

6

| | | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
|---|---|---|---|---|---|---|---|---|
| $-z$ | 32 | 1 | 0 | 0 | 2 | 0 | 0 | 4 |
| $x_2$ | 1 | -1/2 | 1 | 0 | -1/2 | 0 | 0 | 1/2 |
| $x_3$ | 3 | 3/2 | 0 | 1 | 3/2 | 0 | 0 | -1/2 |
| $x_6$ | 0 | -3/2 | 0 | 0 | -3/2 | 0 | 1 | 1/2 |
| $x_5$ | 2 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |

- This sequence corresponds to the following sequence of vertices in the associated polytope with variables $x_1$, $x_2$, $x_3$




We will analyze this observation now more closely.

Adjacency of vertices and basic feasible solutions

Two vertices $x'$, $y'$ of a polyhedron $P$ are called adjacent



:<=> the line segment $[x', y']$ is an edge of $P$

two basic feasible solutions $x, y$ of $Ax = b, x \geq 0$ are called adjacent

:<=> if one obtains $B_y$ from $B_x$ by a single pivot operation according to Theorem 3.16

Observe:

(1) One then also obtains $B_x$ from $B_y$ by a single pivot, i.e., the neighborhood is symmetric

(2) Then there are columns $j$ and $k$ with $B_y = (B_x - \{A_j\}) \cup \{A_k\}$

(3) This does not exclude that $x = y$, i.e., that the basic solution is degenerate

According to this definition, the simplex algorithm traverses a sequence of pairwise adjacent basic feasible solutions $x^1, x^2, \ldots, x^N$ with

$c^T x^1 \geq c^T x^2 \geq \ldots \geq c^T x^N = z_{opt}$

3.18 Theorem (Interpretation of edges in the three views)

Let $P$ be a polytope and let $S = \{x \mid Ax = b, x \geq 0\}$ be the associated feasible set of an LP in standard form.

Let $x' = (x_1', \ldots, x_{n-m}')^T$, $y' = (y_1', \ldots, y_{n-m}')^T \in P$ be different vertices and let $x = (x_1, \ldots, x_n)^T$, $y = (y_1, \ldots, y_n)^T \in S$ be the associated basic feasible solutions according to (3.7).

Then the following statements are equivalent:

(1) $[x', y']$ is an edge of $P$

(2) If $z' \in [x', y']$ is a strictly convex combination of points $u', v' \in P$, then $u', v' \in [x', y']$

(3) $x, y$ are (different) adjacent basic feasible solutions of $S$

Proof

Similar to Theorem 3.10, we use again the transformation $P \leftrightarrow S$

(1) => (2)

Let $[x', y']$ be an edge of $P$ => there is a supporting hyperplane $H$ with $H \cap P = [x', y']$, say $H = \{ w' \mid h^T w' = g \}$.

Assume that (2) is wrong for $z' \in [x', y']$, and let w.o.l.g. $u' \notin [x', y']$ and $h^T u' < g$.

Then $h^T v' \leq g$, since $P$ is contained in a halfspace induced by $H$

=> $g = h^T z' = h^T(\lambda u' + (1-\lambda)v') = \lambda h^T u' + (1-\lambda)h^T v' < g$ as $\lambda \neq 0$, a contradiction

(2) => (3)

Assume that $x', y'$ fulfill (2) but not (3), i.e., $x$ and $y$ are not adjacent

Let $M_x, M_y$ be the sets of columns $A_j$ of $A$ with $x_j > 0$ and $y_j > 0$, respectively.

Claim: there is a basic feasible solution $w \neq x, y$ such that: $w_j > 0$ => $A_j \in M_x \cup M_y$

Suppose not. Let w.o.l.g. $y \neq 0$ (possible since $x \neq y$). Then choose



$$c_j := \begin{cases} 0 & A_j \in M_y \\ 1 & A_j \in M_x - M_y \\ nM & \text{otherwise} \end{cases}$$

with $M$ large enough so that $c_j u_j > n$ for every basic feasible solution $u$ (such an $M$ exists because of Lemma 3.4).

=> $y$ is the only optimal solution for $c$ and every basic feasible solution $u$ with entries $u_j > 0$ not in $M_x \cup M_y$ has higher cost than $x$

=> if we start the simplex algorithm in $x$ w.r.t. $c$, it would report that no improvement is possible (since $x$ and $y$ are not adjacent) and thus claim that $x$ is optimal, a contradiction

Consider now the vertex $w' \in P$ corresponding to $w$. Since $w'$ is a vertex, $w'$ is not in $[x', y']$

=> $w$ is not in $[x, y]$

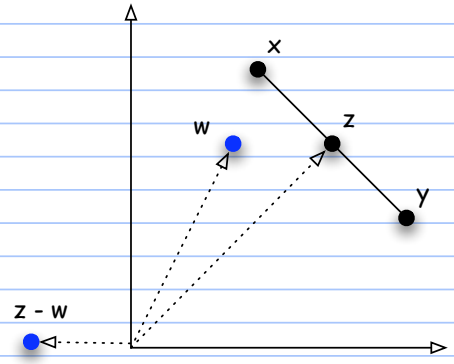
Let $z := 1/2 (x + y)$ => $z_j > 0$ for all $A_j \in M_x \cup M_y$ => each entry of $z$ can be decreased a little without reaching 0

Let $d := z - w$ => $d_j \neq 0$ only for entries in $M_x \cup M_y$

$z_j > 0$ for all entries in $M_x \cup M_y$ => there is $\theta > 0$ with $u := z + \theta d$, $v := z - \theta d \geq 0$

With these definitions, $Au = b$ and $Av = b$, i.e., $u, v \in S$.

Moreover: w not in [ x, y ] => u, v not in [ x, y ]




=> z = 1/2 (u + v) is a convex combination of the feasible points u, v ∉ [ x, y ]

=> (transformation to P) z´ = 1/2 (u´ + v´) is a convex combination of the points u´, v´ ∉ [ x´, y´ ], a contradiction to (2)

(3) => (1)

Let $B_x$, $B_y$ be the bases for x and y with $B_y = ( B_x - \{ A_j \} ) \cup \{ A_k \}$

Define c by



$$c_j := \begin{cases} 0 & \text{if } A_j \in B_x \cup B_y \\ 1 & \text{otherwise} \end{cases}$$

Claim: x and y are the only basic optimal feasible solutions w.r.t. c

clearly: x, y are optimal w.r.t. c

assume there is another basic optimal feasible solution z

construction of c => z fulfills $z_j > 0$ => $A_j \in B_x \cup B_y$

=> $B_z \subseteq B_x \cup B_y$

$A_j \notin B_x$ => (with $| B_x \cup B_y | = m + 1$, $z \neq x$) $A_j \in B_z$

=> $B_z = ( B_x - \{ A_q \} ) \cup \{ A_j \}$

=> there are single pivots that change x into y ($A_j$ enters, $A_k$ leaves) and to z ($A_j$ enters, $A_q$ leaves), respectively. In both cases, $A_j$ enters the basis.

Let $B_x(r) = k$ and $B_x(s) = q$

$$\Rightarrow \quad \theta_0 = \frac{x_{r0}}{x_{rj}} = \frac{x_{s0}}{x_{sj}}$$

=> y = z, a contradiction

Claim => only convex combinations of x, y fulfill

$Aw = b$

$w \geq 0$

$c^T w \leq c^T x$

The transformation of S to P as in the proof of Theorem 3.10 yields:

Only points $w'$ in $[x', y']$ fulfill the inequality $d^T w' \leq d^T x'$ ($d'$ = transformed cost vector) in P

=> $[x', y']$ is the intersection of a halfspace with P

=> $[x', y']$ is an edge of P ❑

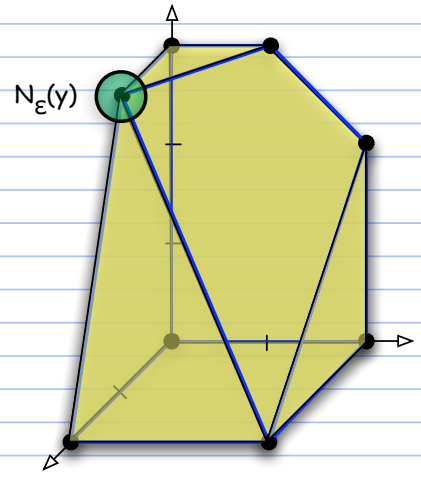
3.19 Remarks

(1) LP is a convex optimization problem. So the Euclidean neighborhood

$$N_\varepsilon(y) := \{ x \in S_I : || y-x || \leq \varepsilon \}$$

is exact because of Theorem 2.16.

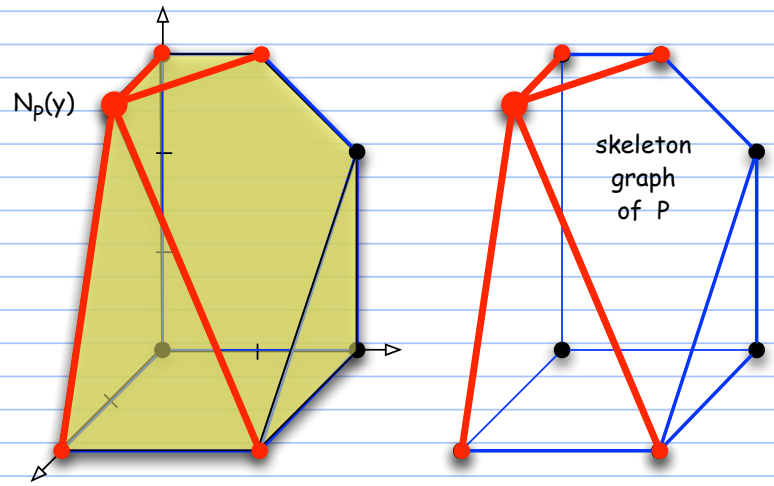





(2) The geometric interpretation of the simplex algorithm defines another neighborhood via adjacency of vertices:

$N_P(y) := \{ x \mid x$ vertex of P, x adjacent to $y \}$

This neighborhood is also exact because of Theorem 3.18, Theorem 3.10 and Theorem 3.14. It corresponds to the graph neighborhood in the skeleton graph of the polytopes.



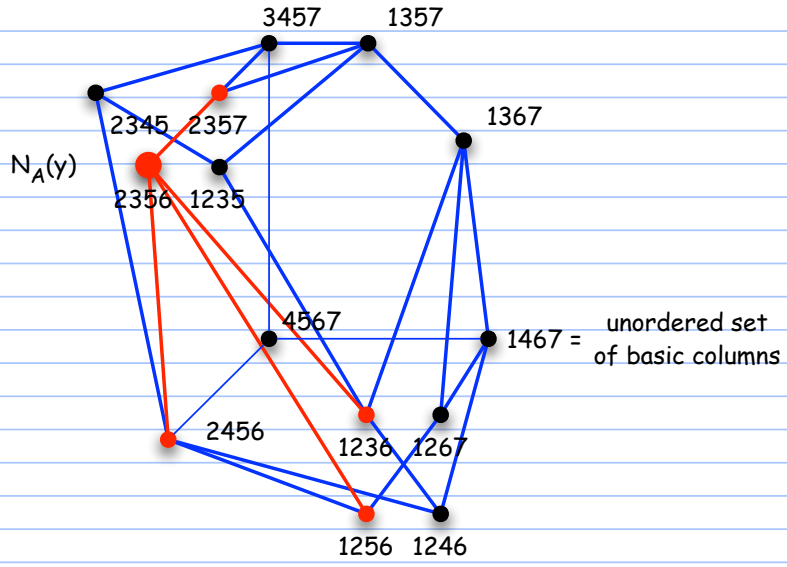

(3) The algebraic interpretation of the simplex algorithm defines a third neighborhood via adjacency of basic feasible solutions:

$N_A(y) := \{ x \mid x$ basic feasible solution, $x$ adjacent to $y \}$

This is also exact in the sense of Theorem 3.14 (no negative reduced cost) and corresponds also to a graph neighborhood. This graph $G$ results from the skeleton graph by refining every polytope vertex by a set of graph vertices (= all basic feasible solutions corresponding to the polytope vertex). $G$ coincides with the skeleton graph



iff P has no degenerate vertices.

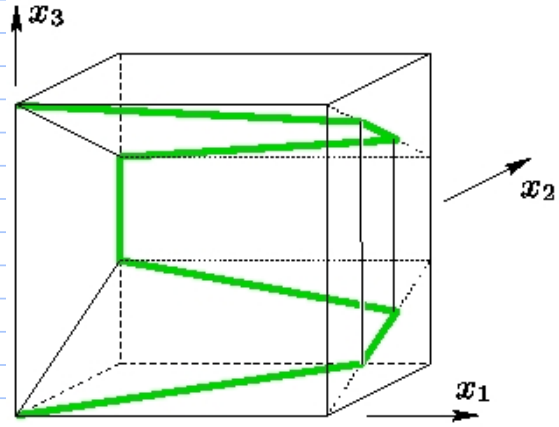



(4) So the simplex algorithm can be seen as a local search on the graph defined by neighborhood $N_A$. Checking for better neighbors can be done by checking the sign of the reduced costs and thus is very simple.

(5) It is not known if this local search is polynomial. There are counterexamples for all known pivot rules that

require exponentially many pivots.

- These counterexamples are in most cases so-called Klee-Minty cubes, i.e., slightly distorted cubes on which the simplex algorithm all visits all vertices, while it could reach the optimal vertex in one step.




@ - http://www.mathematik.de/ger/information/forschungsprojekte/zieglergeometrie/zieglergeometrie.html

- (6) Luckily, these counterexamples are not practically relevant. Empirically, the runtime of the simplex algorithm is linear in the number of rows.

## The dual of an LP in general form

### Derivation of the dual

Consider an LP in general form: (4.1)

$$\min c^T x \qquad x \in \mathbb{R}^n, c \in \mathbb{R}^n$$

$$\begin{array}{llll} \text{s.t.} & a_i^T x = b_i & i \in M & a_i \in \mathbb{R}^n \\ & a_i^T x \geq b_i & i \in \overline{M} & \\ & x_j \geq 0 & j \in N & \\ & x_j \text{ unconstrained} & j \in \overline{N} & \end{array}$$

we transform it to standard form according to Lemma 3.2 with

surplus variables $x_i^s$ for the inequalities

split variables $x_j = x_j^+ - x_j^-$ with $x_j^+, x_j^- \geq 0$

This gives



$$\left.\begin{array}{rcl} \min & \hat{c}^T \hat{x} & \\ \text{s.t.} & \hat{A}\hat{x} = b, \hat{x} \geq 0 \quad \text{with} & \\ \hat{A} & = & \left( A_j, j \in N \;\middle|\; (A_j, -A_j), j \in \overline{N} \;\middle|\; \begin{array}{l} 0, \quad i \in M \\ -I, \quad i \in \overline{M} \end{array} \right) \\ \hat{x} & = & \left( x_j, j \in N \mid (x_j^+, x_j^-), j \in \overline{N} \mid x_i^s, i \in \overline{M} \right)^T \\ \hat{c} & = & \left( c_j, j \in N \mid (c_j, -c_j), j \in \overline{N} \mid 0, i \in \overline{M} \right)^T \end{array}\right\} \quad (4.2)$$

where, w.o.l.g., matrix $\hat{A}$ has full row rank, and where $A_j$ denotes the column of $x_j$ in (4.1)

The previous results on the simplex algorithm give:

If (4.2) has an optimal solution, then there is a basis $\hat{B}$ of $\hat{A}$ with

$$\hat{c}^T - \underbrace{(\hat{c}_{\hat{B}}^T \hat{B}^{-1})}_{=:\pi^T} \hat{A} \leq 0$$

i.e., reduced cost $\leq 0$

Let m be the number of constraints in (4.1). Then

$$\pi^T := \hat{c}_{\hat{B}}^T \hat{B}^{-1} \in \mathbb{R}^m$$

is a feasible solution for inequalities

$$\pi^T \hat{A} \leq \hat{c}^T \qquad (4.3)$$

- Inequalities (4.3) have 3 groups w.r.t. their columns:
  - Group 1

    $\pi^T A_j \leq c_j, \quad j \in N$ (4.4)
  - Group 2

    $$\left.\begin{array}{rcl} \pi^T A_j & \leq & c_j \\ -\pi^T A_j & \leq & -c_j \end{array}\right\} \Leftrightarrow \pi^T A_j = c_j, \quad j \in \overline{N} \qquad (4.5)$$
  - Group 3

    $-\pi_i \leq 0 \Leftrightarrow \pi_i \geq 0, \quad i \in \overline{M}$ (4.6)

- Definition of the dual of LP
  - (4.4) - (4.6) define constraints for a new LP with variables $\pi_1, \ldots, \pi_m$. These constraints, together with the



objective function $\max \pi^T b$ constitute the dual LP of (4.1). The initial problem (4.1) is called the primal LP.

- Transformation rules primal -> dual (follow from (4.4) - (4.6))

| primal | | dual |
|---|---|---|
| $\min\ c^T x$ | | $\max\ \pi^T b$ |
| $a_i^T x = b_i$ | $i \in M$ | $\pi_i$ unconstrained |
| $a_i^T x \geq b_i$ | $i \in \overline{M}$ | $\pi_i \geq 0$ |
| $x_j \geq 0$ | $j \in N$ | $\pi^T A_j \leq c_j$ |
| $x_j$ unconstrained | $j \in \overline{N}$ | $\pi^T A_j = c_j$ |

- Observe: The dual LP is obtained from the optimality criterion of the primal. The variables $\pi_1, \ldots, \pi_m$ correspond to multipliers of the rows of $\hat{A}$ that fulfill the primal optimality criterion.

- 4.1 Theorem (dual dual = primal)
  - The dual of the dual is the primal.
  - We therefore speak of primal-dual pairs of LPs
  - Proof

Write the dual in primal form:

$$\begin{array}{llll} \min\ \pi^T(-b) & & \text{such that} & \\ (-A_j^T)\pi & \geq & -c_i & j \in N \\ (-A_j^T)\pi & = & -c_i & j \in \overline{N} \\ \pi_i & \geq & 0 & j \in \overline{M} \\ \pi_i & & \text{unconstrained} & j \in M \end{array}$$

The transformation rules yield the following dual LP

$$\begin{array}{llll} \max\ x^T(-c) & & \text{such that} & \\ x_j & \geq & 0 & j \in N \\ x_j & & \text{unconstrained} & j \in \overline{N} \\ -a_i^T x & \leq & -b_i & i \in \overline{M} \\ -a_i^T x & = & -b_i & i \in M \end{array}$$

which is the primal LP ❑

**The Duality Theorem**

4.2 Theorem (Weak and Strong Duality Theorem)



Let x be a primal feasible solution and π be a dual feasible solution. Then (Weak Duality Theorem)

$c^T x \geq \pi^T b$ (4.7)

If an LP has an optimal solution, so has its dual, and the optimal objective values are the same (Strong Duality Theorem)

Proof

Let x be a primal feasible solution and π be a dual feasible solution. Then

$$c^T x \overset{\pi \text{ dual feasible}}{\geq} (\pi^T A)x = \pi^T(Ax) \overset{x \text{ primal feasible}}{\geq} \pi^T b$$

Assume w.o.l.g. that the LP is in primal form (4.2) and has an optimal solution

=> has an basic optimal feasible solution $\hat{x}$ with associated basis $\hat{B}$ and $\pi^T = \hat{c}_{\hat{B}}^T \hat{B}^{-1}$ is feasible for the dual by construction

For this π we obtain

$$\pi^T b = (\hat{c}_{\hat{B}}^T \hat{B}^{-1})b = \hat{c}_{\hat{B}}^T(\hat{B}^{-1}b) = \hat{c}_{\hat{B}}^T \hat{x}_B = \hat{c}^T \hat{x}$$

So π and $\hat{x}$ have the same objective function value.

Weak Duality (4.7) then implies that π is a dual optimal solution ❑

4.3 Theorem (Possible primal-dual pairs)

Primal-dual pairs exist exactly in one of the following cases:

(1) both LPs have a finite optimal solution and their objective values are equal

(2) both LPs have no feasible solution

(3) one LP has an unbounded objective function and the other has no feasible solution



| primal \ dual | finite optimal solution | feasible solution, unbounded objective | no feasible solution |
|---|---|---|---|
| finite optimal solution | (1) | | |
| feasible solution, unbounded objective | | | (3) |
| no feasible solution | | (3) | (2) |

Proof

Strong Duality Theorem => Case (1) occurs in row 1 and column 1 of the table, and this is the only table entry in which it occurs

Consider now row 2 of the table, i.e., $x$ is a primal feasible solution but $c^T x$ unbounded from below.

If there is a dual feasible solution $\pi$, we obtain $\pi^T b \leq c^T x$ with the Weak Duality Theorem

=> $c^T x$ is bounded from below, a contradiction.

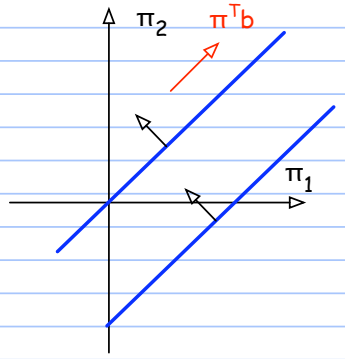
Therefore case (3) can only occur at positions (2,3) and (3,2)

An example for (3)

(P) min $x_1$ s.t. $x_1 + x_2 \geq 1$, $-x_1 - x_2 \geq 1$, $x_1, x_2 \geq 0$

=> (P) has no feasible solution

(D) max $\pi_1 + \pi_2$ s.t. $\pi_1 - \pi_2 \leq 1$, $\pi_1 - \pi_2 \leq 0$, $\pi_1, \pi_2 \geq 0$



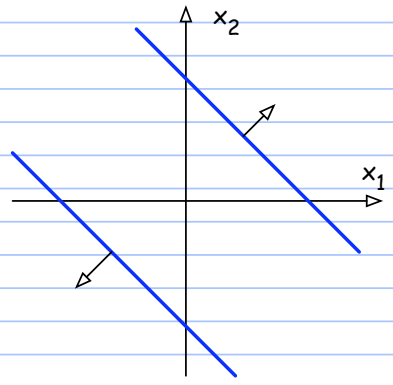

=> $\pi^T b$ is unbounded

So only entry (3,3) remains. This case can occur



An example for (2)

(P) min $x_1$ s.t. $x_1 + x_2 \geq 1$, $-x_1 - x_2 \geq 1$, $x_1, x_2$ unconstrained




=> (P) has no feasible solutions

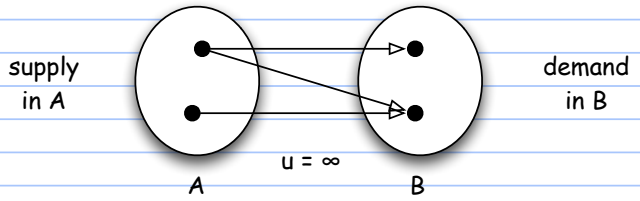
(D) max $\pi_1 + \pi_2$ s.t. $\pi_1 - \pi_2 = 1$, $\pi_1 - \pi_2 = 0$, $\pi_1, \pi_2 \geq 0$

=> (D) has no feasible solution ❑

**The transportation problem and its dual**

Hitchcock problem or transportation problem (Hitchcock 1941) is a special minimum cost flow problem, see ADM

I




G "bipartite"

We want to transport a good (oil, grain, coal) at minimum cost from the supply locations to the demand locations

Vertex $i \in A$ $(i = 1, \ldots, m)$ supplies $a_i$ units

Vertex $j \in B$ $(j = 1, \ldots, n)$ demands $b_j$ units, total supply = total demand.

Edges $(i,j) \in A \times B$ have cost $c_{ij}$ per transported unit and infinite capacity $u_{ij}$

An LP formulation for the transportation problem

$x_{ij}$ = number of units transported from $i$ to $j$

$\min \sum_{i,j} c_{ij}x_{ij}$ s.t.

$\sum_j x_{ij} = a_i$ for all $i$ (pick up supply $a_i$ from vertex $i$)



$\sum_i x_{ij} = b_j$ for all $j$ (deliver demand $b_j$ to vertex $j$)

$x_{ij} \geq 0$ for all $i, j$

The associated matrix $A$ of coefficients has the form

| | $x_{11}$ | $x_{12}$ | … | $x_{1n}$ | $x_{21}$ | $x_{22}$ | … | $x_{2n}$ | … | $x_{m1}$ | $x_{m2}$ | … | $x_{mn}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $i = 1,\ldots,m$ | 1 | 1 | … | 1 | 0 | 0 | … | 0 | … | 0 | 0 | … | 0 |
| | 0 | 0 | … | 0 | 1 | 1 | … | 1 | … | 0 | 0 | … | 0 |
| | | | ⋱ | | | | ⋱ | | ⋱ | | | ⋱ | |
| | 0 | 0 | … | 0 | 0 | 0 | … | 0 | … | 1 | 1 | … | 1 |
| $j = 1,\ldots,n$ | 1 | 0 | … | 0 | 1 | 0 | … | 0 | … | 1 | 0 | … | 0 |
| | 0 | 1 | … | 0 | 0 | 1 | … | 0 | … | 0 | 1 | … | 0 |
| | | | ⋱ | | | | ⋱ | | ⋱ | | | ⋱ | |
| | 0 | 0 | … | 1 | 0 | 0 | … | 1 | … | 0 | 0 | … | 1 |

The dual of the transportation problem

Introduce dual variables $u_i$, $v_j$ for the constraints as follows

$u_i$ $-\sum_j x_{ij} = -a_i$ for all $i$

$v_j$ $\sum_i x_{ij} = b_j$ for all $j$

The dual LP reads

$\max \sum_i -a_i u_i + \sum_j b_j v_j$ s.t.

$-u_i + v_j \le c_{ij}$ for all $i, j$

$u_i, v_j$ unconstrained

Interpretation of the dual LP

"Dual" entrepreneur offers to do the transportation for pairs (i,j)

He can buy the supply $a_i$ at location $i$ from the primal entrepreneur, transport it to $j$ and sell it there

$u_i$ = price to buy a unit of the good at vertex $i$

$v_j$ = returns per unit at vertex $j$

$v_j - u_i$ = profit per unit bought in $i$ and sold in $j$

$v_j - u_i \le c_{ij}$ dual entrepreneur must stay below primal transportation cost in order to get the transport (i,j) from the primal entrepreneur (otherwise primal entrepreneur will do it himself)

Dual entrepreneur wants to maximize his total profit $\sum_j b_j v_j - \sum_i a_i u_i$ under these conditions

**The dual of the diet problem**

The primal problem (see example 3.1)

$\min c^T x$

s.t. $Ax \ge r$



$x \ge 0$

The associated dual problem

$\max \pi^T r$

s.t. $\pi^T A \le c^T$

$\pi^T \ge 0$

Interpretation

The dual entrepreneur makes nutrient pills for each of the $m$ ingredients (magnesium, vitamin C, ...)

He asks the price $\pi_i$ per unit of nutrient $i$

$\pi^T A_j \le c_j$ <=> the total price of all pills substituting one unit of food $j$ must not exceed the price $c_j$ of one unit of food $j$ (pills will not be bought otherwise)

$\max \pi^T r$ <=> maximizing total profit of the dual entrepreneur

Dual LPs often have a natural interpretation in practice

Complementary slackness provides simple necessary and sufficient conditions for optimality of a pair of primal feasible and dual feasible solutions. They have far reaching consequences for the design of algorithms (primal-dual algorithms, primal-dual approximation algorithms)

4.4 Theorem (Complementary slackness)

Let $x$ be a primal feasible solution and $\pi$ be a dual feasible solution. The following statements are equivalent:

$x$, $\pi$ are optimal (in the primal and the dual, respectively)

$u_i := \pi_i \cdot (a_i^T x - b_i) = 0$ for all $i = 1, \ldots, m$ (4.8)

$v_j := (c_j - \pi^T A_j) \cdot x_j = 0$ for all $j = 1, \ldots, n$ (4.9)

i.e.,: (slack of primal or dual constraint)·(value of associated dual or primal variable) = 0

Proof

$u_i \geq 0$

since

$a_i^T x - b_i = 0 \Rightarrow u_i = 0$

$a_i^T x - b_i \geq 0 \Rightarrow \pi_i \geq 0 \Rightarrow u_i \geq 0$

$v_i \geq 0$



since

$x_j$ unconstrained $\Rightarrow \pi^T A_j = c_j \Rightarrow v_j = 0$

$x_j \geq 0 \Rightarrow \pi^T A_j \leq c_j \Rightarrow v_j \geq 0$

Set $u := \sum_i u_i$, $v := \sum_j v_j \Rightarrow u, v \geq 0$. Then

$u = 0 \Leftrightarrow$ (4.8) holds

$v = 0 \Leftrightarrow$ (4.9) holds

Then

$$u + v = \sum_i \pi_i \cdot (a_i^T x - b_i) + \sum_j (c_j - \pi^T A_j) \cdot x_j$$
$$= -\sum_i \pi_i b_i + \sum_j c_j x_j + \sum_i \pi_i a_i^T x - \sum_j \pi^T A_j x_j$$
$$= -\pi^T b + c^T x + (\pi^T A) x - \pi^T (A x)$$
$$= -\pi^T b + c^T x$$

Hence: $u + v = -\pi^T b + c^T x$

Suppose (4.8) and (4.9) hold $\Rightarrow u + v = 0 \Rightarrow c^T x = \pi^T b$

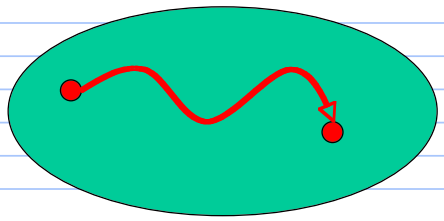
Weak Duality Theorem $\Rightarrow$ $x$, $\pi$ are optimal

Suppose that $x$ and $\pi$ are optimal

Strong Duality Theorem $\Rightarrow$ $c^T x = \pi^T b \Rightarrow u + v = 0 \Rightarrow$ (4.8) and (4.9) ❑

**The shortest path problem as primal LP**

- Shortest Path Problem (SP)
  - Instance
    - Digraph $G$
    - Rational edge weights $c(e)$, $e \in E(G)$
    - Vertices $s, t \in V(G)$
  - Task
    - Determine an elementary $s,t$-path $P$ of minimum weight $c(P)$ (shortest $s,t$-path)
      - $c(W) = \sum_{e \in E(W)} c(e) c(W) = \sum_{e \in E(P)} c(e)$



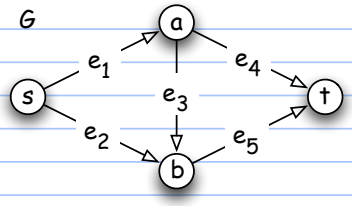
- (SP) is an instance of (LP)

The vertex-edge-incidence matrix $A = (a_{ij})$ of $G$ is defined as

$$a_{ij} := \begin{cases} +1 & \text{if } e_j \text{ enters } i \\ -1 & \text{if } e_j \text{ leaves } i \\ 0 & \text{otherwise} \end{cases}$$

where $V(G) = \{ 1, \ldots, n \}$ and $E(G) = \{ e_1, \ldots, e_m \}$

Example




A

| | $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ |
|---|---|---|---|---|---|
| s | 1 | 1 | 0 | 0 | 0 |
| t | 0 | 0 | 0 | −1 | −1 |
| a | −1 | 0 | 1 | 1 | 0 |
| b | 0 | −1 | −1 | 0 | 1 |

The vertex-edge-incidence matrix of a digraph has per column exactly one 1, exactly one -1, and 0 otherwise

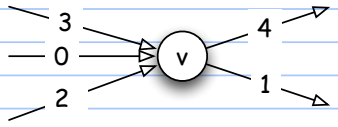
=> sum of rows is 0 => rank(A) < n



Later: rank(A) = n-1 if $G$ is connected (in the undirected sense)

Let $f_j$ be a variable representing the amount of flow on edge $e_j$, and let $f := (f_1, \ldots, f_m)^T$

Flow conservation in node $i$ is then expressed as $a_i^T f = 0$




inflow in v = 5 = outflow from v

An s,t-path is a flow of flow value 1 from $s$ to $t$ (all $f_j = 1$ on the path and 0 otherwise)

=> every s,t-path is a solution of the linear system

$$Af = b \text{ with } b = \begin{pmatrix} +v \\ -v \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{matrix} \text{row } s \\ \text{row } t \\ \\ \\ \text{flow conservation} \end{matrix}$$

with $v = 1$

Of course, this linear system has also solutions that do not correspond to s,t-paths. But we have

4.6 Lemma

(1) If

$$\min\ c^T f$$
$$Af = b$$
$$f \geq 0$$

has an optimal solution, then also one with $f_j \in \{0, 1\}$. Every such solution corresponds to an s,t-path

(2) The simplex algorithm finds such a solution

Proof:

(1) follows from the algorithm for minimum cost s,t-flows in ADM I

(2) can easily be shown directly, but follows also from the fact that matrix $A$ is totally unimodular and $b$ is integer. Then all basic feasible solutions of the LP are integer. We will show this more general result in Chapter 7.2. ❑

**Solving (SP) with the simplex algorithm**

We formulate (SP) as (LP)



$$\min\ c^T f$$
$$Af = b \quad (A = \text{vertex-edge-incidence matrix})$$
$$f \geq 0$$

and solve it with the simplex algorithm.

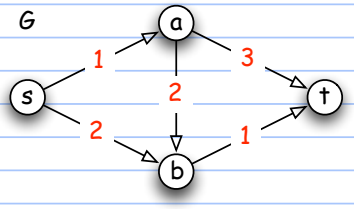
Since rank(A) < n, we may delete a row

=> delete the row for vertex t, this yields $b \geq 0$

In the example we obtain the following tableau for cost vector $c = (1, 2, 2, 3, 1)$

Initial tableau, not yet transformed w.r.t. a basis, and graph $G$ with edge costs

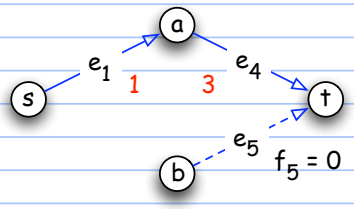
| | | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 2 | 3 | 1 |
| s | 1 | 1 | 1 | 0 | 0 | 0 |
| a | 0 | -1 | 0 | 1 | 1 | 0 |
| b | 0 | 0 | -1 | -1 | 0 | 1 |




Choose {1, 4, 5} as basis and transform the tableau w.r.t. that basis.

Interpret the associated basic feasible solution in the graph.

| | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ |
|---|---|---|---|---|---|
| | -4 | 0 | -1 | 0 | 0 | 0 |
| $f_1$ | 1 | 1 | 1 | 0 | 0 | 0 |
| $f_4$ | 1 | 0 | 1 | 1 | 1 | 0 |
| $f_5$ | 0 | 0 | -1 | -1 | 0 | 1 |

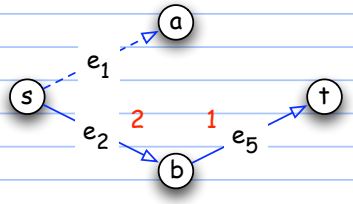



The basic solution has n-1 = |V| - 1 variables, but not every s,t-path has so many edges

=> many basic feasible solutions are degenerate (a common phenomenon in combinatorial optimization problem)

Next tableau and basic feasible solution in the graph

| | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ |
|---|---|---|---|---|---|
| | -3 | 0 | 0 | 1 | 1 | 0 |
| $f_1$ | 0 | 1 | 0 | -1 | -1 | 0 |
| $f_2$ | 1 | 0 | 1 | 1 | 1 | 0 |
| $f_5$ | 1 | 0 | 0 | 0 | 1 | 1 |




=> optimal solution found, shortest path has length 3

**The dual of the shortest path problem**



We formulate it w.r.t. the full tableau containing also the row for vertex t

=> dual variables $\pi_i$ correspond to a node potential in graph $G$

Tableau in the example:

| | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ | b |
|---|---|---|---|---|---|---|
| c | 1 | 2 | 2 | 3 | 1 | |
| $\pi_s$ | 1 | 1 | 0 | 0 | 0 | +1 |
| $\pi_t$ | 0 | 0 | 0 | -1 | -1 | -1 |
| $\pi_a$ | -1 | 0 | 1 | 1 | 0 | 0 |
| $\pi_b$ | 0 | -1 | -1 | 0 | 1 | 0 |

Dual LP:

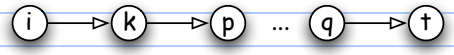
$\max \pi_s - \pi_t$

$\pi_i - \pi_j \le c_{ij}$ for all edges $(i, j) \in E(G)$

$\pi_i$ unconstrained

Interpretation of the dual LP

Along any path




from i to t we have

$(\pi_i - \pi_k) + (\pi_k - \pi_p) + \ldots + (\pi_q - \pi_t) = \pi_i - \pi_t$

$\leq c_{ik} \qquad \leq c_{kp} \qquad \leq c_{qt}$

$\Rightarrow \pi_i - \pi_t \leq c_{ik} + c_{kp} + \ldots + c_{qt}$ = length of the path from i to t

Since this holds for every such path,

$\pi_i - \pi_t \leq$ length of a shortest path from i to t

$\Rightarrow \max \pi_s - \pi_t$ is equivalent to

finding the greatest lower bound for the length of a shortest path from s to t

**Complementary slackness conditions**

Path f and node potential $\pi$ are primal-dual optimal $\Leftrightarrow$

(1) $f_{ij} > 0 \Rightarrow \pi_i - \pi_j = c_{ij}$

i.e., edge (i,j) lies on a shortest path $\Rightarrow$ potential difference = cost

(2) $\pi_i - \pi_j < c_{ij} \Rightarrow f_{ij} = 0$



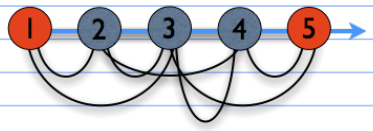
i.e., potential difference < cost $\Rightarrow$ edge (i,j) does not lie on a shortest path

Interpretation:

the lower bounds $\pi_i - \pi_t$ are tight along any shortest path

The cord model (for $c_{ij} \geq 0$)

edge (i, j) $\leftrightarrow$ cord with length $c_{ij}$

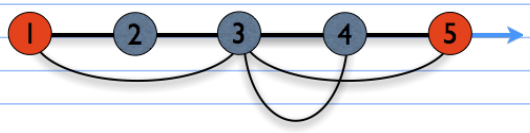



$\pi_i - \pi_j \leftrightarrow$ pulling vertices i and j apart

$\pi_i - \pi_j \leq c_{ij} \leftrightarrow$ pulling is bounded from above by length $c_{ij}$

$\max \pi_s - \pi_t \leftrightarrow$ pull s and t apart as far as possible

complementary slackness: the cords on shortest paths are the tight ones



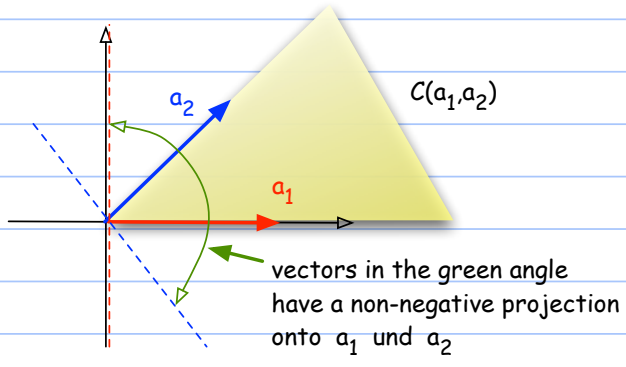
- Remarks
  - Deleting the row for vertex $t$ => we have no variable $\pi_t$ => the dual objective function is $\max \pi_s$
    But: edges $(i,t)$ yield the dual constraint $\pi_i \le c_{it}$, so that $\pi_s$ cannot get arbitrarily large.
    We obtains the same dual constraint $\pi_i \le c_{it}$ if we set $\pi_t = 0$ (which we may do w.o.l.g. since we only have potential differences in the dual).
  - Dijkstra's algorithm (ADM I) applied to the dual graph (in which the direction of all edges of $G$ is reversed) iteratively computes the $\pi_i$, where $\pi_t$ is set to 0.



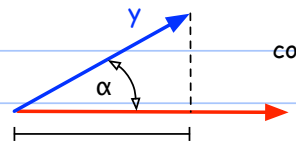
- This is a central and very useful lemma in duality theory. It has several variants also known as Theorems of the Alternative.
- Cones and projections
  - The cone $C(a_1, \ldots, a_m)$ generated by $a_1, \ldots, a_m$
    - Let $a_1, \ldots, a_m \in R^n$ (e.g. the rows of matrix $A$). The cone $C(a_1, \ldots, a_m)$ generated by $a_1, \ldots, a_m$ is defined as

$$C(a_1, \ldots, a_m) := \{ x \in \mathbb{R}^n \mid x = \sum_{i=1}^{m} \pi_i a_i,\ \pi_i \ge 0 \}$$

= set of non-negative linear combinations of $a_1, \ldots, a_n$



The projection of y onto a




$$\cos\alpha = \frac{y^T a}{\|y\| \cdot \|a\|}$$

projection of y onto a = $\|y\| \cos\alpha$

=> the projection of y onto a is non-negative <=> $y^T a$ is non-negative

**4.5 Theorem (Farkas' Lemma)**

Let $a_1, \ldots, a_m \in R^n$ and $c \in R^n$. The following are equivalent

(1) for all $y \in R^n$ : $y^T a_i \geq 0$ for all $i = 1,\ldots,m$ => $y^T c \geq 0$

i.e., for all y :

y has a non-negative projection onto each $a_i$

=> y has a non-negative projection onto c

(2) $c \in C(a_1, \ldots, a_m)$

i.e., c lies in the cone generated by $a_1, \ldots, a_m$



Proof

(1) => (2)

Consider the LP

min $c^T y$

$a_i^T y \geq 0 \quad i = 1,\ldots,m$

y unconstrained

=> y = 0 is a feasible solution of the LP

The objective function is bounded from below since the constraints of the LP imply $c^T y \geq 0$ because of (1),.

=> LP has a finite optimal solution

=> the dual LP

max 0

$\pi^T A_j = c_j$

$\pi \geq 0$

has a feasible solution

=> there are numbers $\pi_1, \ldots, \pi_m \geq 0$ with $c = \pi^T A = \sum_i \pi_i a_i$

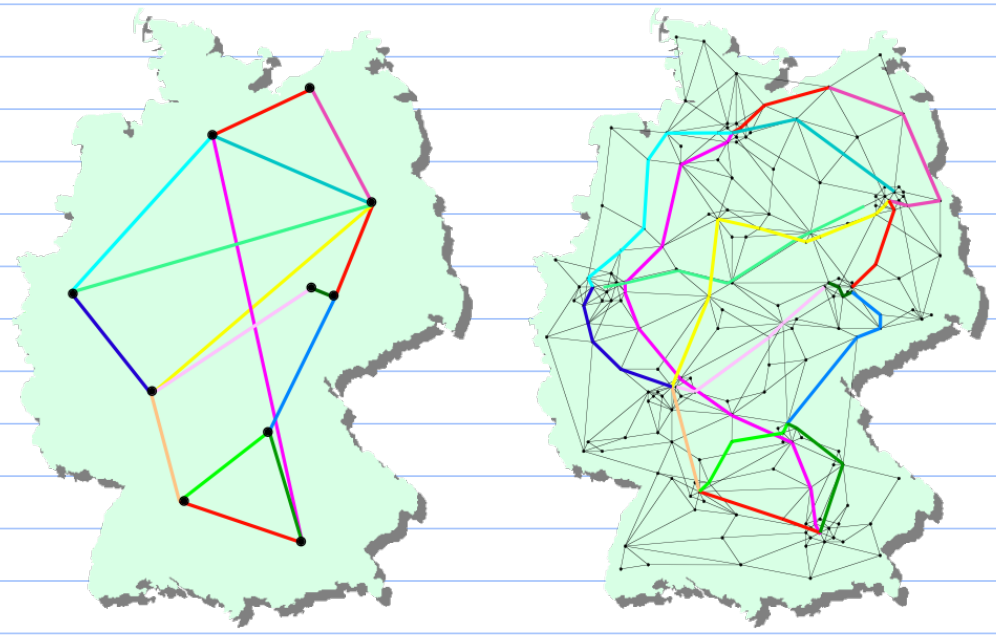
=> $c \in C(a_1, \ldots, a_m)$

- (2) => (1)
  - $c \in C(a_1, \ldots, a_m)$ => there are numbers $\pi_i \geq 0$ with $c = \sum_i \pi_i a_i$
    consider $y$ with $y^T a_i \geq 0$ for all $i = 1,\ldots m$
    => $y^T c = \sum_i \pi_i y^T a_i \geq \sum_i \pi_i \cdot 0 = 0$ ❑
- There are many equivalent formulations of Farkas' Lemma. Examples are
  - (A) $\forall y \; (y^T a_i \geq 0 \; \forall i \Rightarrow y^T b \geq 0) \Leftrightarrow \exists \; x \geq 0$ with $A^T x = b$ (original version by Farkas 1894)
  - (B) $\forall y \geq 0 \; (y^T a_i \geq 0 \; \forall i \Rightarrow y^T b \geq 0) \Leftrightarrow \exists \; x \geq 0$ with $A^T x \leq b$
  - More in Chapter 7.5

- **An application of Farkas' Lemma: necessary conditions for the disjoint path problem**
  - Disjoint Path Problem
    - Instance
      - Undirected graph $G$
      - Pairs of vertices $\{ s_1, t_1 \}, \ldots, \{ s_k, t_k \}$
    - Task
      - Determine pairwise edge disjoint paths from $s_i$ to $t_i$ $(i = 1, \ldots, k)$



- An example: minimum cost embeddings of VPNs into the base net of Telekom



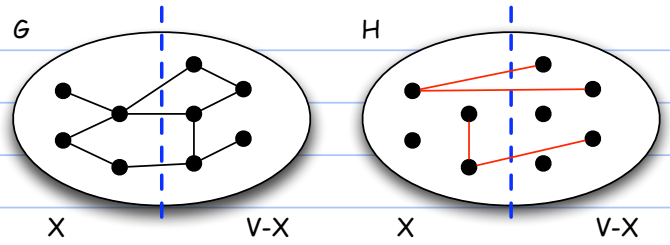
The decision version of the disjoint path problem is NP-complete. We therefore look for strong necessary and hopefully also sufficient criteria for the existence of a solution.

Cut criterion

Let $H$ be the graph with $V(H) := V(G)$ and $E(H) := \{ \{ s_1, t_1 \}, \ldots, \{ s_k, t_k \} \}$. A necessary condition for the existence of a solution is the cut criterion

$$|\delta_G(X)| \geq |\delta_H(X)| \text{ for all } \emptyset \neq X \subseteq V(G)$$

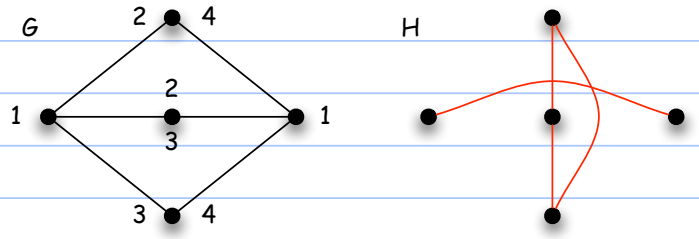
i.e., there are at least as many edges leaving $X$ in $G$ as there are pairs in $H$ to be connected




The cut criterion is not sufficient

4.6 Example






Cut criterion holds, but there is no solution

Distance criterion

Let $\text{dist}_{G,z}(s,t)$ be the length of a shortest path from $s$ to $t$ in $G$ w.r.t. edge weights $z(e) \geq 0$, $e \in E(G)$.

An instance of the disjoint path problem fulfills the distance criterion

:<=> for any choice of edge weights $z(e) \geq 0$, $e \in E(G)$,

$$\sum_{\{s,t\} \in E(H)} \text{dist}_{G,z}(s,t) \leq \sum_{e \in E(G)} z(e)$$

The cut criterion reduces to the distance criterion for edge weights

$$z(e) := \begin{cases} 1 & \text{if } e \in \delta(X) \\ 0 & \text{otherwise} \end{cases}$$

4.7 Theorem (The distance criterion is necessary)

The distance criterion is necessary and sufficient for the existence of a fractional solution of the disjoint path problem.

In particular, it is necessary for the existence of a solution of a disjoint path problem

Proof

Consider the disjoint path problem as a cycle packing problem

cycles = all elementary cycles in $G + H$ that contain exactly one edge of $H$

k := number of these cycles

integer cycle packing = union of pairwise edge disjoint cycles that contain every edge of $H$ in exactly one cycle

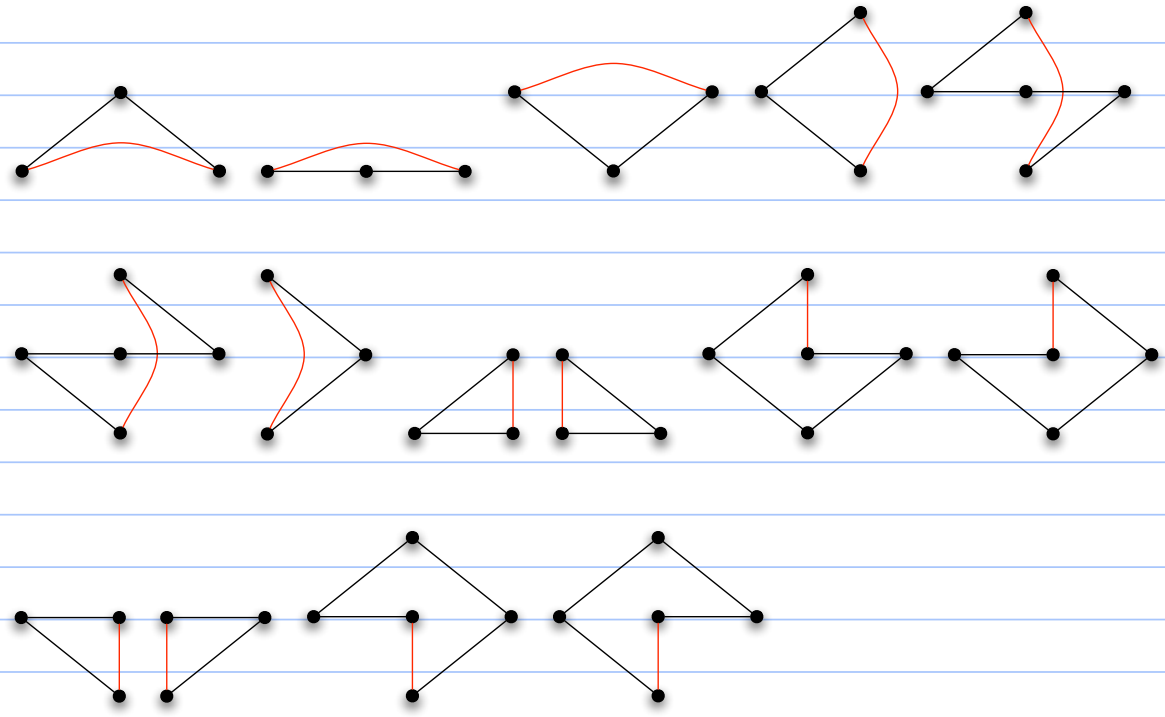
(existence <=> feasibility of the disjoint path problem)

fractional cycle packing = non-negative linear combination (of incidence vectors) of all these cycles such that the resulting vector has the value 1 at the entries corresponding to the edges of $H$, and is at most 1 at every entry corresponding to an edge of $G$.

(they contain integer cycle packings as special case)

Example 4.6 has the following cycles in the cycle packing problem





A formulation of the fractional cycle packing

Let $M$ be the $E(G)$-cycle-incidence matrix, i.e.,

rows of $M$ <-> edges of $G$

columns of $M$ <-> incidence vectors of all cycles of $G + H$

$M_{e,C} = 1$ <=> $e$ lies on cycle $C$

Let $N$ be the $E(H)$-cycle-incidence matrix, i.e.,

rows of $N$ <-> edges of $H$

columns of $N$ <-> incidence vectors of all cycles of $G + H$

$N_{e,C} = 1$ <=> $e$ lies on cycle $C$

Observe: every column of $N$ contains exactly one 1

=> fractional cycle packing = $\pi' \in R^k$ with $\pi' \geq 0$, $M\pi' \leq \mathbf{1}$, $N\pi' = \mathbf{1}$

Add slack variables to obtain a linear system and denote the enlarged vector again by $\pi$

=> fractional cycle packing = $\pi \in R^{k+m}$ ($m = |E(G)|$) with $\pi \geq 0$, $M\pi = \mathbf{1}$, $N\pi = \mathbf{1}$

Write it as

$$A\pi = \mathbf{1},\ \pi \geq 0 \qquad \text{with } A = \left(\begin{array}{c|c} M & I \\ \hline N & \mathbf{0} \end{array}\right)$$

i.e., the all ones vector $\mathbf{1}$ lies in the cone $C(A_1, \ldots, A_{k+m})$ generated by the columns $A_j$ = of $A$



Applying Farkas' Lemma gives condition (3)

Farkas' Lemma yields: there is such a vector $\pi$

<=> for all $y \in R^{|E(G)|+|E(H)|}$ : $y^T A_j \geq 0$ for all $j = 1,\ldots,k+m$ => $y^T \mathbf{1} \geq 0$

Partition $y$ into $(z,v)^T$, such that $z$ corresponds to the rows of $M$ (edges of $G$) and $v$ to the rows of $N$ (edges of $H$).

We then get:

$y^T A_j \geq 0$ => $z_j \geq 0$ for columns $A_j$ of slack variables

$y^T A_j \geq 0$ => $z^T M_j + v^T N_j \geq 0$ for the other columns $A_j$

Let $C_j$ be the cycle of column $A_j$

=> $C_j$ decomposes into a path $P_j$ in $G$ and an edge $f$ from $H$

Then

$z^T M_j$ = length $z(P_j)$ of the path $P_j$ w.r.t. edge weights $z(e)$

$v^T N_j$ = edge weight $v(f)$, where $f$ is the edge of $H$ lying on cycle $C_j$

Hence

$y^T A_j \geq 0$ => $z(P_j) + v(f) \geq 0$ for all cycles $C_j$ containing edge $f$

So $z(P_j) + v(f) \geq 0$ is equivalent to

$dist_{G,z}(s,t) + v(f) \geq 0$ with $f = \{ s, t \}$ (1)

The constraint $y^T \mathbf{1} \geq 0$ becomes

$\sum_{e \in E(G)} z(e) + \sum_{e \in E(H)} v(e) \geq 0$ (2)

Farkas' Lemma then yields for arbitrary (z,v) (3)

$z(e) \geq 0$, $dist_{G,z}(s,t) + v(f) \geq 0$ for all edges $f = \{ s, t \}$ in H

$\Rightarrow \sum_{e \in E(G)} z(e) + \sum_{f \in E(H)} v(f) \geq 0$

Condition (3) is equivalent to the distance criterion is (by proving that their negations are equivalent)

(3) violated => distance criterion violated

(3) violated => there are z, v with

$z(e) \geq 0$,

$dist_{G,z}(s,t) + v(f) \geq 0$ for all edges $f = \{ s, t \}$ in H and

$\sum_{e \in E(G)} z(e) + \sum_{f \in E(H)} v(f) < 0$

$\Rightarrow 0 \leq \sum_{f \in E(H)} dist_{G,z}(s,t) + \sum_{f \in E(H)} v(f) < \sum_{f \in E(H)} dist_{G,z}(s,t) - \sum_{e \in E(G)} z(e)$

$\Rightarrow \sum_{e \in E(G)} z(e) < \sum_{f \in E(H)} dist_{G,z}(s,t)$

=> distance criterion violated

distance criterion violated => (3) violated



distance criterion violated

$\Rightarrow$ there is $z \geq 0$ with $\sum_{e \in E(G)} z(e) < \sum_{f \in E(H)} dist_{G,z}(s,t)$

choose $v(f) := - dist_{G,z}(s,t)$ for edge $f = \{s, t\}$ in H

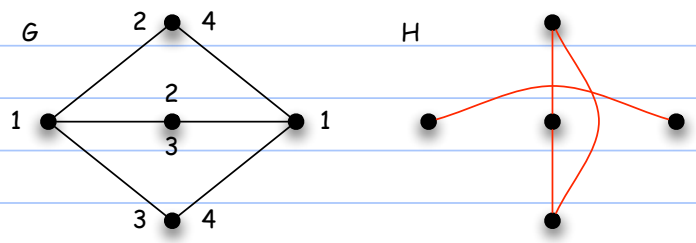
$\Rightarrow dist_{G,z}(s,t) + v(f) \geq 0$ for all edges $f = \{ s, t \}$ in H and

$\sum_{e \in E(G)} z(e) + \sum_{f \in E(H)} v(f) = \sum_{e \in E(G)} z(e) - \sum_{f \in E(H)} dist_{G,z}(s,t) < 0$

=> (3) is violated ❑

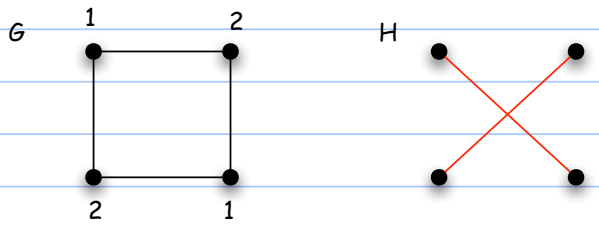
The distance criterion is stronger than the cut criterion

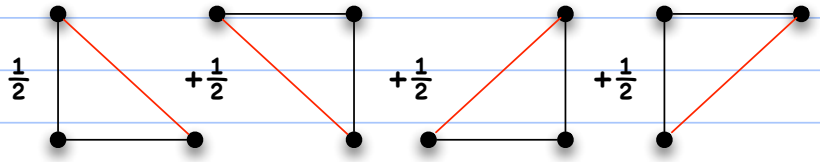
Example 4.6 does not fulfill the distance criterion




Set $z(e) = 1$ for all e in G $\Rightarrow \sum_{f = \{s,t\} \in E(H)} dist_{G,z}(s,t) = 8$, $\sum_{e \in E(G)} z(e) = 6$

## 4.4 Farkas' Lemma

- The distance criterion is not sufficient for the existence of a solution of the disjoint path problem
  - The instance of the disjoint path problem
    - 

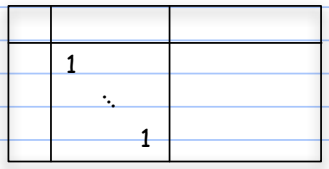
  - A fractional cycle packing
    - 

    - So the distance criterion holds because of Theorem 4.7
  - There is no solution for the disjoint path problem



## 4.5 Dual information in the tableau

- How to get dual information from the optimal primal tableau?
  - Suppose w.o.l.g. that the initial tableau (possibly with artificial variables from Phase I) has columns 1,...,m as basic columns and that the tableau is transformed w.r.t. to this basis




- Then the following properties hold in the optimal tableau with basis B
  - rows 1,...,m are obtained from the initial tableau by multiplying it with $B^{-1}$ from the left
  - the reduced cost are obtained as

$$\bar{c}_j = c_j - \pi^T A_j \geq 0 \qquad (4.10)$$

where $\pi$ is an optimal solution of the dual problem (Proof of the Strong Duality Theorem)

- In columns 1,...,m (which are unit vectors in the initial tableau) we get

$$\bar{c}_j = c_j - \pi^T A_j = c_j - \pi_j \qquad (4.11)$$

Hence an optimal dual solution is obtained from the optimal tableau of the primal as

$\pi_j = c_j - \bar{c}_j$ (j = 1,. . . ,m) (4.12)

Observe: this holds for the dual problem of the initial tableau (and not for dual versions of other, equivalent primal formulations).

Moreover, the first m columns contain $B^{-1} = B^{-1} I$ (4.13)

| | $c_j - \pi_j$ | |
|---|---|---|
| | $B^{-1}$ | |

4.8 Example (Example for the Two-Phase-Method continued)

Initial tableau



| | | $x_1^a$ | $x_2^a$ | $x_3^a$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|---|---|---|---|
| -z | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| -ξ | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| $x_1^a$ | 1 | 1 | 0 | 0 | 3 | 2 | 1 | 0 | 0 |
| $x_2^a$ | 3 | 0 | 1 | 0 | 5 | 1 | 1 | 1 | 0 |
| $x_3^a$ | 4 | 0 | 0 | 1 | 2 | 2 | 1 | 0 | 1 |

Optimal tableau

| | | $x_1^a$ | $x_2^a$ | $x_3^a$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|---|---|---|---|
| -z | -9/2 | 5/2 | -1 | -1 | 3/2 | 0 | 3/2 | 0 | 0 |
| -ξ | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| $x_2$ | 1/2 | 1/2 | 0 | 0 | 3/2 | 1 | 1/2 | 0 | 0 |
| $x_4$ | 5/2 | -1/2 | 1 | 0 | 7/2 | 0 | 1/2 | 1 | 0 |
| $x_5$ | 3/2 | -5/2 | 0 | 1 | -11/2 | 0 | -1/2 | 0 | 1 |

(4.12) gives $\pi_1 = 0 - 5/2 = -5/2$

$\pi_2 = 0 - (-1) = 1$

$\pi_3 = 0 - (-1) = 1$

for the values of the dual variables w.r.t. the dual problem obtained from the primal formulation with artificial variables $x_i^a$

4.9 Example (Example for the shortest path problem continued)

Solving the primal problem

Initial tableau has no identity matrix, but 2 unit vectors

=> add one artificial variable in Phase I

| | | | $x^a$ | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ |
|---|---|---|---|---|---|---|---|---|
| | | $-\xi$ | | 1 | 0 | 0 | 0 | 0 | 0 |
| | | $-z$ | | 0 | 1 | 2 | 2 | 3 | 1 |
| s | $x^a$ | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| a | $f_4$ | 0 | 0 | -1 | 0 | 1 | 1 | 0 |
| b | $f_5$ | 0 | 0 | 0 | -1 | -1 | 0 | 1 |

Transform cost coefficients of $\xi$ and $z$ to reduced form (must become 0 for basic variables)



| | | | $x^a$ | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ |
|---|---|---|---|---|---|---|---|---|
| | | $-\xi$ | -1 | 0 | -1 | -1 | 0 | 0 | 0 |
| | | $-z$ | 0 | 0 | 4 | 3 | 0 | 0 | 0 |
| s | $x^a$ | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| a | $f_4$ | 0 | 0 | -1 | 0 | 1 | 1 | 0 |
| b | $f_5$ | 0 | 0 | 0 | -1 | -1 | 0 | 1 |

Pivot step

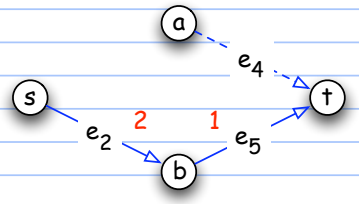
| | | | $x^a$ | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ |
|---|---|---|---|---|---|---|---|---|
| | | $-\xi$ | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | | $-z$ | -3 | -3 | 1 | 0 | 0 | 0 | 0 |
| s | $f_2$ | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| a | $f_4$ | 0 | 0 | -1 | 0 | 1 | 1 | 0 |
| b | $f_5$ | 1 | 1 | 1 | 0 | -1 | 0 | 1 |

=> $\xi = 0$ and $x^a$ is a non-basic variable

=> optimal w.r.t. z

basic columns of the initial tableau

Primal information (visualized in the graph)




the primal optimal solution displays the edges on the shortest path

Dual information (obtained from the primal optimal tableau and displayed in the graph)

$\pi_s = c_{x^a} - \bar{c}_{x^a} = 0 - (-3) = 3$

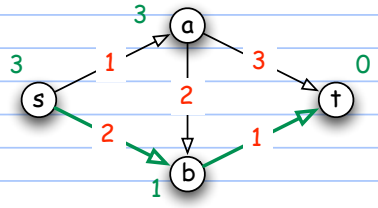
$\pi_a = c_4 - \bar{c}_4 = 3 - 0 = 3$

$\pi_b = c_5 - \bar{c}_5 = 1 - 0 = 1$

$\pi_t = 0$

$\pi_t = 0$ since row t is not in the primal LP






the dual solution displays the shortest distance from a vertex to t

- Goal: use the primal tableau to solve the dual LP

- Characteristics of the dual LP
  - The primal optimality condition $\bar{c} \geq 0$ becomes a dual constraint

    => the primal simplex algorithm has a primal feasible solution

    and fulfills the dual constraint $\bar{c} \geq 0$ only at termination when the optimum is reached

- This suggests the following characteristics for the dual simplex algorithm
  - generate a sequence of dual feasible solutions

    establish primal feasibility only at termination when the optimum is reached

- Deriving the operations in the tableau
  - Tableau X with basic solution

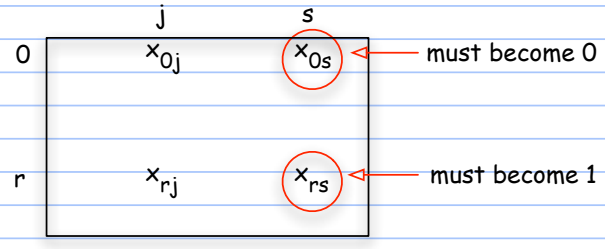


| | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ | |
|---|---|---|---|---|---|---|
| | −3 | 1 | 0 | 0 | 0 | 0 | => dual feasible, i.e., $\bar{c} \geq 0$ |
| $f_2$ | 1 | 1 | 1 | 0 | 0 | 0 | |
| $f_4$ | 1 | 0 | 0 | 0 | 1 | 1 | |
| $f_3$ | −1 | −1 | 0 | 1 | 0 | −1 | |

primal infeasible, i.e., $x_B \not\geq 0$

- Choose a pivot row $r$ (instead of a pivot column) with $x_{r0} < 0$ (i.e., an infeasible entry $x_{r0} < 0$ in the primal basic solution)
- Choose a pivot column in row $r$ the by considering entries $x_{rj} < 0$ (as to obtain $x_{r0} \geq 0$ after the pivot)
  - Pivoting with $x_{rs} < 0$ changes the cost row to

$$x'_{0j} = x_{0j} - \frac{x_{rj}}{x_{rs}} x_{0s} \qquad j = 1, \ldots, n$$




To stay feasible in the dual, $x_{0j}' \geq 0$ for all j

$$\Rightarrow \frac{x_{0j}}{x_{rj}} \leq \frac{x_{0s}}{x_{rs}} \quad \text{for } x_{rj} < 0$$

=> choose column s is such a way that

$$\frac{x_{0s}}{x_{rj}} = \max\{ \frac{x_{0j}}{x_{rj}} \mid x_{rj} < 0,\ j = 1, \ldots, n \}$$

Observe the symmetry with the primal simplex algorithm

in particular: all $x_{rj} > 0$ => dual LP has an unbounded objective function

4.10 Theorem (Interpretation of the dual simplex algorithm)



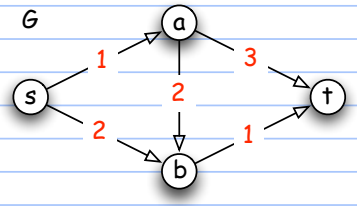
The dual simplex algorithm is the primal simplex algorithm applied to the primal formulation of the dual LP

Proof: Check ❑

4.11 Example (Example for the shortest path problem continued)

initial tableau, not yet transformed w.r.t. a basis and graph with costs

| | | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 2 | 3 | 1 |
| s | 1 | 1 | 1 | 0 | 0 | 0 |
| a | 0 | -1 | 0 | 1 | 1 | 0 |
| b | 0 | 0 | -1 | -1 | 0 | 1 |

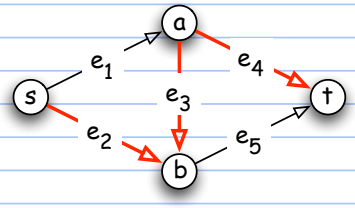



choose B = { 2, 4, 3 } as basis and transform the tableau w.r.t. B, display the basic solution in the graph

| | | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ |
|---|---|---|---|---|---|---|
| | -3 | 1 | 0 | 0 | 0 | 0 |
| $f_2$ | 1 | 1 | 1 | 0 | 0 | 0 |
| $f_4$ | 1 | 0 | 0 | 0 | 1 | 1 |
| $f_3$ | -1 | -1 | 0 | 1 | 0 | -1 |

=> dual feasible

primal infeasible




the basic solution corresponds to the s,t-cut $X = \{ s, a \}$

ADM I: s,t-cuts are "dual structures" of s,t-flows. This is confirmed here by LP duality

Choosing the pivot element

$r = 3$ is the pivot row

choosing the pivot column:

$$\frac{x_{0s}}{x_{rj}} = \max\left\{ \frac{x_{0j}}{x_{rj}} \mid x_{rj} < 0,\ j = 1, \ldots, n \right\} = \max\left\{ \frac{1}{-1}, \frac{0}{-1} \right\} = 0$$

$j = 1$ $\qquad$ $j = 5$

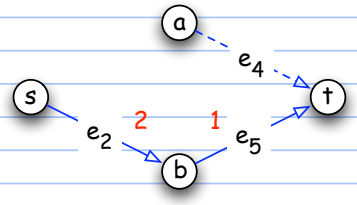
=> $s = 5$ is the pivot column



| | | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ |
|---|---|---|---|---|---|---|
| | -3 | 1 | 0 | 0 | 0 | 0 |
| $f_2$ | 1 | 1 | 1 | 0 | 0 | 0 |
| $f_4$ | 1 | 0 | 0 | 0 | 1 | 1 |
| $f_3$ | -1 | -1 | 0 | 1 | 0 | -1 |

pivot operation

| | | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ |
|---|---|---|---|---|---|---|
| | -3 | 1 | 0 | 0 | 0 | 0 |
| $f_2$ | 1 | 1 | 1 | 0 | 0 | 0 |
| $f_4$ | 0 | -1 | 0 | 1 | 1 | 0 |
| $f_5$ | 1 | 1 | 0 | -1 | 0 | 1 |




=> primal and dual feasible => optimal

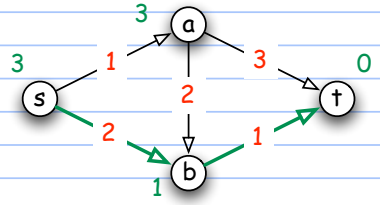
The dual optimal solution can be obtained from the inverse of the optimal basis as $\pi^T = c_B^T B^{-1}$ (Duality Theorem).

The optimal basis is $B = \{ 2, 4, 5 \}$ with inverse

$$B^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

So

$$\pi^T = c_B^T B^{-1} = (2,3,1) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = (3,3,1)$$




Observe: in this case we could not obtain $\pi$ and $B^{-1}$ directly from the optimal tableau, since the dual LP is not the one constructed from the initial tableau with basis { 2, 4, 3 }, but the dual LP of Example 4.9.

## 5. Computational aspects of the Simplex algorithm 28

- The full tableau is redundant (identity matrix!) and may contain very many non-basic columns (all of them are stored at every pivot step, => much memory for every tableau). The revised simplex algorithm uses in every step only essential information in a memory-efficient way.
  - The core of that information is the inverse $B^{-1}$ of the current basis B, from which we can easily compute all information needed for a pivot step
  - The revised simplex algorithm is used in all commercial LP-codes

- Main idea of the revised simplex algorithm: the CARRY matrix
  - consider the initial tableau with identity matrix = initial basis on the left in the tableau

| -z | 0 .... 0 | $\bar{c}_j$ |
|---|---|---|
| b | I | |

$CARRY^{(0)}$

because of (4.13), matrix I will change to the inverse $B^{-1}$ of the current basis B in subsequent pivot steps



- after iteration $\ell$ the tableau contains the following data in den first m+1 columns

| -z´ | $-\pi^T$ | |
|---|---|---|
| b´ | $B^{-1}$ | |

$CARRY^{(\ell)}$

where

- $\pi^T$ = dual solution because of (4.12), in general infeasible.

  The numbers $\pi_i$ are also called simplex multipliers.
- $b' = B^{-1}b$ is the current right hand side = current primal solution
- $z' = c_B^T B^{-1} b$ is the current primal cost

- It suffices to maintain the following data for the simplex algorithm
  1. initial tableau

| | $c^T$ |
|---|---|
| b | A |

2. current CARRY-matrix $CARRY^{(\ell)}$

3. current basis by its column indices $B(1),\ldots,B(m)$

From 1., 2., and 3., one can obtain all data required for a pivot step

(1) Pricing Operation (computing reduced costs)

iteratively compute

$$\bar{c}_j = c_j - \pi^T A_j$$

for non-basic variables until some reduced cost $\bar{c}_j < 0$ or $\bar{c} \geq 0$ (=> termination with an optimal solution)

(2) Generation of the Pivot Column (transforming the pivot column w.r.t. the current basis)

compute

$$X_s = B^{-1} A_s$$



= column s of the current tableau X

the pivot element $x_{rs}$ is obtained as

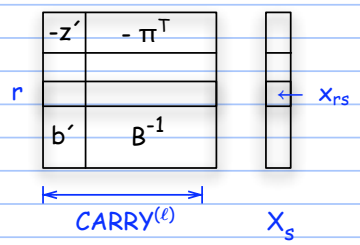
$$\min_{i,\, x_{is} > 0} \frac{b'_i}{x_{is}} \quad \leftarrow \text{ in } CARRY^{(\ell)} \quad \leftarrow \text{ in } X_s$$

or z is unbounded is (if all $x_{is} \leq 0$)

(3) Pivot Operation (pivot step)

compute $CARRY^{(\ell+1)}$

i.e., transform $X_s$ into the unit vector (with 1 at the pivot element) and apply the corresponding row operations to $CARRY^{(\ell)}$




(4) Basis Update

set B(r) := s

The Two-Phase-Method works similarly

one starts with the artificial cost vector

(1, ..., 1, 0, ..., 0)

artificial variables

$x_1^a, \ldots, x_m^a$ form the initial basis

Transformation to reduced cost w.r.t. this basis changes the cost to

$$d_j := -\sum_{i=1}^{m} a_{ij}$$

for each non-basic variable

At the end of Phase I we change over to the original cost vector

=> compute $-\pi^T = -c_B^T B^{-1}$ and put it into $CARRY^{(\ell)}$

compute $-z = -c_B^T b'$ and put it into $CARRY^{(\ell)}$

The required data is available: $c_B$ is stored in the initial data, B is stored, and $B^{-1}$ and $b'$ are stored in



$CARRY^{(\ell)}$

(1) Do not look at every non-basic columns per iteration

one needs all reduced costs

$$\bar{c}_j = c_j - \pi^T A_j$$

only to prove optimality. At other steps, partial pricing is enough (in the primal simplex; it is not possible in the dual simplex).

(2) Columns $A_j$ of non-basic variables come from the initial tableau

this is often sparse (in particular with combinatorial problems, e.g. there are only 2 entries $\neq 0$ in a vertex-edge-incidence matrix)

=> can exploit techniques to save memory usage and runtime (data structures and algorithms for sparse matrices)

(3) Maintain $CARRY^{(\ell)}$ implicitly

since $CARRY^{(\ell)}$ is obtained in a simple way from $CARRY^{(\ell-1)}$, it is not necessary to store the complete matrix $CARRY^{(\ell)}$.

$CARRY^{(\ell)} = P_\ell \cdot CARRY^{(\ell-1)}$



with

$P_\ell = (e_1, \ldots, e_{r-1}, \eta, e_{r+1}, \ldots, e_m)$ models elementary row operations

$e_i$ = i-th unit vector

$$\eta = \begin{pmatrix} -\frac{x_{1s}}{x_{rs}} \\ \vdots \\ \frac{1}{x_{rs}} \\ \vdots \\ -\frac{x_{ms}}{x_{rs}} \end{pmatrix} \leftarrow \text{pivot row}$$

=> store $P_\ell$ by its $\eta$-vector and position r

inductively

$$CARRY^{(\ell)} = P_{\ell-1} \cdot P_{\ell-2} \cdot \ldots \cdot P_1 \cdot CARRY^{(0)}$$

if $\ell$ gets large, one can "re-invert", i.e., the search for an equivalent but shorter sequence of $\eta$-vectors ($\ell \leq m$ suffice)

(4) Combine these techniques with methods for numerical stability

- LU-partition, Cholesky-factorization
  (Class on numerical methods)
  - Every regular matrix $B$ can be written as $B = P \cdot L \cdot U$ with
    $P$ = permutation matrix (then $P^{-1} = P^T$)
    $L$ = lower triangular matrix
    $U$ = upper triangular matrix
  - Linear systems $Bx = b$ can then be easily solved:
    $Bx = b$ transforms to $LUx = P^T b$
    solve $Ly = P^T b$
    solve $Ux = y$
  - i.e., solve 2 linear systems in triangular form

- Some quotations by Bob Bixby, the "father" of Cplex and Gurobi



Bixby
ISMP 2003

- from Solving Real-World Linear Programs: A Decade and More of Progress, Operations Research (50) 2002, 3-15
  - It was thus around 1987 that I became seriously involved in the computational aspects of linear programming.
    The first version of CPLEX, CPLEX 1.0, was released in 1988.
  - Advances in computing machinery

Table 1: **Machine improvements–Simplex algorithms**

| Old machine/processor | New machine/processor | Estimated speedup |
|---|---|---|
| Sun 3/50 | Compaq Server ES40, 667 MHz | 900 |
| Sun 3/50 | Pentium 4, 1.7 GHz | 800 |
| 25 MHz Intel 386 | Compaq Server ES40, 667 MHz | 400 |
| IBM 3090/108S | Compaq Server ES40, 667 MHz | 45 |
| Cray X-MP/416 | Compaq Server ES40, 667 MHz | 10 |

Table 2: **Machine improvements–Barrier algorithms**

| Old machine/processor | New machine/processor | Estimated speedup |
|---|---|---|
| Sun 3/50 | Pentium 4, 1.7 GHz | 13000 |
| Sun 3/50 | Compaq Server ES40, 667 MHz | 12000 |
| 33 MHz Intel 386 | Compaq Server ES40, 667 MHz | 4000 |
| IBM 3090/108S | Compaq Server ES40, 667 MHz | 10 |
| Cray X-MP/416 | Compaq Server ES40, 667 MHz | 5 |

Algorithmic improvements

The dual simplex algorithm with steepest edge.

The dual simplex algorithm was introduced by Lemke [1954]. It is not a new algorithm. However, to my knowledge, commercial implementations of this algorithm were not available in 1987 as full-fledged alternatives to the primal simplex algorithm. [...]



All that has changed. The dual simplex algorithm is now a standard alternative in modern codes. Indeed, computational tests, some of which will be presented later in this paper, indicate that the overall performance of the dual algorithm may be superior to that of the primal algorithm.

There are a number of reasons why implementations of the dual simplex algorithm have become so powerful. The most important is an idea introduced by Goldfarb and Forrest [1992], a so-called "steepest-edge" rule for selecting the "leaving variable" at each dual simplex iteration. This method requires relatively little additional computational effort per iteration and is far superior to "standard" dual methods, in which the selection of the leaving variable is based only upon selecting a basic variable with large primal infeasibility.

Linear algebra

Linear algebra improvements touch all the parts of simplex algorithms and are also crucial to good implementations of barrier algorithms. Enumerating all such improvements is beyond the scope of this paper. I will mention only a few. For simplex algorithms, two improvements stand out among the rest.

The first of these to be introduced was dynamic LU-factorization using Markowitz threshold pivoting. This approach was perfected by Suhl and Suhl [1990], and has become a standard part of modern codes. In previous-generation codes, "preassigned pivot" sequences were used in the numerical factorization (see

Hellerman and Rarick [1971]). These methods were very effective when no numerical difficulties occurred, but encountered serious difficulties in the alternative case.

The second major linear algebra improvement is that LP codes now take advantage of certain ideas for solving large, sparse linear systems, ideas that have been known in the linear-algebra community for several years (see Gilbert and Peierls [1988]). At each major iteration of a simplex algorithm, several sizeable linear systems must be solved. The order of these systems is equal to the number of constraints in the given LP. Typically these systems take as input a vector with a very small number of nonzero entries, say between one and ten – independent of overall model size – and output a vector with only a few additional nonzeros. Since it is unlikely that the sparsity of the output is due to cancellation during the solve, it follows that only a small number of nonzeros in the LU-factorization (and update) of the basis could have been touched during the solve. The trick then is to carry out the solve so that the work is linear in this number of entries, and hence, in total, essentially a constant time operation, even as problem size grows. The effect on large linear programs can be enormous.

Presolve

This idea is made up of a set of problem reductions: Removal of redundant constraints, fixed variables, and other extraneous model elements. The seminal reference on this subject is Brearley et al [1975]. Presolve



was available in MPS III, but modern implementations include a much more extensive set of reductions, including so-called aggregation (substituting out variables, such as free variables, the satisfaction of the bounds of which are guaranteed by the satisfaction of the bounds on the variables that remain in the model). The effects on problem size can be very significant, in some cases yielding reductions by factors exceeding an order of magnitude. Modern presolve implementations are seamless in the sense that problem input and solution output occur in terms of the original model.

Examples of performance improvements

Table 10: **Solution times–Best simplex**

| Model | CPLEX 1.0 | CPLEX 2.2 | CPLEX 5.0 | CPLEX 7.1 | Algorithm |
|---|---|---|---|---|---|
| car | 1555.0 | 701.1 | 275.8 | 120.6 | primal |
| continent | 364.7 | 110.5 | 104.4 | 46.7 | primal |
| energy1 | 1217.4 | 275.0 | 260.5 | 22.6 | dual |
| energy2 | 10130.1 | 736.0 | 664.0 | 693.9 | dual |
| energy3 | 21797.1 | 271.9 | 229.1 | 161.7 | dual |
| fuel | 5619.5 | 1123.2 | 698.6 | 675.0 | primal |
| initial | 3832.2 | 102.2 | 51.3 | 15.5 | dual |
| schedule | 152404.0 | 252.3 | 220.8 | 64.6 | dual |

full paper

# SOLVING REAL-WORLD LINEAR PROGRAMS: A DECADE AND MORE OF PROGRESS

**ROBERT E. BIXBY**

*ILOG, Inc. and Rice University, bixby@ilog.com or bixby@rice.edu*

This paper is an invited contribution to the 50th anniversary issue of the journal *Operations Research*, published by the Institute of Operations Research and Management Science (INFORMS). It describes one person's perspective on the development of computational tools for linear programming. The paper begins with a short personal history, followed by historical remarks covering the some 40 years of linear-programming developments that predate my own involvement in this subject. It concludes with a more detailed look at the evolution of computational linear programming since 1987.

## 1. INTRODUCTION

I am a relative newcomer to computation. For the first half of my scientific career, my research focused exclusively on the theoretical aspects of operations research and discrete mathematics. That focus began to change in the early 1980s with the appearance of personal computers.

My first PC was used primarily to implement elementary algorithms used in teaching. At first these algorithms did not include a simplex algorithm; eventually, however, I concluded that it would be useful to incorporate computation in the LP courses that I was teaching. As a result, I started writing my own code, initially a simple tableau code.

At that time, in the early 1980s, I knew nothing about the computational aspects of linear programming (LP). I knew a great deal of theory, but numerical analysis and the computational issues associated with numerical algorithms were not subjects that were part of my graduate education. I had no idea that tableaus were numerically unstable.

Fortunately for me, by the time my interests in computation had started, the Department of Industrial Engineering and Management Sciences at Northwestern University had hired Bob Fourer, one of the creators of the AMPL modeling language. Bob had worked for several years at the National Bureau of Economic Research doing practical linear programming, followed by a graduate career at Stanford. He knew a lot about the computational aspects of mathematical programming, and he passed on a great deal of that knowledge to me in informal conversations.

Linear programming become more central to what I was doing when a friend of mine, Tom Baker, founded Chesapeake Decision Sciences (now a part of Aspen Technologies). Shortly thereafter, Tom asked if I had an LP code that he could use in the LP module of the product he was building. I said yes, converted my code to C (that was one of Tom's conditions), and delivered it to him.

To this day, I'm not quite sure why Tom thought my code would eventually be reasonably good. Initially it certainly was not.

After the code was delivered to Chesapeake, there followed a period of about two years during which I received a steady stream of practical LPs from Chesapeake, LPs on which my code did not do very well. In each case, I poked around in my code and the LP itself to see what ideas I could come up with, never looking in the literature (this wasn't my area of research). Slowly the code got better, until some time around 1986, one of Tom's colleagues informed me that my code had actually gotten good enough that one of their customers was interested in obtaining it separately. I was, to say the least, surprised, and immediately set about doing my first actual comparisons to other LP codes. I chose Roy Marsten's (1981) quite successful and portable (that was key for me) XMP code. I discovered, to my amazement, that for a substantial subset of the *netlib*[1] testset my code was indeed pretty good, running on average two times faster than XMP. In addition, it appeared that my code was significantly more stable than XMP.

This comparison to XMP was an important part of what transformed LP computation into a serious part of my scientific research. Equally important was integer programming.

This was the mid-1980s, and integer-programming computational research was beginning to flower, with important contributions by people such as Martin Grötschel, Ellis Johnson, Manfred Padberg, and Laurence Wolsey. Linear programming was an essential component in that work, but the tools available at that time were proving to be inadequate. The then state-of-the-art codes, such as MPSX/370, simply were not built for this kind of application; in addition, they did not deal well with issues such as degeneracy. The situation at the time is well described by some remarks of Grötschel and Holland (1991), commenting on their use of MPSX/370 in work on the traveling salesman problem: They note that if the LP-package they were using had been

*Subject classification:* Professional: comments on
*Area of review:* Anniversary Issue (Special).

0030-364X/02/5001-0003 $05.00
1526-5463 electronic ISSN



**Operations Research** © 2002 INFORMS
Vol. 50, No. 1, January–February 2002, pp. 3–15

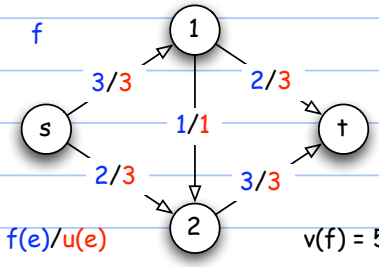
bixby-or2002.pdf

- Goals of this chapter
  - Illustrate the revised simplex algorithm
  - Illustrate how to handle LPs with (exponentially) many columns: Column Generation
  - We use a path-based formulation of the max-flow problem

- The max-flow problem (see ADM I)
  - Maximum Flow problem (MFP)
    - Instance
      - network $(G, u, s, t)$ where
        - $G$ is a digraph
        - $s, t$ are vertices of $G$, called the source and the sink, respectively
        - $u(e) \geq 0$ is the capacity of edge $e$
    - Task
      - Find an $s,t$-flow $f$ with maximum flow value $v(f)$
        - $s,t$-flow $f$ = edge weight $f(e)$ for every edge with
          - $0 \leq f(e) \leq u(e)$ for all edges $e$



        - flow conservation in all vertices $v \neq s, t$
        - flow value $v(f)$ = net outflow out of $s$

    - Example: a flow f
      - 


- A formulation of the max-flow problem as LP with edge-variables (edge-based formulation)
  - we use the vertex-edge-incidence matrix as for the shortest path problem

    max $v$ (maximize the flow value $v$) such that

$$Af = b \text{ with } b = \begin{pmatrix} +v \\ -v \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \begin{matrix} \text{row } s \\ \text{row } t \text{ (redundant)} \\ \\ \\ \text{flow conservation} \end{matrix}$$

$f \leq u$

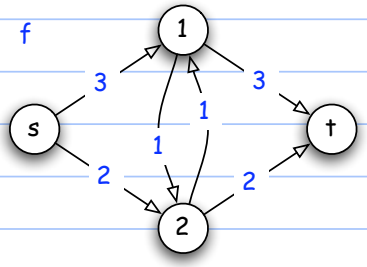
$f \geq 0$

Here $E(G) = \{ e_1, \ldots, e_m \}$ and $f = (f_1, \ldots, f_m)^T$, i.e.,, we have a variable per edge for the flow on that edge

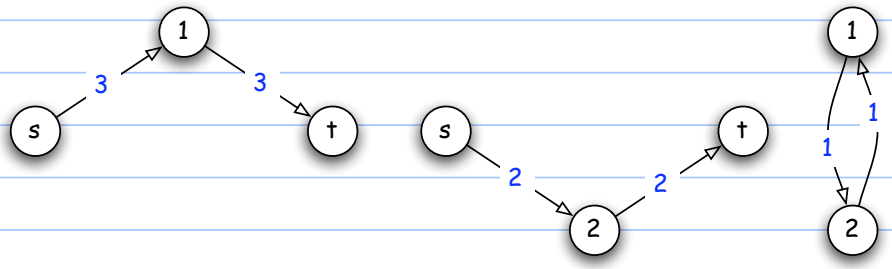
- A formulation of the max-flow problem as LP with path-variables (path-based formulation)
  - is based on the Flow Decomposition Theorem of ADM I
    - ADM I, Theorem 5.2 (Flow Decomposition Theorem for s,t-flows, Ford-Fulkerson 1962)
      - Let $f \neq 0$ be an s,t-flow in $(G, u, s, t)$. Then
        - f is a positive linear combination of (incidence vectors of) directed (elementary) s,t-paths and directed (elementary) cycles
        - the number of these paths and cycles can is at most m
        - if f is integer, then there is such a linear combination with integer coefficients



- Example
  - s,t-flow
    - f




  - decomposition into directed paths and cycles (not unique)




  - corresponding linear combination

$$\begin{pmatrix} f_{s1} \\ f_{s2} \\ f_{12} \\ f_{1t} \\ f_{21} \\ f_{2t} \end{pmatrix} \qquad f = \begin{pmatrix} 3 \\ 2 \\ 1 \\ 3 \\ 1 \\ 2 \end{pmatrix} = 3 \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + 2 \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} + 1 \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

- Observe: the s,t-paths determine the flow value, cycles play no role
- The path-based LP
  - Let $P_1, \ldots, P_p$ be all directed elementary s,t-paths in $G$ (Observe: p may be exponential in n).

    The edge-path-incidence matrix $D = (d_{ij})$ is defined by

$$d_{ij} := \begin{cases} 1 & \text{edge } e_i \text{ lies on path } P_j \quad i = 1, \ldots, m \\ 0 & \text{otherwise} \qquad\qquad\qquad\quad j = 1, \ldots, p \end{cases}$$

    $f = (f_1, \ldots, f_p)^T$ is a flow vector that has an entry $f_j$ for s,t-path $P_j$ denoting the amount of flow that is sent along path $P_j$
  - The capacity constraints read



$$Df \le u$$

i.e.

$$(d_{i1}, \ldots, d_{ip}) \cdot \begin{pmatrix} f_1 \\ \vdots \\ f_p \end{pmatrix} \le u_i$$

$\Sigma$ of flows on all paths containing $e_i$
= flow in edge $e_i$

for every row $e_i$

- Flow conservation holds trivially and the flow value $v$ is obtained as

$$v = \Sigma_j f_j$$

- Then the path-based LP is

$$\min c^T f \text{ with } c_j = -1$$

$$Df \le u$$

$$f \ge 0$$

- We transform it into standard form by adding slack variables $s_i$

=> $\min\ c'^{T}f' \quad \text{with} \quad f' = (f \mid s)^{T},\ c' = (c \mid 0)^{T}$

$D'f' = u \quad \text{with} \quad D' = (D \mid I)$ (5.1)

$f' \geq 0$

Slack variable $s_i$ represents the residual capacity on edge $e_i$

Solving the path-based formulation with the revised simplex algorithm

a basic feasible solution is given by $f = 0$ and $s = u$

=> Phase I is not necessary

column $D_j$ of path $P_j$ has (Pricing Rule) reduced cost $\bar{c}_j < 0$

$\Leftrightarrow \quad \bar{c}_j = c_j - \pi^{T}D_j < 0$

$\Leftrightarrow \quad (-\pi)^{T}D_j < 1$ because of $c_j = -1$

Interpretation

$-\pi$ = vector of edge weights $-\pi_i$ of edge $e_i$

$(-\pi)^{T}D_j$ = length of the path $P_j$ w.r.t. edge weights $-\pi_i$

Idea



Compute the shortest s,t-path P w.r.t. $-\pi$. Then (5.2)

P has length $\geq 1$ => optimality criterion holds for all columns belonging to paths

P has length $< 1$ => have found a column (path) to enter the basis

So it is not necessary to store all (exponentially many) columns of paths explicitly

In general, the idea

Find a column that violates the optimality criterion at most

is again an LP, whose solution is often much simpler than all generating all columns in the revised simplex algorithm explicitly.

This idea is called Column Generation

For the column of slack variable $s_i$ we obtain

$s_i$ has negative reduced cost <=> $-\pi_i < 0$ (5.3)

since:

reduced cost of column $s_i$ is $0 - (\pi^{T}I)_i = -\pi_i < 0$

5.1 Theorem (Solving the path-based max-flow problem with column generation)

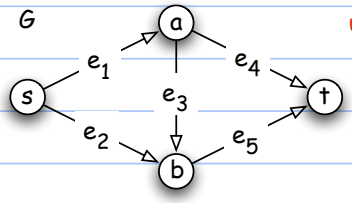
- A basic feasible solution of (5.1) fulfills the optimality criterion
  <=> $-\pi \geq 0$ and the shortest s,t-path w.r.t. $-\pi$ has length $\geq 1$
- The pivot steps in the revised simplex algorithm are shortest-path computations with edge lengths $-\pi \geq 0$ or letting a slack variable $s_i$ enter the basis if $-\pi_i < 0$
- Proof
  - If $-\pi_i < 0$ for some i, then $s_i$ is because of (5.3) a non-basic variables with negative reduced cost => optimality criterion violated
  - If $-\pi \geq 0$, then pricing reduces because of (5.2) and (5.3) to computing a shortest s,t-path with non-negative edge weights $-\pi$. ❑

- Consequence
  - The revised simplex algorithm solves the max-flow problem (essentially) as a sequence of shortest path problems
  - Need only a (m+1)x(m+1) CARRY matrix in each iteration

- 5.1 Example



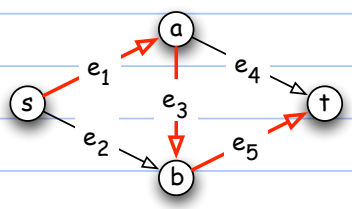
- Prepare data for the revised simplex algorithm




$CARRY^{(0)}$

| | 0 | 0 | 0 | 0 | 0 | 0 | ← $-\pi^T$ |
|---|---|---|---|---|---|---|---|
| $s_1$ | 1 | 1 | | | | | |
| $s_2$ | 1 | | 1 | | | | |
| $s_3$ | 1 | | | 1 | | | |
| $s_4$ | 1 | | | | 1 | | |
| $s_5$ | 1 | | | | | 1 | |

- $-\pi^T = 0$ => every s,t-path is a shortest path with cost $0 < 1$
  => choose w.o.l.g. $P_1 = \{ e_1, e_3, e_5 \}$ as entering column




The associated column $D_1$ of the initial LP is $(1, 0, 1, 0, 1)^T$ and we obtain the transformed column $X_1$ as

$X_1 = B^{-1}D_1 = D_1$

The reduced cost of column $X_1$ is then

$c_1 - \pi^T D_1 = -1 + \text{length of the path} = -1 + 0 = -1$

Data for pivot operation:

| | 0 | 0 | 0 | 0 | 0 | 0 | -1 |
|---|---|---|---|---|---|---|---|
| $s_1$ | 1 | 1 | | | | | 1 |
| $s_2$ | 1 | | 1 | | | | 0 |
| $s_3$ | 1 | | | 1 | | | 1 |
| $s_4$ | 1 | | | | 1 | | 0 |
| $s_5$ | 1 | | | | | 1 | 1 |

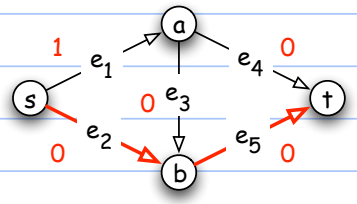
1st pivot



$CARRY^{(1)}$

| | 1 | 1 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|
| $f_1$ | 1 | 1 | | | | |
| $s_2$ | 1 | | 1 | | | |
| $s_3$ | 0 | -1 | | 1 | | |
| $s_4$ | 1 | | | | 1 | |
| $s_5$ | 0 | -1 | | | | 1 |

η-vector




graph with edge weights -π and shortest path

The associated column $D_2$ of the initial LP is $(0, 1, 0, 0, 1)^T$ and we obtain the transformed column $X_2$ as

$X_2 = B^{-1}D_2 = D_2$

The reduced cost is $c_2$ + length shortest path = -1 + 0 = -1

Data for next pivot:

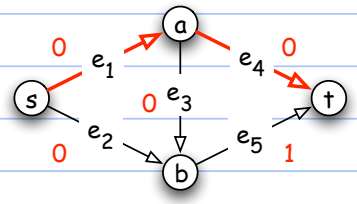
| | 1 | 1 | 0 | 0 | 0 | 0 | -1 |
|---|---|---|---|---|---|---|---|
| $f_1$ | 1 | 1 | | | | | 0 |
| $s_2$ | 1 | | 1 | | | | 1 |
| $s_3$ | 0 | -1 | | 1 | | | 0 |
| $s_4$ | 1 | | | | 1 | | 0 |
| $s_5$ | 0 | -1 | | | | 1 | 1 |

2nd pivot

$CARRY^{(2)}$

| | 1 | 0 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|
| $f_1$ | 1 | 1 | | | | |
| $s_2$ | 1 | 1 | 1 | | | -1 |
| $s_3$ | 0 | -1 | | 1 | | |
| $s_4$ | 1 | | | | 1 | |
| $f_2$ | 0 | -1 | | | | 1 |

η-vector




graph with edge weights -π and shortest path

The associated column $D_3$ of the initial LP is $(1, 0, 0, 1, 0)^T$ and we obtain the transformed column $X_2$ as

$X_3 = B^{-1}D_3 = (1, 1, -1, 1, -1)^T$



The reduced cost is $c_3$ + length shortest path = -1 + 0 = -1

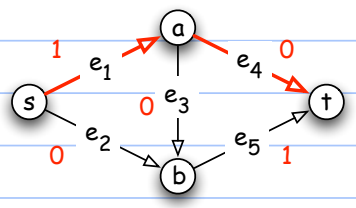
Data for next pivot:

| | 1 | 0 | 0 | 0 | 0 | 1 | -1 |
|---|---|---|---|---|---|---|---|
| $f_1$ | 1 | 1 | | | | | 1 |
| $s_2$ | 1 | 1 | 1 | | | -1 | 1 |
| $s_3$ | 0 | -1 | | 1 | | | -1 |
| $s_4$ | 1 | | | | 1 | | 1 |
| $f_2$ | 0 | -1 | | | | 1 | -1 |

3rd pivot

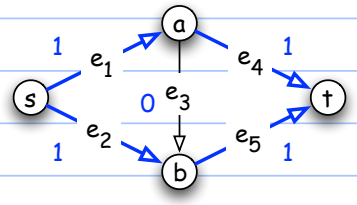
$CARRY^{(3)}$

| | 2 | 1 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|
| $f_3$ | 1 | 1 | | | | |
| $s_2$ | 0 | 0 | 1 | | | -1 |
| $s_3$ | 1 | 0 | | 1 | | |
| $s_4$ | 0 | -1 | | | 1 | |
| $f_2$ | 1 | 0 | | | | 1 |




length shortest path ≥ 1

=> have reached an optimal solution with maximum flow $f_3 + f_2$




maximum flow

- Remarks
  - Column generation maintains only a (changing) subproblem of the initial problem (called the master problem) with few columns. It generates new columns when needed for improvement by solving the pricing operation as a separate optimization problem over all possible (and only implicitly represented) columns.
  - In the max-flow example, the pricing reduces to a simple shortest path problem. In general, the pricing will be more difficult and often even NP-hard (e.g. if the flow carrying paths must respect additional constraints, e.g. in traffic applications where routes must not be too long, or only main streets ...).
  - Column generation is one of the workhorses for solving complex network problems. More in ADM III



- Goal of this chapter
  - modify the simplex algorithm so that lower and upper bounds can be handled implicitly
  - serves as preparation for the network simplex (programming exercise)
  - lower bounds are easy
  - upper bounds need a modified definition of a basic feasible solution, this then leads to a more efficient and rather easy variation of the simplex algorithm

- LP in standard form with lower and upper bounds
  - min $c^T x$ s.t. $Ax = b$, $\ell \le x \le u$

$$\ell = \begin{pmatrix} \ell_1 \\ \vdots \\ \ell_n \end{pmatrix} \geq 0 \qquad \text{vector of lower bounds}$$

$$u = \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} \geq \ell \qquad \text{vector of upper bounds}$$

- Approaches
  - (1) treat bounds as additional constraints and transform the resulting LP into standard form with slack variables for the bound constraints
    => matrix $A$ gets larger
    unwanted, as bounds are very simple constraints
  - (2) treat bounds implicitly by a slight variation of the simplex algorithm
    -> in this chapter

- Treat lower bounds by substituting variables
  - Approach: $x_j = y_j + \ell_j$
  => $y_j = x_j - \ell_j$
  => write initial LP as
    $\min c^T y$ s.t. $Ay = b'$, $0 \le y \le u'$
    with $b' := b - A\ell$ and $u' := u - \ell$
  Solution $y$ of the modified LP yields a solution $x$ of the initial LP by
    $x = y + \ell$



- So assume w.o.l.g. $\ell = 0$ in the sequel
  i.e., $\min c^T x$
  s.t. $Ax = b$ (5.4)
  $0 \le x \le u$
  and w.o.l.g. rank(A) = m
- Basic feasible solutions for LPs with upper bounds
  - Extended partition of the variables
    - Instead of a partition of the variables/columns of $A$ into $B$ (basic variables) and $N$ (non-basic variables) we now consider an extended partition into
      B basic variables
      L non-basic variables with value = lower bound = 0
      U non-basic variables with value = upper bound = $u_j$
    for such a partition we want that
      $Bx_B + Lx_L + Ux_U = Bx_B + Uu_U = b$ (5.5)
    which gives
      $x_B = B^{-1}(b - Uu_U) = B^{-1}b - B^{-1}Uu_U$

A basic feasible solution with basis B of an LP with upper bounds is every basic solution of the form (5.5)

5.2 Theorem (Fundamental Theorem for LPs with upper bounds)

If LP (5.4) has an optimal solution, then also an optimal solution that is a basic feasible solution of the form (5.5).

Proof

Model $x \leq u$ with slack variables $s$ as linear system $x + s = u$

Then we obtain the following large LP in standard form (LPSU)

$$\begin{aligned} \min\ & c^T x \\ \text{s.t.}\ & Ax = b \\ & x + s = u \\ & x \geq 0,\ s \geq 0 \end{aligned}$$

Since $A$ has full row rank, also (LPSU) has full row rank.

Let $(x,s)^T$ be a basic feasible solution of (LPSU) with basis $B'$.

Set

$$B := \{ j \in \{1,\dots,n\} \mid x_j \in B' \text{ and } s_j \in B' \} \text{ and } p := |B|$$



$$L := \{ j \in \{1,\dots,n\} \mid x_j \notin B' \text{ and } s_j \in B' \} \text{ and } s := |L|$$

$$U := \{ j \in \{1,\dots,n\} \mid x_j \in B' \text{ and } s_j \notin B' \} \text{ and } q := |U|$$

Identifying $B = A_B$ and $U = A_U$ we can permute $B'$ such that:

$$PB'Q^T = \begin{pmatrix} B & B_q & 0 & 0 \\ I_p & 0 & I_p & 0 \\ 0 & I_q & 0 & 0 \\ 0 & 0 & 0 & I_s \end{pmatrix} \begin{matrix} \\ \\ \uparrow \\ \downarrow \\ \\ \end{matrix} \begin{matrix} \\ \\ \text{rows for} \\ x + s = u \\ \\ \end{matrix}$$

with permutation matrices $P$ and $Q$

Counting the rows of $B'$ gives $m + n = m + p + q + s$

Counting the columns of $B'$ gibes $m + n = 2p + q + s$

$\Rightarrow p = m \Rightarrow B$ is an $m \times m$-matrix

Laplace expansion of $\det(B') = \det(PB'Q)$ along the last $n$ rows gives $|\det(B')| = |\det(B)|$

$B'$ is a basis of the large LP $\Rightarrow \det(B') \neq 0 \Rightarrow \det(B) \neq 0$

$\Rightarrow B$ is basis of $A$

Since $B'(x,s)^T = (b,u)^T$, the permuted form of $B'$ transforms into (5.6)

$$Bx_B + Ux_U = b$$

$$x_B + s_B = u_B$$
$$x_U = u_U$$
$$s_L = u_L$$

So the partition B, L, U defines a basic solution $Bx_B + Ux_U = b$ of the form (5.5)

=> every basic feasible solution of the large LP corresponds to a basic feasible solution of (5.4) in the form of (5.5)

=> statement with the Fundamental Theorem of linear optimization (Theorem 3.12) ❑

Optimality criterion for LPs with upper bounds

5.3 Theorem (Optimality criterion for LPs with upper bounds)

A basic feasible solution of the form (5.5) is optimal

<=> the reduced costs in the tableau fulfill

$$\left.\begin{array}{lcll} \bar{c}_j & = & 0 & \text{for } x_j \in B \\ \bar{c}_j & \geq & 0 & \text{for } x_j \in L \\ \bar{c}_j & \leq & 0 & \text{for } x_j \in U \end{array}\right\} \quad (5.7)$$

Proof:



By transforming the optimality conditions of the large LP ❑

Choosing the pivot element for LPs with upper bounds

Choosing the pivot column

Choose column $A_j$ with $j \in L$ and $\bar{c}_j < 0$ or column $A_j$ with $j \in U$ and $\bar{c}_j > 0$ (5.8)

Choosing the pivot row

$x_s \in L$ => increase $x_s$ as much as possible

$x_s \in U$ => decrease $x_s$ as much as possible

Use the representation of $x(\theta)$ according to (3.16)

$$x_\ell(\theta) = \begin{cases} y_{B(i)} - \theta x_{is} & \ell = B(i),\ i = 1, \ldots, m \\ \theta & \ell = s \\ 0 & \text{sonst} \end{cases}$$

and take into account that $x_s = u_s$ is possible

$x_{is} < 0$ => $x_{B(i)}$ increases

$$\Rightarrow \text{bound } \theta_i = \frac{u_i - x_{i0}}{-x_{is}} \text{ for } \theta$$

$x_{is} > 0 \Rightarrow x_{B(i)}$ decreases

$$\Rightarrow \text{bound } \theta_i = \frac{x_{i0}}{x_{is}} \text{ for } \theta$$

(as in the ordinary simplex algorithm)

- Choose $\theta$ as minimum of the $\theta_i$ and $u_s$. There are 2 cases
  - The minimum is attained at $u_s$
    - leave the basis unchanged, variable $x_s$ moves from U to L or vice versa
  - The minimum is attained at $\theta = \theta_r$
    - pivot with $x_{rs}$

      the new value of $x_s$ is $\theta$ (if $x_s = 0$) or $u_s - \theta$ (if $x_s = u_s$)
- Termination
  - requires additional arguments



- A more detailed treatment of the network simplex algorithm will be done in the exercises

  Here: a proof that it is a specialization of the simplex algorithm with upper bounds

- Network problem (Input for the network simplex)

  min $c^T x$ s.t. $Ax = b$, $0 \le x \le u$

  $A$ = vertex-edge-incidence matrix of a digraph $G = (V, E)$

  In the terminology of ADM I this is a Minimum Cost Flow Problem (MCFP)

- Basic solutions of the network problem
  - we assume in the sequel that $V = \{ 1,\dots,n \}$ and that $G$ is connected.

    rows of $A$ add up to 0

    => delete w.o.l.g. the row for vertex 1 and denote the resulting matrix again by $A$

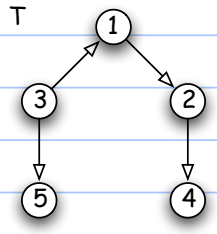
    => $A$ has n-1 rows

  - Consider a spanning tree $T$ of $G$

    => $T$ has n-1 edges

Let B be the set of the associated columns of A

Example:




B

| | (1,2) | (3,1) | (3,5) | (2,4) |
|---|---|---|---|---|
| 1 | 1 | -1 | | |
| 2 | -1 | | | 1 |
| 3 | | 1 | 1 | |
| 4 | | | | -1 |
| 5 | | | -1 | |

Consider T as undirected tree with root 1

Order the vertices of T in preorder traversal (i.e., root-left-right recursively)

in the example: 1, 3, 5, 2, 4

Order the edges according to this vertex order,

i.e., for each vertex j ≠ 1 take the (unique) last edge on the path from 1 to j.

in the example: (3,1) (3,5) (1,2) (2,4)

consider the permuted matrix B´ according to these row and column orders



| | (3,1) | (3,5) | (1,2) | (2,4) |
|---|---|---|---|---|
| 1 | -1 | | 1 | |
| 3 | 1 | 1 | | |
| 5 | | -1 | | |
| 2 | | | -1 | 1 |
| 4 | | | | -1 |

5.4 Lemma

The permuted matrix B´ obtained from the preorder traversal of the tree is (after deleting row 1) an upper triangular matrix with entries ≠ 0 on the diagonal.

Proof

Suppose that the preorder traversal just visits vertex i.

Let j be the father of i in T.

=> (i,j) or (j,i) is tree edge, say (i,j)

Permutation of the rows and columns => (i, (i,j)) is an entry on the diagonal of B´

Preorder traversal => j was visited from i

=> column (i,j) contains

-1 in row j

+1 in row i

0 in all later rows ❑

5.5 Consequence

The rows and columns of the vertex-edge-incidence matrix of a spanning trees of $G$ can be permuted by a preorder traversal so that the resulting matrix is non-singular and upper triangular.

The linear systems $Bx = b$ and $\pi^T B = c_B^T$ of the revised simplex algorithm can easily be solved by exploiting the upper triangular form and the fact that every column contains at most 2 entries ≠ 0. This amounts to simple iterative substitution along the triangular form.

5.6 Theorem (correspondence basis <-> spanning tree)

Every spanning tree of $G$ defines a basis of the network problem (which need not be feasible w.r.t. $0 \le x \le u$).

Every basis of the network problem defines a spanning tree of $G$.

Proof



=>:

Lemma 5.4.

In particular, every basis has n-1 columns.

<=:

Let B be a basis of the network problem

=> the associated columns correspond to a subgraph $G'$ with n-1 edges

Claim: $G'$ has no undirected cycles

Assume that $G'$ has an undirected cycle $C$.

choose an orientation of $C$ and let $C^+$ be the set of forward edges and $C^-$ be the set of backward edges of $C$ w.r.t. the chosen orientation.

every vertex i in $C$ is incident to exactly 2 edges from $C$

$$\Rightarrow \quad \sum_{e \in C^+} A_e - \sum_{e \in C^-} A_e = 0$$

this contradicts the fact that B is a basis.

ADM I, Theorem 2.3 => every undirected graph with n vertices and n-1 edges and without cycles is a tree ❑

Steps in the revised simplex algorithm

Basis B = tree together with partition B, L, U.

1. Computing the right hand side

It is the solution of

$$Bx_B = b - Uu_U =: b'$$

=> solve the linear system $Bx_B = b'$

iteratively along the triangular form of B according to Consequence 5.4

2. Computing the simplex multipliers $\pi_i$

They are the solution of

$$\pi^T B = c_B^T$$

=> $\pi_i - \pi_j = c_{ij}$ for column/edge $(i,j) \in B$

=> iteratively along the triangular form of B , $\pi_i$ of the last row can be read of directly, then iterate backwards

3. Optimality criterion



reduced cost of column/edge (i,j) is

$$\bar{c}_{ij} = c_{ij} - \pi^T A_{ij} = c_{ij} - \pi_i + \pi_j$$

Section 5.4 =>

$$\bar{c}_{ij} \geq 0 \quad \text{for } (i,j) \in L$$
$$\bar{c}_{ij} \leq 0 \quad \text{for } (i,j) \in U$$

4. Computing the transformed column $X_{rs}$

$-X_{rs}$ corresponds to the change of the basic variables when the value $x_{rs}$ of the non-basic pivot column is set to 1

so (r,s) enters the basis, B is a tree

=> B + (r,s) contains a unique cycle C

Orienting C according to (r,s) partitions C into a set of forward edges and a set of backward edges.

Setting $x_{rs}$ to 1 then implies in C a change by

+1 on forward edges

-1 on backward edges

=> pivot operation corresponds exchanging edges on this cycle.

If the non-basic variable $x_{rs}$ is the "bottleneck", then there it changes only from L to U or vice versa (see Section 5.4).

- 5. Computing an initial basic feasible solution
  - = Phase I, see Exercises

- 6. Anti-cycling
  - by considering only strong tree solutions, see Exercises.

- Literature on the network simplex algorithm
  - Chapter 11 in

    K. Ahuja, T.L. Magnanti, J.B. Orlin

    Network Flows: Theory, Algorithms, and Applications

    Prentice Hall, 1993

## 6. Primal-dual algorithms 34

- Background
  - Primal-dual algorithms are based on complementary slackness.
  - They were originally developed for network problem [Dantzig, Ford, Fulkerson 1956]
  - They provide a general method to derive "specialized" algorithms for combinatorial optimization problems, exact and approximate.

- Basic idea
  - Start with an LP in standard form

    (P) min $z = c^T x$

    $Ax = b \geq 0$ (w.o.l.g.)

    $x \geq 0$
  - The associated dual LP is

    (D) max $w = \pi^T b$

    $\pi^T A \leq c^T$

    $\pi$ unrestricted
  - Complementary slackness yields



$x \in S_P$, $\pi \in S_D$ are optimal

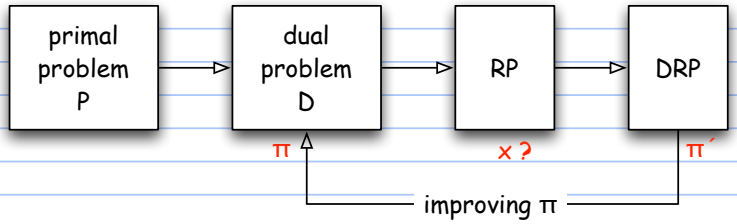
<=> $\pi_i(a_i^T x - b_i) = 0$ for all i (this holds since $Ax = b$)

$(c_j - \pi^T A_j)x_j = 0$ for all j (6.1)

So: (6.1) = is the only condition for optimality

- Primal-dual algorithm

  Given $\pi \in S_D$, find $x \in S_P$ such that x and $\pi$ fulfill (6.1)
  - We search for such an $x \in S_P$ solving an auxiliary problem, called the restricted primal (RP), determined by the given dual feasible solution $\pi \in S_D$.
  - If no such x exists, we use information from the dual (DRP) of the restricted primal (RP) in order to construct a "better" dual solution $\pi \in S_D$.
  - We iterate this process until we (hopefully) find an optimal pair x, $\pi$




Remark: this is essentially a dual algorithm, since we have a dual feasible solution $\pi$ in every step and obtain an optimal primal feasible solution $x$ only at termination. It is nevertheless called primal-dual because of the role of the complementary slackness conditions.



Constructing a dual feasible start solution $\pi$

All $c_j \geq 0$

=> $\pi = 0$ is dual feasible, as $\pi^T A \leq c^T$

Some $c_j < 0$

Use a trick:

Introduce another primal variable $x_{n+1} \geq 0$

Introduce another primal constraint

$$x_1 + x_2 + \ldots + x_{n+1} = b_{m+1}$$

with $b_{m+1} \geq n \cdot M$ (M from Lemma 3.4) and $c_{m+1} = 0$

Lemma 3.4 => this constraint does not change $S_P$

The dual problem then is

$$\max w = \pi^T b + \pi_{m+1} b_{m+1}$$

$$\pi^T A_j + \pi_{m+1} \leq c_j \quad j = 1,\ldots,n$$

$$\pi_{m+1} \leq 0$$

$\pi_i$ unrestricted, $i = 1,\ldots,m$

A feasible solution of this dual LP is given by

$\pi_i = 0 \quad i = 1,\dots,m$

$\pi_{m+1} = \min_j c_j < 0$ (since at least one $c_j < 0$)

=> a dual feasible solution can be constructed quite easily (much simpler than a primal with the Two-Phase-method)

The Restricted Primal (RP)

Assume that we have a dual feasible solution $\pi$ of (D)

To fulfill (6.1), set

$J := \{ j \mid \pi^T A_j = c_j \}$

We call $J$ the set of admissible columns

(6.1) => $x \in S_P$ is optimal <=> $x_j = 0$ for all $j \notin J$

So we are looking for an $x$ with

$\sum_{j \in J} A_j x_j = b$

$x \geq 0, \; x_j = 0$ for all $j \notin J$

This search is a pure feasibility problem, which we will solve with Phase I of the simplex algorithm. The Phase I problem is called the Restricted Primal (RP):



$$
\begin{array}{lll}
\min & \xi = \sum_{i=1}^{m} x_i^a & \\
\text{unter} & \sum_{j \in J} a_{ij} x_{ij} + x_i^a = b_i & i = 1,\dots,m \\
& x_j \geq 0 & j \in J \\
& x_j = 0 & j \notin J \\
& x_i^a \geq 0 & i = 1,\dots,m
\end{array}
\qquad \left.\begin{array}{l} \\ \\ \\ \\ \\ \end{array}\right\} \text{(RP)}
$$

$\longleftarrow$ may delete these $x_j$

We can solve (RP) with the simplex algorithm. (RP) searches for a feasible solution of (P) without the columns $A_j$ with $j \notin J$. The artificial variables define the initial basis of (RP).

If $\xi_{opt} = 0$, then each artificial variable is 0 and $x$ is a feasible solution of (RP)

=> $x$ is an optimal solution of (P)

If $\xi_{opt} > 0$, then there is no $x$ in (RP) fulfilling (6.1)

=> we investigate the dual LP of (RP)

The dual (DRP) of the restricted primal

(DRP) reads

$$
\left.
\begin{array}{lllll}
\max & w = \pi^T b & & & (6.2) \\
s.t. & \pi^T A_j \leq 0 & & j \in J & (6.3) \\
& \pi_i \leq 1 & & i = 1, \ldots, m & (6.4) \\
& \pi_i & \text{unrestricted} & i = 1, \ldots, m & (6.5)
\end{array}
\right\} (DRP)
$$

Let π´ be an optimal solution of (DRP) (it exists because of Strong Duality)

Idea: combine π´ with the original dual solution π to

$\pi^* := \pi + \theta\pi'$ (6.6)

where θ is chosen such that π* stays feasible in (D) and the dual objective function of (D) strictly increases

Consequence for the dual objective function of (D):

$(\pi^*)^T b = \pi^T b + \theta(\pi')^T b$

$\theta(\pi')^T b = \xi_{opt} > 0$ as (RP) and (DRP) are a primal dual pair

Hence $\theta > 0$ is required for a strict increase of the dual objective function

Consequences for dual feasibility in (D)



dual feasibility means

$(\pi^*)^T A_j = \pi^T A_j + \theta(\pi')^T A_j \leq c_j$ for $j = 1, \ldots, n$

this is no problem if $(\pi')^T A_j \leq 0$ (this holds for all $j \in J$ since $\pi' \in S_{DRP}$)

There are 2 cases

$(\pi')^T A_j \leq 0$ for all $j = 1, \ldots, n$

=> θ can be made arbitrarily large

=> the dual objective function of (D) is unbounded

Theorem 4.3 => (P) has no feasible solution

$(\pi')^T A_j > 0$ for some $j \notin J$

Then we obtain a constraint for θ:

$\pi^T A_j + \theta(\pi')^T A_j \leq c_j$

$\theta(\pi')^T A_j > 0$

$$\text{so } \theta \leq \frac{c_j - \pi^T A_j}{(\pi')^T A_j}$$

We summarize

6.1 Theorem (Infeasibility of (P) in the primal-dual algorithm)

If $\xi_{opt} > 0$ in (RP) and $(\pi')^T A_j \leq 0$ for all $j = 1, \ldots, n$ w.r.t. the optimal solution $\pi'$ of (DRP), then (P) has no feasible solution.

Proof: clear from the above ❑

6.2 Theorem (Improvement of the dual solution in the primal-dual algorithm)

If $\xi_{opt} > 0$ in (RP) and $(\pi')^T A_j > 0$ for some $j \notin J$, then

$$\theta_1 := \min \{ \frac{c_j - \pi^T A_j}{(\pi')^T A_j} \mid j \notin J, (\pi')^T A_j > 0 \} \qquad (6.7)$$

is the largest $\theta$, such that $\pi^* := \pi + \theta\pi'$ is dual feasible. Then

$$w^* := (\pi^*)^T b = \pi^T b + \theta_1 (\pi')^T b > w \ (= \pi^T b)$$

Proof: clear from the above ❑

The primal-dual algorithm

Algorithm (Primal-Dual)



Input

Primal LP (P) in standard form

Associated dual LP (D) with feasible solution $\pi$ (possibly constructed by the above trick)

Output

At termination : an optimal solution or a message that (P) has no feasible solution

Termination can be guaranteed by anti-cycling rules

Method

repeat

Construct (RP) by computing $J := \{ j \mid \pi^T A_j = c_j \}$

call Phase I with cost vector $\xi = \sum x_i^a$ for (RP)

if $\xi_{opt} > 0$ then

call dual Simplex for (DRP) and take the computed optimal solution $\pi'$

if $(\pi')^T A_j \leq 0$ for all $j = 1, \ldots, n$

then return "(P) has no feasible solution"

else

compute $\theta_1$ according to (6.7)

set $\pi := \pi + \theta_1 \pi'$

until $\xi_{opt} = 0$

return solution $x$ of (RP)



(1) Restart: The basic optimal solution of the previous (RP) is a basic feasible solution for the new (RP)

6.3 Theorem (Keeping admissible basic columns)

Every admissible column of the optimal basis of (RP) remains admissible at the start of the next iteration of the primal-dual algorithm

Proof

Let $A_j$ be an admissible column of the optimal basis of (RP)

Definition of admissible column => $A_j$ is a column of $A$, i.e., does not belong to an artificial variable

reduced cost of a basic column is 0, $\pi'$ is a dual optimal solution of (RP)

$\Rightarrow 0 = \bar{c}_j = c_j - (\pi')^T A_j = 0 - (\pi')^T A_j$

$\Rightarrow (\pi')^T A_j = 0$

Then

$$(\pi^*)^T A_j = \pi^T A_j + \theta_1 (\pi')^T A_j = \pi^T A_j + 0 = \pi^T A_j = c_j$$

since $A_j$ is an admissible column w.r.t. $\pi$

=> $A_j$ remains admissible w.r.t. $\pi^*$

An optimal basis of (RP) is composed of

admissible columns => stay admissible because of Theorem 6.3

columns of artificial variables => stay in the new (RP)

=> Theorem and (1) ❑

(2) (RP) can be solved with the revised simplex algorithm

this follows from Theorem 6.3. We only need to update the set $J$ for the non-basic columns

(3) Termination can be achieved by anti-cycling rules

6.4 Theorem (Termination of the primal-dual algorithm)

The primal-dual algorithm solves (P) in finitely many steps

Proof

Interpret (RP) as a sequence of pivots of variables $x_1^a, \ldots, x_m^a, x_1, \ldots, x_n$

(possible since $x_j = 0$ for $j \notin J$ and thus can be interpreted as a non-basic variable)

=> (RP) traverses a sequence of basic feasible solutions of $(I \mid A)$

Claim: The objective function decreases monotonically along that sequence (not necessarily strictly)

this is clear within the repeat-loop, because then the algorithm is just the ordinary (revised) simplex



algorithm.

consider now a new entry into the repeat-loop

=> we compute $\theta_1$

Let $r$ be the index at which the minimum is attained in the computation of $\theta_1$

Sub-Claim: column $r$ is admissible and has negative reduced cost in the new (RP)

$$(\pi^*)^T A_r = \pi^T A_r + \theta_1 (\pi')^T A_r = \pi^T A_r + \frac{c_r - \pi^T A_r}{(\pi')^T A_r} \cdot (\pi')^T A_r = c_r$$

=> $A_r$ is admissible w.r.t. $\pi^*$ => $r$ admissible in the new (RP)

in the new (RP) column $r$ has reduced cost (see Proof of Theorem 6.3 in this subsection)

$$0 - (\pi')^T A_r < 0$$

as $(\pi')^T A_r > 0$ by definition of $\theta_1$

Sub-Claim => when entering the repeat-loop, we may choose column $r$ as pivot column in the sense of the ordinary simplex algorithm with monotonically decreasing cost

Claim => adapting the lexicographic rule to the sequence of basic solutions of $(I \mid A)$ yields termination ❑

Deriving (P), (D), (RP), (DRP)

We consider the formulation of (SP) from Section 4.3

(P) min $c^T f$

$Af = b$ ($A$ = vertex-edge-incidence matrix)

$f \geq 0$

where the row of vertex $t$ is deleted

The dual LP is

(D) max $\pi_s - \pi_t$

$\pi_i - \pi_j \leq c_{ij}$ for all edges $(i, j) \in E(G)$

$\pi_i$ unrestricted

$\pi_t = 0$ (corresponds to deleted row t)

The set of admissible columns is

$IJ = \{ (i,j) \in E \mid \pi_i - \pi_j = c_{ij} \}$



(RP) then is

min $\xi = \sum_{i=1,\ldots,n-1} x_i^a$

$$x^a + Af = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \leftarrow \text{row } s$$

$f_{ij} \geq 0$ for all edges $(i,j) \in E(G)$

$f_{ij} = 0$ for all edges $(i,j) \notin IJ$

$x_i^a \geq 0$ for $i = 1, \ldots, n-1$

The associated dual (DRP) is

max $w = \pi_s$

$\pi_i - \pi_j \leq 0$ for all edges $(i, j) \in IJ$

$\pi_i \leq 1$ for $i = 1, \ldots, n-1$ (obtained from the columns of the $x_i^a$)

$\pi_t = 0$

Interpretation of the primal-dual algorithm

(1) $\xi_{opt} = 0$ in (RP) <=> there is a path from s to t using only edges from IJ.

Each such path is an optimal solution of (P), i.e., a shortest s,t-path

Proof

"=>"

Let $\xi_{opt} = 0$

=> every basic optimal solution of (RP) is an s,t-path with $f_{ij} = 0$ for all edges $(i,j) \notin IJ$

=> this path uses only edges from IJ.

"<="

every s,t-path containing only edges from IJ is feasible in (RP) and has $\xi = 0$

=> this path is optimal for (P) because of the primal-dual method (is satisfies complementary slackness) ❑

(2) If there is no path from s to t with edges only from IJ, then $\pi'$ with

$$\pi'_i := \begin{cases} 0 & t \text{ can be reached from } i \text{ via edges from } IJ \text{ or } i = t \\ 1 & \text{otherwise} \end{cases}$$

is optimal for (DRP)

Proof



$\pi'$ is feasible for (DRP)

$\pi_i \le 1$ and $\pi_t = 0$ holds by definition

Assume that $\pi_i' - \pi_j' \le 0$ is violated for edge $(a,b) \in IJ$

$\pi'$ has only values 0 and 1 => $\pi_a' = 1$ and $\pi_b' = 0$

Definition of $\pi'$ => t can be reached from b via edges from IJ

$(a,b) \in IJ$ => t can be reached from a via edges from IJ

=> $\pi_a' = 0$, a contradiction

$\pi'$ is optimal for (DRP)

The objective function is $\max w = \pi_s$

Constraint $\pi_s \le 1$ => every $\pi$ with $\pi_s = 1$ is optimal

=> $\pi'$ is optimal ❑

(3) For $\xi_{opt} > 0$ and $\pi'$ defined in (2) we obtain

$\theta_1 = \min \{ c_{ij} - (\pi_i - \pi_j) \mid (i, j) \notin IJ, \ \pi_i' - \pi_j' = 1 \}$

Proof

Let $\xi_{opt} > 0$ and $\pi'$ be defined as in (2), so in particular optimal for (DRP).

(6.7) implies

$$\theta_1 = \min \{ \frac{c_{ij} - \pi^T A_{ij}}{(\pi')^T A_{ij}} \mid (i,j) \notin IJ,\ (\pi')^T A_{ij} > 0 \}$$

$(\pi')^T A_{ij} = \pi_i' - \pi_j' > 0 \iff \pi_i' = 1$ and $\pi_j' = 0 \Rightarrow (\pi')^T A_{ij} = \pi_i' - \pi_j' = 1$ ❑

(4) The primal-dual algorithm reduces (SP) to a sequence of reachability problems

Can t be reached from i via edges from IJ ?

or., after inverting the orientation of all edges,

Which vertices can be reached from t via edges from IJ ?

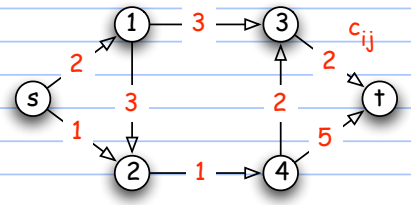
Proof

Follows from (1)-(3) ❑

6.5 Example

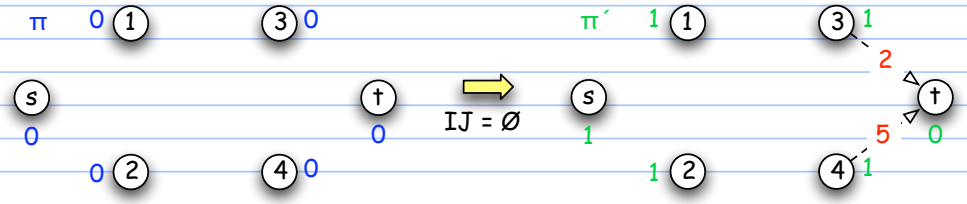
Input data






$c_{ij} \geq 0 \Rightarrow \pi = 0$ is feasible in (D)

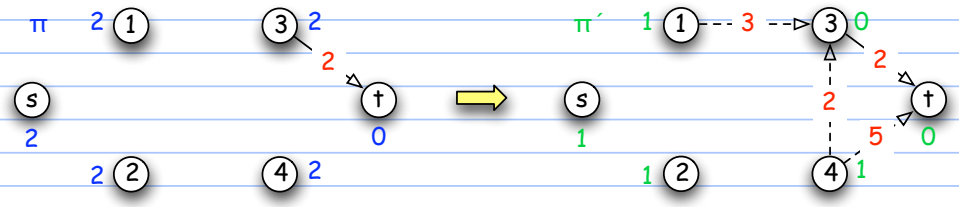
Iteration 1




$\theta_1 = \min \{ c_{ij} - (\pi_i - \pi_j) \mid (i,j) \notin IJ,\ \pi_i' - \pi_j' = 1 \} = 2$ for edge (3,t)

$\Rightarrow \pi^* = \pi + \theta_1 \pi' = (0,\ldots,0)^T + 2 \cdot (1,1,1,1,1,0)^T = (2,2,2,2,2,0)^T$

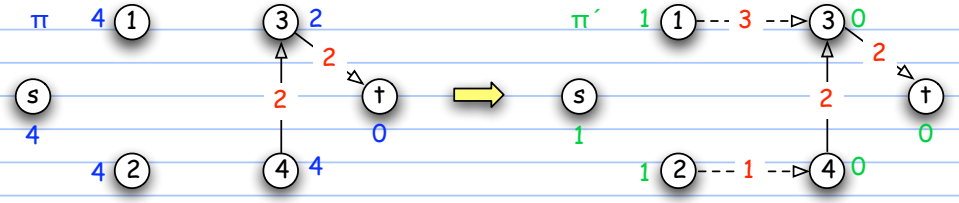
Iteration 2




$\theta_1 = \min\{ 3 - (2 - 2), 2 - (2 - 2), 5 - (2 - 0) \} = 2$ for edge (4,3)

$\Rightarrow \pi^* = \pi + \theta_1 \pi' = (2,2,2,2,2,0)^T + 2 \cdot (1,1,1,0,1,0)^T = (4,4,4,2,4,0)^T$

Iteration 3

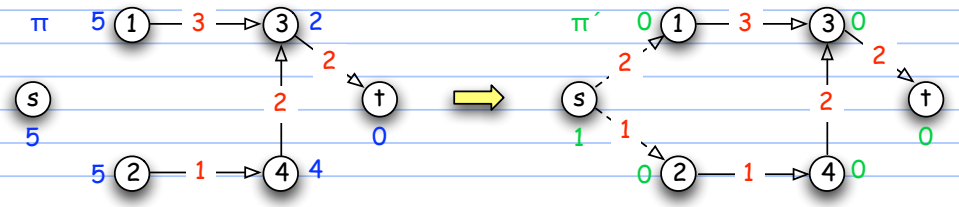



$\theta_1 = \min\{ 3 - (4 - 2), 1 - (4 - 4) \} = 1$ for edges (1,3) and (2,4)

$\Rightarrow \pi^* = \pi + \theta_1 \pi' = (4,4,4,2,4,0)^T + 1 \cdot (1,1,1,0,0,0)^T = (5,5,5,2,4,0)^T$

Iteration 4

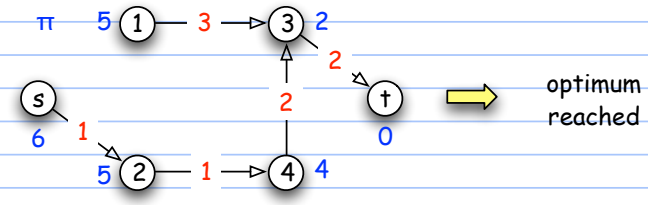





$\theta_1 = \min\{ 2 - (5 - 5), 1 - (5 - 5) \} = 1$ for edge (s,2)

$\Rightarrow \pi^* = \pi + \theta_1 \pi' = (5,5,5,2,4,0)^T + 1 \cdot (1,0,0,0,0,0)^T = (6,5,5,2,4,0)^T$

Iteration 5




$\pi = (6,5,5,2,4,0)^T$ is dual optimal = vector of shortest distances to t

Detailed interpretation of the different steps

(1) Define W as

$W := \{ i \in V \mid t$ can be reached from $i$ via edges from $IJ \} = \{ i \in V \mid \pi_i' = 0 \}$

$\pi_i$ remains unchanged as soon as $i \in W$, since $\pi_i' = 0$ afterwards

(2) When an edge $(i, j)$ enters $IJ$, it stays in $IJ$,

because $\pi_i$ and $\pi_j$ change by the same amount $\Rightarrow$ $\pi_i - \pi_j$ stays the same

(3) $i \in W \Rightarrow \pi_i$ = length of a shortest path from $i$ to $t$

(inductive proof)

In every iteration of the algorithm, one adds those vertices from $V - W$ to $W$ that are closest to $t$

(inductive Proof)

Consequence

The primal-dual algorithm for (SP) with $c \geq 0$ is essentially Dijkstra's algorithm, as in the chord model in Section 4.3



Deriving (P), (D), (RP), (DRP)

Primal LP (P) and dual LP (D)

We consider the formulation of the transportation problem from Section 4.1

(P) min $\sum_{i,j} c_{ij} f_{ij}$ s.t.

$\sum_j f_{ij} = a_i$ for all $i$ (pick up supply $a_i$ from vertex $i$)

$\sum_i f_{ij} = b_j$ for all $j$ (deliver demand $b_j$ to vertex $j$)

$f_{ij} \geq 0$ for all $i, j$

with (w.o.l.g.) $\sum_i a_i = \sum_j b_j$

Introduce dual variables $\alpha_i$, $\beta_j$ for the two groups of constraints

$\alpha_i$ $\sum_j f_{ij} = a_i$ for all $i$

$\beta_j$ $\sum_i f_{ij} = b_j$ for all $j$

The dual (D) then is

(D) max $\sum_i a_i \alpha_i + \sum_j b_j \beta_j$ s.t.

$\alpha_i + \beta_j \leq c_{ij}$ for all $i, j$

$\alpha_i, \beta_j$ unconstrained

A feasible solution of (D) is

$\alpha_i = 0$ for all $i$

$\beta_j = \min_i c_{ij}$ for all $j$ does not require that all $c_{ij} \geq 0$)

Restricted primal (RP)

The set of admissible columns is

$IJ = \{ (i,j) \in E \mid \alpha_i + \beta_j = c_{ij} \}$

Restricted primal (RP)

$\min \xi = \sum_{i=1,\dots,m+n} x_i^a$

$\sum_j f_{ij} + x_i^a = a_i$ for $i = 1, \dots, m$

$\sum_i f_{ij} + x_{m+j}^a = b_j$ for $j = 1, \dots, n$

$f_{ij} \geq 0$ for all edges $(i,j) \in IJ$

$f_{ij} = 0$ for all edges $(i,j) \notin IJ$

$x_i^a \geq 0$ for $i = 1, \dots, m+n$

We modify (RP) by substituting the artificial variables $x_i^a$ in the objective function and obtain (with $f_{ij} = 0$ for all edges $(i,j) \notin IJ$)



$$\xi = \sum_i a_i + \sum_j b_j - 2 \sum_{(i,j) \in IJ} f_{ij}$$

($\sum_i a_i + \sum_j b_j$: constant)

=> minimizing $\xi$ <=> maximizing $\sum_{(i,j) \in IJ} f_{ij}$

Deleting the artificial variables then yields (because of $x_i^a \geq 0$)

(RP') max $\sum_{(i,j) \in IJ} f_{ij}$

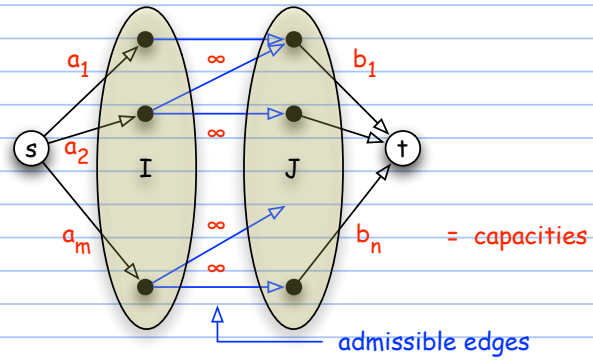
$\sum_j f_{ij} \leq a_i$ for $i = 1, \dots, m$

$\sum_i f_{ij} \leq b_j$ for $j = 1, \dots, n$

$f_{ij} \geq 0$ for all edges $(i,j) \in IJ$

$f_{ij} = 0$ for all edges $(i,j) \notin IJ$

=> (RP') corresponds to a max-flow problem in the graph $G$ of admissible edges




The primal-dual algorithm yields:

f is optimal in (P) <=> the maximum flow value fulfills $v(f) = \sum_i a_i = \sum_j b_j$

The dual (DRP) of (RP)

Introduce dual variables $u_i$, $v_j$ for the two groups of constraints

$u_i$ $\sum_j f_{ij} + x_i^a = a_i$ for $i = 1, \dots, m$

$v_j$ $\sum_i f_{ij} + x_{m+j}^a = b_j$ for $j = 1, \dots, n$

The dual of (RP) is

(DRP) max $w = \sum_i a_i u_i + \sum_j b_j v_j$ s.t.



$u_i + v_j \le 0$ for all $(i,j) \in IJ$

$u_i, v_j \le 1$

$u_i, v_j$ unrestricted

6.6 Lemma (Optimal solution of (DRP))

Let $\xi_{opt} > 0$ in (RP) and let $f$ be a maximum $s,t$-flow in $G$.

Let $I^* \subseteq I$ be the set of vertices that can be reached from $s$ in $G_f$ (i.e. there is a flow augmenting path from $s$ to these vertices).

Let $J^* \subseteq J$ be the set of vertices that can be reached from $s$ in $G_f$ (i.e. there is a flow augmenting path from $s$ to these vertices).

Then

$\alpha_i := 1$ if $i \in I^*$ $\alpha_i := -1$ if $i \notin I^*$

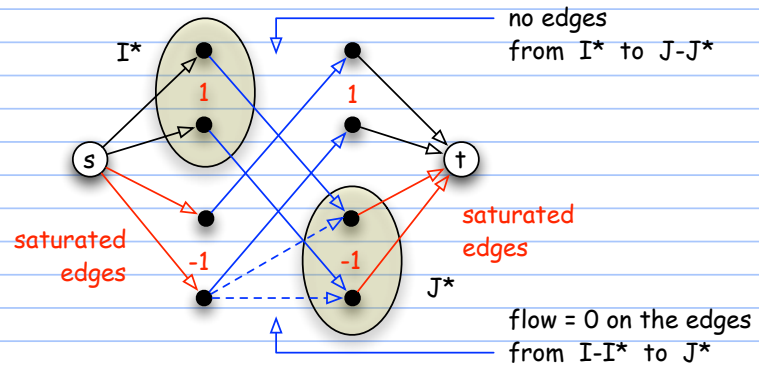
$\beta_j := -1$ if $j \in J^*$ $\beta_j := 1$ if $j \notin J^*$

is an optimal solution of (DRP).

Proof

From ADM I we know that $X := \{s\} \cup I^* \cup J^*$ is a cut of minimum capacity of $G$, and that every max-flow algorithm computes such a cut. Hence the sets $I^*$ and $J^*$ can be determined efficiently.

We analyze this cut of $G$




(1) there is no edge $(i,j)$ from $I^*$ to $J-J^*$

otherwise $j \in J^*$ because of the infinite capacity of $(i,j)$

(2) $f_{ij} = 0$ for all edges $(i,j)$ from $I-I^*$ to $J^*$

otherwise $(j,i)$ is a backward edge in $G_f$, implying $i \in I^*$

(3) edges $(s,i)$ from $s$ to $I-I^*$ are saturated

otherwise $i \in I^*$

(4) the edges $(j,t)$ from $J^*$ to $t$ are saturated



otherwise there is a flow augmenting $s,t$-path

(5) the flow value is $v(f) = \sum_{i \in I-I^*} a_i + \sum_{j \in J^*} b_j$

since: $v(f)$ = net outflow out of $X = \{s\} \cup I^* \cup J^*$ into $V(G) - X$

=> (5) follows from (1) - (4)

(A) $\alpha_i$ and $\beta_j$ are feasible for (DRP)

Show that $\alpha_i + \beta_j \le 0$

If $\alpha_i + \beta_j > 0$ => $\alpha_i = 1$ and $\beta_j = 1$ => $i \in I^*$ and $j \in J-J^*$

=> a contradiction to (1)

(B) $\alpha_i$ and $\beta_j$ are optimal for (DRP)

The objective function value for $\alpha_i$ and $\beta_j$ is

$$w = \sum_i a_i\alpha_i + \sum_j b_j\beta_j$$

$$= \sum_{i \in I^*} a_i - \sum_{i \in I-I^*} a_i - \sum_{j \in J^*} b_j + \sum_{j \in J-J^*} b_i$$

Because of (DRP') and (5)

$$\xi_{opt} = \sum_i a_i + \sum_j b_j - 2v(f)$$

$$= \sum_i a_i + \sum_j b_j - 2\left(\sum_{i \in I-I^*} a_i + \sum_{j \in J^*} b_j\right) = w$$

Weak duality => $\alpha_i$ and $\beta_j$ are optimal ❑

Updating the dual solution

Let $\alpha_i$ and $\beta_j$ be a dual solution of (D) and let $\alpha_i'$ and $\beta_j'$ be the optimal solution of (DRP)

If $\xi_{opt} > 0$, then there are 2 cases in the primal-dual algorithm (Theorem 6.2)

Case 1: $\alpha_i' + \beta_j' \leq 0$ for all $(i,j) \notin IJ$

=> (P) is infeasible by Theorem 6.2

this cannot happen as (P) has a feasible solution,

e.g. $f_{ij} = (1/\sum_k a_k) \cdot a_i \cdot b_j$

Case 2: $\alpha_i' + \beta_j' > 0$ for some $(i,j) \notin IJ$

So this case is the standard case. (6.7) yields

$$\theta_1 = \min \{ \frac{c_{ij} - \pi^T A_{ij}}{(\pi')^T A_{ij}} \mid (i,j) \notin IJ,\ (\pi')^T A_{ij} > 0 \}$$

$$= \min \{ \frac{c_{ij} - \alpha_i - \beta_j}{\alpha_i' + \beta_j'} \mid (i,j) \notin IJ,\ \alpha_i' + \beta_j' > 0 \}$$

$$= \min \{ \frac{c_{ij} - \alpha_i - \beta_j}{2} \mid i \in I^*,\ j \notin J^* \}$$

We summarize



6.7 Lemma (Updating the dual solution)

Let $\xi_{opt} > 0$ in (RP) and let $f$ be a maximum s,t-flow in $G$.

Let $I^* \subseteq I$ be the set of vertices that can be reached from $s$ in $G_f$

Let $J^* \subseteq J$ be the set of vertices that can be reached from $s$ in $G_f$

Then

$$\theta_1 = \min \{ \frac{c_{ij} - \alpha_i - \beta_j}{2} \mid i \in I^*,\ j \notin J^* \}$$

and the new dual solution is obtained as

$\alpha_i^* = \alpha_i + \theta_1$ if $i \in I^*$ $\alpha_i^* = \alpha_i - \theta_1$ if $i \notin I^*$

$\beta_j^* = \beta_j - \theta_1$ if $j \in J^*$ $\beta_j^* = \beta_j + \theta_1$ if $i \notin J^*$

Every optimal flow of the old (RP') stays feasible in the new (RP').

Proof

The new dual solution is obtained as $\pi^* := \pi + \theta_1 \pi'$

Hence the values of $\alpha_i^*$ and $\beta_j^*$ follow from the value of $\theta_1$ and Lemma 6.6

It remains to show that the optimal flow stays feasible. This follows already from Theorem 6.3, since the

optimal flow is a basic feasible solution of (RP'). We give a direct proof below.

Claim: edges (i,j) with positive flow stay admissible in the new (RP)

$f_{ij} > 0$ => edge (i,j) is admissible in the old (RP) => $\alpha_i + \beta_j = c_{ij}$

analysis of the cut of $G$ (Proof of Lemma 6.6) => two possible cases

Case 1: $i \in I^*$ and $j \in J^*$

=> $\alpha_i^* + \beta_j^* = \alpha_i + \theta_1 + \beta_j - \theta_1 = \alpha_i + \beta_j = c_{ij}$

Case 2: $i \in I - I^*$ and $j \in J - J^*$

=> $\alpha_i^* + \beta_j^* = \alpha_i - \theta_1 + \beta_j + \theta_1 = \alpha_i + \beta_j = c_{ij}$ ❑

The primal-dual algorithm for the transportation problem

Algorithm alpha-beta

Input

Instance of the transportation problem, i.e., numbers $a_i > 0$, $b_j > 0$, and $c_{ij}$ with $\sum_i a_i = \sum_j b_j$

Output

A minimum cost transportation plan $f_{ij}$

Method



Determine a feasible solution of (D) by

$\alpha_i := 0$ for all i

$\beta_j := \min_i c_{ij}$ for all j

repeat

construct graph $G$ of admissible edges from $IJ := \{ (i,j) \in E \mid \alpha_i + \beta_j = c_{ij} \}$

compute a maximum s,t-flow $f$ in $G$ // warmstart with the flow from the previous iteration is possible

if $v(f) < \sum_i a_i$ then

set $I^* := \{ i \in I \mid$ there is a flow augmenting s,i-path in $G_f \}$

set $J^* := \{ j \in J \mid$ there is a flow augmenting s,j-path in $G_f \}$

set

$$\theta_1 := \min \{ \frac{c_{ij} - a_i - \beta_j}{2} \mid i \in I^*, j \notin J^* \}$$

and take as new dual solution

$\alpha_i := \alpha_i + \theta_1$ if $i \in I^*$ $\quad$ $\alpha_i := \alpha_i - \theta_1$ if $i \notin I^*$

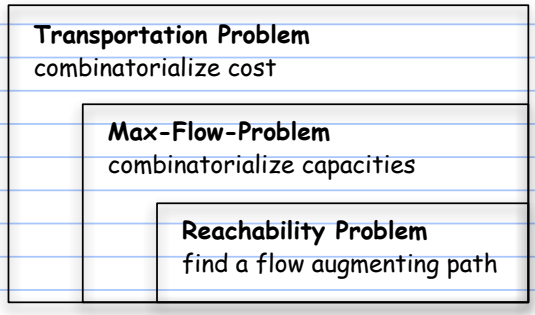
$\beta_j := \beta_j - \theta_1$ if $j \in J^*$ $\quad$ $\beta_j := \beta_j + \theta_1$ if $i \notin J^*$

until $v(f) = \sum_i a_i$

return flow f

Interpretation

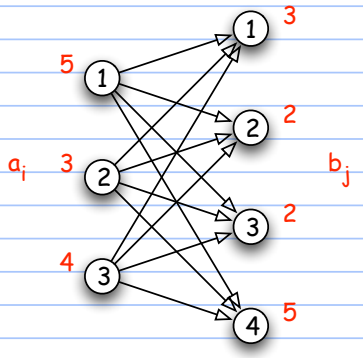
For the transportation problem, the paradigm of the primal-dual algorithm leads to two nested loops of reachability problems . The loops "combinatorialize" cost and capacities.




6.8 Example

Input data






| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 2 | 1 | 3 | 4 |
| 2 | 1 | 2 | 2 | 3 |
| 3 | 3 | 1 | 4 | 2 |

$c_{ij}$ matrix

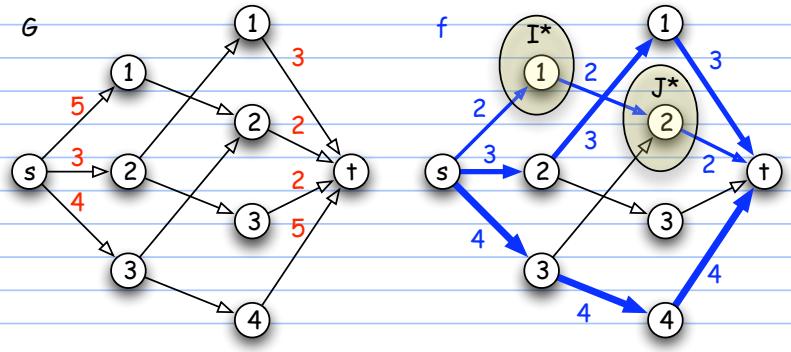
Calculating the dual solution

| $\alpha_i$ \ $\beta_j$ | 1 | 1 | 2 | 2 |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| 0 1 | 2 | (1) | 3 | 4 |
| 0 2 | (1) | 2 | (2) | 3 |
| 0 3 | 3 | (1) | 4 | (2) |

(○) = $\min_i c_{ij}$ marks the admissible edges (i,j)

Iteration 1

Construct graph $G$ of admissible edges and compute a maximum s,t-flow in $G$ and the sets $I^*$ and $J^*$




flow value $v(f) = 9 < \sum_i a_i = 12$ => update the dual solution

Updating the dual solution

Determine $\theta_1$

$$\theta_1 := \min \{ \frac{c_{ij} - \alpha_i - \beta_j}{2} \mid i \in I^*, j \notin J^* \}$$

| | $\beta_j$ | 1 | 1 | 2 | 2 |
|---|---|---|---|---|---|
| $\alpha_i$ | | 1 | 2 | 3 | 4 |
| 0 | 1 | 2 | 1 | 3 | 4 |
| 0 | 2 | 1 | 2 | 2 | 3 |
| 0 | 3 | 3 | 1 | 4 | 2 |

$c_{ij}$ matrix



$I^* = \{ 1 \}$, $J^* = \{ 2 \}$

edge (1,1) -> $(2 - 0 - 1) / 2 = 1/2$

edge (1,3) -> $(3 - 0 - 2) / 2 = 1/2$ => $\theta_1 = 1/2$

edge (1,4) -> $(4 - 0 - 2) / 2 = 1$

Computing the new dual solution

$\alpha_i := \alpha_i + \theta_1$ if $i \in I^*$ $\quad$ $\alpha_i := \alpha_i - \theta_1$ if $i \notin I^*$

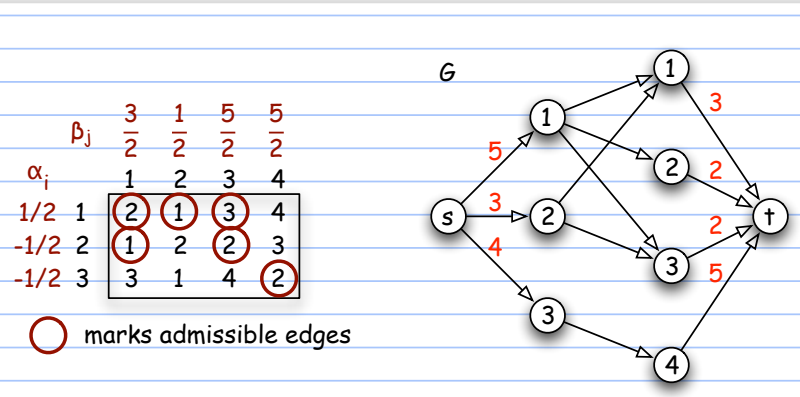
$\beta_j := \beta_j - \theta_1$ if $j \in J^*$ $\quad$ $\beta_j := \beta_j + \theta_1$ if $i \notin J^*$

So $\alpha_1 = 1/2$ $\quad$ $\alpha_2 = -1/2$ $\quad$ $\alpha_3 = -1/2$

$\beta_1 = 3/2$ $\quad$ $\beta_2 = 1/2$ $\quad$ $\beta_3 = 5/2$ $\quad$ $\beta_4 = 5/2$

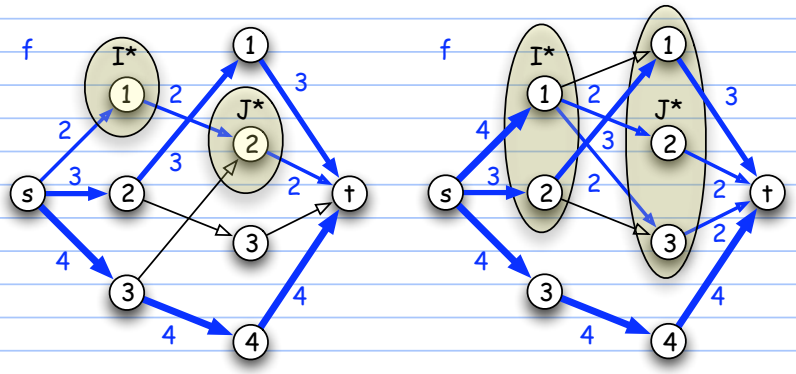
Iteration 2

Construct graph $G$ of admissible edges




Compute a maximum s,t-flow in $G$ and the sets $I^*$ and $J^*$






flow value $v(f) = 11 < \sum_i a_i = 12$ => update the dual solution

Updating the dual solution

Determine $\theta_1$

$$\theta_1 := \min\left\{ \frac{c_{ij} - \alpha_i - \beta_j}{2} \mid i \in I^*,\ j \notin J^* \right\}$$

| $\alpha_i$ | | $\beta_j$ | $\frac{3}{2}$ | $\frac{1}{2}$ | $\frac{5}{2}$ | $\frac{5}{2}$ |
|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 |
| 1/2 | 1 | | 2 | 1 | 3 | 4 |
| -1/2 | 2 | | 1 | 2 | 2 | 3 |
| -1/2 | 3 | | 3 | 1 | 4 | 2 |

$I^* = \{ 1, 2 \},\ J^* = \{ 1, 2, 3 \}$

edge (1,4) -> (4 - 1/2 - 5/2) / 2 = 1/2

edge (2,4) -> (3 + 1/2 - 5/2) / 2 = 1/2 => $\theta_1 = 1/2$

Compute the new dual solution

$\alpha_i := \alpha_i + \theta_1$ if $i \in I^*$ $\quad \alpha_i := \alpha_i - \theta_1$ if $i \notin I^*$

$\beta_j := \beta_j - \theta_1$ if $j \in J^*$ $\quad \beta_j := \beta_j + \theta_1$ if $i \notin J^*$

So $\alpha_1 = 1 \quad \alpha_2 = 0 \quad \alpha_3 = -1$

$\beta_1 = 1 \quad \beta_2 = 0 \quad \beta_3 = 2 \quad \beta_4 = 3$

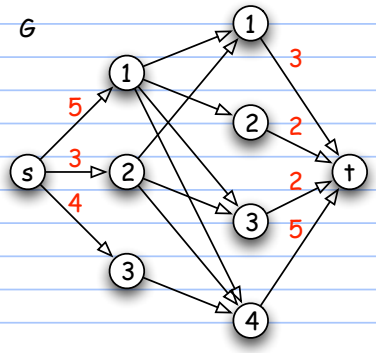
Iteration 3

Construct graph $G$ of admissible edges

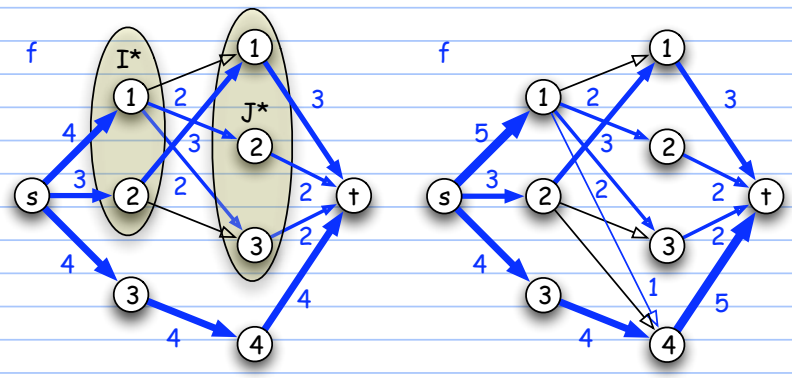


| $\alpha_i$ | | $\beta_j$ | 1 | 0 | 2 | 3 |
|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 |
| 1 | 1 | | (2) | (1) | 3 | (4) |
| 0 | 2 | | (1) | 2 | (2) | (3) |
| -1 | 3 | | 3 | 1 | 4 | (2) |

(○) marks admissible edges




Compute a maximum s,t-flow in $G$ and the sets $I^*$ and $J^*$




previous flow  new flow

flow value $v(f) = 12 = \sum_i a_i = 12$ => have constructed an optimal solution

Runtime of the algorithm

Let w.o.l.g. $m \le n$ => $G$ has $O(n)$ vertices and $O(n^2)$ edges

The primal objective of (RP') increases with every flow augmenting path and is bounded from above by $\sum_i a_i$ .

=> total runtime for flow augmentations is $(\sum_i a_i) \cdot O(\text{breadth first search}) = (\sum_i a_i) O(n^2)$



All other computations (constructing $G$, $\theta_1$, the new dual solution) are in $O(n^2)$ and happen at most $(\sum_i a_i)$ times

=> Total runtime of the primal-dual algorithm is $(\sum_i a_i) O(n^2)$

=> the primal-dual algorithm is only pseudo-polynomial

It can be improved by Capacity Scaling for the $a_i$ and $b_j$ (see ADM I, Section 6.4, for capacity scaling)

=> essentially only $\log(\max\{a_i, b_j\})$ many max-flow problems with runtime $O(n^3)$

An interesting special case is the assignment problem, which is defined by $a_i = b_j = 1$ (then $n = m$)

=> $\sum_i a_i = n$ => runtime $O(n^3)$

The alpha-beta algorithm was first developed for maximum weighted matching in bipartite graph (Paul Kuhn 1955) and is known as Hungarian method, see e.g.

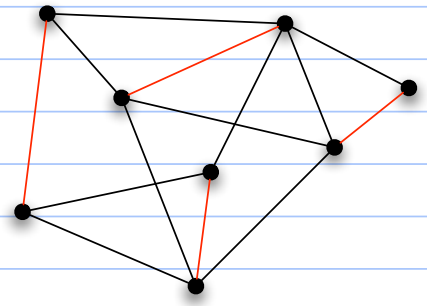
A. Schrijver, Combinatorial Optimization: Polyhedra and Efficiency

Volume 1, Chapter 17.2

- Goal of this section
  - Sketch of a primal dual algorithm for weighted (perfect) matching.

    This closes the gap from ADM I, Section 7.1

- Matching
  - Let $G$ be an undirected graph. A matching of $G$ is
    - a set $M \subseteq E(G)$ of edges such that no 2 edges of M share an endpoint
    - a matching M is called perfect if every vertex of $G$ is incident to an edge of M
    - a graph with a perfect matching (red edges)





- Maximum weight matching problem (MWMP)
  - Instance
    - Undirected graph $G$, edge weights $c(e)$
  - Task
    - Find a matching M with maximum weight c(M)

      $c(M) = \sum_{e \in M} c(e)$

- Minimum weight perfect matching problem (MWPMP)
  - Instance
    - Undirected graph $G$, edge weights $c(e)$
  - Task
    - Find a perfect matching $M$ with minimum weight c(M)

      or report that there is no perfect matching

- 6.8 Lemma (Equivalence of matching problems)

MWMP and MWPMP are equivalent in the sense that there is a simple transformation from one problem to the other such that one can construct from an optimal solution of one problem an optimal solution of the other.

Proof

"=>"

Let (G,c) be an instance of the minimum weight perfect matching problem.

Choose K large enough so that c'(e) := K - c(e) > 0 for all edges e, and only maximum cardinality matchings of G have maximum weight w.r.t. c'. ( $K := 1 + \sum_{e \in M} |c(e)|$ suffices)

Let M be an optimal solution of the maximum weight matching problem for (G, c')

M has maximum cardinality => M is a perfect matching for (G,c) or there is no perfect matching in G

If M is perfect, then $c'(M) = Kn/2 - \sum_{e \in M} c(e)$. So M has maximum weight w.r.t. c' iff M has minimum weight w.r.t. c.

"<="

Let (G,c) be an instance of the maximum weight matching problem.

Add |V(G)| many new vertices to G and so many edges that the new graph G' is complete.

Set c'(e) := - c(e) if $e \in E(G)$ and c'(e) := 0 if e is a new edge.

Let M' be an optimal solution of the minimum weight perfect matching problem for (G', c')



=> $M := M' \cap E(G)$ is an optimal solution of the maximum weight matching problem ❑

The primal-dual algorithm for the minimum weight perfect matching problem

Primal LP (P)

There is no obvious LP-formulation. The following theorem was one of the pioneering results of Edmonds.

Given an instance (G,c) with G = (V, E), he considers the LP (P)

$\min \sum_{e \in E} c(e)x_e$

$x(\delta(v)) = 1$ for all $v \in V$

$x(\delta(S)) \geq 1$ for all odd vertex sets S of G

$x \geq 0$

Here $x(\delta(S)) := \sum_{e \in \delta(S)} x_e$

6.9 Theorem (Matching polytope Theorem, Edmonds 1965)

Let (G,c) be an instance of the minimum weight perfect matching problem. Then:

(1) G has a perfect matching <=> (P) has a feasible solution

(2) In this case the minimum weight of a perfect matching of G is equal to the optimal value of (P).

Proof

The primal-dual algorithm constructs an optimal solution $x$ of (P) (if there is one) that is a perfect matching of $G$.

=> Theorem 6.9 and Lemma 3.5 imply that all basic feasible solutions of (P) correspond to perfect matchings. ❑

Dual LP (D)

We do not transform (P) into standard form. The primal-dual algorithm can be adapted also to other forms of (P).

Let $U$ be the set of all odd vertex sets of $G$. The dual LP of (P) is (D)

$\max \Sigma(y_v : v \in V) + \Sigma(Y_S : S \in U)$

$y_v + y_w + \Sigma(Y_S : e \in S \in U) \leq c(e)$ for all edges $e = vw \in E$

$Y_S \geq 0$ for all $S \in U$

$y_v$ unrestricted

Complementary slackness conditions

$x_e > 0 \Rightarrow c(e) - (y_v + y_w + \Sigma(Y_S : e \in S \in U)) = 0$



$Y_S > 0 \Rightarrow x(\delta(S)) = 1$

If $x$ is the incidence vector of a perfect matching $M$, then the complementary slackness conditions are equivalent to

$e \in M \Rightarrow c(e) - (y_v + y_w + \Sigma(Y_S : e \in S \in U)) = 0$ (6.8)

$Y_S > 0 \Rightarrow |M \cap \delta(S)| = 1$ (6.9)

Remarks on the primal-dual algorithm

(6.8) defines "admissible" edges for the matching that we look for in the (RP)

(6.9) corresponds to $|M \cap \delta(S)| = 1$, if $S$ has been shrunk to a pseudo-node $v$ (a blossom)

=> a solution of (RP) corresponds essentially to searching a perfect (so a maximum cardinality) matching in the graph of admissible edges in which all sets $S$ with $Y_S > 0$ are shrunk

=> we can compute such a solution with the algorithm for a maximum cardinality matching (e.g. the one from ADM I).

An optimal solutions of (DRP) can (similar to the transportation problem) be obtained directly from the best matching in (RP), but this is more complicated than for the transportation problem.

Altogether, the minimum weight perfect matching problem reduces to a sequence of maximum cardinality

matching problems.

- The algorithm can be implemented to run in $O(n^2m)$ time. In particular, at most $n$ variables $y_S > 0$ throughout the algorithm.

  There are improvements for dense graphs (that work only on sparse subsets of the edges)

- For details see Chapter 5.3 in

  W. J. Cook, W. H. Cunningham, W. R. Pulleyblank and A. Schrijver

  Combinatorial Optimization

  Wiley 1998

## 7. Integer linear optimization 41

- Integer linear programs (Integer Linear Program, ILP, IP) require all variables to be integer, i.e., $x_j \in \mathbb{Z}$ for all j.
  Mixed integer linear programs (Mixed Integer Linear Program, MILP, MIP) may require only some of them to be integer.
- In this section:
  - Integer variables add much modeling power. Many non-linear effects can be modeled by IPs.
  - The drawback is that IPs are NP-hard in general.

- LP relaxation of an IP
  - standard form of an IP

    $\min\ c^T x$

    $s.t.\ Ax = b$

    $x \geq 0$ and integer

    Special case: 0/1 IP or Binary Integer Program

    $\min\ c^T x$

    $s.t.\ Ax = b$

    $x_i \in \{0, 1\}$



- The LP relaxation of an IP is obtained be dropping the integrality constraints, i.e.,

  $\min\ c^T x$

  $s.t.\ Ax = b$

  $x \geq 0$

  in the general case, and

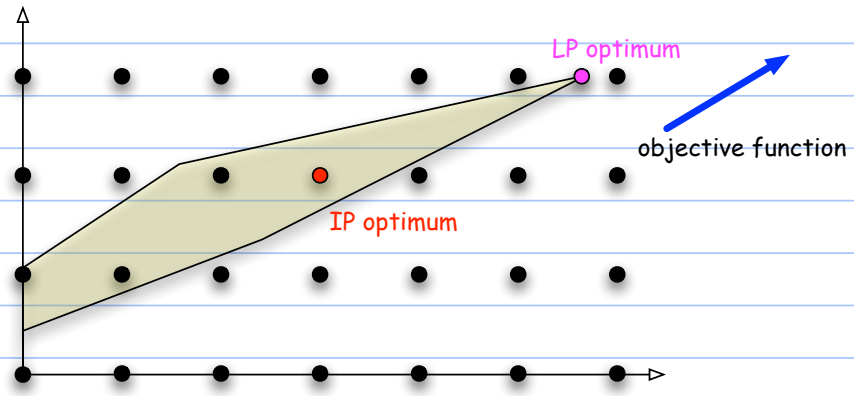
  $\min\ c^T x$

  $s.t.\ Ax = b$

  $0 \leq x_j \leq 1$

  in the 0/1 case.

- Solving the LP relaxation and rounding
  - need not yield a feasible solution
  - may work when variables have large values, but even then large errors can occur.




If $x_j \in \{0, 1\}$ (decision variables)

e.g. the $f_{ij}$ in the shortest path problem

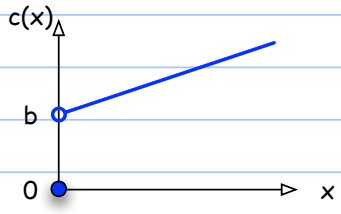
then rounding a fractional solution need not make sense

(we will use it nevertheless later with some care for approximation algorithms)

Modeling with IPs

(1) Fixed Charge Cost






$$c(x) = \begin{cases} ax + b & if \quad x > 0 \\ 0 & if \quad x = 0 \end{cases}$$

$x \in \mathbb{R}^1$, $a, b > 0$

Introduce a 0/1-variable $\delta$ and the constraint $x \leq \delta \cdot U$,

where $U$ is an upper bound for the values of $x$

Claim: the cost function can now be modeled as $c(x,\delta) = ax + b\delta$

$x > 0 \Rightarrow \delta = 1 \Rightarrow c(x) = ax + b$

$x = 0 \Rightarrow \delta = 0$ in the optimum, as $a, b > 0 \Rightarrow c(x) = 0$ ❑

(2) Disjunctive constraints

$x \geq a$ or $y \geq b$ with $a, b \geq 0$ and $x, y \geq 0$

Introduce a 0/1-variable $\delta$

Claim: The inequalities $x \geq \delta a$ and $y \geq (1-\delta)b$ model the disjunctive constraint

clear since $\delta$ is a 0/1 variable ❑

(3) Conditional constraints

if $x < a$ then $y \geq b$ else $y \geq 0$ with $a, b > 0$

Claim: A conditional constraint can be reduced to case (2)

the conditional constraint is equivalent to

$y \geq 0$

$x \geq a$ or $y \geq b$ ❑

(4) Discrete variables

$x \in \{ s_1, \ldots, s_m \}$

Claim: a discrete variable $x \in \{ s_1, \ldots, s_m \}$ can be modeled as

$x = s_1\delta_1 + \ldots + s_m\delta_m$ with $\delta_j \in \{ 0, 1 \}$ and $\delta_1 + \ldots + \delta_m = 1$

clear ❑

7.1 Example (Minimum weight perfect matching problem as IP)

Every solution of the IP



$\min \sum_{e \in E} c(e)x_e$

$x(\delta(v)) = 1$ for all $v \in V$

$x_e \in \{ 0, 1 \}$

is a perfect matching

Complexity of ILPs

7.2 Theorem (Complexity of ILPs)

(1) SATISFIABILITY (SAT) is reducible to ILP

(2) Deciding if an ILP has a feasible solution is an NP-hard problem

(3) It is NP-hard to round a feasible solution of the LP relaxation of an ILP to a feasible solution of the ILP

Proof

Consider an instance of SAT given by $m$ clauses $C_1, \ldots, C_m$ with Boolean variables $x_1, \ldots, x_n$

Introduce for every Boolean variable $x_i$ a 0/1-variable $z_i$ with $z_i = 1$ if $x_i =$ TRUE

Satisfying a clause can then be written as a linear inequality, and the existence of a satisfying truth assignment is equivalent to the existence of a feasible solution of the ILP.

Example:

$$\underbrace{x_1 \vee x_2 \vee x_3}_{C_1},\ \underbrace{x_1 \vee \bar{x}_2}_{C_2},\ \underbrace{x_2 \vee \bar{x}_3}_{C_3},\ \underbrace{x_3 \vee \bar{x}_1}_{C_4},\ \underbrace{\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3}_{C_5}$$

is equivalent to

$$z_1 + z_2 + z_3 \geq 1$$
$$z_1 + (1-z_2) \geq 1$$
$$z_2 + (1-z_3) \geq 1$$
$$z_3 + (1-z_1) \geq 1$$
$$(1-z_1) + (1-z_2) + (1-z_2) \geq 1$$
$$z_i \in \{ 0, 1 \}$$

- If every clause has $\geq 2$ literals (this is the non-trivial case), then $z_i = 1/2$ is a feasible solution of the LP relaxation. So rounding to a feasible solution of the ILP is as hard as finding a satisfying truth assignment for the given SAT instance. ❑

- Remark:
  - The proof does not show that testing for feasibility is NP-complete. To that end we need a certificate for feasibility of polynomial length (see ADM I). It is not directly obvious if such a certificate exists.



- One can, however, show that the entries $x_j$ of an integer feasible solution $x$ are not too large (a statement similar to Lemma 3.4). Thus $x$ itself can serve as a certificate. So NP-hard can be replaced by NP-complete in Theorem 7.2.

**Main question of this section**

When does an LP have integer basic solutions?

=> Then the corresponding ILP can be solved by solving the LP relaxation with the simplex algorithm.

We consider here the following special case:

When does $Ax = b$ have only integer basic solutions for an arbitrary choice of integer right hand side $b$?

This is then a property of the matrix $A$

**Totally unimodular matrices**

A quadratic matrix $B$ with integer entries is called unimodular

:<=> $\det B \in \{ -1, 1 \}$

A matrix $A$ with integer entries is called totally unimodular (TUM)

:<=> every quadratic non-singular submatrix is unimodular

First properties

$A$ TUM => $A$ has only entries $a_{ij} \in \{ -1, 0, 1 \}$

the smallest non-unimodular matrix is



$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

Given a basis $B$ of $A$ with $B = (A_{B(1)}, A_{B(2)}, \ldots, A_{B(m)})$, Cramer's rule yields

$$x_{B(i)} = \frac{\det B^i}{\det B} \quad \text{with } B^i = (A_{B(1)}, \ldots, A_{B(i-1)}, b, A_{B(i+1)}, \ldots, A_{B(m)})$$

=> $x_{B(i)}$ is integer if $A$ is TUM and $b$ is integer

**Polyhedra of linear optimization problems with integer vertices**

Let $R_1(A) := \{ x \in \mathbb{R}^n \mid Ax = b, x \geq 0 \}$ be the polyhedron of the standard form of the LP

Let $R_2(A) := \{ x \in \mathbb{R}^n \mid Ax \leq b, x \geq 0 \}$ be the polyhedron of the canonical form of the LP

Remark:

Both polyhedra are are defined here as subsets of $\mathbb{R}^n$.

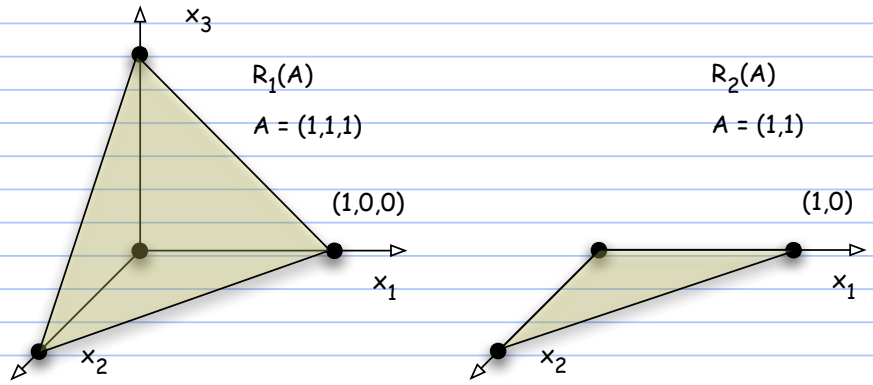
The definition of $R_2(A)$ follows the correspondence between geometric and algebraic interpretation of LPs in Section 3.3. Thus the vertices of $R_2(A)$ correspond to the basic feasible solutions of the LP $\{ Ax + s = b, x, s \geq 0 \}$ enhanced by slack variables $s$.

The vertices of $R_1(A)$ correspond also to the basic feasible solutions of the LP $\{ Ax = b, x \geq 0 \}$ because:

If $x$ is a basic feasible solution with basis $B$, then $x_N = 0$, i.e., $x$ lies in the intersection of the n-m hyperplanes $x_i = 0$ with $i \in N$ and the m hyperplanes $a_i x = b_i$, $i = 1, ..., m$.

The opposite direction follows with arguments similar to those in the proof of Theorem 3.6.

Example:




Here $x_3$ takes the role of a slack variable for $R_2(A)$



7.3 Theorem (Integrality of $R_1(A)$)

If $A$ is totally unimodular, then all vertices of $R_1(A)$ are integer for any integer right hand side $b$.

In particular, for an LP in standard form with totally unimodular matrix $A$ and integer right hand side $b$, the simplex algorithm always terminates with an integer optimal solution.

Proof:

follows from section "first properties" ❑

7.4 Theorem (Integrality of $R_2(A)$)

If $A$ is totally unimodular, then all vertices of $R_2(A)$ are integer for any integer right hand side $b$.

In particular, for an LP in canonical form with totally unimodular matrix $A$ and integer right hand side $b$, the simplex algorithm applied to the corresponding standard form with slack variables always terminates with an integer optimal solution.

Proof:

Adding slack variables gives the matrix $(A|I)$.

Let $C$ be a non-singular quadratic submatrix of $(A|I)$

=> after a suitable permutation of the rows, $C$ has the form

$$\left(\begin{array}{c|c} B & 0 \\ \hline D & I_k \end{array}\right)$$

with $B$ = quadratic submatrix of $A$

$I_k$ = (k,k)-identity matrix

=> $|\det(C)| = |\det(B)| = 1$, since $A$ is TUM

=> $(A|I)$ is TUM

=> statement with Theorem 7.3 and Theorem 3.10 ❑

So Theorems 7.3 and 7.4 say that the polyhedra $R_1(A)$ and $R_2(A)$ have integer vertices, if $A$ is totally unimodular and the right hand side $b$ is integer.

**Recognizing totally unimodular matrices**

The complexity of recognizing totally unimodular matrices has been open for a long time and was solved by Seymour only in 1980. He proved a "Decomposition Theorem" stating that every totally unimodular matrix can be constructed from "simple" totally unimodular matrices by certain construction rules. His Decomposition



Theorem leads to a polynomial algorithm for recognizing totally unimodular matrices. It has a runtime of $O((m+n)^4 m)$.

For details see Chapters 19 and 20 in

A. Schrijver

Theory of Linear and Integer Programming

Wiley 1986

We will only show a sufficient criterion

7.5 Theorem (A sufficient criterion for total unimodularity)

A matrix $A$ with entries $a_{ij} \in \{ -1, 0, 1 \}$ is totally unimodular if it fulfills (1) and (2) below:

(1) $A$ has at most 2 entries $\neq 0$ per column

(2) The rows of $A$ can be partitioned into two disjoint sets $I_1, I_2$ such that

for every column with 2 entries $\neq 0$ and the same sign, the associated rows lie in different sets $I_j$

for every column with 2 entries $\neq 0$ and different signs, the associated rows lie in the same set $I_j$

Proof by induction on the size $k$ of the quadratic submatrix

Base case k = 1

obvious, as A has only entries $a_{ij} \in \{ -1, 0, 1 \}$

Inductive step to k

Let $C$ be a quadratic non-singular (k,k)-submatrix of $A$

=> each column of $C$ has at least one entry ≠ 0

Case 1: $C$ has a column with exactly one entry $a_{ij} \neq 0$

Laplace expansion of $\det(C)$ along this column yields

$$|\det(C)| = |a_{ij}| \cdot |\det(C')|$$

where $C'$ is the submatrix of $C$ after deleting row i and column j

$C$ non-singular => $|\det(C')| \neq 0$

inductive assumption => $|\det(C')| = 1$

$a_{ij} \in \{ -1, 1 \}$ => $|\det(C)| = 1$

Case 2: all columns of $C$ have at least 2 entries ≠ 0

(1) => all columns have exactly 2 entries ≠ 0

consider the partition of the rows in $I_1, I_2$ according to (2)

=> $\sum_{i \in I_1} a_{ij} = \sum_{i \in I_2} a_{ij}$ for every column j



=> $\sum_{i \in I_1} a_i - \sum_{i \in I_2} a_i = 0$

i.e., a linear combination of the row vectors of $C$ yields the null vector

=> this contradicts that $C$ is non-singular

=> this case cannot occur ❑

7.6 Corollary (Important classes of totally unimodular matrices)

Every LP in standard form or canonical form, whose matrix of coefficients is the

1. vertex-edge incidence matrix of a digraph
2. vertex-edge incidence matrix of a bipartite graph

has only integer basic optimal solutions (for an integer right hand side b).

This covers LP formulations of

shortest path problems

max-flow problems

transportation problems

Proof

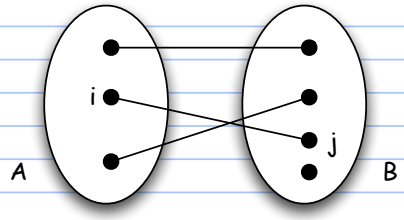
Case 1

In this case, A contains exactly one +1 and one -1 per column

=> set $I_1$ = set of all rows, $I_2$ = Ø

Case 2

Let G be a bipartite graph with bipartition A and B




=> the column of edge ij contains exactly 2 entries ≠ 0, a +1 for vertex i, and a +1 for vertex j

=> set $I_1$ = A, $I_2$ = B ❑

**The Theorem of Birkhoff & von Neumann for doubly stochastic matrices**

We show with our results a famous theorem on doubly stochastic matrices

An nxn-matrix with entries $0 \le a_{ij} \le 1$ is called doubly stochastic



:<=> each of its rows and columns sums to 1

An nxn-matrix with entries $a_{ij} \in \{ 0, 1 \}$ is called a permutation matrix

:<=> each of its rows and columns contains exactly one 1

7.7 Theorem (Birkhoff 1946, von Neumann 1953)

Every doubly stochastic nxn-matrix is a convex combination of nxn-permutation matrices

Proof

A doubly stochastic matrix M can be seen as a feasible solution of the assignment problem

$\min \sum_{i,j} c_{ij} f_{ij}$ s.t.

$\sum_j f_{ij} = 1$ for all i = 1, ..., n

$\sum_i f_{ij} = 1$ for all j = 1, ..., n

$f_{ij} \ge 0$ for all i, j

Let A be the associated matrix of coefficients and let $R_1(A)$ be the associated polyhedron of the standard form

=> $R_1(A)$ is a polytope, as the feasibility domain is bounded because of $0 \le f_{ij} \le 1$

Minkowski's Theorem (Theorem 3.9) => M is a convex combination of the vertices of $R_1(A)$

- $A$ is the vertex-edge incidence matrix of the complete bipartite graph $K_{n,n}$

  => $A$ is totally unimodular because of Corollary 7.6

  => The vertices von $R_1(A)$ are integer because of Theorem 7.3

  $0 \le f_{ij} \le 1$ => the vertices of $R_1(A)$ are permutation matrices ❑

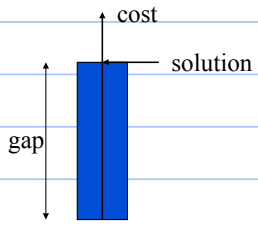


- **Goal of this section**
  - Introduction of Branch and Bound as a standard technique for solving NP-complete problems exactly, in particular IPs.

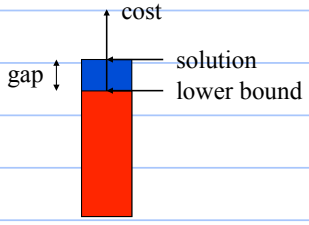
    Although quite simple, Branch and Bound is the basis and the workhorse for all commercial IP solvers, but of course improved by quite a number of additional methods and tricks.

- **The basic idea of Branch and Bound**
  - Branch and Bound (B&B) = problem dependent, cleverly organized systematic search in the set of feasible solutions for an optimal solution, or until termination with a "good" solution (i.e., one with an instance-dependent performance guarantee)

  - The use of lower bounds for a minimization problem




we do not know how good a feasible solution is, if it was computed with a heuristic (e.g. with local search)




lower bounds for the optimal value narrow the "optimality gap"

So: If the optimum is unknown, then lower bounds yield quality guarantees for solutions of a hard optimization problem

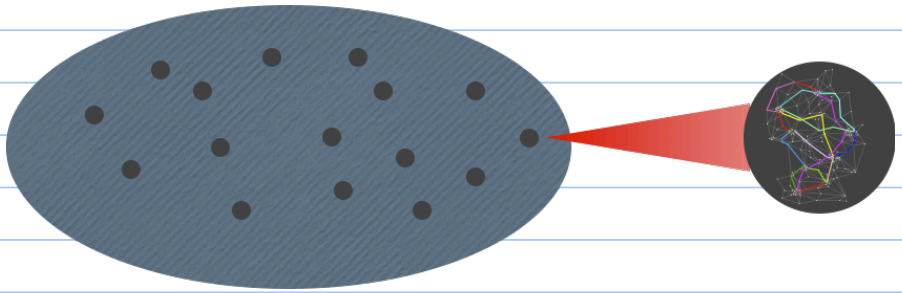
Branch & Bound



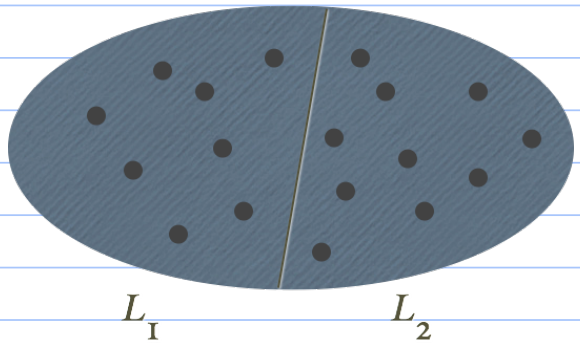
illustrated for the disjoint path problem (see Section 4.4)

We imagine the solution space (= set of feasible solutions) as a set of points

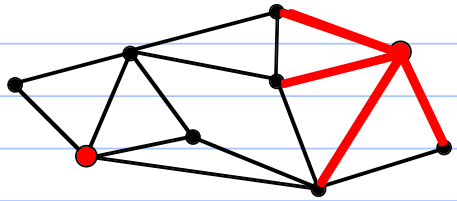
Every point represents a feasible solution



Branching = partition the current set of solutions into $\geq 2$ subsets (not necessarily disjoint)

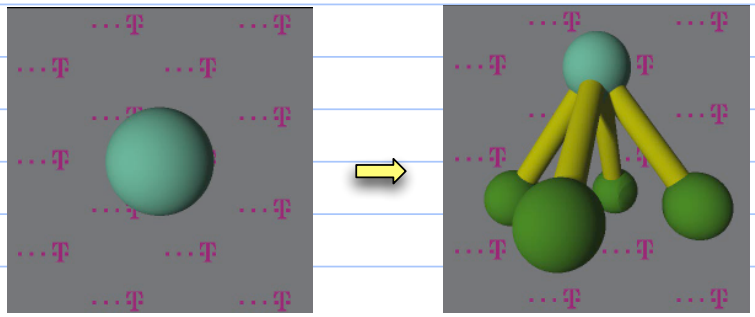



- branching is usually displayed in a tree (Branch and Bound Tree)



here: partition the set of solutions into 4 subsets depending on which red edge is used on a path between the red terminals

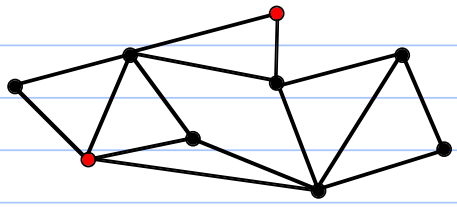




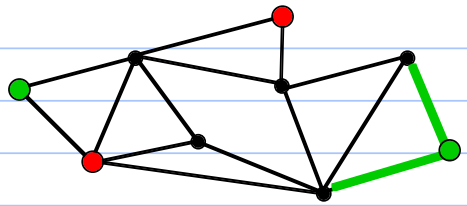
the subsets are displayed as children of the parent

- branching is iterated, this generates the B&B tree

here: the choice of the topmost red edge generates the subproblem given by the graph below, which no longer contains the topmost red edge, as it is reserved for den path connecting the red terminals above. So in the subproblem, the red terminal has been moved.

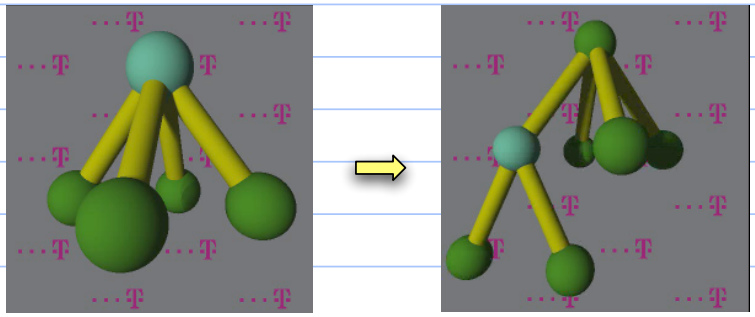


the set of feasible solutions of that subproblem can then be partitioned again w.r.t. to edges of the same or another pair of terminals

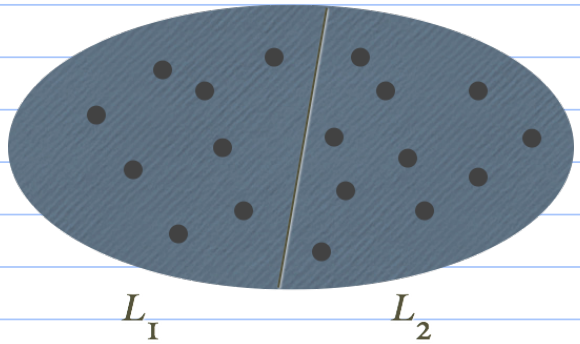


here: choose the green terminals and partition the set of solutions into 2 subsets representing the 2 ways to choose a green edge for the path





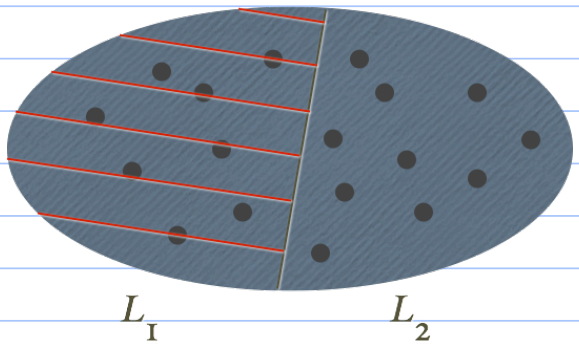
- Bounding is used for pruning branches of the B&B tree with the use of lower bounds



Assumption 1: we know a feasible solution with cost k

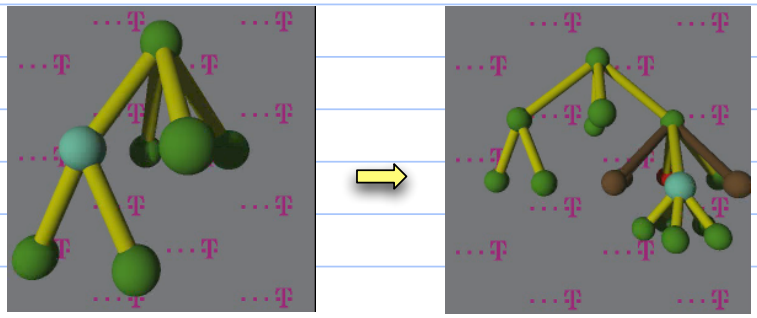
Assumption 2: we know a lower bound s for the optimal value in $L_1$

=> we need not search $L_1$ if $s \geq k$ and thus may delete the subtree of the B&B tree rooted in $L_1$




This way of deleting subtrees of the B&B tree is called pruning, and is depicted here by brown (withered) branches. The node (and its subtree) is then called fathomed.





- Branching and Bounding is used together with
  - Good search strategies for choosing the next node (= subset of feasible solutions) in the B&B tree
    - depth-first search
    - breadth-first search
    - best-first-search (node with best ( = smallest) lower bound)
    - combinations of the above
  - The tree is of course maintained implicitly and will never be generated explicitly
  - Techniques for generating good lower bounds (next chapter)

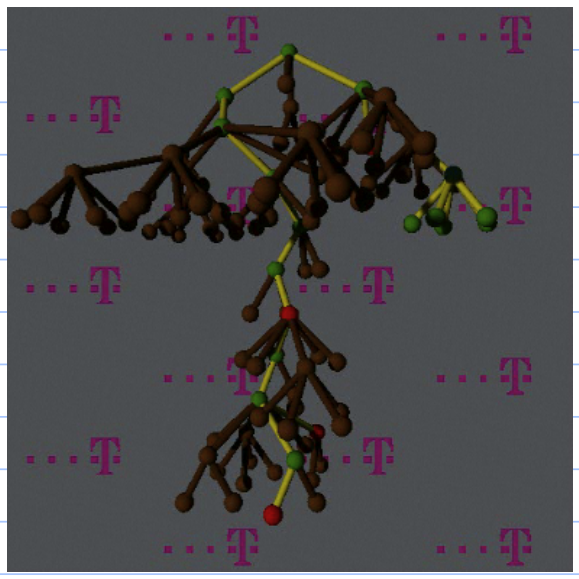
Lagrangian relaxation

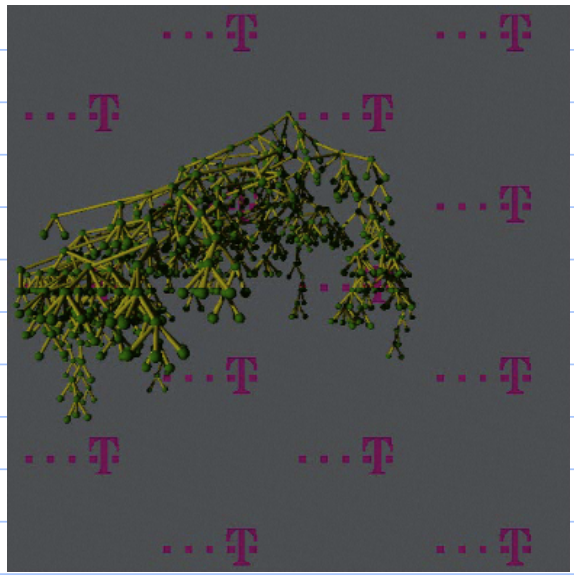
LP-relaxation (natural for IPs)

- Techniques for constructing feasible solutions in tree nodes (they provide upper bounds on the optimum)

- Runtime is exponential, depends very much on the quality of the lower bounds







good lower bounds

small B&B tree

bad lower bounds

huge B&B tree

**Generic Branch and Bound Algorithm**

- Input
  - instance I of a problem
- Output
  - feasible solution $x \in S_I$ with performance guarantee given by the objective function value c(x) and a lower bound $\ell$ for the optimum
- Ingredients
  - lower bounding strategy
  - branching strategy
  - search strategy
- Method
  - 1. Work in the root
    - consider a slightly modified, easier to solve instance I' (a relaxation) for computing a lower bound for I;
      compute the optimal solution x' of I', let z' be the objective function value;
      if $x' \in S_I$ then return x' // x' is optimal
    - set $\ell := z'$ // initial global lower bound
    - // initialize data structure D for maintaining the already generated still unsearched nodes of the B&B tree



    - add I with $\ell(I) := \ell$ to D
    - use heuristics to generate feasible solutions
      set x* := best feasible solution found
      set u := c(x*) // initial upper bound
  - 2. Main loop
    - while performance guarantee $(u-\ell)/\ell$ is not small enough and we have not run out of time or memory do
      - choose next node v of the B&B tree from D for searching // search strategy
      - if $\ell(v) \geq u$ then delete v from D // pruning
      - else
        - generated the children $v_1, \ldots, v_k$ of v // branching rule
          // union of the feasibility domains of the children = feasibility domain of v
        - for each child $v_i$ do
          - compute the optimal solution x' of (the relaxation of) the associated subproblem, let z' be its objective function value // bounding rule
          - if $x' \in S_I$ and $z' < u$ then
            - x* := x' // update the best known feasible solution

u := z' // update the global upper bound

else

if z' < u then add $v_i$ with $\ell(v_i)$ := z' to D // new subproblem

delete v from D // v is fathomed

$\ell$ := min { $\ell(w)$ | w in D } // update global lower bound

return x* and $\ell$

**Branch and Bound for IPs**

natural for bounding: LP relaxation

natural for branching: branch w.r.t. fractional variables in the LP relaxation

7.8 Example (The KNAPSACK problem, see ADM I)

KNAPSACK

Instance

n items with weight $w_i$ and profit $c_i$

a knapsack with capacity (= total weight) W



Task

find a subset $S \subseteq \{ 1, \ldots, n \}$ with

maximum value $c(S) := \sum \{ c_j \mid j \in S \}$

capacity of the knapsack is respected, i.e., $w(S) := \sum \{ w_j \mid j \in S \} \leq W$

An IP formulation of KNAPSACK

Introduce 0/1-variable $x_j$ with $x_j = 1$ if item j is put into the knapsack

$$\min \sum_j -c_j x_j \quad \text{s.t.}$$

$$\sum_j w_j x_j \leq W$$

$$x_j \in \{ 0, 1 \}$$

7.9 Lemma (Optimal solutions of the LP relaxation of KNAPSACK)

An optimal solution of the LP-relaxation

$$\min \sum_j -c_j x_j$$

$$\sum_j w_j x_j \leq W$$

$$0 \leq x_j \leq 1$$

of the IP formulation of KNAPSACK is obtained as follows

- sort and number the items is such a way that $c_1/w_1 \geq c_2/w_2 \geq \ldots c_n/w_n$ (largest profit per unit weight first)
- compute in this order the smallest k, such that $w_1 + w_2 + \ldots + w_{k+1} > W$
- set $x_1 = x_2 = \ldots = x_k = 1$

  $x_{k+1} = (W - w_1 - w_2 - \ldots - w_k)/w_{k+1}$

  $x_j = 0$ otherwise

Proof by checking complementary slackness

- the primal dual pair is given by

| | $c_1$ | $\ldots$ | $c_n$ | |
|---|---|---|---|---|
| u | $w_1$ | $\ldots$ | $w_n$ | W |
| $v_1$ | 1 | | | 1 |
| | | $\ddots$ | | |
| $v_n$ | | | 1 | 1 |
| | $x_1$ | $\ldots$ | $x_n$ | |

- complementary slackness conditions give
  - (1) $x_j > 0 \Rightarrow w_j u + v_j = c_j$
  - (2) $u > 0 \Rightarrow \sum_j w_j x_j = W$ (is satisfied by x from the lemma)



  - (3) $v_j > 0 \Rightarrow x_j = 1$
- Define a dual feasible solution that, together with x, satisfies conditions (1) and (3)
  - (3) $\Rightarrow v_{k+1} = v_{k+2} = \ldots = v_n := 0$

    $\Rightarrow$ (with (1) for $j = k+1$) $w_{k+1}u = c_{k+1} \Rightarrow u = c_{k+1}/w_{k+1}$

    $\Rightarrow$ (with (1) for $j = 1, \ldots k$) $w_j(c_{k+1}/w_{k+1}) + v_j = c_j$

    $\Rightarrow v_j := c_j - w_j(c_{k+1}/w_{k+1})$ for $j = 1, \ldots k$

    $\Rightarrow$ we have defined values for all dual variables from observing conditions (1) and (3)
  - show: this defines a dual feasible solution

    need only show $v_j \geq 0$, i.e., $c_j - w_j(c_{k+1}/w_{k+1}) \geq 0$ for $j = 1, \ldots k$.

    This follows from $c_j/w_j \geq c_{k+1}/w_{k+1}$ for $j = 1, \ldots k$ ❑

- Use the generic B&B algorithm with the following ingredients
  - lower bounding strategy = LP relaxation solved with Lemma 7.9
  - branching strategy = branch on fractional variables $x_{k+1}$
  - search strategy = best first

Instance

| j | $w_j$ | $c_j$ | $\frac{c_j}{w_j}$ |
|---|---|---|---|
| 1 | 16 | 112 | 7 |
| 2 | 15 | 90 | 6 |
| 3 | 3 | 15 | 5 |
| 4 | 3 | 12 | 4 |
| 5 | 3 | 9 | 3 |
| 6 | 4 | 12 | 3 |
| 7 | 13 | 26 | 2 |

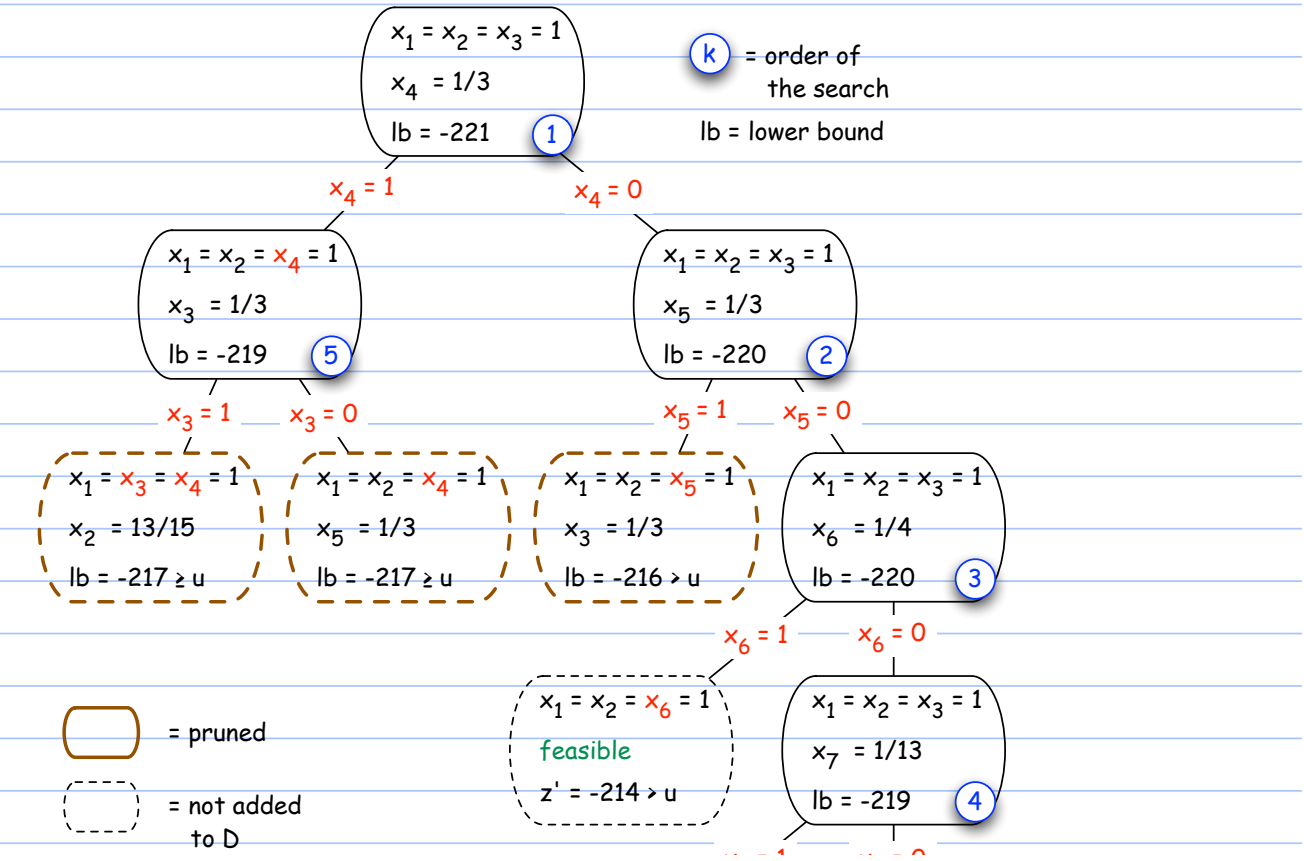
W = 35

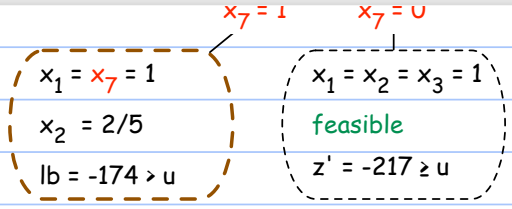
heuristic solution $x_1 = x_2 = x_3 = 1$, $x_j = 0$ otherwise => upper bound $u = -217$

LP relaxation gives $x_1 = x_2 = x_3 = 1$, $x_4 = 1/3$, $x_j = 0$ otherwise => lower bound $\ell = -221$

Branch and Bound Tree








**Using other relaxations than the LP-relaxation**

This is possible, e.g. by deleting constraints

=> the feasibility domain gets larger => minimum gets smaller

7.10 Example (TSP in a digraph)

An IP formulation

Introduce 0/1-variable $x_{ij}$ with $x_{ij} = 1$ <=> edge (i,j) is in the TSP tour

$\min \sum_{ij} c_{ij}x_{ij}$

$\sum_j x_{ij} = 1$ for all $i = 1, \ldots, n$ (7.1)

$\sum_i x_{ij} = 1$ for all $j = 1, \ldots, n$ (7.2)

$\sum_{i,j \in S} x_{ij} \leq |S|-1$ for all $\emptyset \neq S \subset \{1, \ldots, n\}$ (7.3)



$x_{ij} \in \{0, 1\}$ (7.4)

The Cycle Cover Relaxation of the TSP

Is obtained by deleting constraints (7.3).

The remaining constraints define an assignment problem in which edges (i,i) are not permitted. One can model this in objective function with high costs $c_{ii}$. Such assignment problems can be solved efficiently, e.g. with the primal-dual method of Section 6.5.

Using the cycle cover relaxation in a Branch and Bound algorithm

Take the cycle cover relaxation as lower bounding strategy

The optimal assignment $(x_{ij})$ is a tour if it fulfills constraint (7.3). Otherwise branch as follows:

choose a cycle with smallest number of edges and branch by setting every edge to 0

=> each edge of the cycle generates a child in the B&B tree

**Main statements of this chapter**

Lagrangian relaxation is an important technique to generate "good" lower bounds for IPs. It relaxes side constraints, but punishes their violation in the objective function. By varying the penalty costs, the lower bound can be improved.

A systematic improvement of the penalty costs leads to subgradient optimization. This is a method to maximize a non-differentiable concave function.

The lower bound obtained in this way is at least as good as that obtained by LP relaxation, and both are equal under certain conditions. The advantage of Lagrangian relaxation over LP relaxation is due to a quicker (approximate) lower bound computation by combinatorial methods instead of solving an LP as in the LP-relaxation.

Lagrangian relaxation is one of the workhorses in branch and bound algorithms

**The basics of Lagrangian relaxation**

Consider the integer linear program

(P) $\min c^T x$

s.t. $Ax \geq b$ (k "difficult" side constraints)



$Bx \geq d$ (m-k "easy" side constraints)

$x$ integer

Relax the "difficult" side constraints $Ax \geq b$ and punish their violation in the objective function.

To this end, introduce Lagrange multipliers $\lambda_1, \ldots, \lambda_k$ for the relaxed side constraints. They form a kind of dual variable for these side constraints and must fulfill the conditions

$a_i x \geq b_i \Rightarrow \lambda_i \geq 0$ (7.5)

$a_i x = b_i \Rightarrow \lambda_i$ unrestricted (7.6)

For fixed such $\lambda = (\lambda_1, \ldots, \lambda_k)^T$ the Lagrangian relaxation $(LR_\lambda)$ of (P) is defined as

$(LR_\lambda)$ $\min c^T x + \lambda^T(b - Ax) =: L(\lambda, x)$

s.t. $Bx \geq d$

$x$ integer

$L(\lambda,x)$ is called the Lagrange function, $\lambda = (\lambda_1, \ldots, \lambda_k)^T$ is also called Lagrange vector and can be seen as vector of penalty costs.

We denote the feasibility domains of (P) and $(LR_\lambda)$ with $S(P)$ and $S(LR_\lambda)$ and the associated optimal values with $z(P)$ and $z(LR_\lambda)$, respectively.

7.11 Lemma (Lagrangian relaxation yields lower bounds)

For every Lagrange vector $\lambda$:

(1) $S(LR_\lambda) \supseteq S(P)$

(2) $z(LR_\lambda) \leq z(P)$

Proof

(1) is trivial, as side constraints have been deleted

(2):

Let $x$ be optimal w.r.t. (P)

$\Rightarrow b_i - a_i x \leq 0$, or $b_i - a_i x = 0$ for equality constraints

$\Rightarrow \lambda_i(b_i - a_i x) \leq 0$ for all $i$ $\Rightarrow \lambda^T(b - Ax) \leq 0$

$\Rightarrow z(P) = c^T x \geq c^T x + \lambda^T(b - Ax) \geq z(LR_\lambda)$ as $x \in S(P) \subseteq S(LR_\lambda)$ ❑

7.12 Lemma (Optimality criterion)

If $x$ and $\lambda$ fulfill

(1) $x$ is optimal w.r.t. $(LR_\lambda)$



(2) $a_i x \geq b_i$, or $a_i x = b_i$ for equality constraints

(3) $\lambda^T(b - Ax) = 0$

then $x$ is optimal w.r.t. (P). If (3) is violated, then $x$ is $\varepsilon$-optimal with $\varepsilon = \lambda^T(b - Ax)$

Proof

(1), (2) $\Rightarrow x \in S(P)$

$\Rightarrow z(LR_\lambda) = c^T x + \lambda^T(b - Ax) = c^T x \geq z(P)$ because of (3) and $x \in S(P)$

$\Rightarrow z(LR_\lambda) = z(P)$ because of Lemma 7.11.

If $x$ violates (3), then $\lambda^T(b - Ax)$ is the error term ❑

The aim of Lagrangian relaxation

Partition the constraints of (P) is such a way that $(LR_\lambda)$ is much easier to solve than (P)

Make $z(P) - z(LR_\lambda)$ as small as possible (duality gap of Lagrangian relaxation)

i.e., make $L(\lambda) := z(LR_\lambda)$ as large as possible by varying the Lagrange multipliers

$\Rightarrow$ this leads to the optimization problem $\max_\lambda L(\lambda)$

When used for B&B, it is not required to solve this optimization problem optimally. A good value of $L(\lambda)$ usually suffices, as each such value provides a lower bound for $z(P)$.

**Lagrangian relaxation of the symmetric TSP via 1-trees**

IP formulation of the symmetric TSP

Introduce 0/1-variable $x_e$ with $x_e = 1$ <=> edge $e$ is in the tour

(P) min $\sum_e c_e x_e$

$x(\delta(i)) = 2$ for all $i = 1, \dots, n$ (7.7)

$x(S) \leq |S|-1$ for all $\emptyset \neq S \subseteq \{ 2, \dots, n \}$ (7.8)

observe: $S \subseteq \{ 2, \dots, n \}$ suffices to exclude short cycles

$x_e \in \{ 0, 1 \}$ (7.9)

Here $x(S) := \sum_{e = ij,\, i,j \in S} x_e$ and $x(\delta(i)) := \sum_{e \in \delta(i)} x_e$

A variation of (P) gives ($LR_\lambda$)

Partition (7.7) into

$\sum_e x_e = n$ (7.10) redundant in (P)

$x(\delta(i)) = 2$ for $i = 2, \dots, n$ (7.11)



$x(\delta(1)) = 2$ (7.12)

($LR_\lambda$) is defined by relaxing (7.11)

($LR_\lambda$) min $\sum_e c_e x_e + \sum_{i = 2,\dots,n} \lambda_i (2 - x(\delta(i)))$

s.t. (7.8), (7.9), (7.10), (7.12)

Observe: (7.10) is not redundant in ($LR_\lambda$)

Combinatorial structure of the feasible solutions of ($LR_\lambda$)

7.13 Lemma (Feasible solutions von ($LR_\lambda$) are 1-trees)

$x$ is a feasible solution von ($LR_\lambda$) <=> $x$ is a 1-tree, i.e.,

$x$ is a spanning tree on the vertex set $\{ 2, \dots, n \}$

with 2 additional edges out of vertex 1

Proof

"=>"

let $x$ be a feasible solution of ($LR_\lambda$)

(7.9), (7.10), (7.12) => $x$ has n-2 edges on vertices 2, ..., n

(7.8) => $x$ is connected

ADM I => a connected graph with n-2 edges and n-1 vertices is a spanning tree

(7.12) => 2 additional edges out of vertex 1

=> x is a 1-tree

"<="

every 1-tree fulfills conditions (7.8), (7.9), (7.10), (7.12) ❑

The Lagrange function $L(\lambda,x)$

$L(\lambda,x) = \sum_e c_e x_e + \sum_{i=2,\dots,n} \lambda_i(2 - x(\delta(i)))$, $\lambda_i$ unrestricted

=> replace w.o.l.g. $\lambda_i$ by $-\lambda_i$ (this gives a better combinatorial interpretation)

=> $L(\lambda,x) = \sum_e c_e x_e + \sum_{i=2,\dots,n} \lambda_i(x(\delta(i) - 2)$

with $x(\delta(i)) - 2$ = deviation from the desired degree 2 of vertex i

With $\lambda_1 := 0$ we obtain

$L(\lambda,x) = \sum_e c_e x_e + \sum_{i=1,\dots,n} \lambda_i(x(\delta(i) - 2)$

$= \sum_e c_e x_e + \sum_{i=1,\dots,n} \lambda_i x(\delta(i)) - 2\sum_{i=1,\dots,n} \lambda_i$

$= \sum_e c_e x_e + \sum_{e=ij} (\lambda_i + \lambda_j) x_e - 2\sum_{i=1,\dots,n} \lambda_i$

$= \sum_{e=ij} (c_e + \lambda_i + \lambda_j) x_e - 2\sum_{i=1,\dots,n} \lambda_i$



This gives new edge costs $c_e' = c_e + \lambda_i + \lambda_j$ for $e = ij$ minus the constant term $2\sum_{i=1,\dots,n} \lambda_i$

Interpretation of the Lagrangian relaxation

Relaxed problem

= computing a 1-tree with minimum weight w.r.t. edge costs $c_e + \lambda_i + \lambda_j$ for edge $e = ij$

Varying the Lagrange multipliers $\lambda_i$

= varying the edge costs $c_e$ via node values $\lambda_i$

This variation of edge costs has no influence on the optimality of a tour, but may change the 1-tree

because:

$\sum_{e=ij} (c_e + \lambda_i + \lambda_j) x_e - 2\sum_{i=1,\dots,n} \lambda_i = \sum_e c_e x_e$ if x is a tour

If the minimum 1-tree is a tour, then this tour is optimal for (P) because of Lemma 7.12, as $\lambda^T(b - Ax) = 0$ for any tour

A minimum 1-tree can be constructed in polynomial time as follows:

(1) Compute a MST on the vertices 2, ..., n with an algorithm from ADM I (Kruskal or Prim)

(2) Choose the two cheapest edges out of vertex 1

Algorithm for improving the lower bound (varying the $\lambda_i$)

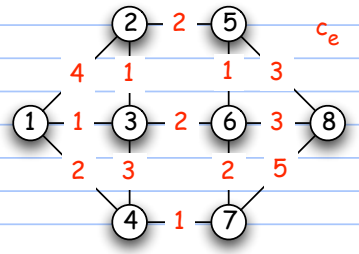
- Input
  - graph $G = (V, E)$ with $V = \{ 1, \ldots, n \}$
  - edge costs $c_e$
- Output
  - optimal tour or 1-tree with "good" lower bound $z(LR_\lambda)$
- Method
  - // initialize the $\lambda_i$
  - set $\lambda_i := 0$ for every vertex $i$
  - // initialize a step length $w > 0$ for varying the $\lambda_i$
  - set $w := 1$
  - repeat
    - compute a minimum 1-tree $x$ for edge costs $c_{ij} + \lambda_i + \lambda_j$
    - if $x$ is tour then return $x$ // $x$ is an optimal tour
    - // varying the $\lambda_i$
    - for all vertices $i \neq 1$ do
      - determine the degree $d_i$ of vertex $i$



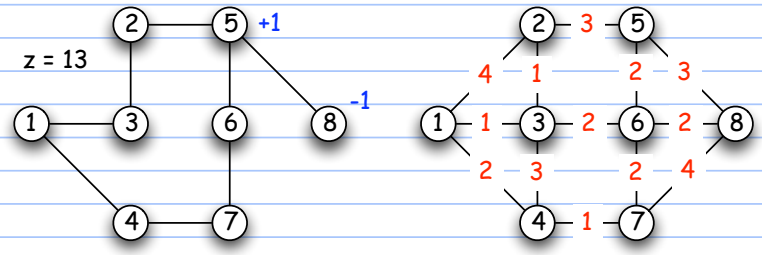
      - if $d_i \neq 2$ then $\lambda_i := \lambda_i + (d_i - 2)w$
    - vary the step length $w$ if appropriate
  - until $z(LR_\lambda) = z(x)$ is "good" enough
  - return best $x$ found and the associated $\lambda$

7.14 Example (1-tree relaxation of the symmetric TSP)

- Step length $w$ is always 1
- Graph with edge costs

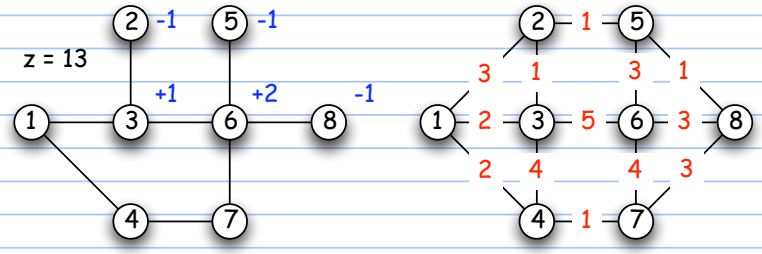



- Iteration 1
  - minimum 1 tree, varying the $\lambda_i$ and new edge costs




Iteration 2

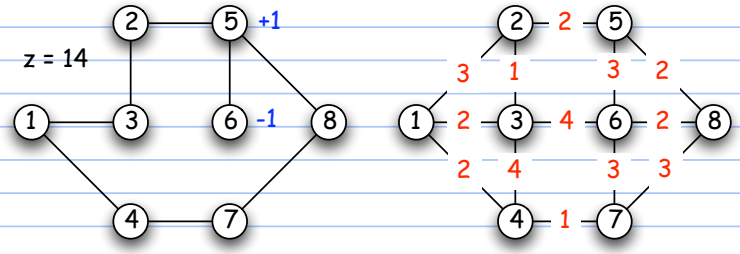
minimum 1 tree, varying the $\lambda_i$ and new edge costs




Iteration 3

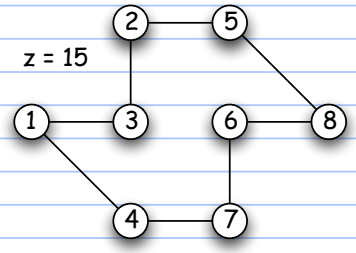
minimum 1 tree, varying the $\lambda_i$ and new edge costs





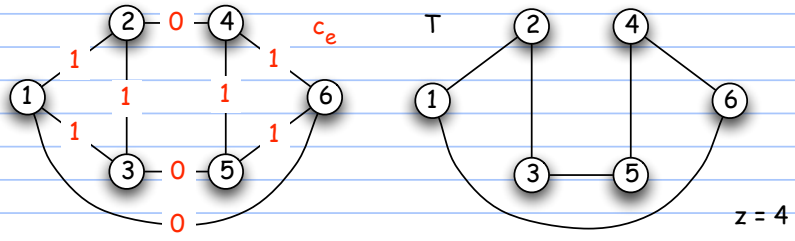

Iteration 4

minimum 1 tree is a tour => optimal tour constructed



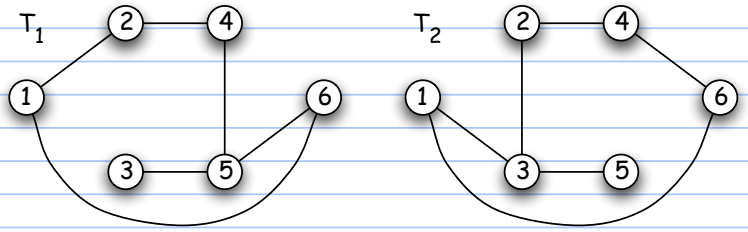

7.15 Example (In general there is no $\lambda$ such that an optimal 1-tree is a tour)

Graph with edge costs and optimal tour




Claim: for every choice of $\lambda_i$ (with $\lambda_1 = 0$), there is no tour w.r.t. $c_{ij} + \lambda_i + \lambda_j$ that is a minimum 1-tree

Consider the 1-trees




The values of these 1-trees w.r.t. edge costs $c_{ij} + \lambda_i + \lambda_j$ is

value of $T_1 = 3 + 2\lambda_2 + 1\lambda_3 + 2\lambda_4 + 3\lambda_5 + 2\lambda_6 =: z_1$



value of $T_2 = 3 + 2\lambda_2 + 3\lambda_3 + 2\lambda_4 + 1\lambda_5 + 2\lambda_6 =: z_2$

The value of an optimal tour w.r.t. $c_{ij} + \lambda_i + \lambda_j$ is

$4 + 2\lambda_2 + 2\lambda_3 + 2\lambda_4 + 2\lambda_5 + 2\lambda_6 =: z_0$

$\Rightarrow z_0 - z_1 = 1 + \lambda_3 - \lambda_5$ and $z_0 - z_2 = 1 - \lambda_3 + \lambda_5$

$\Rightarrow$ either $z_0 > z_1$ or $z_0 > z_2$

since $z_0 < z_1$ and $z_0 < z_2$ imply that $1 + \lambda_3 - \lambda_5 < 0$ and $1 - \lambda_3 + \lambda_5 < 0$

$\Rightarrow \lambda_3 - \lambda_5 > 1$ and $-\lambda_3 + \lambda_5 > 1$, a contradiction ❑

Observe: What we observe here for the TSP, viz. that $\max_\lambda L(\lambda) \neq z(P)$, is generally the case. Lagrangian relaxation provides in general only lower bounds for z(P). But these are very valuable in a Branch & Bound algorithm.

For more information about Lagrangian relaxations of the TSP see

E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, eds.

The Traveling Salesman problem: A Guided tour of Combinatorial Optimization

John Wiley & Sons, New York, 1985.

**Computing $\max_\lambda L(\lambda)$ by subgradient optimization**

$\max_\lambda L(\lambda) = \max_\lambda \min_x L(\lambda,x) = \max_\lambda \min\{ L(\lambda,x) \mid x \in S(LR_\lambda) \}$

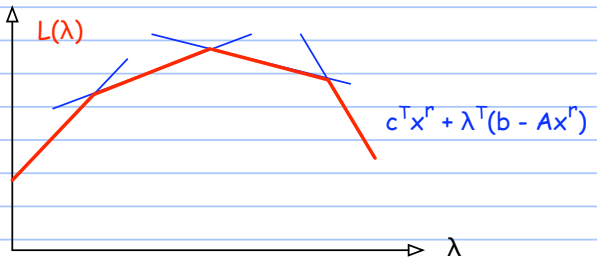
Subgradient optimization uses the fact that $S(LR_\lambda)$ is finite when $S(LR_\lambda)$ is a polytope. This follows from the integrality of $x$, and we will assume it in the sequel.

$S(LR_\lambda)$ finite => we can write $S(LR_\lambda)$ as $S(LR_\lambda) = \{ x^1, x^2, \ldots, x^R \}$

=> $L(\lambda) = \min\{ c^T x^r + \lambda^T(b - Ax^r) \mid r = 1, \ldots, R \}$

=> $L(\lambda)$ is the minimum of finitely many affine linear functions $c^T x^r + \lambda^T(b - Ax^r)$ of $\lambda$

=> $L(\lambda)$ is piecewise linear and concave, but in general not differentiable






Subgradient optimization

~ gradient method for maximizing a concave continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}^1$

Gradient and subgradient

Gradient of a continuously differentiable function in $u$

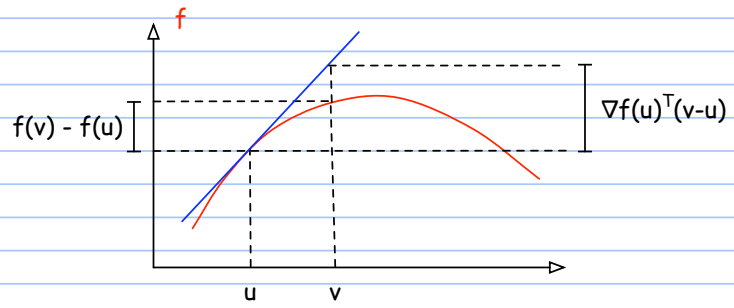
= vector of partial derivatives in u:

$$\nabla f(u) = \left( \frac{\partial f}{\partial x_1}(u), \ldots, \frac{\partial f}{\partial x_n}(u) \right)$$

From calculus we know:

f is concave <=>

$f(v) - f(u) \leq \nabla f(u)^T (v-u)$ for all $v, u$




- Subgradient of a continuous concave function in $u$
  - = vector $d$ with $f(v) - f(u) \leq d^T(v-u)$ for all $v$

    The set of subgradients in $u$ is called the subdifferential of $f$ in $u$ and is denoted by $\partial f(u)$

    Then: $f$ is differentiable in $u$ $\Rightarrow$ $\partial f(u) = \{ \nabla f(u) \}$

- Conditions for the maximum of a concave function
  - The continuously differentiable case
    - From calculus we know:

      $f$ attains its maximum at $\lambda^*$ $\Leftrightarrow$ $\nabla f(\lambda^*) = 0$



  - The non-differentiable case
    - 7.16 Lemma (Condition for the maximum of a continuous concave function)
      - Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^1$ be continuous and concave. Then

        $f$ attains its maximum at $\lambda^*$ $\Leftrightarrow$ $0 \in \partial f(\lambda^*)$
      - Proof
        - "<="
          - let $0 \in \partial f(\lambda^*)$

            $\Rightarrow$ $0 = 0^T(v - \lambda^*) \geq f(v) - f(\lambda^*)$ for all $v$ $\Rightarrow$ $f$ attains its maximum at $\lambda^*$
        - "=>"
          - let $f$ attain its maximum at $\lambda^*$

            $\Rightarrow$ $0 = 0^T(v - \lambda^*) \geq f(v) - f(\lambda^*)$ for all $v$ $\Rightarrow$ $0 \in \partial f(\lambda^*)$ ❑

- Generic subgradient optimization
  - Input
    - a continuous concave function $f : \mathbb{R}^n \rightarrow \mathbb{R}^1$

Output

a point $\lambda^*$ at which $f$ attains its maximum, or a point $\lambda$ with a "good" value $f(\lambda)$

Method

choose a starting point $u_0$

initialize a counter $i := 0$

repeat

if $0 \in \partial f(u_i)$ then return $u_i$ // f attains its maximum at $u_i$

// this step my be skipped if the test "$0 \in \partial f(u_i)$" is computationally too expensive

compute a subgradient $d_i \in \partial f(u_i)$ and a step length $w_i > 0$

set $u_{i+1} := u_i + w_i \cdot d_i$

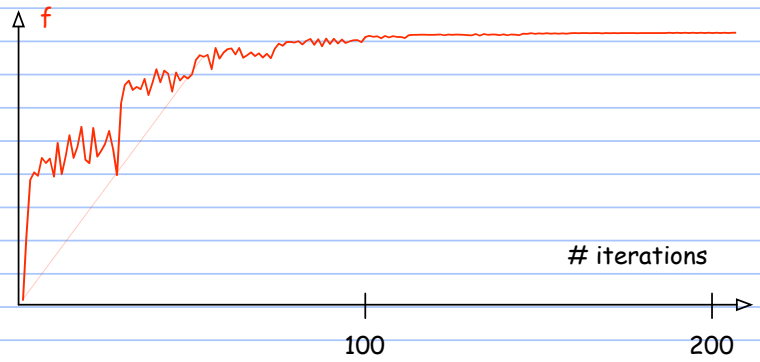
$i := i+1$

until no more computing time or hardly any progress

return the best point of the sequence $u_0, \ldots, u_i$

A typical run of the algorithm






The run shows that improvement need not be monotone

Main ingredients of the subgradient optimization

determine a step length $w_i > 0$

solved in theory by a theorem of Polyak, but still difficult in practice, usually requires experiments

7.17 Theorem (Polyak 1967)

Let $f : \mathbb{R}^n \to \mathbb{R}^1$ be concave and continuous and let $f$ attain its maximum at $\lambda^*$.

Let $(w_i)_{i \in N}$ be a sequence of step lengths with

(1) $w_i \geq 0$ for all i

(2) $(w_i)_{i \in N}$ is a monotonically decreasing null sequence

(3) the series $\sum w_i$ is divergent

Then the sequence of points $u_i$ generated by subgradient optimization fulfills

$\lim_{i \to \infty} f(u_i) = f(\lambda^*)$

without proof ❑

This theorem ensures convergence under relatively weak conditions, which can easily be met in practice. The only problem is to control the speed of convergence. But this is not that important for the use in B&B.

Computing a subgradient $d_i \in \partial f(u_i)$

This is simple, subgradients come for free in Lagrangian relaxation

7.18 Lemma (Subgradients in Lagrangian relaxation)



Let $x^*$ be an optimal solution of $(LR_\lambda)$ in $\lambda = u$.

Then $b - Ax^*$ is a subgradient of $L(\lambda) = \min_x L(\lambda,x)$ in $\lambda = u$, i.e., $b - Ax^* \in \partial f(u)$.

Proof by checking the definition of subgradient

$L(v) - L(u) = \min_x L(v,x) - \min_x L(u,x)$

$= \min_x L(v,x) - L(u,x^*)$ since $x^*$ is optimal for $(LR_u)$

$\leq L(v,x^*) - L(u,x^*)$ since $x^*$ is feasible for $(LR_v)$

$= (c^T x^* + v^T(b - Ax^*)) - (c^T x^* + u^T(b - Ax^*))$

$= (v^T - u^T)(b - Ax^*) = (b - Ax^*)^T(v - u)$ ❑

Remark: In the 1-tree relaxation of the symmetric TSP, a transition of $\lambda_i$ to $-\lambda_i$ reveals $x(\delta(i)) - 2$ as subgradient. The change of multipliers $\lambda_i$ in this example are therefore an application of subgradient optimization.

**Lagrangian relaxation vs. LP relaxation**

There is a relationship between the optimal value of a Lagrangian relaxation and the optimal value of the LP relaxation of an IP.

We consider:

- The initial problem

  (P) $\min c^T x$

  s.t. $Ax \geq b$

  $Bx \geq d$

  $x$ integer

  we do not consider sign constraints for $x$, but assume that these are contained in $Bx \geq d$

- The Lagrangian relaxation of (P)

  ($LR_\lambda$) $\min c^T x + \lambda^T(b - Ax) = \min_x L(\lambda, x) = L(\lambda)$

  s.t. $Bx \geq d$

  $x$ integer

- The LP relaxation of (P)

  (LP) $\min c^T x$

  s.t. $Ax \geq b$

  $Bx \geq d$



$x$ unconstrained

with optimal value $z(LP)$

7.19 Theorem (Relationship between Lagrangian relaxation and LP relaxation)

- $\max_\lambda L(\lambda) \geq z(LP)$
- Equality holds if the polyhedron defined by $Bx \geq d$ is integer (so that the integrality condition in ($LR_\lambda$) may be dropped).

Proof

we show this for side constraints of the form $Ax \geq b$ (=> $\lambda \geq 0$), the proof can easily be adapted to equations ($\lambda$ unconstrained).

$$\max_{\lambda \geq 0} L(\lambda) = \max_{\lambda \geq 0} \min_{\substack{x \\ Bx \geq d \\ x \text{ gzz}}} L(\lambda,x) = \max_{\lambda \geq 0} \min_{\substack{x \\ Bx \geq d}} L(\lambda,x)$$

holds if $Bx \geq d$ induces an integer polyhedron, otherwise we have $\geq$

$$= \max_{\lambda \geq 0} \min_{Bx \geq d} (c^T x + \lambda^T (b - Ax))$$

$$= \max_{\lambda \geq 0} [\lambda^T b + \min_{Bx \geq d} (c^T - \lambda^T A)x] = \max_{\lambda \geq 0} [\lambda^T b + \max_{y \geq 0, B^T y = c - A^T \lambda} d^T y]$$

LP duality

$$= \max_{\lambda \geq 0, y \geq 0, B^T y = c - A^T \lambda} [b^T \lambda + d^T y] = \min_{x \text{ unconstrained}, Ax \geq b, Bx \geq d} c^T x$$

LP duality

$$= z(LP) \quad \square$$

7.20 Remark

The 1-tree relaxation bound is because of Theorem 7.19 just the LP-relaxation bound of the TSP-polytope.



Since LPs can in principle be solved in polynomial time (by interior point methods), it seems that the LP-relaxation should be preferred above the Lagrangian relaxation if $Bx \geq d$ defines an integer polyhedron. But in practice one very often favors subgradient optimization, since it is usually much faster (very often, $L(\lambda)$ con be computed combinatorially), and since approximate values of $\max_\lambda L(\lambda)$ are usually sufficient.

**Main statements of this chapter**

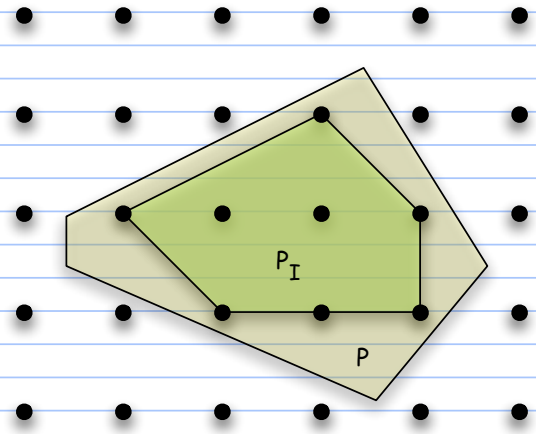
We introduce cutting plane algorithms as another method to solve IPs exactly.

We show that that these are in principle finite methods.

From the proofs we learn more about integer polytopes (Gomory-Chvátal-Cuts, Chvátal Closure)

**The integer hull of a polyhedron**

The integer hull $P_I$ of a polyhedron $P$ is the convex hull of all its integer points.






A polyhedron $P$ is called integer (or integral) if all its vertices are integer.

Then a polyhedron $P$ fulfills:

(1) $P$ is integer <=> $P = P_I$

(2) integer optimization on $P$ <=> linear optimization on $P_I$

Therefore, one is interested in linear descriptions of $P_I$ (= description of $P_I$ by linear inequalities)

One difficulty here is that $P_I$ need not be a polyhedron any more in general.

An example is given by

$$P := \{(y, x) \in \mathbb{R}^2 \mid \frac{y}{x} \leq \sqrt{2}\}$$

(exercise)

One can show, however, that $P_I$ is a polyhedron when $P$ is rational, and we will do this for rational polytopes $P$.

A polyhedron $P = \{ x \in \mathbb{R}^n \mid Ax \leq b \}$ is called rational, if all entries of $A$ and $b$ are rational numbers. We will assume in this chapter that all polyhedra are rational. For the sake of completeness, we will mention this as an assumption in all theorems.

**Criteria for the existence of feasible points and valid inequalities**

These criteria are alternative formulations of Farkas' Lemma (Lemma 4.5).

An inequality $w^T x \leq t$ is called valid for polyhedron $P$, if all points $x \in P$ fulfill that inequality.

7.21 Lemma (Farkas' Lemma for the existence of feasible solutions)

Consider a polyhedron $P = \{ x \in \mathbb{R}^n \mid Ax \leq b \}$. Then:

(1) $P \neq \emptyset$ <=> $y^T b \geq 0$ for all $y \in \mathbb{R}^m$ with $y \geq 0$ and $y^T A = 0$

(2) $P = \emptyset$ <=> there is $y \in \mathbb{R}^m, y \geq 0$ with $y^T A = 0$ and $y^T b \leq -1$

(3) $P = \emptyset$ <=> the inequality $0^T x \leq -1$ can be obtained as non-negative linear combination of the inequalities in $Ax \leq b$

Proof

(1)

"=>"

Consider the LP $\max \{ 0^T x \mid Ax \leq b \}$

$P \neq \emptyset$ => every $x \in P$ is an optimal solution of the LP

Duality theorem => the dual LP has an optimal solution and

$0 = \max \{ 0^T x \mid Ax \leq b \} = \min \{ y^T b \mid y^T A = 0, y \geq 0 \}$



=> $y^T b \geq 0$ for all $y \geq 0$ with $y^T A = 0$

"<="

Consider the LP $\min \{ y^T b \mid y^T A = 0, y \geq 0 \}$

0 is a feasible solution of this LP

Assumption => the objective function $y^T b$ is bounded from below by 0

Duality theorem => $P$ has an optimal solution, so in particular a feasible solution

(2)

follows from the negation of (1)

$P = \emptyset$ <=> there is $y' \in \mathbb{R}^m, y' \geq 0$ with $(y')^T A = 0$ and $(y')^T b < 0$.

Let $g := (y')^T b < 0$

With $y := y'/|g|$ we obtain

$P = \emptyset$ <=> there is $y \in \mathbb{R}^m, y \geq 0$ with $y^T A = 0$ and $y^T b \leq -1$

(3)

"<="

clear

"=>"

take $y$ from (2) and multiply $Ax \le b$ by $y$ from the left =>

$0^T x = y^T A x \le y^T b \le -1 \Rightarrow 0^T x \le -1$ ❑

**7.22 Lemma (Farkas' Lemma for valid inequalities)**

The following statements are equivalent for a non-empty polyhedron $P = \{ x \in \mathbb{R}^n \mid Ax \le b \}$:

(1) $w^T x \le t$ is a valid inequality for $P$

(2) There is $y \in \mathbb{R}^m, y \ge 0$ with $y^T A = w^T$ and $y^T b \le t$

Proof

(1) => (2)

Consider the LP $\max \{ w^T x \mid Ax \le b \}$

$P \ne \emptyset$, $w^T x \le t$ => the LP has an optimal solution

duality theorem => the dual LP has an optimal solution $y^*$ and

$t \ge \max \{ w^T x \mid Ax \le b \} = \min \{ y^T b \mid y^T A = w^T, y \ge 0 \} = (y^*)^T b$

=> $y^*$ fulfills (2)

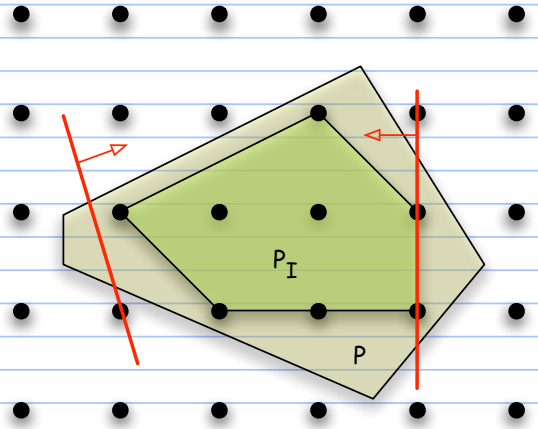
(2) => (1)

$Ax \le b, y \ge 0 \Rightarrow w^T x = (y^T A)x = y^T(Ax) \le y^T b \le t$ ❑



**Cutting planes and cutting plane algorithms**

Idea: use hyperplanes to cut off parts of polyhedron $P$, but without cutting off points from $P_I$. So these hyperplanes are valid for $P_I$ (possibly even a supporting hyperplane or a facet). Such a hyperplane is called a cutting plane. A cutting plane $H$ that cuts off a point $x^* \in P - P_I$, is called an $x^*$ separating hyperplane.




Cutting plane algorithm (Idea)

- Input
  - Integer linear program (IP)
    $\min \{ c^T x,\ x \in P_I \}$ with $P$ = polyhedron of the LP relaxation of (IP)
- Output
  - optimal solution of (IP)
- Method
  - repeat forever
    - solve the LP $\min \{ c^T x,\ x \in P \}$
    - let $x^*$ be the computed optimal solution of the LP
    - if $x^*$ is integer then return $x^*$ // $x^*$ is an optimal solution of (IP)
    - compute a cutting plane that cuts off $x^*$ from $P$ and is valid for $P_I$ // $x^*$ separating hyperplane
    - let $H$ be the associated halfspace containing $P_I$
    - set $P := P \cap H$

- Obvious questions
  - (1) how does one prove that an inequality is a cutting plane?



  - (2) do cutting plane algorithms terminate?
  - (3) how does one compute an $x^*$ separating hyperplane for a given $x^* \in P - P_I$ ?
  - We will answer here only (1) and (2) and show that there are proof techniques for (1), and that there is a finite set of cutting planes of a special structure such that cutting plane algorithms using them terminate with $P = P_I$.
  - (3) depends very much on the specific problem, we will show some examples in Chapter 8.

- **Cutting plane proofs**
  - For a polytope $P = \{ x \in \mathbb{R}^n \mid Ax \le b \}$, the validity of an inequality $w^T x \le t$ can be shown by Farkas' Lemma (Lemma 7.22). This is more complicated for cutting planes.

- 7.23 Example (Example of a cutting plane proof)
  - Consider the system of linear inequalities

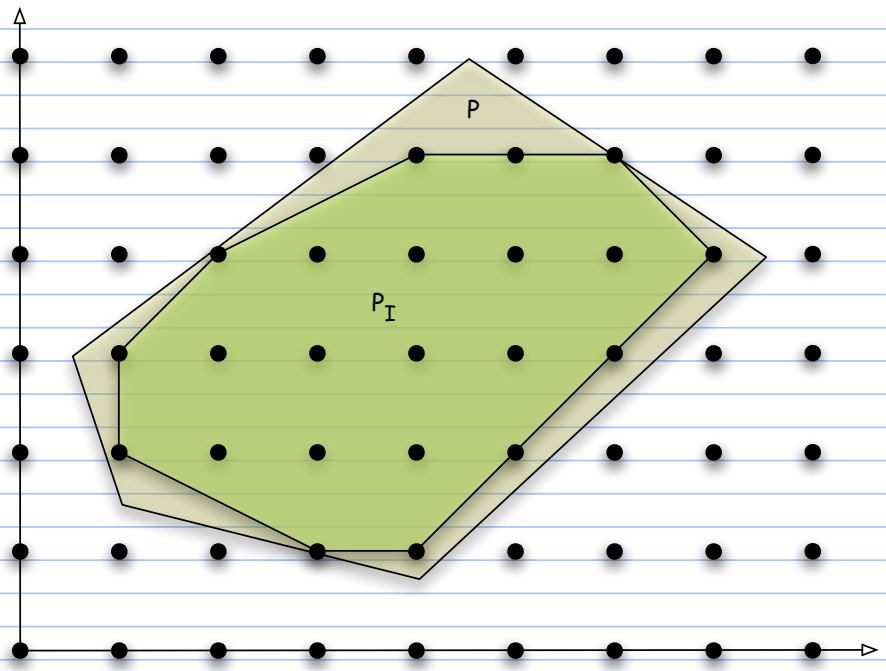
$$2x_1 + 3x_2 \le 27 \quad (1)$$
$$2x_1 - 2x_2 \le 7 \quad (2)$$
$$-6x_1 - 2x_2 \le -9 \quad (3)$$

$-2\,x_1 - 6\,x_2 \le -11$ (4)

$-6\,x_1 + 8\,x_2 \le 21$ (5)

The associated polytope $P$ and its integer hull






$x_2 \le 5$ is a valid inequality for $P_I$

how can we derive it from the given inequalities for $P$ ?

Multiply (5) with 1/2

=> $-3x_1 + 4x_2 \leq 21/2$

=> $-3x_1 + 4x_2 \leq \lfloor 21/2 \rfloor = 10$ is valid for $P_I$ (as there are only integer coefficients on the left hand side)

=> this gives the new inequality $-3x_1 + 4x_2 \leq 10$ (6) for $P_I$

Multiply (6) with 2, (1) with 3 and add the resulting inequalities

=> $-6x_1 + 8x_2 \leq 20$

$6x_1 + 9x_2 \leq 81$

=> $17x_2 \leq 101$

=> we obtain the wanted inequality $x_2 \leq \lfloor 101/17 \rfloor = 5$ by rounding down the right hand side

In general, these inequalities have the form

$y^T Ax \leq \lfloor y^T b \rfloor$ with $y \geq 0$ and $y^T A$ integer

where $Ax \leq b$ is the system of inequalities after the "previous" step.

This observation leads to the general definition of a cutting plane proof.

General definition

Let $Ax \leq b$ be a system of $m$ linear inequalities.



A cutting plane proof for the inequality $w^T x \leq t$ with integer $w$ and $t$ starting from $Ax \leq b$ is a finite sequence of inequalities of the form

$$a_{m+k}^T x \leq b_{m+k} \quad (k = 1, \ldots, M)$$

together with non-negative numbers

$$y_{k_j} \quad (1 \leq k \leq M,\ 1 \leq j \leq m + k - 1)$$

such that, for each $k = 1, \ldots, M$, the inequality

$$a_{m+k}^T x \leq b_{m+k}$$

is obtained as non-negative linear combination

$$(y_{k_1}, \ldots, y_{k_{m+k-1}}) \begin{pmatrix} a_{11}x_1 + \ldots + a_{1n}x_n \\ \vdots \\ a_{m+k-1,1}x_1 + \ldots + a_{m+k-1,n}x_n \end{pmatrix} \leq \lfloor (y_{k_1}, \ldots, y_{k_{m+k-1}}) \begin{pmatrix} b_1 \\ \vdots \\ b_{m+k-1} \end{pmatrix} \rfloor$$

of the previous inequalities (initial and already generated), where

- the coefficients of the variables on the left hand side are integer
- the right hand side is not integer is and is rounded down
- the last inequality of the sequence is the inequality $w^T x \leq t$

Inequalities of the form

$y^T Ax \leq \lfloor y^T b \rfloor$ with $y \geq 0$ and $y^T A$ integer

are called Gomory-Chvátal cuts.

Gomory has shown in 1960 that these cuts lead to finite cutting plane algorithms.

Chvátal has introduced cutting plane proofs in 1973. These proofs are similar to Farkas' Lemma in the variants Lemma 7.21 (2) and 7.22.

7.24 Theorem (Cutting plane proofs for rational polytopes, Chvátal 1973)

Let $P = \{ x \in \mathbb{R}^n \mid Ax \leq b \}$ be a rational polytope and let $w^T x \leq t$ be an inequality with integer $w$ and $t$ that is valid for $P_I$. Then there exists a cutting plane proof of $w^T x \leq t'$ from $Ax \leq b$, for some $t' \leq t$.

Proof: see below. ❑

7.25 Theorem (Cutting plane proofs for rational polytopes without integer points, Chvátal 1973)

Let $P = \{ x \in \mathbb{R}^n \mid Ax \leq b \}$ be a rational polytope without integer points. Then there exists cutting plane proof of $0^T x \leq -1$ from $Ax \leq b$.

Proof: see below. ❑



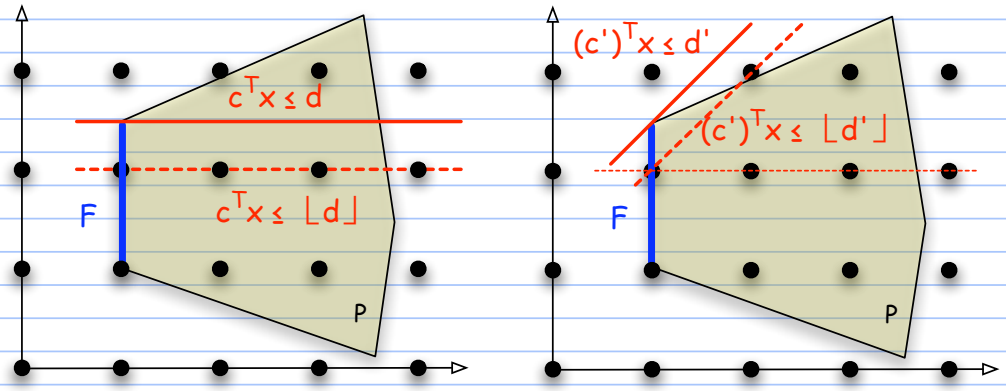
For the proof we need a lemma that enables an inductive argument on the dimension of $P$. It shows that Gomory-Chvátal cuts for a face of a rational polyhedron can be lifted to the the polyhedron itself "by rotation".

7.26 Lemma (Rotation of Gomory-Chvátal cuts)

Let $F$ be a non-empty face of a rational polytope $P$. Assume that $F$ is given by a linear system and that $c^T x \leq \lfloor d \rfloor$ is a Gomory-Chvátal cut for $F$.

Then there exists a Gomory-Chvátal cut $(c')^T x \leq \lfloor d' \rfloor$ for $P$ with

$F \cap \{ x \mid c^T x \leq \lfloor d \rfloor \} = F \cap \{ x \mid (c')^T x \leq \lfloor d' \rfloor \}$ (equality on $F$)




Proof

Let w.o.l.g. $P = \{ x \mid A'x \le b',\ A''x \le b'' \}$ with $A'', b''$ integer.

Let $F := \{ x \mid A'x \le b',\ A''x = b'' \}$ (equations $A''x = b''$ describes F)

Let $c^T x \le \lfloor d \rfloor$ be the given Gomory-Chvátal cut for F

and let w.o.l.g. $d = \max \{ c^T x \mid x \in F \}$ (deepest cut with $c^T x$; it exists since P is a polytope).

Duality theorem => the dual LP has an optimal solution

=> there are $y' \ge 0$ and $y''$ unconstrained with



$(y')^T A' + (y'')^T A'' = c^T$ (*)

$(y')^T b' + (y'')^T b'' = d$ (**)

We now construct $c'$ and $d'$ from $y''$

$(c')^T := c^T - (\lfloor y'' \rfloor)^T A'' = (y')^T A' + (y'' - \lfloor y'' \rfloor)^T A''$ because of (*)

integer $\ge 0$ $\ge 0$

$d' := d - (\lfloor y'' \rfloor)^T b'' = (y')^T b' + (y'' - \lfloor y'' \rfloor)^T b''$ because of (**)

c is integer as part of a Gomory-Chvátal cut, $(\lfloor y'' \rfloor)^T A''$ is integer => $c'$ integer

$(c')^T x \le d'$ is a valid inequality for P

because $(c')^T x = (y')^T A' x + (y'' - \lfloor y'' \rfloor)^T A'' x \le (y')^T b' + (y'' - \lfloor y'' \rfloor)^T b'' = d'$

$\ge 0$ $\ge 0$

Definition of $d'$ => $d = d' + (\lfloor y'' \rfloor)^T b''$

integer since $b''$ is integer

=> $\lfloor d \rfloor = \lfloor d' \rfloor + (\lfloor y'' \rfloor)^T b''$

Equality on F

$F \cap \{ x \mid (c')^T x \le \lfloor d' \rfloor \}$

$= F \cap \{ x \mid (c')^T x \le \lfloor d' \rfloor,\ (\lfloor y'' \rfloor)^T A'' x = (\lfloor y'' \rfloor)^T b'' \}$

fulfilled in $F$ because of $A''x = b''$

$= F \cap \{ x \mid (c' + (\lfloor y'' \rfloor)^T A'')^T x \leq (\lfloor d' \rfloor + \lfloor y'' \rfloor)^T b'' \}$

$= F \cap \{ x \mid c^T x \leq \lfloor d \rfloor \}$ ❑

Proof of Theorem 7.25

(cutting plane proofs for rational polytopes $P = \{ x \in \mathbb{R}^n \mid Ax \leq b \}$ without integer points)

Induction on dim(P)

Inductive base

$P = \emptyset$

=> statement follows from Farkas' Lemma 7.21 (3)

dim(P) = 0

=> $P = \{x^*\}$ and $x^*$ is not integer.

=> there is an integer vector $w$ such that $w^T x^*$ is not an integer (set $w_i := 1$ for one non-integer entry of $x^*$ and $w_i := 0$ otherwise).

Let $t$ be such that the hyperplane $H = \{ x \mid w^T x = t \}$ contains $x^*$ (can easily be achieved by translation).



$w^T x^*$ not integer => $t = w^T x^*$ not integer

=> $w^T x \leq t$ is valid for $P$, but $P' := P \cap \{ x \mid w^T x \leq \lfloor t \rfloor \} = \emptyset$

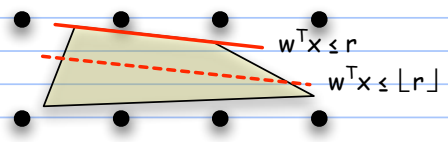
Farkas Lemma 7.22 => there is cutting plane proof for $w^T x \leq \lfloor t \rfloor$ from $Ax \leq b$

$P' = \emptyset$ => (Farkas Lemma 7.21 (3)) there is cutting plane proof for $0^T x \leq -1$ from $Ax \leq b$ and $w^T x \leq \lfloor t \rfloor$

Inductive step to $\dim(P) \geq 1$

Let $w^T x \leq r$, $w$ integer, be an inequality that defines a proper face of $P$.

Let $P' := \{ x \in P \mid w^T x \leq \lfloor r \rfloor \}$



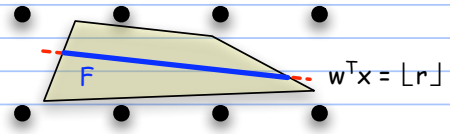

Case 1: $P' = \emptyset$

Farkas Lemma 7.22 => we can prove $w^T x \leq r$ from $Ax \leq b$

Farkas Lemma 7.21 => we can prove $0^T x \leq -1$ from $Ax \leq b, w^T x \leq \lfloor r \rfloor$

Case 2: $P' \neq \emptyset$

Let $F := \{ x \in P' \mid w^T x = \lfloor r \rfloor \}$ => $F$ is a face of $P'$




Claim: $\dim(F) < \dim(P)$

because:

either $F$ is a proper face of $P$ (if $r$ is integral)

or $w^T x \leq \lfloor r \rfloor$ cuts off something from $P$, i.e., $P$ contains vectors that do not satisfy $w^T x = \lfloor r \rfloor$

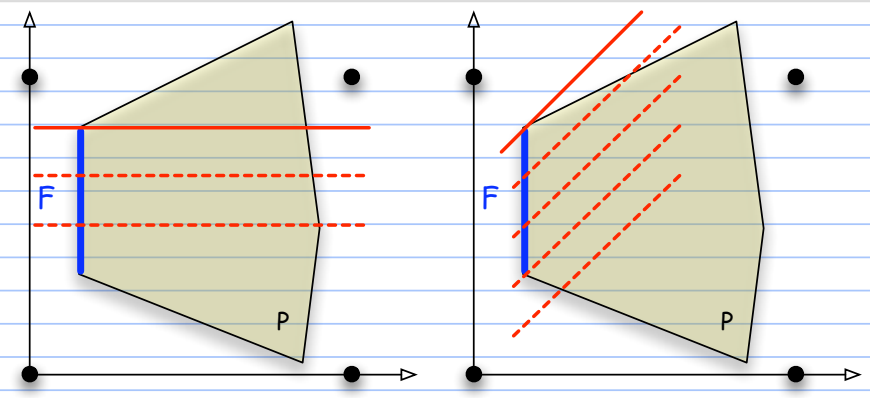
=> we have a lower dimension in both cases

$F_I$ empty, inductive assumption

=> there is a cutting plane proof for $0^T x \leq -1$ from $Ax \leq b, w^T x \leq \lfloor r \rfloor, -w^T x \leq -\lfloor r \rfloor$

Use the cutting plane $0^T x \leq -1$ for $F$ several times and apply the Rotation Lemma each time






$P$ polytope => $\min \{ w^T x \mid x \in P \}$ is finite =>

Repeating the arguments with $P'' := \{ x \in P \mid w^T x \leq \lfloor r \rfloor - 1 \}$ etc. leads after finitely many steps to a cutting plane proof for an inequality $w^T x \leq t$ with $\{ x \in P \mid w^T x \leq t \} = \emptyset$

=> have shown a reduction to Case 1 ❑

Proof of Theorem 7.24 (cutting plane proofs for rational polytopes)

Case 1: $P_I = \emptyset$

Theorem 7.25 => there is a cutting plane proof for $0^T x \leq -1$

$P$ is a polytope => $r := \max\{ w^T x \mid x \in P \}$ is finite

=> $w^T x \le r$ is a valid inequality for $P$

Farkas' Lemma 7.22 => there is a cutting plane proof for $w^T x \le r$

$w$ integer => $w^T x \le \lfloor r \rfloor$ is Gomory-Chvátal cut

adding $w^T x \le \lfloor r \rfloor$ and $0^T x \le -1$ gives $w^T x \le \lfloor r \rfloor - 1$

repeated addition of $0^T x \le -1$ gives $w^T x \le t' \le t$ in finitely many steps

Case 2: $P_I \neq \emptyset$

$P$ is a polytope => $r := \max\{ w^T x \mid x \in P \}$ is finite

$w$ integer => $w^T x \le \lfloor r \rfloor$ is a Gomory-Chvátal cut for $P$

let $P' := \{ x \in P \mid w^T x \le \lfloor r \rfloor \}$

we are done if $\lfloor r \rfloor \le t$

So assume $\lfloor r \rfloor > t$

Let $F := \{ x \in P' \mid w^T x = \lfloor r \rfloor \}$ => $F$ is a face of $P'$

$F$ contains no integer points, as $w^T x \le t$ is valid for $P_I$ and $t < \lfloor r \rfloor$

Theorem 7.25 => for $F$, there is a cutting plane proof of $0^T x \le -1$ from $Ax \le b$, $w^T x = \lfloor r \rfloor$

Rotation Lemma for $F$ and $P'$ => there is a cutting plane proof of an inequality $c^T x \le \lfloor d \rfloor$ for $P$ from $Ax \le$



$b$, $w^T x \le \lfloor r \rfloor$ such that $P' \cap \{ x \mid c^T x \le \lfloor d \rfloor, w^T x = \lfloor r \rfloor \} = \emptyset$

So, after applying this sequence of cuts to $P'$, we have $w^T x \le \lfloor r \rfloor - 1$.

Repeating this argument eventually gives $w^T x \le t' \le t$, completing the proof. ❑

**Chvátal closure and Chvátal rank**

Cutting plane proofs may use already generated cutting planes. We consider now what happens, if one can only use the initially given cutting planes $Ax \le b$

Let $P = \{ x \in \mathbb{R}^n \mid Ax \le b \}$ be a rational polytope. If one adds to $P$ all Gomory-Chvátal cuts $y^T Ax \le \lfloor y^T b \rfloor$ with $y \ge 0$, $y^T A$ integer, one obtains the Chvátal closure $P'$ of $P$.

7.27 (Properties of the Chvátal closure)

The Chvátal closure of a rational polytope is again a rational polytope. In particular, it has a linear description using only the given inequalities $Ax \le b$ and finitely many Gomory-Chvátal cuts.

Proof

Let $P = \{ x \mid Ax \le b \}$ with $A$ and $b$ integer

Set $P' := P \cap \{ x \mid y^T A x \le \lfloor y^T b \rfloor$ with $y \ge 0$, $y^T A$ integer $\}$. Then:

(7.13) $P' = P \cap \{ x \mid y^T A x \le \lfloor y^T b \rfloor$ with $y \ge 0$, $y^T A$ integer, $0 \le y < 1 \}$

Proof of (7.13):

Let $w^T x \le \lfloor t \rfloor$ be a Gomory-Chvátal cut with $y \ge 0$, $y^T A = w$, $y^T b = t$

Let $y' := y - \lfloor y \rfloor$ be the fractional part of y. Then $0 \le y' < 1$

Let $w' := (y')^T A = y^T A - (\lfloor y \rfloor)^T A = w - (\lfloor y \rfloor)^T A$

=> w' is integer, as w and A are integer.

Let $t' := (y')^T b = y^T b - (\lfloor y \rfloor)^T b = t - (\lfloor y \rfloor)^T b$

=> t and t' differ by an integer number, namely by $(\lfloor y \rfloor)^T b$

=> $w^T x \le \lfloor t \rfloor$ is obtained as sum of

$(w')^T x \le \lfloor t' \rfloor$ <-- formed according to (7.13)

$+ (\lfloor y \rfloor)^T A x \le (\lfloor y \rfloor)^T b$ <-- redundant as

non-negative linear combination of the rows of $Ax \le b$

=> the inequalities specified in (7.13) form the Chvátal closure

There are only finitely many inequalities of the form (7.13)

Denote the entries of matrix A by $a_{ij}$ and let $A_j$ be the j-th column of A



$0 \le y < 1$ => $y^T A_j \in [ -\Sigma_i |a_{ij}|, \Sigma_i |a_{ij}| ]$

$y^T A_j$ integer => there are only finitely many such $y^T A_j$

All inequalities of the form (7.13) have integer coefficients

=> They are again rational ❑

Iterating the Chvátal closure operation defines a sequence of Chvátal closures

$P = P^{(0)} \supseteq P^{(1)} \supseteq P^{(2)} \supseteq \dots \supseteq P_I$

7.28 Theorem (The Chvátal closure operation terminates)

Let P be a rational polytope. Then there is $k \in \mathbb{N}$ with $P_I = P^{(k)}$.

In particular, cutting plane algorithms with a good choice of Gomory Chvátal cuts terminate after finitely many steps.

Proof

$P_I$ is a polytope and can thus be described by finitely many inequalities (Minkowski's Theorem).

Each of these inequalities has a cutting plane proof of some finite length r with inequalities only from finitely many $P^{(i)}$ $(i \le r)$

=> the maximum of these r shows the statement ❑

Gomory has specified such a good choice of Gomory Chvátal cuts already in 1960.

The smallest $k$ with $P_I = P^{(k)}$ is called the Chvátal rank of P. It can be interpreted as a measure of complexity of the integer hull of polytopes. Already in $\mathbb{R}^2$ there are examples that the Chvátal rank can become arbitrarily large. For 0/1-polytopes in $\mathbb{R}^n$ is is bounded by $6n^3 \log n$.

7.29 Example (A polytope with Chvátal rank 2)

The initial polytope P

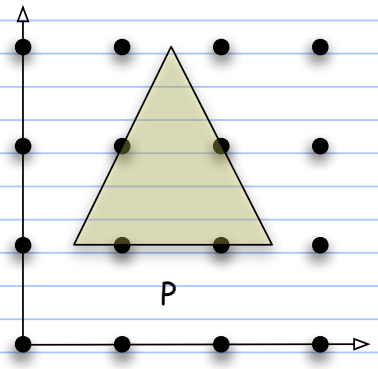
P is given by

$$-2x_1 + x_2 \leq 0 \quad (1)$$

$$2x_1 + x_2 \leq 6 \quad (2)$$

$$-x_2 \leq -1 \quad (3)$$





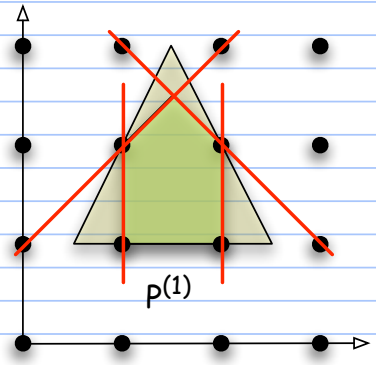

The first Chvátal closure $P^{(1)}$

$y^T = (0, 1/2, 1/2)$ gives $x_1 \leq 5/2 \Rightarrow x_1 \leq 2$

$y^T = (1/2, 0, 1/2)$ gives $-x_1 \leq -1/2 \Rightarrow -x_1 \leq -1$

$y^T = (5/6, 1/3, 1/6)$ gives $-x_1 + x_2 \leq 11/6 \Rightarrow -x_1 + x_2 \leq 1$

$y^T = (1/3, 5/6, 1/6)$ gives $x_1 + x_2 \leq 29/6 \Rightarrow x_1 + x_2 \leq 4$




$x_2 \leq 2$ can not be derived from $Ax \leq b$

the general form of a Gomory-Chvátal cut is

$(-2y_1 + 2y_2)x_1 + (y_1 + y_2 - y_3)x_2 \leq \lfloor 6y_2 - y_3 \rfloor$

$\Rightarrow -2y_1 + 2y_2 = 0,\ y_1 + y_2 - y_3 = 1,\ \lfloor 6y_2 - y_3 \rfloor = 2$

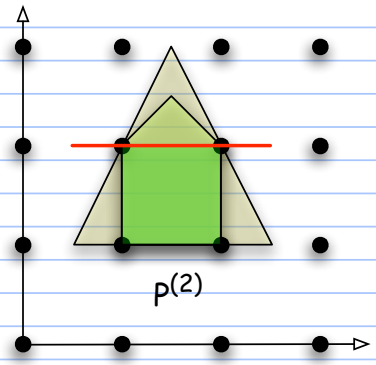
$\Rightarrow y_1 = y_2,\ y_3 = 2y_2 - 1,\ \lfloor 4y_2 + 1 \rfloor = 2 \Leftrightarrow 1 \leq 4y_2 < 2$

$\Rightarrow y_2 < 1/2 \Rightarrow y_3 < 0 \Rightarrow$ a contradiction

The second Chvátal closure $P^{(2)}$

$x_2 \leq 2$ is obtained from $-x_1 + x_2 \leq 1,\ x_1 + x_2 \leq 4$ with $y^T = (1/2, 1/2)$






**Concluding remarks**

All statements on cutting plane proofs and Chvátal rank hold also for arbitrary rational polyhedra (see Korte & Vygen), but in general not for non-rational polyhedra.

Gomory-Chvátal cuts are a standard tool for solving IPs in CPLEX. They are generated automatically in the Branch & Bound algorithm. Some of them directly in the root node from the LP-relaxation. Others for the subproblems in nodes of the Branch & Bound tree.

For many combinatorial optimization problems, there are results on the Chvátal rank of particular inequalities:

The odd-set inequalities of the matching polytopes have Chvátal Rang 1 w.r.t. the LP formulation consisting only

of the degree constraints:

$x(\delta(v)) = 1$ for all $v \in V$ degree constraints

$\sum_{e \subseteq R} x_e \leq r$ for all sets $R \subseteq V(G)$ with $|R| = 2r+1$ odd-set inequalities

$x \geq 0$

- The comb-inequalities of the TSP have Chvátal Rang 1 w.r.t. the 2-matching polytope (see Section 8.2)



- Separation is the problem to compute for a given point $x^*$ and a given polyhedron $Q$ a hyperplane $H$ that separates $x^*$ from $Q$.

  Separation is needed in cutting plane algorithms. Then $Q = P_I$ (integer hull of a rational polyhedron $P$) and $x^*$ is the LP optimum over $P$.

  There is a strong relationship between separation and optimization. We define:

- Optimization (OPT)
  - Input:
    - rational polyhedron $Q$,

      $c \in \mathbb{R}^n$ such that $c^T x$ is bounded from below on $P$
  - Output:
    - $x^* \in Q$ with $x^* = \min \{ c^T x \mid x \in Q \}$
- Separation (SEP)
  - Input:
    - rational polyhedron $Q$,

      $y \in \mathbb{R}^n$

Output:

"Yes" if $y \in Q$

$a^T x \leq d \in \mathbb{R}^n$ with $a^T x \leq d$ for all $x \in Q$ but $a^T y > d$ if $y \notin Q$ (a separating hyperplane)

7.30 Theorem (Polynomial equivalence of separation and optimization; Grötschel, Lovász, Schrijver 1984)

(OPT) can be solved in polynomial time <=> (SEP) can be solved in polynomial time.

This holds also for ε-approximations.

Without proof.

The following techniques are used for full-dimensional polyhedra:

"<=" ellipsoid method and duality theorem

"=>" antiblocking of polyhedra

For details see

M. Grötschel, L. Lovász, and A. Schrijver,

Geometric Algorithms and Combinatorial Optimization,

Springer, Berlin, 2nd ed., 1993. ❑



Remarks

If (OPT) can be solved in polynomial time, then cutting planes can be found in polynomial time.

If (OPT) is NP-hard, then we will (if P ≠ NP) not be able to find all cutting planes in polynomial time. But we may still be able to find many.

Therefore one uses in practice polynomial algorithms for constructing cutting planes until no more are found. Then one branches on fractional variables and looks again for cutting planes for the resulting subproblems, etc. This combination of Branch & Bound with cutting plane algorithms is called Branch & Cut. We will see examples in Chapter 8.

Instead of the ellipsoid method (which has proven to be inefficient in practice) one uses the dual simplex algorithm. The dual simplex algorithm can easily accommodate new cutting planes as additional constraints.

## 8.1 Introduction

- Goal of this chapter
  - Develop an abstract view on combinatorial optimization problems in order to describe induced polytopes in a unified way.
- An instance of an (abstract) combinatorial optimization problem is a triple $(E, \mathcal{F}, c)$ where
  - $E$ is a finite set, the ground set (e.g. the set of edges of a graph)
  - $\mathcal{F}$ is the set system of feasible solutions $F \subseteq E$ (e.g. the set system of all matchings $M \subseteq E(G)$)
  - $c : E \to \mathbb{R}$ with $c(F) := \sum_{e \in F} c(e)$ computes the value of feasible solution $F$ (e.g. the weight of a matching M)
- The polytope $P_{\mathcal{F}}$ induced by $(E, \mathcal{F}, c)$ is obtained as follows.
  - For a set $F \subseteq E$ we consider the incidence vector $x^F \in \mathbb{R}^E$ defined by

$$x^F_e := \begin{cases} 1 & e \in F \\ 0 & e \notin F \end{cases}$$

  We interpret $c$ as vector $c \in \mathbb{R}^E$ with $c_e := c(e)$

  and write for arbitrary vectors $x \in \mathbb{R}^E$

$$x(F) := \sum_{e \in F} x_e$$

  - Then $P_{\mathcal{F}} := \text{conv}\{ x^F \mid F \in \mathcal{F} \}$,

i.e., it is the convex hull of all incidence vectors $x^F$ of feasible solutions $F \in \mathcal{F}$

Minkowski's Theorem yields:

$P_{\mathcal{F}}$ is a polytope, and the vertices of $P_{\mathcal{F}}$ are incidence vectors of feasible solutions

=> there is a system of linear inequalities $Ax \leq b, x \geq 0$ whose set of solutions is $P_{\mathcal{F}}$

(the so-called linear description of $P_{\mathcal{F}}$)

=> the optimum of $(E, \mathcal{F}, c)$ is attained in a vertex of $Ax \leq b, x \geq 0$

=> we can solve $(E, \mathcal{F}, c)$ with linear optimization, if we have a linear description $Ax \leq b, x \geq 0$ of $P_{\mathcal{F}}$

Problems:

How does one find linear descriptions?

The proof of the Minkowski's Theorem is constructive, but "unsuited".

How large is the number of constraints?

In general exponential, see the matching polytope,

but we only need a partial description for a vertex attaining the optimum

These questions are studied in combinatorial polyhedral theory (polyhedral combinatorics)



Goal of this section

Illustration of some (partial) linear descriptions and of techniques how to prove them

**The matching polytope**

$(E, \mathcal{F}, c)$ is given by

$E$ = edge set $E(G)$ of an undirected graph

$\mathcal{F} = \{ M \subseteq E \mid M \text{ is a matching} \} =: \mathcal{M}$

$c(e)$ = non-negative edge weight of $e$

$P_{\mathcal{M}} := \text{conv}\{ x^M \mid M \in \mathcal{M} \}$ is called the matching polytope of graph $G$

8.1 Theorem (linear description of the matching polytopes)

In bipartite graphs, $P_{\mathcal{M}}$ has the linear description

$x(\delta(v)) \leq 1$ for all vertices $v$

$x \geq 0$

In arbitrary graphs, $P_{\mathcal{M}}$ has the linear description

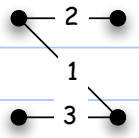
$x(\delta(v)) \leq 1$ for all vertices $v$

$\sum_{e \subseteq R} x_e \le r$ for all odd sets $R \subseteq V(G)$ with $|R| = 2r+1$, $r \ge 1$

$x \ge 0$

Example

$G$ bipartite

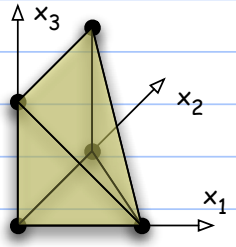



$\mathcal{M} = \{ \emptyset, \{1\}, \{2\}, \{3\}, \{2,3\} \}$

we have variables $x_1, x_2, x_3$ for the edges 1, 2, 3

then $P_{\mathcal{M}} := \text{conv}\{ x^M \mid M \in \mathcal{M} \}$ is a tilted pyramid with quadratic ground set in the $x_2$-$x_3$ plane





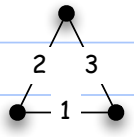

A linear description of $P_{\mathcal{M}}$ is obtained as (redundant inequalities for vertices of degree 1 deleted)

$x_1 + x_2 \le 1$

$x_1 + x_3 \le 1$

$x_i \ge 0$

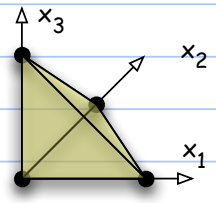
$G$ not bipartite




$\mathcal{M} = \{ \emptyset, \{1\}, \{2\}, \{3\} \}$

we have variables $x_1, x_2, x_3$ for the edges 1, 2, 3

then $P_{\mathcal{M}} := \text{conv}\{ x^M \mid M \in \mathcal{M} \}$ is a tetrahedron




- The inequalities for the bipartite case are fulfilled by $x := (1/2, 1/2, 1/2)^T$, but $x \notin P_{\mathcal{M}}$

  So the additional inequalities for odd sets are needed, in this case

  $x_1 + x_2 + x_3 \leq 1$

The proof uses in both cases the following ideas

- (1) The inequalities are valid for $P_{\mathcal{M}}$, i.e., $P_{\mathcal{M}} \subseteq \{ x \mid Ax \leq b, x \geq 0 \}$
- (2) The optimization problem

  $\max c^T x$ subject to inequalities $Ax \leq b, x \geq 0$

always attains its optimum in an incidence vector $x^M$, $M \in \mathcal{M}$

(this can e.g. by complementary slackness or the primal-dual algorithm)



- (3) Lemma 3.5 (for each vertex $x$ there is an objective function $c$ such that the optimum is obtained in $x$ only)
- (2), (3) => every vertex of the linear description is an incidence vector $x^M$, $M \in \mathcal{M}$

  => $\{ x \mid Ax \leq b, x \geq 0 \} \subseteq P_{\mathcal{M}}$

**The polytope of antichains of a partial order**

Basic facts about partial orders

A (finite) partial order $(E, <)$ is given by

a (finite) ground set $E$ and

a binary relation $<$ on $E$ with

$a < b$ and $b < c$ => $a < c$ ($<$ is transitive)

$a < b$ => $a \neq b$ ($<$ is irreflexive)

We represent partial orders by edge diagrams

= acyclic digraph $G = (V, E)$ with

$E(G)$ = ground set of the partial order

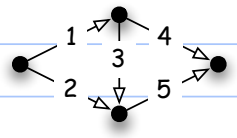
$e < e'$ :<=> there is a directed path from head(e) to tail(e')

head(e) = head(e') for all minimal elements $e, e'$ of the partial order $(E, <)$

tail(e) = tail(e') for all maximal elements e, e' of the partial order (E, <)

(every partial order is an induced suborder of an edge diagram)

Example:




E = { 1, 2, 3, 4, 5 } and 1 < 4, 1 < 3 < 5, 2 < 5

a, b ∈ E are called comparable :<=> a < b or b < a

chain = set of pairwise comparable elements

in the example, Ø {3} {1, 5} {1,4} {1,3,5} are chains, the last two are maximal chains (i.e., ⊆-maximal)

a, b ∈ E are called incomparable :<=> a, b are not comparable

antichain = set of pairwise incomparable elements

in the example, Ø {3} {2, 3} {4,5} {2,3,4} are antichains, the last two are maximal antichains (i.e., ⊆-maximal)

Combinatorial optimization problem:



Compute an antichain of maximum weight w.r.t. weights c(e) ≥ 0

Application: project scheduling

partial order = processing structure of a construction project

chain = must be done sequentially

antichain = may be done simultaneously

weight of an antichain = amount of required resources if all jobs are processed simultaneously

maximum weight = maximum amount of required resources

(E, $\mathcal{F}$, c) is given by

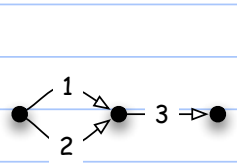
E = edge set E(G) of the edge diagram of the partial order

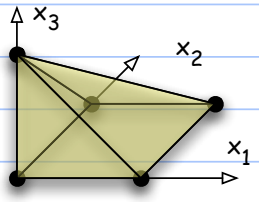
$\mathcal{F}$ = { A ⊆ E | A is an antichain } =: $\mathcal{A}$

c(e) = non-negative edge weights of e

$P_{\mathcal{A}}$ := conv { $x^A$ | A ∈ $\mathcal{A}$ } is called the antichain polytope of the partial order

Example




"pyramid"

Linear description of $P_{\mathcal{A}}$

Necessary condition: every chain contains at most one element of any antichain.

As inequality: $x(K) \leq 1$ for every chain $K$

**8.2 Theorem (Linear description of the antichain polytope)**

$P_{\mathcal{A}}$ has the linear description

$x(K) \leq 1$ for every chain $K$

$x \geq 0$

Proof

(1) all inequalities are valid for the antichain polytope

Let $x \in P_{\mathcal{A}}$ and $K$ be a chain



=> $x$ is a convex combination of vertices of $P_{\mathcal{A}}$, i.e, of incidence vectors $x^A$ of antichains $A$

=> each incidence vector $x^A$ fulfills $x^A(K) \leq 1$

=> the convex combination $x$ fulfills $x(K) \leq 1$

(2) The LP given by the inequalities attains its optimum on incidence vectors of antichains

The LP is given by

(P) max $c^T x$ s.t. $x(K) \leq 1$ for every chain $K$

$x \geq 0$

The dual LP is

(D) min $\mathbf{1}^T y = \sum_K y_K$ s.t. $\sum_{K : e \in K} y_K \geq c_e$ for all $e \in E$

$y_K \geq 0$ for all chains $K$

Interpreting $y_K$ as "multiple occurrences" of chain $K$, the dual (D) says:

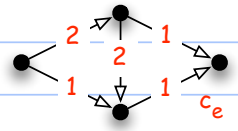
find as few as possible chains (multiple occurrences permitted) such that each element $e$ is "covered" at least $c_e$ times

=> consider only maximal chains

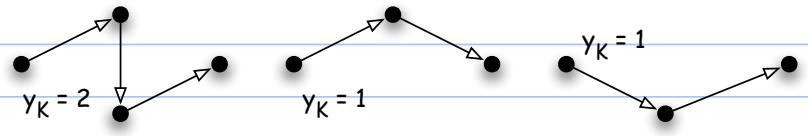
=> $\sum_K y_K$ is a flow in the edge diagram $G$ w.r.t. lower capacities $c_e$

Example:

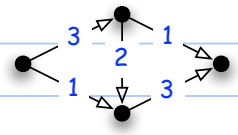
G with lower capacities




a covering with chains




the interpretation of $\Sigma_K y_K$ as flow




=> we can apply the Min-Flow-Max-Cut Theorem



Similar to the Max-Flow-Min-Cut Theorem of ADM I:

The minimum flow value in an s,t-network with lower capacities $c_e$ and upper capacities $u_e$ equals the maximum capacity cap(X) of an s,t-cut X

In this case, the capacity of an s,t-cut X with $s \in X$ defined as

$$cap(X) = \sum_{e \in \delta^+(X)} c_e - \sum_{e \in \delta^-(X)} u_e$$

In the edge diagram all upper capacities are $\infty$ and the flow value is bounded from below

=> the maximum capacity of a cut is finite

=> a cut of maximum capacity has only forward edges, i.e., $\delta^-(X) = \emptyset$

=> $A := \delta^+(X)$ is an antichain (otherwise $\delta^-(X) \neq \emptyset$) with weight = maximum capacity of a cut

Conversely, every maximal antichain A defines a cut X as

X := set of head vertices of edges in A and of all vertices of edges e with $e < a$ for some $a \in X$.

Then $A := \delta^+(X)$

So:

optimum value of (P) = optimum value of (D)

= minimum flow value = maximum capacity of an s,t-cut

= maximum weight of an antichain

=> optimum of (P) is attained by an incidence vectors of an antichain

(1) and (2) prove Theorem 8.2 by the same arguments as for the matching polytope. ❑

8.3 Remark

(P) and (D) correspond to a weighted version Dilworth's Theorem

Theorem of Dilworth:

minimum number of chains covering all elements of a partial order

= maximum cardinality of an antichain

**The traveling salesman polytope in the symmetric, complete case**

Consider the TSP on an undirected, w.o.l.g. complete graph $K_n$ with edge costs $c(e) \geq 0$.

Then $(E, \mathcal{F}, c)$ is given by

$E$ = edge set $E(K_n)$

$\mathcal{F}$ = { $T \subseteq E$ | $T$ is a TSP-tour }

$c(e)$ = non-negative edge weight of $e$



$P_{\mathcal{F}}$ := conv { $x^T$ | $T$ is TSP-tour } is called the TSP polytope and is denoted by $Q^n_{TSP}$

The vertices must fulfill

$x(\delta(v)) = 2$ for all vertices $v$ (8.1)

=> $\dim(Q^n_{TSP}) \leq m - n = \frac{n(n-3)}{2}$ One can even show that $dim(Q^n_{TSP}) = m - n$

=> (8.1) defines a linear system of maximum rank whose set of solutions contains $Q^n_{TSP}$

Constraints (8.1) still permit subtours.

These can be avoided by the subtour elimination constraints:

$x(E(W)) \leq |W| - 1$ for all $\emptyset \neq W \subset V$ (8.2)

where $E(W)$ = set of edges with both end points in $W$

Furthermore (non-negativity constraints, upper bounds)

$0 \leq x_e \leq 1$ for all edges $e$ (8.3)

- $Q^n_{TSP}$ is contained in the following polytopes (which are relaxations of $Q^n_{TSP}$)
  - in the 1-tree polytope (= convex hull of the incidence vectors of 1-trees), see Section 7.4
  - in the 2-matching polytope $Q^n_{2M}$ (= convex hull of the incidence vectors of perfect 2-matchings)

    perfect 2-matching = edge set $F$ such that every vertex is incident to exactly 2 edges from $F$

- For the perfect 2-matching polytope, a linear description is known for arbitrary graphs (Edmonds 1965)
  - $Q^n_{2M} = \{ x \in \mathbb{R}^E \mid 0 \le x_e \le 1$ for all edges $e$

    $x(\delta(v)) = 2$ for all vertices $v$

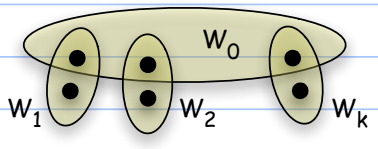
    $\sum_{i=0}^{k} x(E(W_i)) \le |W_0| + \frac{1}{2}(k-1)$ $k$ odd (8.4)

    with $W_0, W_1, \ldots, W_k \subseteq V$

    $|W_0 \cap W_i| = 1 = |W_i - W_0|$,

    $W_i \cap W_j = \emptyset$ for $i, j > 0$ $\}$






- The 2-matching inequalities can be generalized to comb inequalities for the TSP polytope
  - Let $G$ be a complete graph, and $W_0, W_1, \ldots, W_k \subseteq V$ with

    $|W_0 \cap W_i| \ge 1$ $i = 1, \ldots, k$

    $|W_i - W_0| \ge 1$ $i = 1, \ldots, k$

    $W_i \cap W_j = \emptyset$ for $i, j > 0$

    $k \ge 3$, odd

    Then

    $$\sum_{i=0}^{k} x(E(W_i)) \le |W_0| + \sum_{i=0}^{k} (|W_i| - 1) - \frac{k+1}{2} \quad (8.5)$$

    defines valid inequalities for the TSP polytope
  - The comb inequalities have Chvátal rank 1 w.r.t. the 2-matching polytope.

8.4 Theorem (Inequalities for the TSP polytope, Chvátal 1973, Grötschel & Padberg 1979)

For the TSP polytope $Q^n_{TSP}$ the following statements hold for $n \geq 6$

(a) $\dim(Q^n_{TSP}) = m - n$

(b) the inequalities (8.3) define facets for every edge $e$

(c) the subtour elimination constraints (8.3) are facets if $3 \leq |W| \leq n-3$

(d) all comb inequalities (8.5) are facets

without proof ❑

8.5 Remark

(a) - (d) do not constitute a linear description, and none is known.

for $Q^{10}_{TSP}$, more than 50 billion different facets are known, for $Q^{120}_{TSP}$ there are more than $10^{179}$

=> complete linear descriptions become too large to be solved by LP algorithms directly

But one can use partial linear descriptions in Branch & Cut. For a fixed objective function one needs only cutting planes that lead to an optimal vertex. Other parts of the polytope are then of no interest.



Goal of this section

Discussion of a concrete branch & cut algorithm for the TSP problem

Examples for polynomial separation

Examples of how to show that inequalities define facets

**A branch & cut algorithm for the TSP problem**

The algorithm starts with the LP

(P) min $c^T x$ s.t.

$x(\delta(v)) = 2$ for all vertices $v$ (8.1)

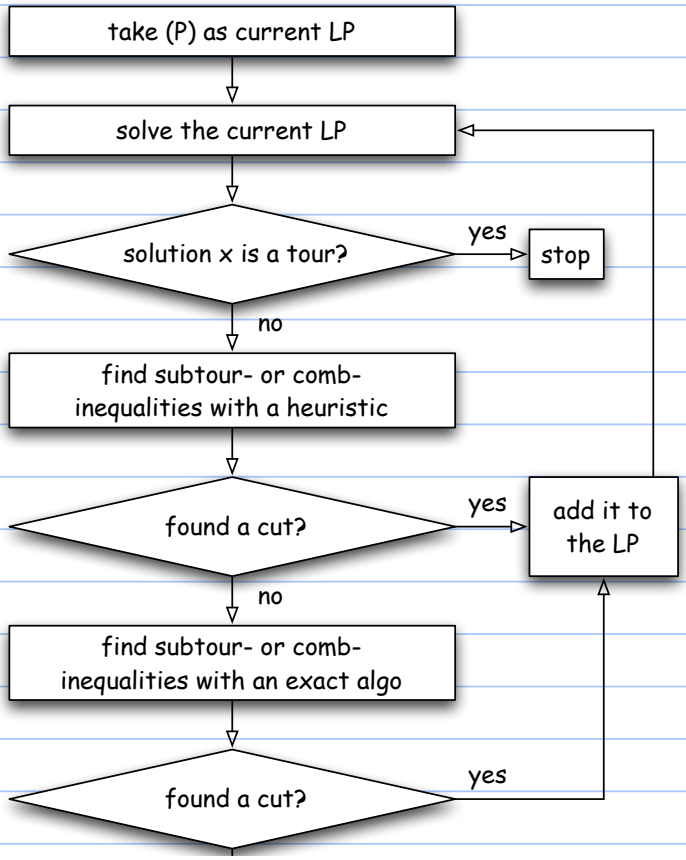
$0 \leq x_e \leq 1$ for all edges $e$ (8.3)

and first adds subtour elimination inequalities and then comb inequalities

(first with a fast heuristic, then with an exact algorithm).

When no more such inequalities are found, one switches to branch and bound or a general cutting plane algorithm

A flow diagram of the algorithm






no

use branch & bound or a general cutting plane algorithm

- This principle is used recursively in the nodes of the B&B tree. When good lower bounds are available one usually needs only a few iterations in the final step.
- This method can solve TSP problems with up to several thousands vertices exactly.

  With additional techniques one can solve TSP problems with 100.000 and more vertices, see Concorde.

@ http://www.tsp.gatech.edu/concorde/index.html

- Branch & Cut motivates
  - the construction of fast separation algorithms
  - the search for valid inequalities or facets

- **Designing fast separation algorithms**
  - this often leads to other combinatorial optimization problems

Example 1: Chain inequalities for the antichain polytope

They have the form

$x(K) \leq 1$ for every chain $K$

For a vector $x^*$, the separation problem for chain inequalities can be solved as follows:

(1) compute the longest chain $K^*$ w.r.t. weights given by $x^*$

(2) if $x^*(K^*) > 1$, then $x(K^*) \leq 1$ is an $x^*$ separating chain inequality

if $x^*(K^*) \leq 1$, then $x^*$ fulfills all chain inequalities

The longest chain can be computed in the edge diagram as longest path (this is polynomial, as the edge diagram is acyclic)

Example 2: Subtour-elimination inequalities for the TSP polytope

we assume that the following inequalities are satisfied

$x(\delta(v)) = 2$ for all vertices $v$ degree constraints (8.1)

$0 \leq x \leq 1$ non-negativity and upper bounds (8.3)

(which can be checked efficiently)



8.6 Lemma (Separation of subtour elimination inequalities)

Assume that $x^*$ fulfills inequalities (8.1) and (8.3)

Then there is an $x^*$ separating subtour inequality

<=> there is a cut $\delta(X)$ with $x^*(\delta(X)) < 2$

Proof

Let $X$ be a vertex set, $\emptyset \neq X \neq V$. Then

$x^*(\delta(X)) = \sum_{v \in X} \sum_{e \in \delta(v)} x_e^* - 2 \sum_{e \subseteq X} x_e^*$

$= \sum_{v \in X} 2 - 2 \sum_{e \subseteq X} x_e^*$ because of (8.1)

$= 2|X| - 2 \sum_{e \subseteq X} x_e^* = 2|X| - 2x^*(E(X))$

So $x^*(\delta(X)) + 2x^*(E(X)) = 2|X|$ (*)

"<=>"

$x^*$ violates the subtour inequality w.r.t. $X$

<=> $x^*(E(X)) > |X| - 1$ <=> $2x^*(E(X)) > 2|X| - 2$ <=> $x^*(\delta(X)) < 2$ with (*) ❑

8.7 Consequence (Separation of subtour elimination inequalities)

Given (8.1), the separation problem for subtour-elimination inequalities leads to computing a cut $\delta(X)$ with minimum capacity. This can be done by solving a sequence of s,t-max-flow problems.

- Without proof ❑

**Searching for valid inequalities or facets**

We demonstrate this for the antichain polytope

8.8 Theorem (Facets of the antichain polytope)

Consider the antichain polytope $P_{\mathcal{A}}$. Then:

$x_e = 0$ defines a facet for every edge $e$.

$x(K) = 1$ defines a facet <=> $K$ is a maximal chain.

Proof

(1) $\dim(P_{\mathcal{A}}) = |E|$, i.e., $P_{\mathcal{A}}$ has full dimension

The unit vectors correspond to singleton antichains and are linearly independent.

Because of (1), a facet is a supporting hyperplane $H$ whose cut with $P_{\mathcal{A}}$ has dimension $|E| - 1$

=> to show this dimension, one must find $m := |E|$ vectors $x^0, \ldots, x^{m-1}$ in $H \cap P_{\mathcal{A}}$ such that $x^i - x^0$ are linearly independent $(i = 1, \ldots, m-1)$

(2) $x_e = 0$ defines a facet for every edge $e$



set $x^0 := 0$, $x^i :=$ i-th unit vector except for $i = e$

=> all these vectors lie on the hyperplane $x_e = 0$

and the $x^i - x^0$ are linearly independent

(3) $x(K) = 1$ defines a facet <=> $K$ is a maximal chain

"<="

let $K = \{ e_1, \ldots, e_k \}$ and $E\text{-}K = \{ e_{k+1}, \ldots, e_m \}$

$K$ is maximal => for every $e_j \in E\text{-}K$ there is $e_{i(j)} \in K$ such that $e_j, e_{i(j)}$ are incomparable

=> $\{e_1\}, \ldots, \{e_k\}, \{e_{k+1}, e_{i(k+1)}\}, \ldots, \{e_m, e_{i(m)}\}$ are antichains

and their incidence vectors $y^0, \ldots, y^{m-1}$ lie on the hyperplane $x(K) = 1$

$y^0, \ldots, y^{m-1}$ are obviously linearly independent

=> $y^i - y^0$ are linearly independent => $x(K) = 1$ is a facet

"=>"

Consider a chain $K$ such that $x(K) = 1$ is facet, but $K$ is not maximal.

Let $K'$ be maximal chain with $K \subset K'$, and let $e' \in K'\text{-}K$.

Claim: $\{ x(K) = 1 \} \cap P_{\mathcal{A}} \subseteq \{ x(K') = 1 \} \cap P_{\mathcal{A}}$

let $x \in P_{\mathcal{A}}$ and $x(K) = 1$

$K \subseteq K'$ => ( $x(K) = 1$ => $x(K') \geq 1$ )

but since $x \in P_{\mathcal{A}}$ we have $x(K') = 1$ => $x \in \{ x(K') = 1 \} \cap P_{\mathcal{A}}$

Facets $H$ define inclusions-maximal sets $H \cap P_{\mathcal{A}}$ among all faces of $P_{\mathcal{A}}$

But the incidence vector of $\{e'\}$ lies in $\{ x(K') = 1 \} \cap P_{\mathcal{A}} - \{ x(K) = 1 \} \cap P_{\mathcal{A}}$,

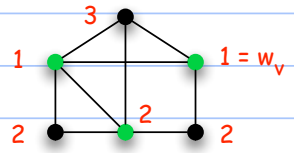
a contradiction ❑

- Goals of this chapter
  - Demonstrate on 3 selected techniques that LP-theory provides advanced methods to design approximation algorithms
  - Please repeat the chapter on approximation algorithm from ADM I

- Goals of this section
  - Approximation algorithms based on solving an LP with subsequent rounding to an integer solution
  - Proving approximation guarantees by the use of LP-duality and dual solutions

- As an example, we consider WEIGHTED VERTEX COVER (WVC)
  - Instance
    - an undirected graph $G$ with vertex weights $w_v \geq 0$
  - Task
    - Determine a vertex cover $C$ of $G$ with minimum weight $\sum_{v \in C} w_v$
  - Example






$C$ = green vertices

- **Simple rounding (for a minimization problem)**
  - Algorithm Simple Rounding
    - Formulate the given instance I as an integer program (IP)
    - solve the LP-relaxation (LP) of (IP) in polynomial time
      - Recall that approximation algorithms must run in polynomial time (in the encoding length <I> of I)
      - If (LP) has only polynomially many variables and inequalities in the encoding length of I, then it can be solved in polynomial time with one of the known polynomial LP-algorithms (see Chapter 10).
      - However, (LP) often has exponentially many inequalities. Then we must show that the separation problem for these inequalities is solvable in polynomial time. Theorem 7.26 then shows that (LP) can be solved in polynomial time.
    - Round the fractional optimal solution of (LP) to a feasible solution of (IP)
      - This rounding is problem dependent. It need not work in general.

9.1 Lemma (Approximation guarantee for simple rounding)

Let A(I) be the feasible solution of (IP) obtained by rounding.

Let OPT(I) be an optimal solution of the given instance I and let LP(I) be an optimal solution of the LP-relaxation.

If

$A(I) \leq \rho \cdot LP(I)$ for every instance I,

then algorithm "Simple Rounding" is a $\rho$-approximation algorithm

Remark: it is common in the theory of approximation algorithms to use A(I), OPT(I) both for the solution itself and for the value of that solution.

Proof

As (LP) is a relaxation of (IP), we have $LP(I) \leq IP(I) = OPT(I)$

$\Rightarrow A(I) \leq \rho \cdot LP(I) \leq \rho \cdot OPT(I)$ ❑

Application to WVC

IP-formulation (IP)



Introduce 0/1-variables $x_v$ with $x_v = 1 :\Leftrightarrow v \in C$

Then WVC is equivalent to the IP

$$\min \sum_v w_v x_v$$

$$\text{s.t.} \quad x_u + x_v \geq 1 \quad \text{for every edge } e = (u,v) \text{ of } G$$

$$x_v \in \{0, 1\} \quad \text{for every vertex } v \text{ of } G$$

The LP-relaxation (LP) of (IP)

$$\min \sum_v w_v x_v$$

$$\text{s.t.} \quad x_u + x_v \geq 1 \quad \text{for every edge } e = (u,v) \text{ of } G$$

$$x_v \geq 0 \quad \text{for every vertex } v \text{ of } G$$

$x_v \geq 0$ is sufficient, since each optimal LP solution fulfills $x_v \leq 1$ because of $w_v \geq 0$

(LP) has only polynomially many inequalities and variables and can thus be solved in polynomial time with a polynomial LP-algorithm (see Chapter 10)

The rounding

let x' be an optimal solution of (LP)

round x' to x* as follows:

$$x_v^* := \begin{cases} 1 & \text{if } x_v' \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Then:

- $x^*$ is feasible for (IP), i.e., a vertex cover
  - let (u,v) be an edge of $G$
    $\Rightarrow x_u' + x_v' \geq 1 \Rightarrow x_u' \geq 0.5$ or $x_v' \geq 0.5 \Rightarrow x_u^* = 1$ or $x_v^* = 1$
    $\Rightarrow$ the edge (u,v) is covered by $x^*$
- $A(I) \leq 2 \cdot LP(I)$
  - $x_v^* \leq 2x_v' \Rightarrow A(I) \leq 2 \cdot LP(I)$ as $w_v \geq 0$
- So algorithm "Simple Rounding" is a 2-approximation algorithm for WVC

**The use of dual solutions (for minimization problems)**

9.2 Lemma (The use of dual solutions in approximation algorithms)

Let (D) be the dual LP of the LP-relaxation (LP) of (IP).

Let dual(I) be a feasible solution of (D) for instance I.

Let A be a polynomial algorithm that constructs a feasible solution A(I) of (IP) with



$A(I) \leq \rho \cdot dual(I)$ for every instance I

Then A is a $\rho$-approximation algorithm

Proof:

- weak duality theorem $\Rightarrow dual(I) \leq LP(I)$
  (LP) is a relaxation of (IP) $\Rightarrow LP(I) \leq OPT(I)$
  So $A(I) \leq \rho \cdot dual(I) \leq \rho \cdot OPT(I)$ ❑

Remark

In contrast to Simple Rounding, the use of dual solutions need not solve an LP. It suffices that the algorithm constructs a feasible solution A(I) of (IP). The dual solution dual(I) is only needed in the proof of the inequality

$A(I) \leq \rho \cdot dual(I)$

but not in the algorithm.

Application to WVC (Bar-Yehuda & Even 1981)

The LP-relaxation (LP) of WVC (see above) is

$\min \sum_v w_v x_v$

s.t. $x_u + x_v \geq 1$ for every edge $e = (u,v)$ of $G$

$x_v \geq 0$ for all vertices $v$ of $G$

The associated dual LP (D) has a variable $y_e$ for every edge $e$ of $G$ and reads

$\max \sum_{e \in E} y_e$

s.t. $\sum_{e \in \delta(v)} y_e \leq w_v$ for every vertex $v$ of $G$

$y_e \geq 0$ for all edges $e$ of $G$

It computes edge weights $y_e \geq 0$ such that the total weight in the graph gets as large as possible, but the weight in every "star" $\delta(v)$ is at most $w_v$ (i.e. it computes a w-packing with maximum value)

The special case $w_v = 1$ for all vertices

Algorithm G (Gavril, 1974, see ADM I)

Input

an instance I of VERTEX COVER

Output

a vertex cover $G(I)$ with $G(I) \leq 2 \cdot OPT(I)$

Method



compute an $\subseteq$-maximal matching $M$ of $G$

set U := both endpoints of every matching edge from M

return U

Proving the approximation guarantee with the use of dual solutions

The set U computed by the algorithm is a vertex cover

otherwise $M$ would not be $\subseteq$-maximal

M is a dual feasible solution, M = dual(I)

clear, since every star $\delta(v)$ contains at most one matching edge

$G(I) = |U| = 2|M| = 2 \cdot dual(I) \Rightarrow G(I) \leq 2 \cdot dual(I) \Rightarrow$ approximation guarantee 2 with Lemma 9.2 ❑

The general case with arbitrary weights $w_v \geq 0$

Call a vertex $v$ saturated if $\sum_{e \in \delta(v)} y_e = w_v$

Algorithm PACK

Input

- an instance $I$ of WVC with $E \neq \emptyset$ and w.o.l.g. $w_v > 0$ for all $v$
  // vertices $v$ with $w_v = 0$ will be taken and PACK is only applied to the graph induced by the remaining vertices
- Output
  - a vertex cover PACK(I) with $PACK(I) \leq 2 \cdot OPT(I)$
- Method
  - set $C := \emptyset$ and $y_e = 0$ for all edges $e$
  - repeat
    - choose an edge $e$
    - increase the value of the dual variable $y_e$ until one (or both) endpoints of $e$ are saturated
    - add the saturated endpoint(s) of $e$ to $C$
    - delete the saturated endpoint(s) of $e$ and all incident edges
  - until no edges are left
  - return $C$

- Proving the approximation guarantee with the use of dual solutions



- The set $C$ computed by the algorithm is a vertex cover
  - an edge $e$ is only deleted if at least one of its endpoints $u$ is saturated
    => $u \in C$ and $u$ covers edge $e$
- The edge weights at the end of the algorithm constitute a dual feasible solution dual(I)
  - clear, since $y_e \geq 0$ and $\sum_{e \in \delta(v)} y_e \leq w_v$ throughout the algorithm
- $PACK(I) = \sum_{v \in C} w_v \leq 2 \cdot \sum_{e \in E} y_e = 2 \cdot dual(I)$
  => approximation guarantee 2 with Lemma 9.2
  - interpret the increase of $y_e$ by k as paying k $ to each endpoint of $e$
    => a vertex $v$ has been paid $w_v$ when it enters the set $C$
    => $\sum_{v \in C} w_v \leq$ total payment to all vertices $= 2 \sum_{e \in E} y_e$ ❑

- Goals of this section
  - Illustrate approximation algorithms based on solving an LP with subsequent randomized rounding, i.e., rounding with probabilities obtained from the optimal LP solution
  - We take MAX SAT as illustrative example, see ADM I, Chapter 9.4

- A trivial randomized rounding for MAX SAT (see ADM I, chapter 9.4)
  - Algorithm Randomize (Johnson 1974)
    - Input
      - an instance of MAX SAT
      - at least $k$ literals ($k \geq 1$) per clause $Z$
      - weight $c(Z)$ per clause $Z$
    - Output
      - a random truth assignment with expected performance

        $E[\sum_Z c(Z)] \geq (1 - 1/2^k) \cdot OPT(I)$
    - Method
      - toss a fair coin for every Boolean variable $x_i$ and set



        - $x_j$ := TRUE if the coin shows head
        - $x_j$ := FALSE if the coin shows number
      - return the resulting random truth assignment

  - This algorithm is good for $k \geq 2$ (it gives at least 3/4 of the optimal weight), but is bad for $k = 1$ (where it gives only 1/2 of the optimal weight).

- Randomized rounding based on an LP relaxation
  - The general principle of randomized rounding (Raghavan & Thompson 1987)
    1. Model the problem as an IP — variables $x_j \in \{0, 1\}$
    2. Relax the IP to an LP — $0 \leq x_j \leq 1$
    3. Solve the LP optimally — values $x_j'$
    4. Round randomized

       set $x_j = 1$ with probability $x_j'$
    5. Show that the resulting vector $x$

       is feasible for the IP

has a good expected approximation guarantee

- Randomized rounding based on an LP relaxation for MAX SAT (Goemans & Williamson 1993)
  - The IP
    - 0/1 variable $y_i$ = truth value of Boolean variable $x_i$
    - 0/1 variable $z_j$ = truth value of clause $C_j$
    - $T_j$ = set of unnegated variables in clause $C_j$
    - $F_j$ = set of negated variables in clause $C_j$
    - The IP then is

$$\max \sum_j w_j z_j$$

$$\text{s.t.} \sum_{i \in T_j} y_i + \sum_{i \in F_j} (1 - y_i) \geq z_j$$

$$y_i \in \{0,1\}, \quad z_j \in \{0,1\}$$

  - Randomized rounding
    - Let $(y^*, z^*)$ be an optimal solution of the LP relaxation of the IP



Use $y^*$ for randomized rounding, i.e., set

$x_i$ := TRUE with probability $y_i^*$

$x_i$ := FALSE with probability $1-y_i^*$

This trivially produced a truth assignment of the given instance.

Of course, not all clauses will be satisfied (i.e., evaluate to TRUE).

- The performance guarantee
  - Consider w.o.l.g. the clause $C_j = x_1 \vee x_2 \vee \ldots \vee x_k$ (it is similar for negated variables and other indices).
  - Then

$$\text{Prob}[C_j \text{ is satisfied}] = 1 - \prod_{i=1}^{k} (1 - y_i^*)$$

$$\geq 1 - \left(1 - \frac{1}{k} \sum_{i=1}^{k} y_i^*\right)^k \quad \text{since geometric mean} \leq \text{arithmetic mean}$$

$$\geq 1 - \left(1 - \frac{1}{k} z_j^*\right)^k \quad \text{because of the LP inequality}$$

  - Now set

$$f(z) := 1 - \left(1 - \frac{1}{k}z\right)^k \quad \text{concave in z}$$

and

$$g(z) := \left(1 - \left(1 - \frac{1}{k}\right)^k\right) z \quad \text{linear in z}$$

Hence $f(z) \geq g(z)$ on the interval [0,1] if $f(z) \geq g(z)$ for the endpoints $z = 0$ and $z = 1$ of the interval

Checking $z = 0$: $f(0) = 0$, $g(0) = 0$

Checking $z = 1$: $f(1) = g(1)$

So

$$\text{Prob}[C_j \text{ satisfied}] \geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right) z_j^*$$

=> E(total weight of all satisfied clauses)

$$\geq \min_k \left(1 - \left(1 - \frac{1}{k}\right)^k\right) \sum_j w_j z_j^* \geq \left(1 - \frac{1}{e}\right) \sum_j w_j z_j^*$$



$$\geq \left(1 - \frac{1}{e}\right) OPT(I) \text{ since } z^* \text{ is an optimal solution of the LP-relaxation}$$

$$\approx 0,623 \cdot OPT(I)$$

So this algorithm achieves in expectation a performance guarantee of at least 0,623 of the optimum weight also for instances with only one literal in some clauses.

Combining both algorithms for MAX SAT

Toss a coin to decide which algorithm to use (Johnson or Randomized Rounding) and run the chosen algorithm.

This is again a randomized algorithm with an expected approximation guarantee of (3/4)

Proof:

Consider a clause $C_j$ with k literals

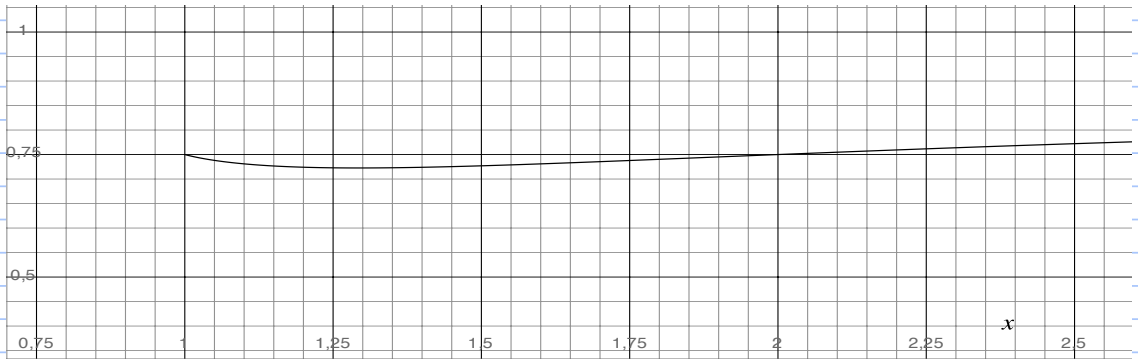
Then

$$\text{Prob}[C_j \text{ satisfied}] \geq \frac{1}{2}\left(1 - \frac{1}{2^k}\right) + \frac{1}{2}\left(1 - \left(1 - \frac{1}{k}\right)^k\right) z_j^*$$

$$\geq \frac{1}{2}\left(1-\frac{1}{2^k}\right)z_j^* + \frac{1}{2}\left(1-\left(1-\frac{1}{k}\right)^k\right)z_j^* \quad \text{since } 0 \leq z_j^* \leq 1$$

$$= f(k)\cdot z_j^* \text{ with } f(k) := \frac{1}{2}\left(1-\frac{1}{2^k}\right) + \frac{1}{2}\left(1-\left(1-\frac{1}{k}\right)^k\right)$$

- Now f(1) = 3/4 and f(x) ≥ 3/4 on the interval [2,∞] (calculus)




=> E(total weight of all satisfied clauses) ≥ 3/4 $\sum_j w_j z_j^*$ ≥ 3/4 OPT(I) ❑



- Goals of this section
  - Introduction of the primal-dual scheme for constructing approximation algorithms
  - We demonstrate this on the network design problem

- **The network design problem**
  - Instance
    - an undirected graph G = (V, E)
    - edge costs $c_e \geq 0$
    - connection requirements $r_{ij}$ for any 2 vertices i, j
  - Wanted
    - an edge set $F \subseteq E$ with minimal cost $\sum_{e \in F} c_e$ such that
    - G' := (V, F) contains at least $r_{ij}$ pairwise edge disjoint paths between i and j for any 2 vertices i, j

  - The network design problem is NP-hard (Karp 1972).

    It arises in designing low-cost networks that can survive edge failures

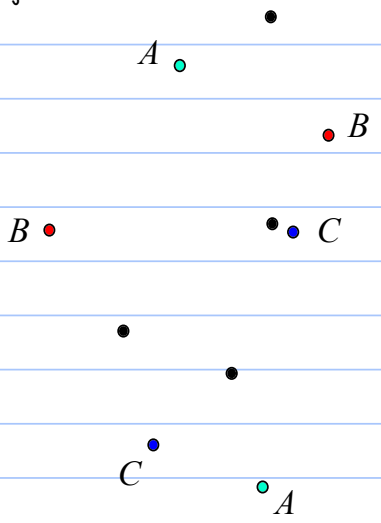
Construction of a VPN as special case (see Section 4.4)

Example

given vertices with requirements

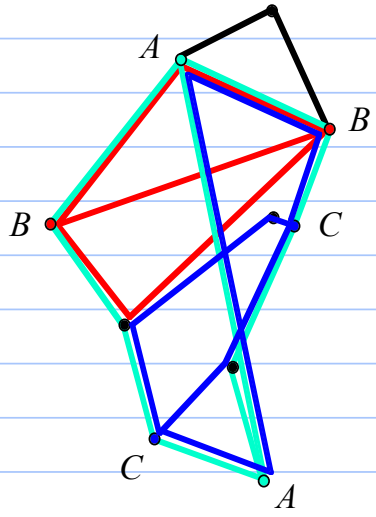
$r_{ij}$ = 3 between colored vertices

$r_{ij}$ = 2 otherwise




a solution of the

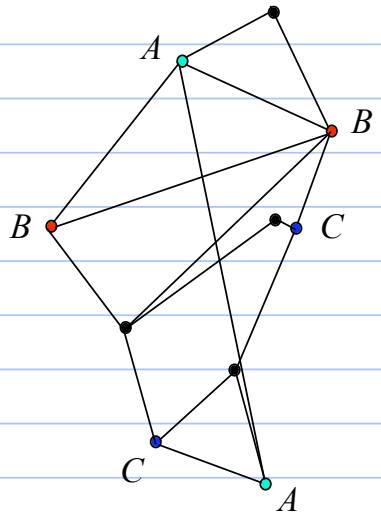
network design problem




The solution "survives" up to 2 edges failures on connections between colored vertices and one edge failure on



connections to a black vertex. The solution was the example VPN for the disjoint path problem in Section 4.4.




A formulation of the network design problem as IP

set $f(S) := \max \{ r_{ij} \mid i \in S, j \notin S \}$ for every vertex set $S \neq \emptyset, V$ (demand of $S$)

introduce a 0/1 variable $x_e$ for choosing edge $e$

(IP):

$$\min \Sigma_e c_e x_e$$

$\Sigma_{e \in \delta(S)} x_e \geq f(S)$ for all $\emptyset \neq S \subset V$ (cut conditions)

$x_e \in \{0, 1\}$ for all edges e

x is a solution of (IP) <=> $F = \{ e \in E \mid x_e = 1 \}$ is a solution of the network design problem

"<="

trivial

"=>"

Max Flow Min Cut Theorem (applied to any pair i, j with edge capacities 1) + cut constraints

=> there is a flow from i to j with value $\geq r_{ij}$

=> there is an integer flow from i to j with value $\geq r_{ij}$

=> there are $r_{ij}$ pairwise edge disjoint paths from i to j

=> the conditions in the (IP) are also sufficient ❑

Some special cases

Shortest s,t-paths

$r_{st} = 1$, $r_{ij} = 0$ otherwise

$f(S) = 1$ if $|S \cap \{s, t\}| = 1$, $f(S) = 0$ otherwise



Minimum spanning trees

$r_{ij} = 1$ for all pairs i, j

$f(S) = 1$ for all $\emptyset \neq S \subset V$, $f(S) = 0$ otherwise

Minimum Steiner tree

$r_{ij} = 1$ for all $i, j \in T$ (T = set of terminals that need to be connected)

$f(S) = 1$ if $S \cap T \neq \emptyset$ and $T - S \neq \emptyset$, $f(S) = 0$ otherwise

Generalized Steiner tree problem

:<=> $f(S) \in \{0, 1\}$

**The primal-dual scheme**

It uses complementary slackness similar to the primal-dual algorithm of Chapter 6.

We recall it below in the form needed here:

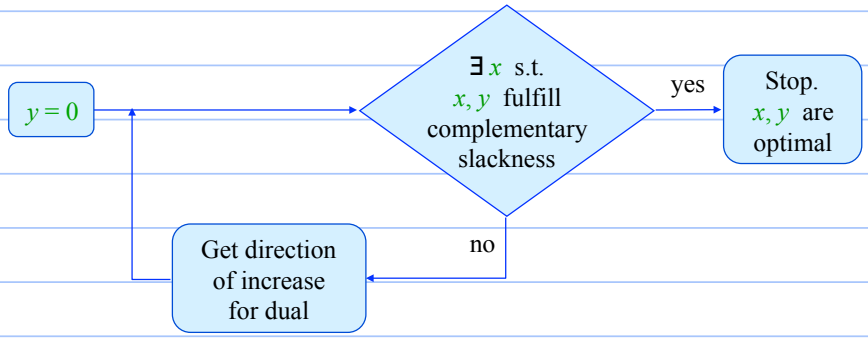
Starting point are an LP

(P) $\min \Sigma_j c_j x_j$ s.t. $\Sigma_j a_{ij} x_j \geq b_i$ for all i, $x_j \geq 0$ for all j

and the associated dual

(D) $\max \Sigma_i b_i y_i$ s.t. $\Sigma_i a_{ij} y_i \leq c_j$ for all j, $y_i \geq 0$ for all i

- Complementary slackness conditions then read
  - $x_j > 0 \Rightarrow \sum_i a_{ij} y_i = c_j$ (primal slackness condition)
  - $y_i > 0 \Rightarrow \sum_j a_{ij} x_j = b_i$ (dual slackness condition)
- A primal feasible $x$ and a dual feasible $y$ are optimal
  - $\Leftrightarrow$ $x$ and $y$ fulfill these conditions (Theorem 4.4)
- The primal dual algorithm of Chapter 6 then runs through the following loop




- The primal-dual scheme for approximation algorithms
  - Model the problem as IP



- Relax the IP to an LP
- Relax the dual slackness condition ($y_i > 0 \Rightarrow \sum_j a_{ij} x_j = b_i$)
- Use the loop to construct a feasible solution $x$ for the IP and a dual feasible solution $y$
- Show that $x$ and $y$ fulfill the inequality $\sum_j c_j x_j \leq \alpha \cdot \sum_i b_i y_i$
  - $\Rightarrow$ $\alpha$-approximation because of Lemma 9.2

- The primal-dual pair for the network design problem with $f(S) \in \{0, 1\}$
  - IP:
    - $\min \sum_e c_e x_e$
      - $\sum_{e \in \delta(S)} x_e \geq f(S)$ for all $\emptyset \neq S \subset V$ (cut constraints)
      - $x_e \in \{0, 1\}$ for all edges $e$
  - LP relaxation = primal LP
    - $\min \sum_e c_e x_e$
      - $\sum_{e \in \delta(S)} x_e \geq f(S)$ for all $S \subset V$ with $f(S) = 1$
      - $x_e \geq 0$ for all edges $e$
    - The condition $x_e \leq 1$ may be dropped, as it will hold in the optimum because of $c_e \geq 0$

- Dual LP
  - max $\sum_{S:f(S)=1} y_S$

    $\sum_{S:e\in\delta(S)} y_S \leq c_e$ for all edges $e$

    $y_S \geq 0$ for all variables $y_S$
  - Call an edge $e$ saturated if $\sum_{S:e\in\delta(S)} y_S = c_e$

    The primal slackness condition then says: $x_e > 0 \Rightarrow e$ saturated

- The primal dual algorithm for $f(S) \in \{0, 1\}$
  - Input
    - Instance of the network design problem with $f(S) \in \{0, 1\}$
  - Output
    - Feasible solution $(V, A)$ of the network design problem with performance guarantee 2
  - Method
    - Initialize all dual variables $y_S := 0$
    - Initialize the primal solution (as edge set $A$) $A := \emptyset$
    - while $A$ is not a feasible solution do

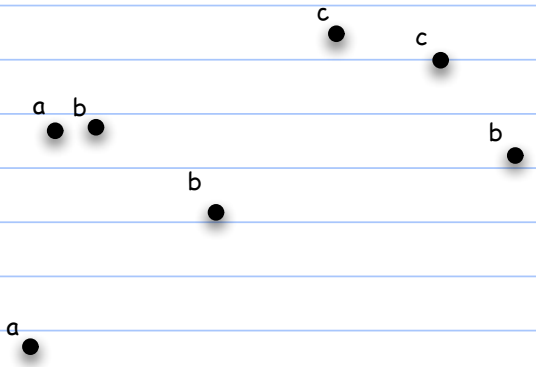


      - let $C$ be the set of all connected components $S$ in the graph $(V, A)$ of the edges of $A$ with $f(S) = 1$
      - increase $y_S$ for all $S \in C$ by the same amount until some edge $e \notin A$ becomes saturated
      - add all saturated edges to $A$
    - remove redundant edges from $A$ ($A$ $\subseteq$-minimal feasible) // cleanup step
    - return $A$

- Example for Euclidean distances

Dual variables are represented by moats around the connected components $S \in C$

Initialize: $A := \emptyset$, $y_S := 0$ for all $S$

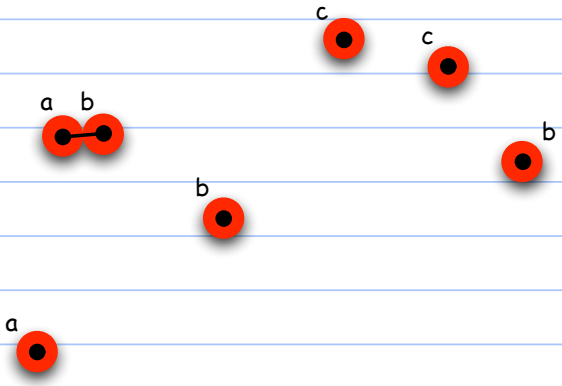



$r_{ij} = 1$ for $i,j = a$, $i,j = b$ and $i,j = c$

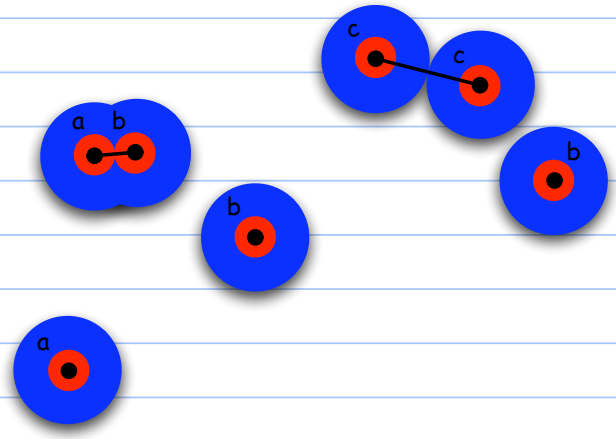
=> initially, all connected components $S \in C$ are singletons

Iteration 1






Iteration 2

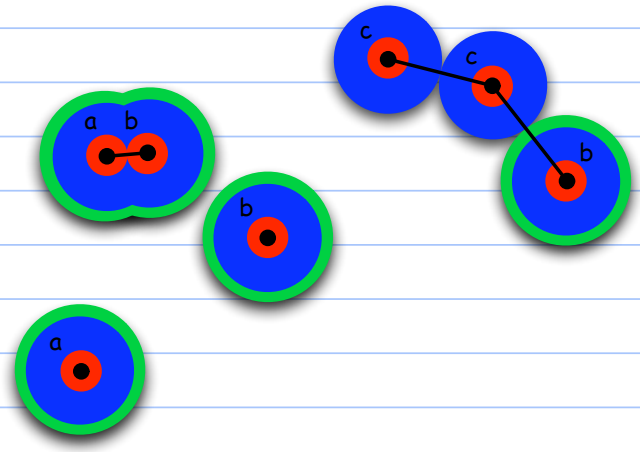



The newly created connected component S (the two c-vertices) has f(S) = 0,

so the associated dual variable $y_S$ will not grow in the next iteration.

Iteration 3

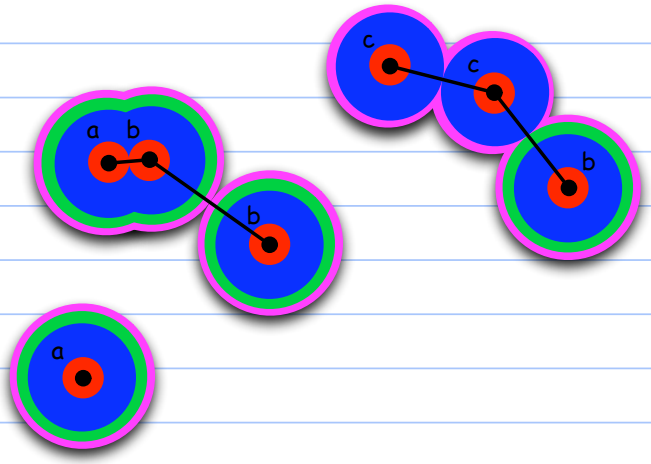





The newly created connected component S (the two c-vertices and the b-vertex) has f(S) = 0,

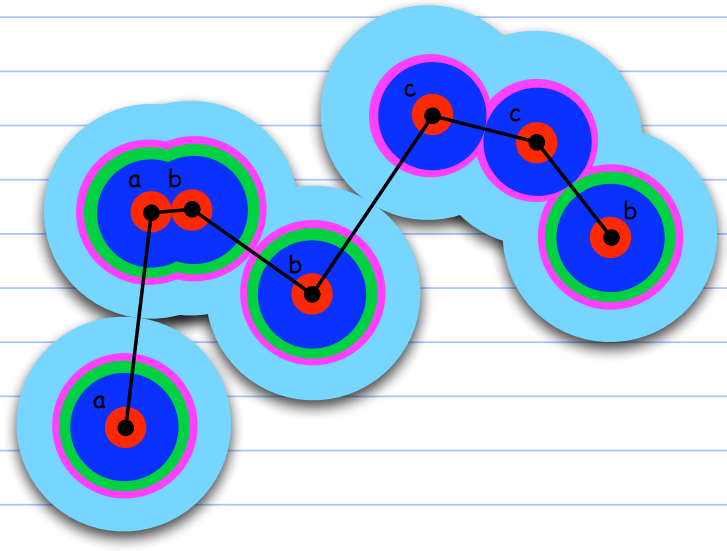
so the associated dual variable $y_S$ will grow in the next iteration.

Iteration 4




Iteration 5

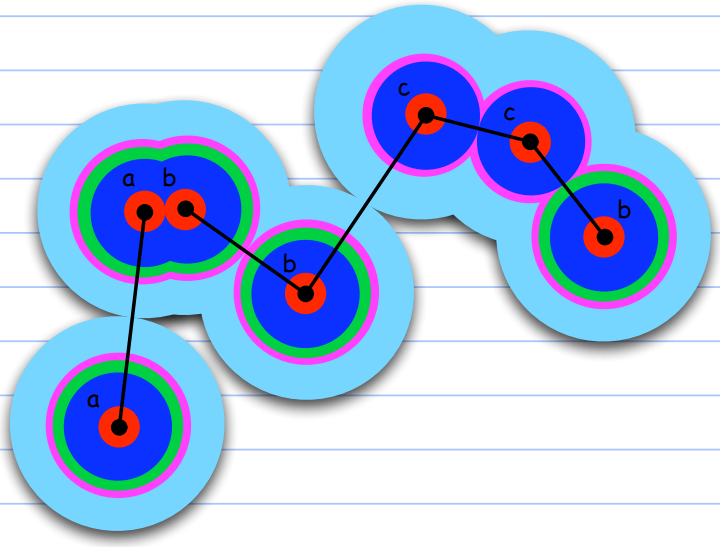





This last iteration added 2 saturated edges to A.

Cleanup step

The edge between a and b is redundant and is removed.




**Performance guarantees of the primal dual scheme for f(S) ∈ { 0, 1 }**

9.3 Theorem (Performance guarantee and runtime of the primal dual algorithm, Agarwal, Klein & Ravi 1991, Goemans & Williamson 1992)

The primal dual algorithm for $f(S) \in \{ 0, 1 \}$ can be implemented with a runtime of $O(n^2 \log n)$.



The computed primal solution $A$ fulfills $\sum_{e \in A} c_e \leq 2 \cdot OPT(I)$, i.e., the algorithm is a 2-approximation algorithm.

Proof

with Lemma 9.4 and 9.5 (only the performance guarantee, not the runtime) ❑

The essential combinatorial inequality

The algorithm runs through iterations $k = 1, 2, \ldots, K$

In iteration $k$, let

$A_k$ be the edge set at the start of that iteration

$C_k$ be the set of connected components that grow in iteration $k$ (i.e. $f(S) = 1$)

$\varepsilon_k$ be the value by which all $y_S$ with $S \in C_k$ grow

9.4 Lemma (Combinatorial inequality giving the approximation guarantee)

The primal dual algorithm is an $\alpha$-approximation algorithm if, in each iteration $k$, the inequality

$$\sum_{S \in C_k} |\delta(S) \cap D| \leq \alpha \cdot |C_k|$$

holds for every $\subseteq$-minimal feasible solution $D$ containing $A_k$

- Interpretation of the combinatorial inequality:
  - Every $\subseteq$-minimal feasible superset $D$ of $A_k$ must add some edges for every cut $\delta(S)$ with $S \in C_k$.
  - The inequality says that the number of these edges is bounded by $\alpha$ times the number of these sets $S$.
  - So every iteration $k$ adds "on average" at most $\alpha$ edges to such a set $S$.
- Proof for $f(S) \in \{0, 1\}$:
  - Let $y$ be the dual solution constructed by the algorithm.
  - Let $A$ be the primal solution constructed by the algorithm.
  - cleanup step => $A$ is contained in a set $D$ for which the combinatorial inequality holds by assumption.
  - => the combinatorial inequality is also true for $A$
  - For the value of $A$ we obtain:

$$\sum_{e \in A} c_e = \sum_{e \in A} \sum_{S : e \in \delta(S)} y_S \quad \text{because of primal slackness}$$

$$= \sum_{S : \delta(S) \cap A \neq \emptyset} |A \cap \delta(S)| \cdot y_S \quad \text{rearranging terms}$$

$$= \sum_{S : \delta(S) \cap A \neq \emptyset} |A \cap \delta(S)| \cdot \sum_{k : S \in C_k} \varepsilon_k \quad \text{total growth of } y_S$$



$$= \sum_{k} \left( \sum_{S \in C_k} |A \cap \delta(S)| \right) \varepsilon_k \quad \text{rearranging terms}$$

$$\leq \sum_{k} (\alpha \cdot |C_k|) \, \varepsilon_k \quad \text{because of the combinatorial inequality}$$

$$= \alpha \sum_{k} |C_k| \varepsilon_k = \alpha \sum_{S} y_S$$

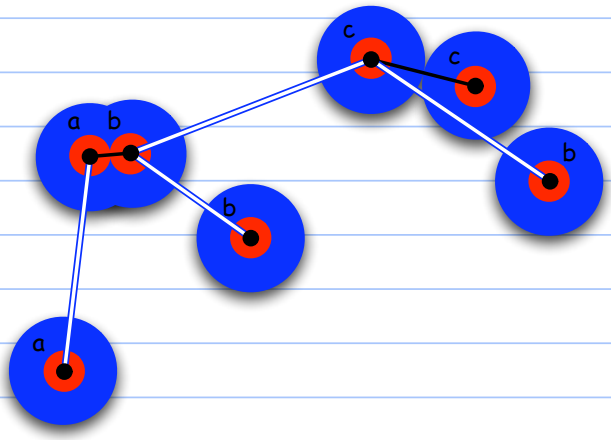
$\leq \alpha \cdot OPT(I)$ because of Lemma 9.2 ❑

**9.5 Lemma (Combinatorial inequality for $f(S) \in \{0, 1\}$)**

For $f(S) \in \{0, 1\}$, the primal dual algorithm fulfills the combinatorial inequality with $\alpha = 2$.

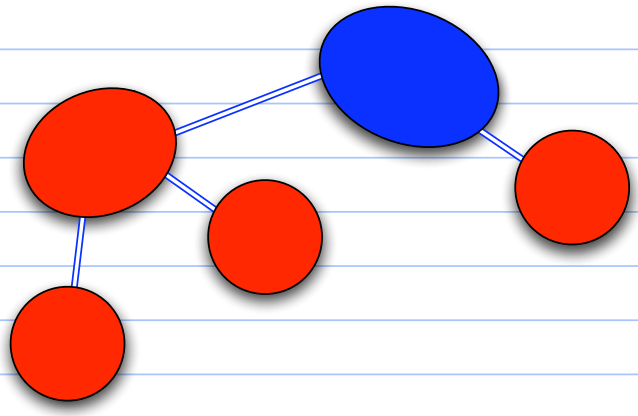
Proof:

- Consider iteration $k$ and an $\subseteq$-minimal feasible superset $D$ of $A_k$




- Contract the connected components of $(V, A_k)$ to "super nodes"
- D is $\subseteq$-minimal feasible, $f(S) \in \{0, 1\}$ => D is a forest in the contracted graph
- Color the super nodes corresponding to sets $S \in C_k$ red, the others blue





Vertex $v = S \in C_k$ => $|\delta(S) \cap D|$ = degree $d(v)$

=> the combinatorial inequality reduces to

$\sum_{v \text{ red}} d(v) \leq 2 \cdot (\#\text{ red vertices})$ in the contracted graph

Claim: no blue vertex has degree 1

- otherwise the corresponding edge would be necessary for feasibility
- => the vertex should be red

Delete blue vertices with degree 0 (they do not influence the inequality that we must show).

The resulting subgraph (which still is a forest) fulfills:

$\Sigma_{v \text{ red}} d(v) = \Sigma_{v \text{ red or blue}} d(v) - \Sigma_{v \text{ blue}} d(v)$

$\leq 2 \cdot (\#\text{ red} + \#\text{ blue}) - \Sigma_{v \text{ blue}} d(v)$ because #edges ≤ #vertices in a forest,

$\leq 2 \cdot (\#\text{ red} + \#\text{ blue}) - 2 \cdot (\#\text{ blue})$ because of the claim

$= 2 \cdot (\#\text{ red})$ ❑

**Performance guarantees in the primal dual scheme for arbitrary values f(S)**

Goemans, Mihail, Vazirani & Williamson 1993

Iteratively use a variation of the primal dual algorithm

This gives a $2 \cdot H(R)$-approximation algorithm with

$R := \max_{ij} r_{ij}$ and

$H(R) := 1 + 1/2 + 1/3 + \ldots + 1/R \sim \log R$

**Experience with the primal dual algorithm in practice**

Steiner trees (Hall 1995)

60 instances from Beasley



500 - 1000 vertices, 600 - 60000 edges

On average only 7% deviation from the optimum

better than heuristics on large instances

Generalized Steiner tree problems (Hu & Wein 1995)

1000 randomly generated instances, 32 - 64 vertices

In general only 5% deviation from the optimum

Network design (Mihail, Mostrel, Dean & Shallcross 1996)

Used in a software package at Bellcore

(ITP/INPLANS CCS Network Topology Analyzer)

Is reported to do well, but details are confidential

**Jain's algorithm for network design**

Is based on simple rounding, gives a 2-approximation for general f(S)

- 9.6 Theorem (Properties of basic solutions of the LP-relaxation of the network design problem, Jain 1998)
  - Every basic feasible solution $x$ of the LP-relaxation of the general network design problem has an entry $e$ with $x_e \geq 1/2$
  - For a proof see Korte & Vygen ❑

- Jain's Algorithm
  - Input
    - instance of the general network design problem
  - Output
    - a feasible solution of the network design problem with performance guarantee 2
  - Method
    - let $Q$ be the LP-relaxation of the IP formulation of the given instance
    - repeat forever
      - compute a basic optimal solution $x$ of $Q$
      - if all $x_e$ are integer then return $x$
      - round $x_e$ to 1 for all edges $e$ with $x_e \geq 1/2$



      - modify $Q$ as follows
        - set the rounded variables $x_e$ to their new value $x_e := 1$
        - adapt the demands $f(S) := f(S) - \sum_{e \in \delta(S)} x_e$

- 9.7 Theorem (Performance guarantee of Jain's algorithm)
  - Jain's algorithm constructs a feasible solution $x$ of the general network design problem with
    $\sum_{e \in A} c_e \leq 2 \cdot OPT(I)$,
    i.e., the algorithm has a performance guarantee of 2.
  - Proof by induction on the number of iterations ❑

- Remarks on Jain's algorithm
  - The LP-relaxation has exponentially many inequalities. It can be solved in polynomial time since
    - separation and optimization are polynomially equivalent (Theorem 7.26)
    - the separation problem for $\sum_{e \in \delta(S)} x_e \geq f(S)$ can be solved in polynomial time by a sequence of min cut problems
  - The algorithm is not so useful in practice since it must solve a sequence of LPs.

- It is open if a "practical" 2-approximation algorithm exists.

## 10. Complexity of linear optimization and interior point methods 56

Most important statements of this chapter (all only with proof sketches)

- Linear programming (LP) is in NP ∩ coNP. Therefore it was conjectured that there is a polynomial algorithm for LP.
- All known variants of the simplex algorithm show an exponential worst-case runtime. However, the average case analysis and the smoothed analysis show a polynomial runtime.
- The ellipsoid method is the historically first method with a polynomial worst-case runtime for LP (Khachiyan 1979). It is, however, not practically relevant.
- Interior points methods have been developed shortly after the ellipsoid method (first by Karmarkar 1984). They also have a polynomial worst-case runtime. Today's variants (log barrier, primal-dual) are competitive with the simplex algorithm and even superior for very large and sparse problems. But they rather unsuited for solving a sequence of related optimization problems (which is important for many algorithms for integer optimization like branch and bound or cutting-plane-methods).

**Encoding length of an LP**

Let the LP be given by

$$\min\ c^T x$$



$$\text{s.t.}\quad Ax = b$$

$$x \geq 0$$

with rational data $A, b, c$

The encoding length (size) of LP w.r.t. to the standard encoding (see ADM I) is

$$\langle LP \rangle = \langle A \rangle + \langle b \rangle + \langle c \rangle$$

Another definition used with interior points methods is

$$L := \langle det_{max} \rangle + \langle b_{max} \rangle + \langle c_{max} \rangle + m + n$$

with

$$det_{max} := \max \{ |\det A'| : A' \text{ is a quadratic submatrix of } A \}$$

$$b_{max} := \max_i |b_i|$$

$$c_{max} := \max_j |c_j|$$

10.1 Lemma (Encoding length of LP)

$L \leq \langle LP \rangle$

The proof is based on

$|\det A|$ = volume of the parallelepiped generated by the columns of $A$

$\Rightarrow\ |\det A| \le \prod_j \|A_j\|$ ❑

**10.2 Lemma (Entries of basic solutions can be represented with $L$ bits)**

Let $x$ be a basic solution of LP in simplified form (gcd of at least one nominator and denominator is 1)

$$x_B^T = \left( \frac{p_{B(1)}}{q}, \ldots, \frac{p_{B(m)}}{q} \right)$$

Then $0 \le p_i < 2^L$ and $1 \le q < 2^L$

The proof is similar to that of Lemma 3.4 ❑

**10.3 Lemma (The objective values of two basic solutions differ sufficiently)**

Let $x, y$ be basic feasible solutions of LP with $c^T x \ne c^T y$.

Then $|c^T x - c^T y| > 1/2^{2L}$

Proof

Let $p$ be the least common multiple of the denominators of $x$, $q$ that of $y$



$$\Rightarrow\ |c^T x - c^T y| = \left| \frac{pc^T x}{p} - \frac{qc^T y}{q} \right| = \left| \frac{pq(c^T x - c^T y)}{pq} \right|$$

$\ge 1/pq$ as $pq(c^T x - c^T y) \ne 0$ and integer

$> 1/(2^L 2^L)$ because of Lemma 10.2 ❑

**10.4 Corollary (It suffices to compute objective values up to an additive error of $1/2^{2L}$)**

Let $z := \min \{ c^T x \mid x \in P \}$ with $P = \{ x \in \mathbb{R}^n \mid Ax = b, x \ge 0 \}$

Let $x \in P$ with $c^T x \le z + 1/2^{2L}$

Then any basic feasible solution $x^*$ with $c^T x^* \le c^T x$ is optimal

Proof

Suppose that $y$ is a basic optimal solution and that $x^*$ is not optimal.

Lemma 10.3 $\Rightarrow |c^T x^* - c^T y| > 1/2^{2L}$

$\Rightarrow c^T x^* > c^T y + 1/2^{2L} = z + 1/2^{2L} \ge c^T x \ge c^T x^*$, a contradiction ❑

**LP ∈ NP ∩ coNP**

To this end we must formulate LP as decision problem:

Input: LP and a rational number $\lambda$

Question: Is $\min \{ c^T x \mid Ax = b, x \geq 0 \} \leq \lambda$ ?

**10.5 Theorem**

$LP \in NP \cap coNP$

Proof

$LP \in NP$

We must provide a certificate of polynomial length for $\min \{ c^T x \mid Ax = b, x \geq 0 \} \leq \lambda$

Case 1: LP has an optimal solution

=> LP has a basic feasible solution $x'$ with $c^T x' \leq \lambda$

Lemma 10.2 => the entries of $x'$ are polynomial in $L$

$Ax' = b, x' \geq 0$ and $c^T x' \leq \lambda$ can be checked in polynomial time (in L)

=> $x'$ is such a certificate

Case 2: LP has a feasible solution but the objective function is not bounded from below

=> the dual program (D) $\max \{ y^T b \mid y^T A \leq c^T, y \text{ unconstrained} \}$ has no feasible solution

Farkas' Lemma for (D) => there is $x^* \geq 0$ with $Ax^* = 0$, $c^T x^* = -1$



Take as certificate

a basic feasible solution of LP to show feasibility

a basic feasible solution of $\{ Ax = b, x \geq 0, c^T x = -1 \}$ to show unboundedness of the primal objective

Both basic solutions are polynomial in $L$ because of Lemma 10.2

Case 3: LP has no feasible solution

then the instance is not a "yes"-instance => no certificate is required

$LP \in coNP$

We must provide a certificate of polynomial length for $\min \{ c^T x \mid Ax = b, x \geq 0 \} > \lambda$

Case 1: LP has an optimal solution

duality theorem => inequality is equivalent to $\max \{ y^T b \mid y^T A \leq c^T, y \text{ unconstrained} \} > \lambda$

this can be certified as in Case 1 above by a basic feasible solution of $\{ y^T A \leq c^T, y \text{ unconstrained} \}$ with value $> \lambda$

Case 2: LP has a feasible solution but the objective function is not bounded from below

then the instance is not a "no"-instance => no certificate is required

Case 3: LP has no feasible solution

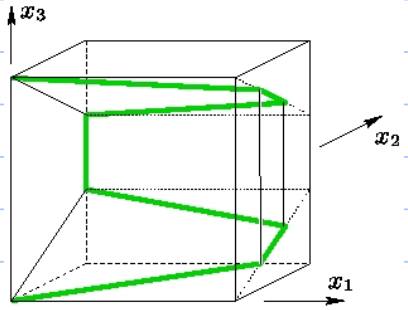
then the instance is a "yes"-instance, as $\min \{ c^T x \mid Ax = b, x \geq 0 \} = \infty$

Farkas' Lemma => there is $y \geq 0$ with $y^T A = 0$, $y^T b = -1$

=> take as certificate a basic feasible solution of $\{ y \geq 0$ with $y^T A = 0$, $y^T b = -1 \}$ ❑



- **Worst-case runtime of the simplex algorithm**
  - The worst-case runtime of the simplex algorithm is exponential
  - The counterexamples are so-called Klee-Minty cubes, i.e., slightly distorted cubes on which the simplex algorithm traverses all vertices, although it could get to the optimal solution with one pivot.




@ http://www.mathematik.de/ger/information/forschungsprojekte/zieglergeometrie/zieglergeometrie.html

- **Average runtime of the simplex algorithm**
  - First results were obtained by Borgwardt 1982
    - Variant of the simplex algorithm: "Schattenecken" algorithm

- The pivot rule is based on a 2-dimensional projection of the polyhedron
- Probabilistic assumptions
  - b = **1** (w.o.l.g.)
  - c and the rows of A are independent, identically distributed random vectors whose distribution is invariant under rotations around the origin
  - The expected number of pivot operations is $O(n^4 m)$

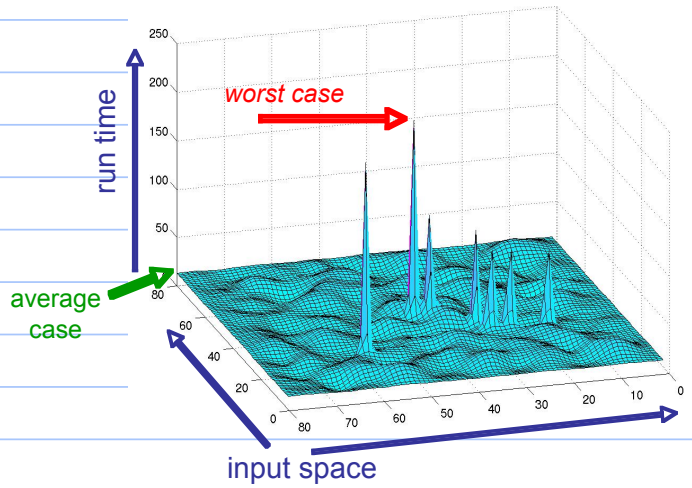
- Improvement by Haimovich 1983
  - $O(n+m)$ with the same algorithm and the same probabilistic assumptions

- Caveats for these results
  - The statement holds only for the arithmetic complexity model with $O(1)$ per operation
  - No statement for the standard simplex algorithm
  - The instances generated from the probabilistic assumptions are not sparse LPs
  - For fixed n, the probability that an instance has a feasible solution rapidly tends to 0 with increasing m
- **Smoothed analysis of the simplex algorithm**

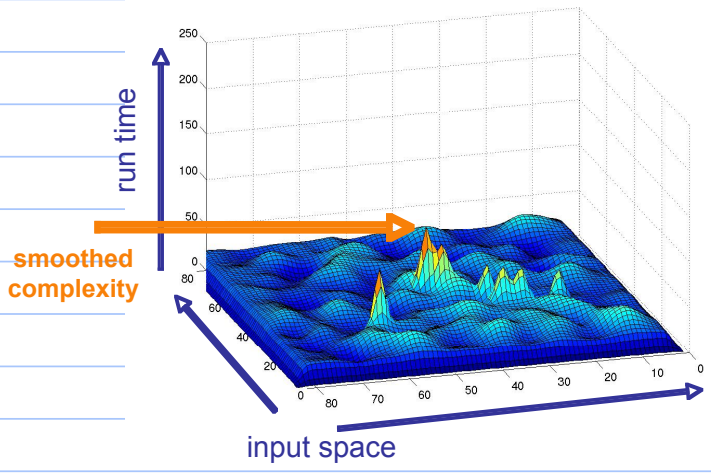


- New complexity model introduced by Spielman & Teng 2002 and first applied to LP (95 page paper)
  - Consider for any instance I a neighborhood N(I) with a probability distribution on N(I), and compute
    $\sup \{ \mathbb{E}_{I^* \in N(I)}[\text{runtime}(I^*)] \mid \text{all instances } I \}$
  - Special cases:
    - Worst case analysis: N(I) = {I}
    - Average case: N(I) = set of all instances with the same dimensions n and m

  Smoothed analysis "interpolates" between these extremes








Results of Spielman & Teng

- Variant of the simplex algorithm: two-stage Schattenecken algorithm
- Instances are normally distributed, i.e., the values of $A$ and $b$, that are $\neq 0$, are "perturbed" by a normal distribution $N(0,\sigma)$
- The neighborhood $N(I)$ is given by the standard deviation $\sigma$ of the normal distribution

- The runtime in the smoothed analysis is polynomial in $n$, $m$, and $1/\sigma$

- Caveats
  - no statement for the standard simplex algorithm
  - the model preserves sparseness, but not degeneracy



- The geometric intuition is simple, the technical details (and the proof of polynomial runtime) are difficult.

  We will illustrate here only the geometric intuition.

- **Reducing linear optimization to finding a feasible point**
  - One possibility: binary search w.r.t. the optimum $d$ with inequality $cx \leq d$
  - Another possibility: use duality
    - Use simultaneously the side constraints of the primal and the dual

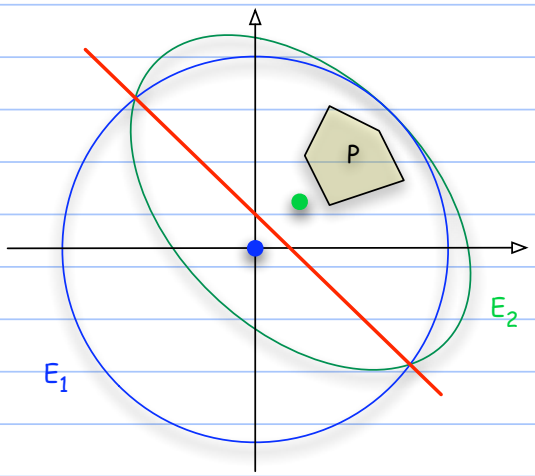
      and the constraint $c^T x \leq b^T y$

      => the only feasible points are $(x,y)^T$ with $x$ optimal in the primal, $y$ optimal in the dual

- **The ellipsoid method computes a feasible point in a polytope $P$**
  - start with a ball $E$ around the origin containing $P$
  - while the volume of $E$ is not too small do // there is still a point in $E \cap P$
    - if the center $x$ of $E$ is in $P$ then return $x$
    - compute a hyperplane that separates $x$ from $P$, let $H$ be the halfspace containing $P$
    - compute the new ellipsoid $E$ with smallest volume containing $H \cap E$ and the intersection points of the

hyperplane with the boundary of the previous ellipsoid

return "there is no feasible solution"




Remarks on the ellipsoid method

there is an analytic formula for the new ellipsoid (efficient update)



the volume per iteration shrinks by the factor $\exp(-1/2n) < 1$

the volume check is done as follows for full dimensional polytopes $P$

translate the inequalities of $P$ by $1/2^{2L+1}$, such that $P$ contains a ball with radius $r = 1/2^{2L}$

=> one can stop if the volume of E is below that of $P$

one need additional techniques for lower dimensional polytopes

the algorithm requires $O(n^2L)$ iterations with $O(n^4L)$ arithmetic operations with numbers of $O(L)$ bits

more information about the ellipsoid method

M. Grötschel, L. Lovász, and A. Schrijver

Geometric Algorithms and Combinatorial Optimization

Springer, Berlin, 2nd ed., 1993

**Goal of this section**

A sketch of the interior point algorithm by Ye with improvements by Freund (both published in Mathematical Programming 1991)

**Starting point and general idea**

given are a primal LP and the associated dual in the form

(P) min $z = c^T x$

s.t. $Ax = b,\ x \geq 0$

(D) max $w = b^T y$

s.t. $A^T y + s = c,\ s \geq 0$ (slack variables), y unrestricted

The algorithm solves (P) and (D) simultaneously

It computes in each phase a primal solution $x^* > 0$ and dual slack variables $s^* > 0$

Basic idea:

Stay away from the boundary $x_j = 0$, $s_j = 0$, i.e. ensure $x_j > 0$, $s_j > 0$

but make the duality gap $c^T x^* - b^T y^* = (A^T y^* + s)^T x^* - (Ax^*)^T y^* = x^{*T} s^* > 0$ small



**The two main ingredients**

Ingredient 1: scaling

Let $x^* > 0$ and $s^* > 0$ be given

Scaling is a function $\mathbb{R}^n \to \mathbb{R}^n$ with

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \to x' = \begin{pmatrix} \frac{x_1}{x_1^*} \\ \vdots \\ \frac{x_n}{x_n^*} \end{pmatrix}$$

Observe:

$$x^* \to \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

Scaling in matrix form:

$$x' = (X^*)^{-1} x \quad \text{with } X^* = \begin{pmatrix} x_1^* & \dots & 0 \\ & \ddots & \\ 0 & \dots & x_n^* \end{pmatrix}$$

With scaling we can write (P) in the transformed space as

(P) min $z = c^T X^* x'$

s.t. $AX^* x' = b,\ x' \geq 0$

Set $c^* := X^* c,\ A^* := X^* A$

=> (P) can be written as

(P) min $z = c^{*T} x'$

s.t. $A^* x' = b,\ x' \geq 0$

Similarly, (D) is written in the transformed space as

(D) max $w = b^T y$

s.t. $A^{*T} y + s' = c^*,\ s' \geq 0$ with

$$s' := X^* \cdot s = \begin{pmatrix} s_1 x_1^* \\ \vdots \\ s_n x_n^* \end{pmatrix}$$

Observe: $x_j s_j = x_j' s_j'$ => the duality gap is invariant under scaling

=> one can do computations in the transformed space



Ingredient 2: potential function

It measures the size of the duality gap. It is a logarithmic barrier function

$G(x,s) := q \cdot \ln(x^T s) - \sum_j \ln(x_j \cdot s_j)$

with a suitable parameter $q > 0$

Observe

$q \cdot \ln(x^T s) \to -\infty$ if the gap $x^T s \to 0$

$-\sum_j \ln(x_j \cdot s_j) \to +\infty$ if $x_j \to 0$ or $s_j \to 0$, i.e., close to the boundary

Question: how to choose $q$ ?

A good choice of $q$ is

$q := n + \sqrt{n}$

This choice leads to $O(\sqrt{n} \cdot L)$ iterations

where L := encoding length of Section 10.1

**Stopping criterion**

The potential function leads to a stopping criterion that is based on Lemma 10.2:

Let x, s be primal-dual feasible with $G(x,s) \leq -k\sqrt{n}L$ for a constant k

Then $x^T s \leq e^{-kL}$

So: stop when

$G(x,s) \leq -k\sqrt{n}L$

with k = 2

Observe: Scaling does not change the value of G(x,s)

=> one may do calculations in the original or in the transformed space

**Ye's algorithm**

Structure of Ye's algorithm

Input

Primal-dual pair of the form

(P) min $z = c^T x$

s.t. $Ax = b,\ x \geq 0$

(D) max $w = b^T y$



s.t. $A^T y + s = c,\ s \geq 0$ (slack variables), y unrestricted

Output

Primal dual pair (x,s) with $G(x,s) \leq -k\sqrt{n}L$

Method

Initialization

i := 0 // counter

choose $x^0, y^0$ primal-dual feasible with $G(x^0,s^0) = O(\sqrt{n}L)$

// idea: modify phase I of the simplex algorithm such that $x \sim 2^L$, $s \sim 2^L$

Iteration

while $G(x^i,s^i) > -2\sqrt{n}L$ do

do a primal step // change only $x^i$

or a dual step // change $s^i$

this gives $(x^{i+1}, s^{i+1})$

i := i+1

Details of the iteration

- Overview
  - Scale the current pair $(x^i, s^i) \rightarrow (e, s')$ with $e = \mathbf{1}$

    => $(e, s')$ is far from the boundary

    the primal or dual step then computes $(\tilde{x}, \tilde{s})$ and reduces $G$

    the re-transformation of $(\tilde{x}, \tilde{s})$ into the original space gives $(x^{i+1}, s^{i+1})$

- Main property of the primal/dual step
  - do it in such a way that $G(x^{i+1}, s^{i+1}) - G(x^i, s^i) \leq -7/120 < 0$

    $\Rightarrow G(x^N, s^N) < -2\sqrt{n}L$ after N steps with

    $$\underbrace{k\sqrt{n}L}_{\approx G(x^0,s^0)} - N\frac{7}{120} \leq 2\sqrt{n}L$$

    $$\Rightarrow N \geq \frac{120}{7}(k+2)\sqrt{n}L = O(\sqrt{n}L)$$

- Computing $(\tilde{x}, \tilde{s})$
  - Compute the x-gradient of $G$ in the point $(e, s')$:



$$g := \nabla_x G(x,s)|_{(e,s')} = \frac{q}{x^T s}s - \left.\begin{pmatrix} \frac{1}{x_1} \\ \vdots \\ \frac{1}{x_n} \end{pmatrix}\right|_{(e,s')} = \frac{q}{x^T s'}s' - e$$

Go into the direction $-g$ to decrease $G$, but stay feasible (i.e., $A^* \cdot \tilde{x} = b$).

- To this end, let $d$ be the projection of $g$ onto the subspace $\{ x \mid A^*x = 0 \}$

  => $d = (I - A^*(A^*A^{*T})^{-1}A^*)g$ (without proof)

  Go into direction $-d$

- A possible problem: $||d||$ is too small

  => the primal step does not decrease $G$ enough

  Therefore: make a primal step if $||d|| \geq 0.4$

  a dual step if $||d|| < 0.4$

- Primal step
  - $$\tilde{x} := e - \frac{1}{4\|d\|}d, \quad \tilde{s} := s$$

- After the primal step, $\tilde{x} > 0$ and $G(\tilde{x}, \tilde{s}) - G(e, s') \leq -7/120$

- Dual step
  - compute the s-gradient of $G$ in the point $(e, s')$:

$$h := \nabla_s G(x,s)|_{(e,s')} = \frac{q}{x^T s'} e - \begin{pmatrix} \frac{1}{s'_1} \\ \vdots \\ \frac{1}{s'_n} \end{pmatrix}$$

  - $\Rightarrow h_j = g_j / s_j$ $\Rightarrow$ $h$ and $g$ show approximately into the same direction
  - go into direction $-(g-d)$ and set

    $\tilde{s} := s' - (g-d)\mu$ with $\mu := e^T s' / q$
  - After the dual step,

$$\tilde{s} = \frac{e^T s'}{q}(d+e), \quad \tilde{x} = x' = e$$

    $\tilde{s} > 0$

    $G(\tilde{x}, \tilde{s}) - G(e, s') \leq -1/6$



- **Analysis of Ye's algorithm**
  - Advance per iteration
    - $G$ decreases in every primal and dual step by a constant amount

    $\Rightarrow$ $O(\sqrt{n}L)$ iterations until $G(x^i, s^i) < -2\sqrt{n}L$
  - Runtime
    - Each iteration can be done with $O(n^3)$ operations
      - the only expensive calculation is that of the projected gradient $d$

        = solution of the linear system $(A^*A^{*T})w = A^*g$

        can be done by Gaussian elimination in $O(n^3)$ operations
    - => $O(n^{3.5}L)$ operations in total

  - Problem: operations cannot be made exactly
    - $||d||$ may be irrational
      - => compute only with fixed number of $L$ bits and round
      - => 19/352 instead of 7/120 by rounding when calculating $||d||$

- The numbers the Gaussian elimination must not become too large
  - Use $\langle \det B \rangle \leq \langle A \rangle$ for every quadratic submatrix of $A$
    => (Cramers' Rule) all numbers in Gaussian elimination can be represented with $L$ bits

10.6 Theorem (Polynomial runtime of Ye's algorithm)

Ye's algorithm runs in $O(n^{3.5}L)$ time.

without proof ❑