Other max flow algorithms

Dinic

- Edmonds-Karp algorithm:
  - Use *shortest augmenting path P*
  - # iterations $\leq ne$ $\Rightarrow$ complexity $O(ne^2)$

VE w/ good data structures $O(VE \lg V)$

- Even faster Algorithms:
  - …
  - $O\left(e^{\frac{10}{7}} U^{\frac{1}{7}} \cdot (\log nU)^{O(1)}\right)$ if max capacity U [Madry'16]

send flow on the path w/ max residual capacity $O(VE \lg U)$

Flows: Extensions, Generalizations

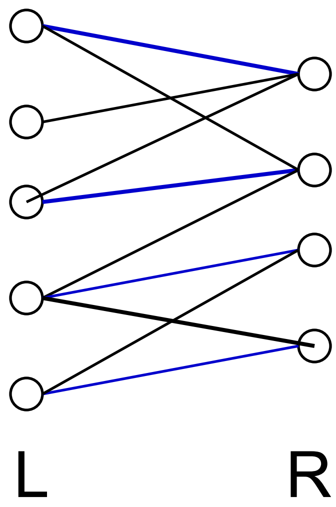
Many variants, extensions can be reduced to the basic max flow and min cut problems

- Undirected Graphs
- Node capacities
- Multiple sources and sinks
- Lower bounds on edge flows

Other Problems

- Graph connectivity problems
- Maximum Bipartite matching
- Minimum bipartite node cover
- ……

Problem: Maximum Bipartite Matching



Bipartite undirected graph

$N = L \cup R$

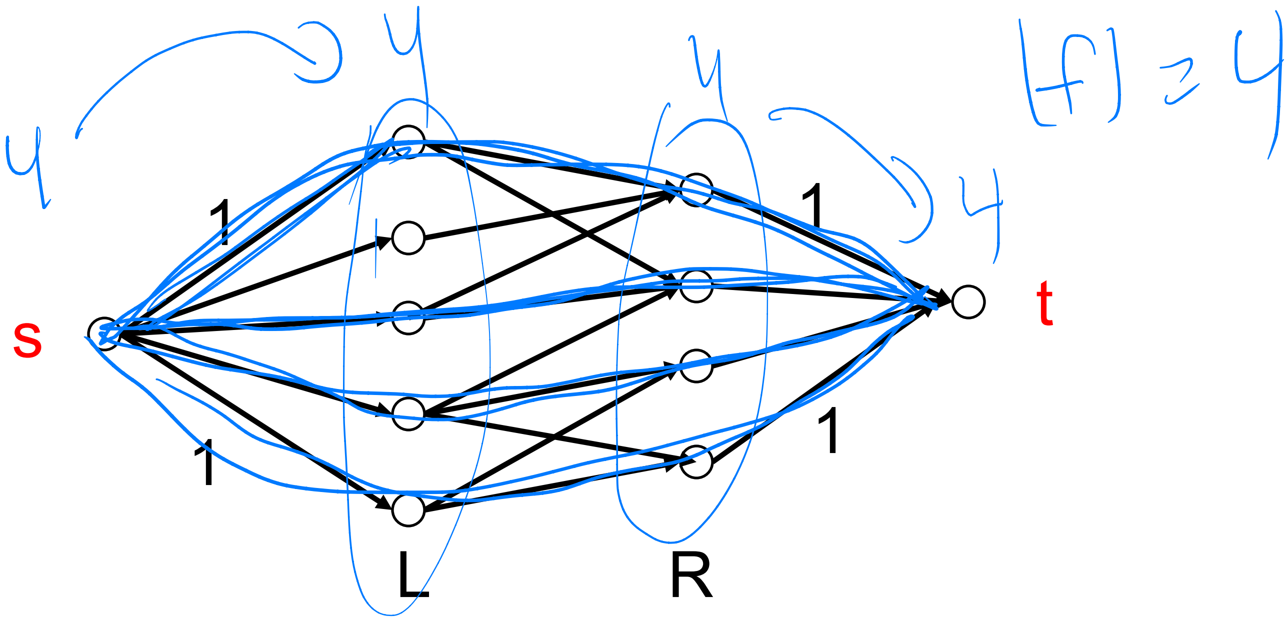
$E \subseteq L \times R$

Matching: Set of disjoint edges (i.e. no common nodes)

(= pairing of some L nodes with distinct R nodes)

Maximum Matching Problem: Find matching with maximum cardinality

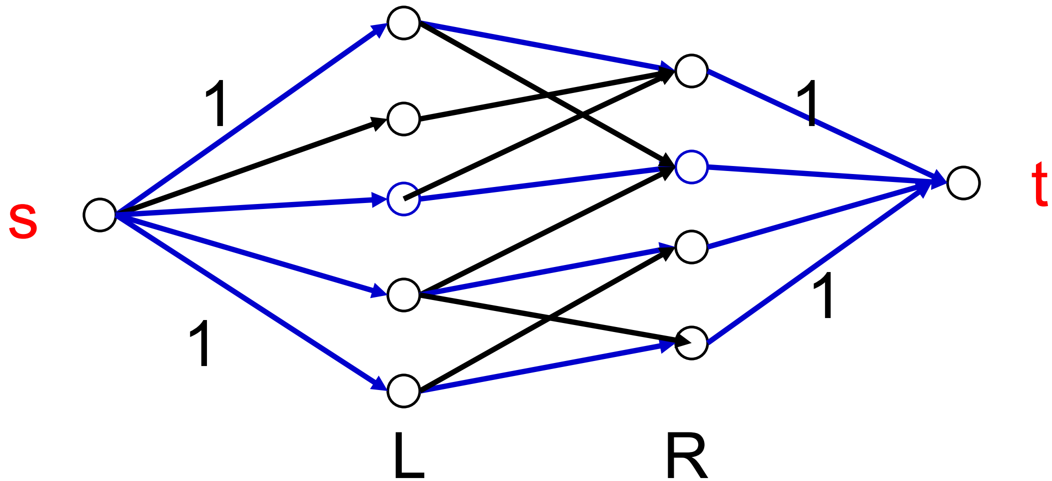
Reduction to Max-Flow problem



All left and right edges have capacity 1

Middle edges can have any capacity ≥1 (eg, 1,2,…, ∝)

Reduction to max flow problem



- Integer capacities $\Rightarrow$ Integer max flow
- Integer-valued flows = 0 -1 valued flows $\leftrightarrow$ matchings
- Max integer flow = max flow $\leftrightarrow$ maximum matching

Complexity of Ford-Fulkerson: $O(ne)$ $O(VE)$

Hopcroft-Karp: $O(\sqrt{n}\, e)$ $O(\sqrt{V}E)$

Linear Programming

- Variables take values in real numbers
- *General Form*: Maximize or minimize a linear function (the *objective function*) subject to a set of linear constraints: linear weak inequalities and equations

Not linear

$x_1 \cdot x_2 \le 6$

$$\max(\min) \sum_{j=1}^{n} c_j x_j$$

subject to

$$\begin{cases} \sum_{j=1}^{n} a_{1j} x_j = b_1 \\ \cdots \\ \sum_{j=1}^{n} a_{ij} x_j \le b_i \\ \cdots \\ \sum_{j=1}^{n} a_{kj} x_j \ge b_k \\ \cdots \end{cases}$$

$\max\ 3x_1 + 2x_2$

s.t.

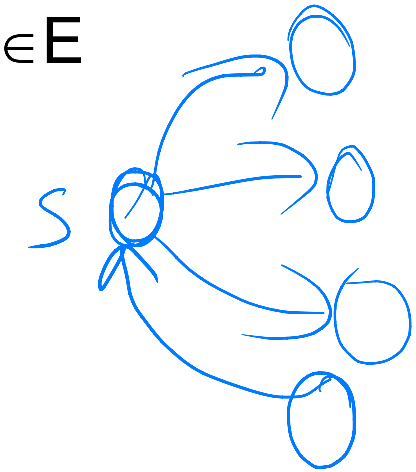
$x_1 + x_2 \le 7$

$2x_1 - x_2 \le 4$

$3x_1 + x_2 = 10$

Max s-t Flow problem as a LP

- LP with edge-flow variables $f(u,v)$, $(u,v) \in E$



- Maximize $\displaystyle\sum_{(s,v)\in E} f(s,v) - \sum_{(v,s)\in E} f(v,s)$

subject to

- $f(u,v) \geq 0, \ \forall (u,v) \in E$ (nonnegativity)
- $f(u,v) \leq c(u,v), \ \forall (u,v) \in E$ (Capacity constraints)
- Flow conservation constraints:

$$\sum_{v:(u,v)\in E} f(u,v) - \sum_{v:(v,u)\in E} f(v,u) = 0, \ \forall u \in N - \{s,t\}$$

Ex 2: Minimum Cost Flow problem

Given flow network $G = (N, E, c)$ with source s, sink t, cost $a(u, v) \geq 0$ for each edge (u, v), and flow demand d, find a flow of value d from s to t that minimizes the total cost

- LP with edge-flow variables $f(u,v)$, $(u,v) \in E$

$$\min \sum_{(u,v)\in E} a(u,v)\cdot f(u,v)$$

$$\text{s.t.} \quad \sum_{(u,v)\in E} f(u,v) - \sum_{(v,u)\in E} f(v,u) = 0, \ \forall u \in N - \{s,t\}$$

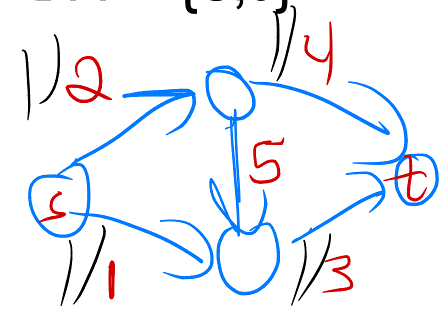
$$\sum_{(s,v)\in E} f(s,v) - \sum_{(v,s)\in E} f(v,s) = d$$

$$f(u,v) \leq c(u,v), \ \forall (u,v) \in E$$

$$f(u,v) \geq 0, \quad \forall (u,v) \in E$$

costs

cap = 1



d = 2

cost = 1·3 + 1·4 + 1·1 + 1·3 = 11

Integrality property: Integer capacities $\Rightarrow$ Integer optimal flow

Shortest s-t path = Min cost flow with all caps=1, demand=1

Ex 3: Fractional Knapsack Problem

- Problem: Given n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, and a knapsack of weight capacity B, choose quantities $x_1, \ldots, x_n$ of the items (possibly fractional) to put in the knapsack so that they all fit and have maximum total value

- LP formulation: variables $x_1, \ldots, x_n$

$$\max \sum_{i=1}^{n} v_i x_i$$

$$\text{s.t } \sum_{i=1}^{n} w_i x_i \le B$$

$$0 \le x_i \le 1, \quad i = 1, \ldots, n$$

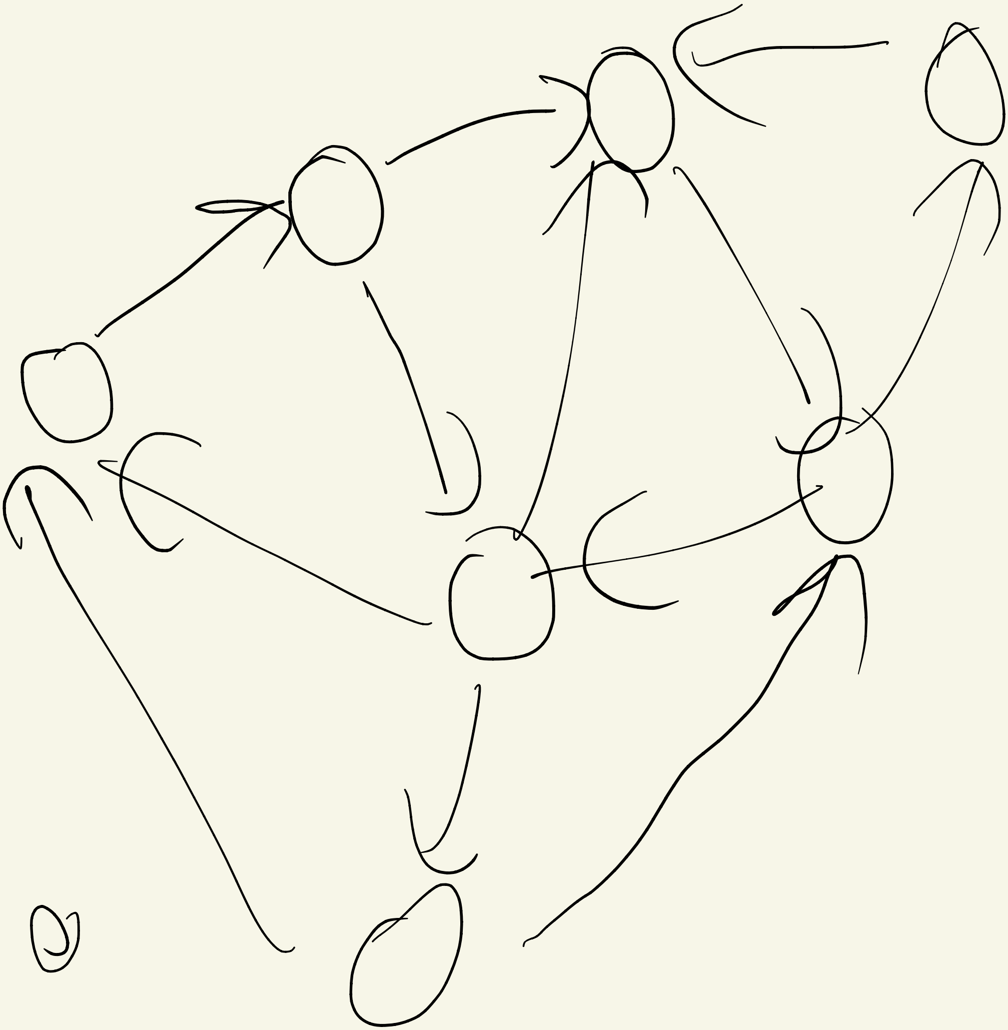
Ex 4: Multicommodity Flow problem

- Flow network $G = (N, E, c)$ with k source-sink pairs (s_i, t_i), one for each commodity $i = 1, \ldots, k$. (Some nodes may be the source or sink for multiple commodities.)
- The flows of different commodities share the edges; total flow through each edge may not exceed the capacity

Different versions:

1. *Maximization version*: Find flows for the commodities that maximize the sum of all the commodities shipped.
2. *Demands version*: Given demand d_i for each commodity, find a flow that ships d_i units from s_i to t_i

   Minimum Cost version: find minimum-cost such flow



Maximum Multicommodity Flow: ver 1

LP formulation: edge flow variables $f_i(u,v), (u,v) \in E, i = 1, \dots, k$

amount of commodity i flowing through edge (u,v)

$$\max \sum_{i=1}^{k} \Big[\sum_{(s_i,v)\in E} f_i(s_i,v) - \sum_{(v,s_i)\in E} f_i(v,s_i) \Big]$$

conservation

$$\text{s.t.} \quad \sum_{(u,v)\in E} f_i(u,v) - \sum_{(v,u)\in E} f_i(v,u) = 0, \ \forall i = 1,\dots,k, \ \forall u \in N - \{s_i, t_i\}$$

$$\sum_{i=1}^{k} f_i(u,v) \le c(u,v), \quad \forall (u,v) \in E$$

joint capacity

$$f_i(u,v) \ge 0, \quad \forall i = 1,\dots,k, \quad \forall (u,v) \in E$$

- No integrality property:

Even if capacities are integer, optimal flow may not be integer

constraints $O(\cancel{N}K + E + KE)$ constr.

N nodes

M edges

K comm.

$O(KE)$ variable

Multicommodity Flow problem: ver 2 (with demands)

2. Demands & costs: variables $f_i(u,v), (u,v) \in E, i = 1, \dots, k$

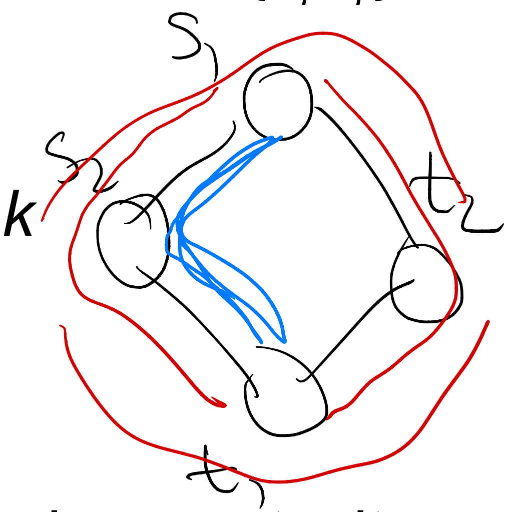
$$\min \sum_{i=1}^{k} \sum_{(u,v)\in E} a(u,v) \cdot f_i(u,v)$$

$$\text{s.t.} \quad \sum_{(u,v)\in E} f_i(u,v) - \sum_{(v,u)\in E} f_i(v,u) = 0, \ \forall i = 1,\dots,k, \ \forall u \in N - \{s_i, t_i\}$$

$$\sum_{i=1}^{k} f_i(u,v) \le c(u,v), \quad \forall (u,v) \in E$$

$$\sum_{(s_i,v)\in E} f_i(s_i,v) - \sum_{(v,s_i)\in E} f_i(v,s_i) = d_i, \quad \forall i = 1,\dots,k$$

$$f_i(u,v) \ge 0, \quad \forall i = 1,\dots,k, \quad \forall (u,v) \in E$$



- Even if capacities and demands are integer and no costs, it may be that the only way to satisfy the demands is with a fractional flow

Three Possibilities for solution of an LP

- Infeasible: Constraint set has no feasible solution

  max x_1 s.t. $x_1, x_2 \geq 0$ and $x_1 + x_2 \leq -2$

- Unbounded: No finite optimum: objective function can be made arbitrarily "good" (large for a maximization problem, small for minimization)

  max x_1 s.t. $x_1, x_2 \geq 0$ and $x_1 + x_2 \geq 7$

- Finite optimum: There is an optimal solution

  (note: the feasible solution set itself may be unbounded in some directions)

  max x_1 s.t. $x_1, x_2 \geq 0$ and $x_1 + x_2 \leq 7$

LP modeling: example

- A steel company can produce two products: bands, coils

Profit: \$25/ton for bands, \$30/ton for coils

Maximum demand/week: 6,000 for bands, 4,000 for coils

Production Rate: for bands 200 tons/hour, coils: 140 tons/hr
Week = 40 hours of production

How many tons of each to maximize profit?

Variables: x = #tons of bands per week, y = #tons of coil/ week

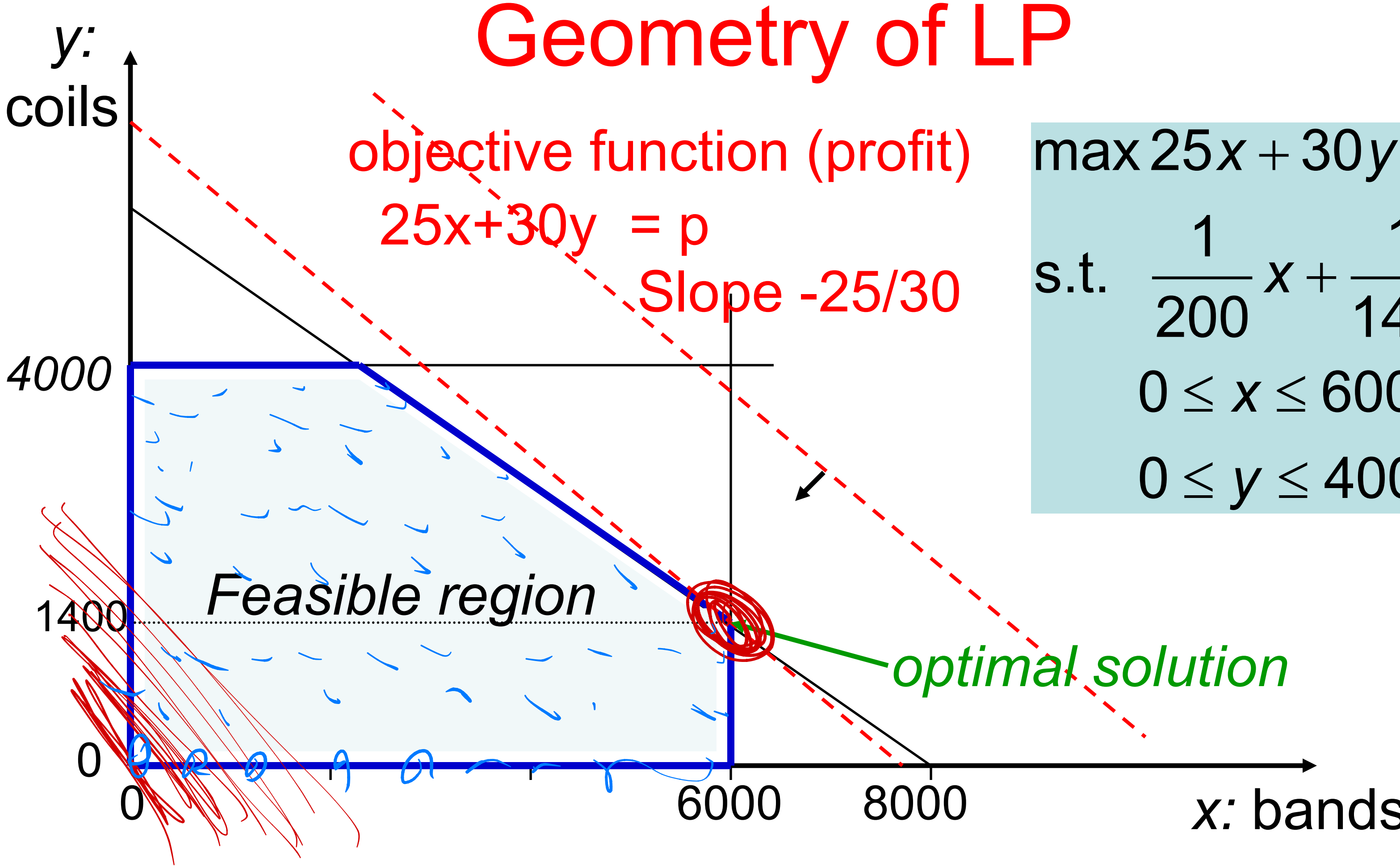
$$\max 25x + 30y$$

$$\text{s.t.} \quad \frac{1}{200}x + \frac{1}{140}y \leq 40$$

$$0 \leq x \leq 6000$$

$$0 \leq y \leq 4000$$

Geometry of LP



$$\max 25x + 30y$$

$$\text{s.t.} \quad \frac{1}{200}x + \frac{1}{140}y \le 40$$

$$0 \le x \le 6000$$

$$0 \le y \le 4000$$

Feasible region: a polyhedron

Optimal solution: a vertex

Vertices of Polyhedron

- Vertex is determined by the intersection of n (=dimension) linearly independent hyperplanes (tight constraints)

- If m constraints and n variables $\rightarrow$ at most $\binom{m}{n}$ vertices

- If all input coefficients in the constraints and the objective function are rationals p/q, where p,q are integers with w bits, then the coordinates of the vertices are also rationals p'/q' where p',q' have polynomial (in n,w) # of bits

Algorithms for Linear Programming

Simplex (Dantzig, 1947)

Method: Starts at a vertex and keeps moving to better adjacent vertex until it reaches an optimum

pivoting rule: how to choose which better adjacent vertex to move to

- In practice, works very well
- In worst case, can lead to exponential (in n,m) iterations
- OPEN if there is a pivoting rule that guarantees polynomial time (polynomial number of iterations)

Ellipsoid Algorithm (Khachian, 1979)

Worst case polynomial time (in n,m,w), but not practical

Interior Point Method (Karmakar,1984)

Worst case polynomial time (in n,m,w),

competitive in practice