CS124 | lecture 17 | 1

LP

ellipsoid method

simplex method

standard format (depends on LP solver)

- minimization
- non-negative
- equality constraints

How to turn a problem into this form!

Format change for specific solvers.

max → min

$$\max\ x_1 + 2x_2 + 3x_3 \iff \min\ -x_1 - 2x_2 - 3x_3$$

inequality → equality

$x_1 + x_2 \le 5$

add a slack variable

$x_1 + x_2 + S = 5$

$S \ge 0$

$x_1 + x_2 \ge 5$

$\Rightarrow x_1 + x_2 - S = 5$

$S \ge 0$

negative → non-negative

allow x to take negative values

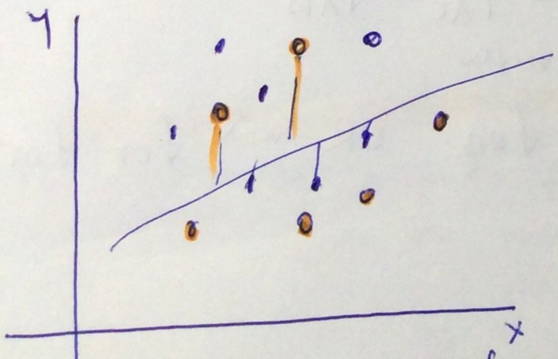
substitute with $x_1 - x_2$

$x_1, x_2 \ge 0$

Reduction

- reduce problems to linear programs
- 1st step, think what are the variables

Linear separator problem.



min sum of errors

- 2 sets of points

white (x_i, y_i)

$i = 1 \dots m$

blue (x_i, y_i)

$i = m+1 \dots m+n$

$ax + by = c$ separator

e_i = "error" ith point $\{e_i \geq 0\}$

$\min \sum_i e_i$

$i = m+1 \ldots m+n$ (blue error) ← below the line

$e_i \geq c - \underbrace{a x_i - b y_i}_{\text{where it is}}$ ← if falls above the line e_i is constrained to be positive

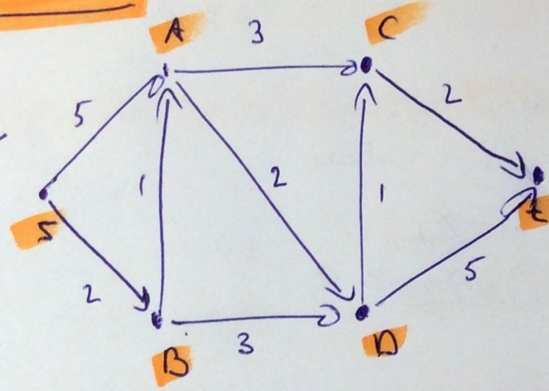
↑ where it should be on the line

$i = 1 \ldots m$ (white error), above the line

$e_i \geq a x_i + b y_i - c$ if it falls below the line, e_i is constrained to be positive

Network Flows

linear program



e.g. road capacities how many trucks can be sent

here max flow is 6

c_e : cap on each edge

f_e : flow on each edge

$f_e \leq c_e$

$f_e, c_e \geq 0$

↑ don't violate capacity constraint

← conservation of flow constraint

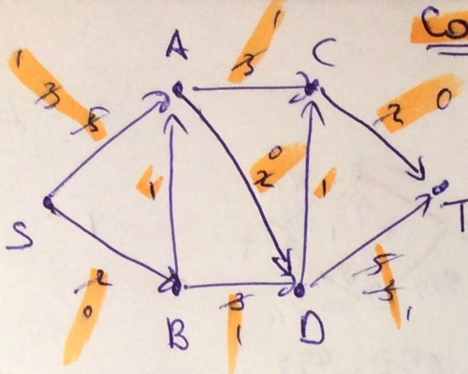
flow in = flow out [redacted]

$f_{SA} + f_{BA} = f_{AC} + f_{AD}$

one eqn. per vertex

$\max f_{SA} + f_{SB}$ OR $\max f_{CT} + f_{DT}$

Combinatorial algorithm



① Find a path from s to t

e.g. S–A–C–T

② then add flow

e.g. $f_{SA} = 2$

$f_{AC} = 2$

$f_{CT} = 2$

③ residual capacity update

$c_e - f_e$

SACT → 2

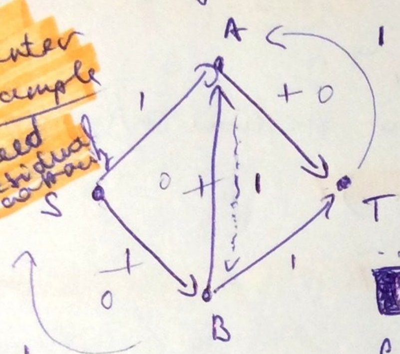
SBDT → 2

SADT → 2

6

Counter example

need residual capacity



residual capacity

SBAT → 1

But max flow should be 2

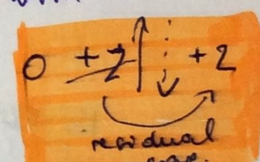
[redacted]

$f_{AB} = -1$

c_{AB} was 0

now 1

To be able to reverse decisions, add edges in opposite direction with the same capacity that was used up in the opposite direction



After running with residual capacity edges

final flows are: $f_{SA} = 1$ $f_{AT} = 1$

$f_{SB} = 1$ $f_{BT} = 1$

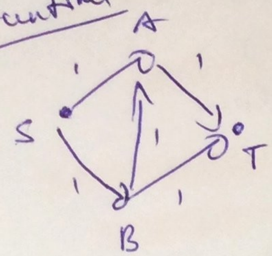
Fix combinatorial algorithm:

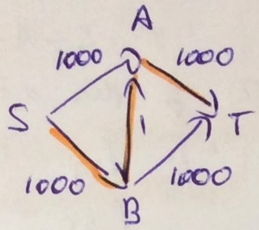
Find a path from s to t on the residual graph with backward edges

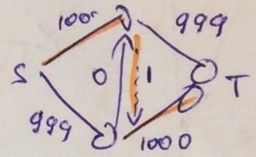
Increment flow

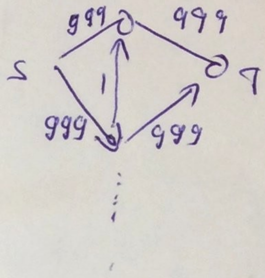
correctness → discussed later

runtime









Total flow

1

2

⋮

2000

Augmenting path:
path increasing flow

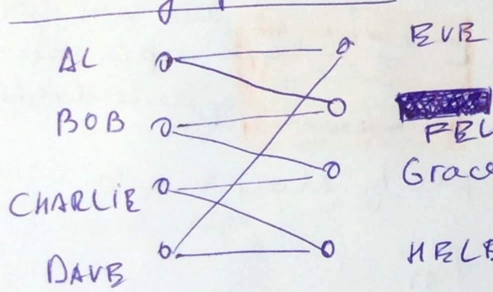
[redacted] finding a path → $O(E)$
DFS, BFS

But # of searches in worst case
could be the size of max flow (assume integers)

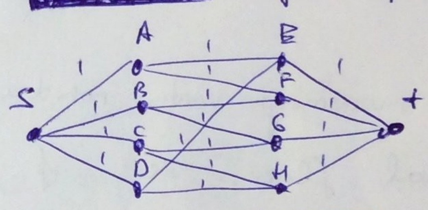
⇒ $O(E \cdot \text{max flow})$

BFS - Edmonds-Karp $O(E^2 V)$

Matching problem



Maximum matching
[redacted] Reduce to a flow problem



int flow ⇒ matching
matching ⇒ int flow

matching ⇒ int flow

any matching is
a disjoint set of
edges in the middle
⇒ flow of the same size

X with LP, may not get integer flow through edges

V with combinatorial, whenever an augmenting path is found, integer flow is sent.

int flow ⇒ matching

because all capacities are 1
a mid-vertex on the left, sends flow
[redacted] to at most one mid-vertex
on the right, and vice versa ← combinatorial alg.