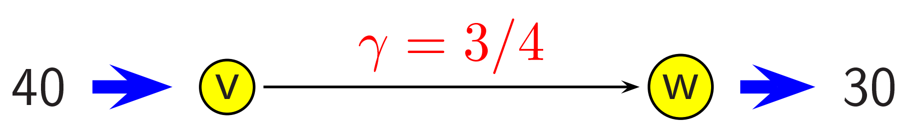
Generalized Max Flows

Kevin Wayne

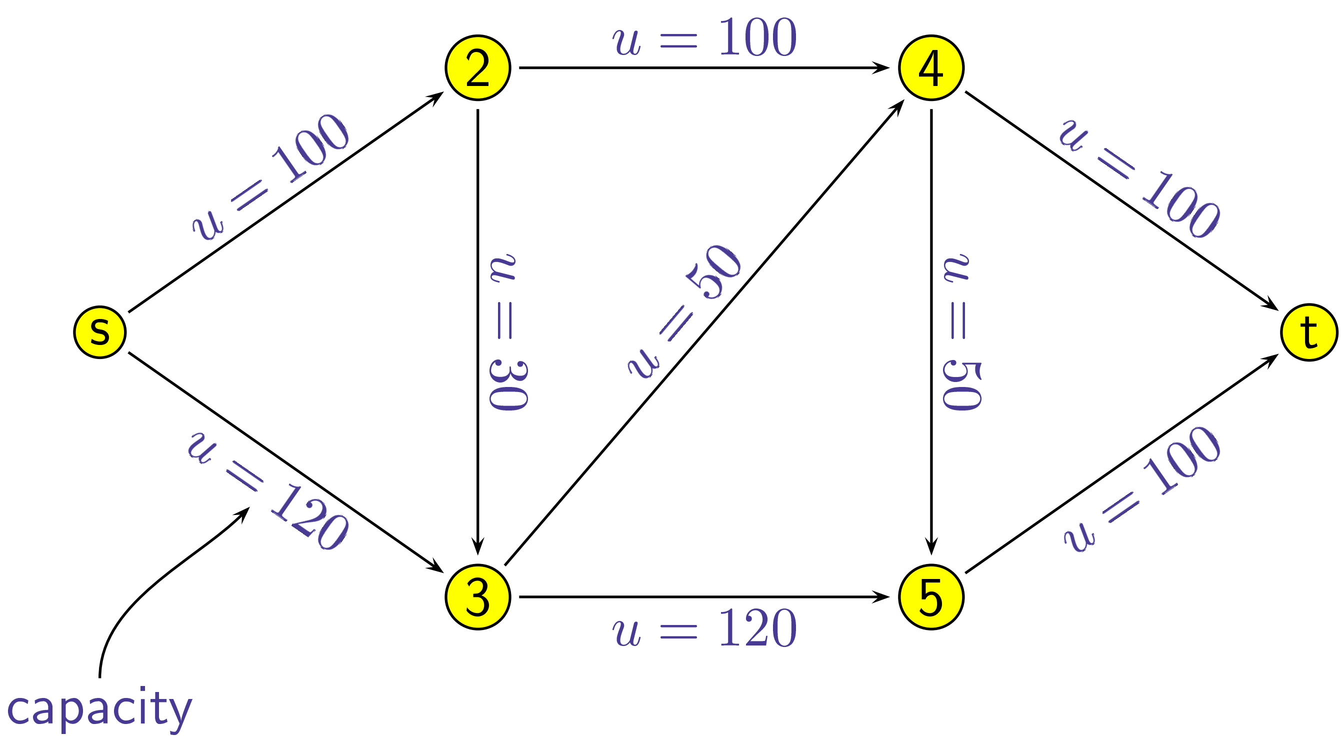
Cornell University

www.orie.cornell.edu/~wayne



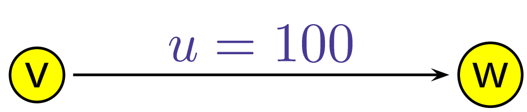
advisor: Éva Tardos

Maximum Flow Problem



Max flow sent to t

- capacity constraints
- flow conservation constraints



Generalized

$\gamma = 3/4$

$\gamma = 1/2$

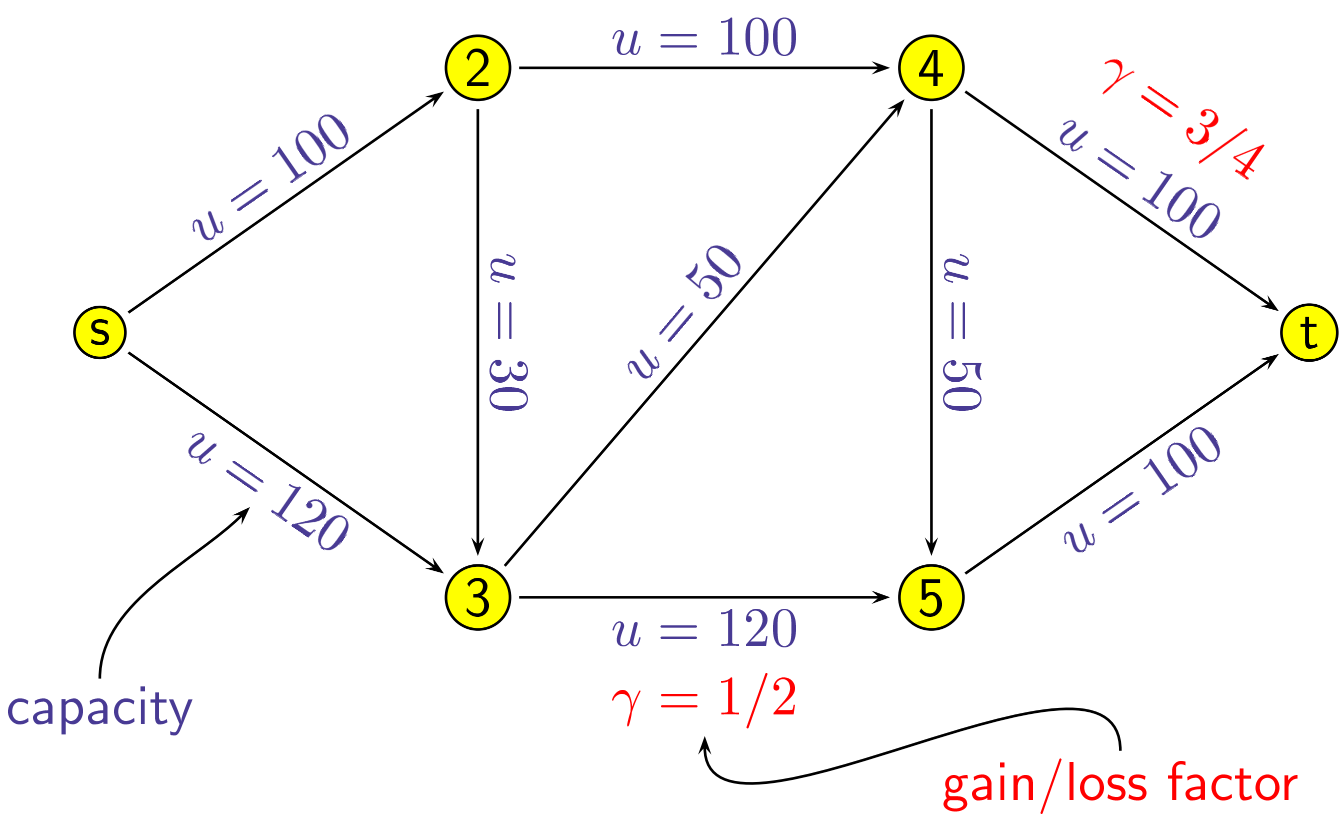
gain/loss factor

(generalized)

$\gamma = 3/4$

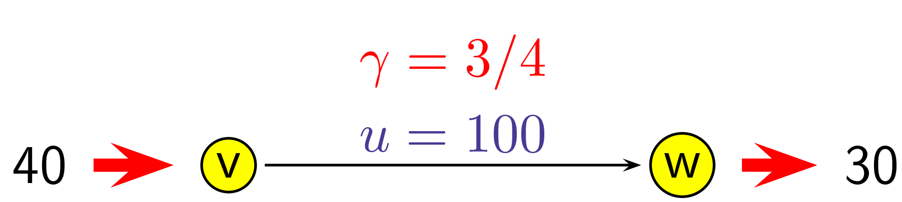
40 ➔ ➔ 30

Generalized Maximum Flow Problem



Max flow sent to t

- capacity constraints
- flow conservation constraints (generalized)



Organization of Talk

1. Applications
2. Previous work
3. Combinatorial structure and optimality conditions
4. Exponential-time augmenting path algorithm
5. Polynomial-time variant using gain-scaling

main part of talk

Applications

"generalized networks are coming to be appreciated as rivaling or even surpassing pure networks in their practical significance."

- *Glover and Klingman*

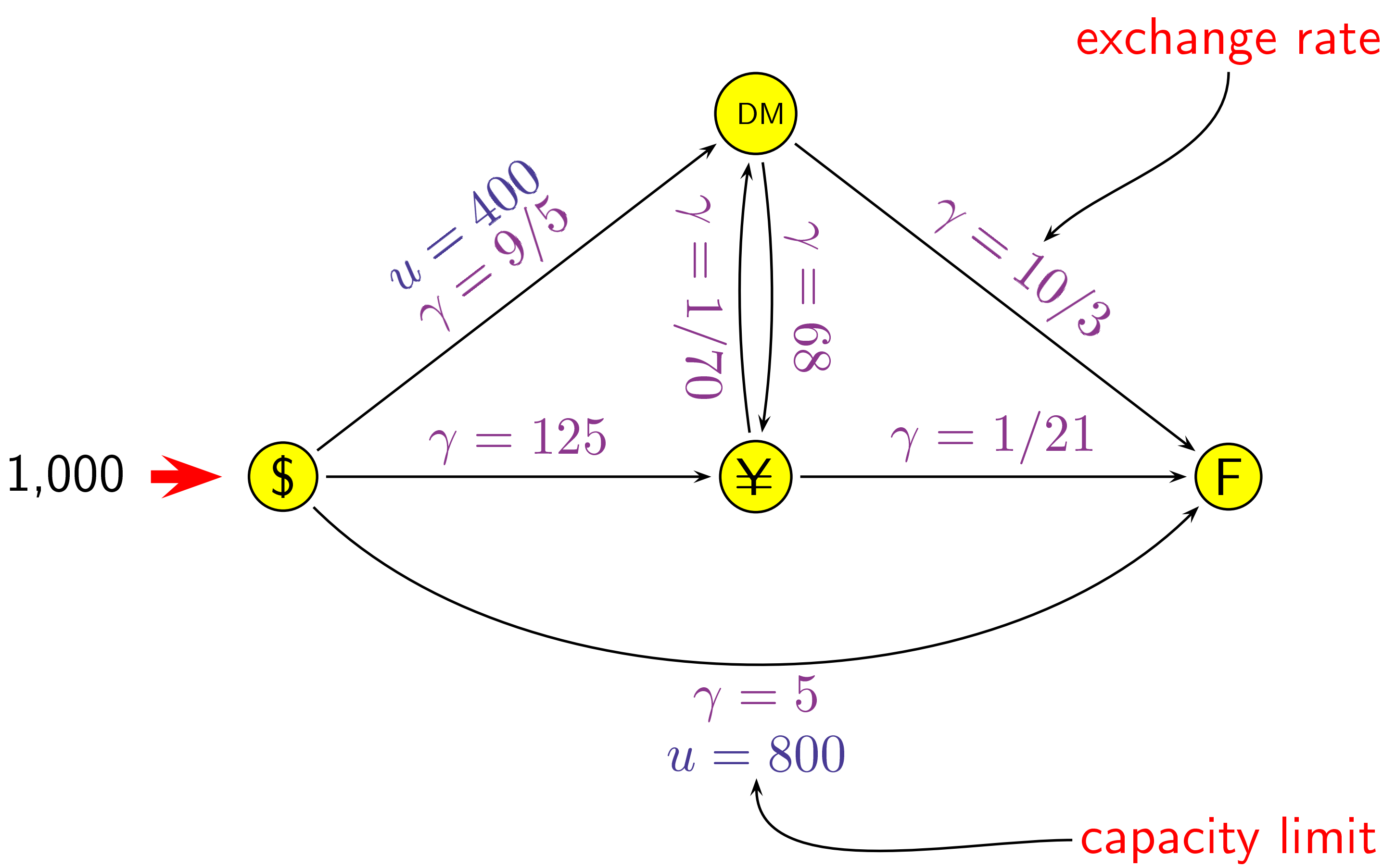
Physical transformations:
leaky pipes, theft, evaporation, attrition, spoilage, taxes, interest

Administrative transformations:
currency conversion, production yields, energy blending, machine scheduling

Optimal Currency Conversion

Convert $1,000 to maximum number of French Francs through sequence of currency conversions

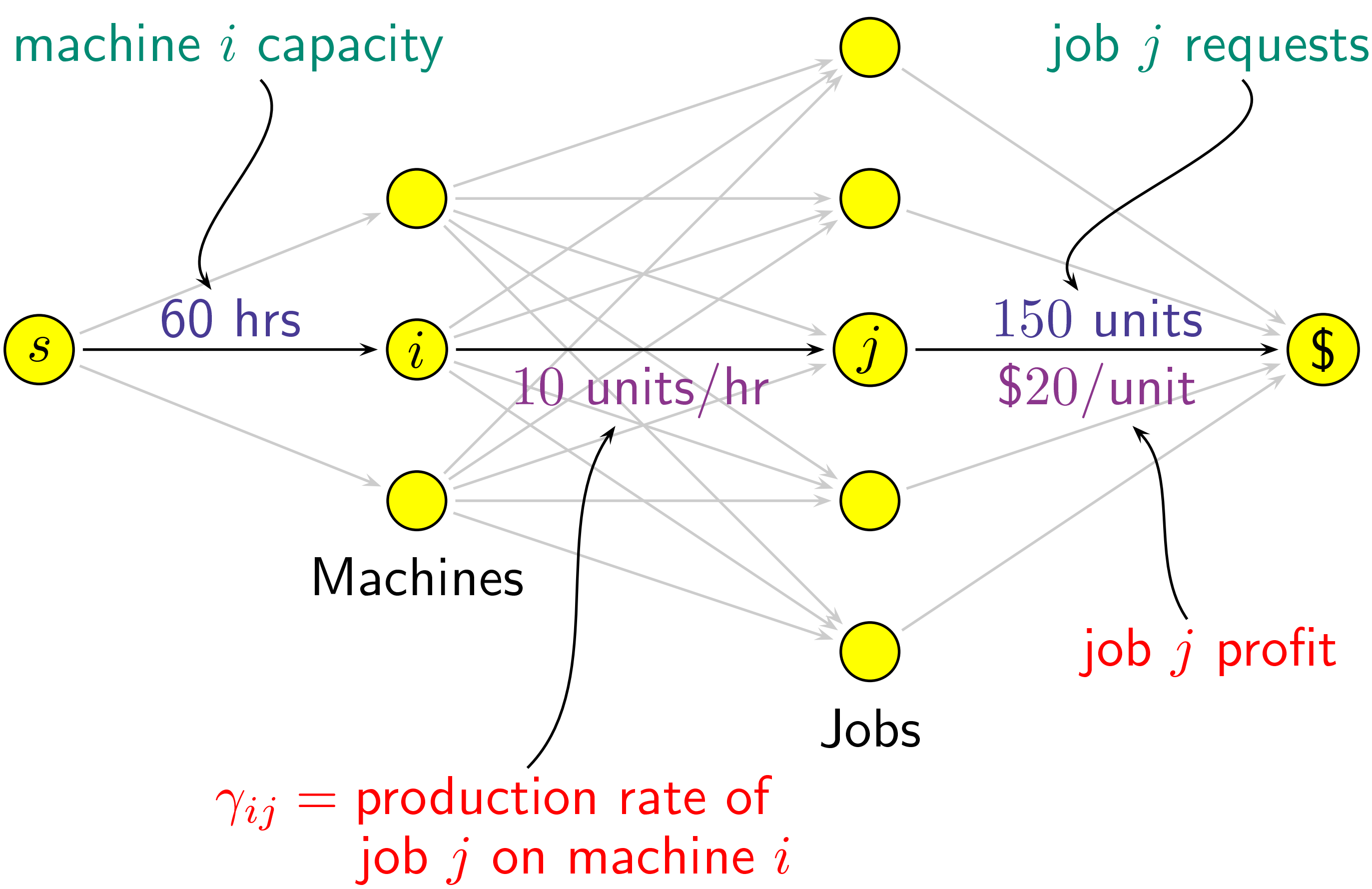
- exchange rates
- limits on trading capacity



Scheduling Unrelated Parallel Machines

Assign jobs to machines to maximize total profit

- machines have speeds and capacities
- profit for completing each job requested
- can split jobs between machines



General Approach

Combinatorial approach (Ford and Fulkerson '50s)

- Exploit underlying network structure
- Superior algorithms for traditional network problems
  - shortest path
  - max flow
  - min cost flow
  - matching
  - minimum spanning tree
- Not so much known about combinatorial algorithms for
  - multicommodity flow
  - generalized flow

Can also be solved by general purpose LP techniques

- simplex, ellipsoid, interior point

Combinatorial Approach

Why generalized flows are harder:

- supply $\neq$ demand
- no integrality theorem
- no max flow - min cut theorem

Can still use:

- linear programming duality

New bit-scaling technique:

- gain-scaling

Problem History

Linear programming

Dantzig '62 network simplex

Onaga '66, Jewell '62 augmenting path

Goldberg, Plotkin, Tardos '88 Fat-Path, MCF

- first polynomial-time combinatorial algorithms
- developed combinatorial machinery

Goldfarb, Jin, Orlin '96

- best worst-case complexity - $\mathcal{O}^*(m^3 \log B)$

$m = \#$ arcs	$\mathcal{O}^* =$ hides polylog(m) factors
$n = \#$ nodes	$B =$ biggest gain/capacity integer

Approximate Problem History

Can find provably good flows faster than optimal flows

Approximate flows An ϵ-optimal flow is a flow with value at least $(1-\epsilon)$ OPT

Cohen, Meggido '92

- strongly polynomial approximation algorithm (# operations depends on size of network only)

Radzik '93a, '93b Fat-Path

- original Fat-Path is strongly polynomial approximation algorithm
- Fat-Path variant
  - $\mathcal{O}^*(m^2+mn\log\log B)$ approximation algorithm
  - complicated

My Work

Gain-scaling technique provides:

this talk →

- Cleanest and simplest polynomial-time algorithm
  - variant of primal-dual method of Truemper '77
- First polynomial-time preflow-push algorithm for generalized flows
  - Goldberg-Tarjan preflow-push is most practical traditional max flow algorithm
  - practical implementation
- Fat-Path variant
  - matches best running time for approximate flows
  - much simpler than Radzik's variant

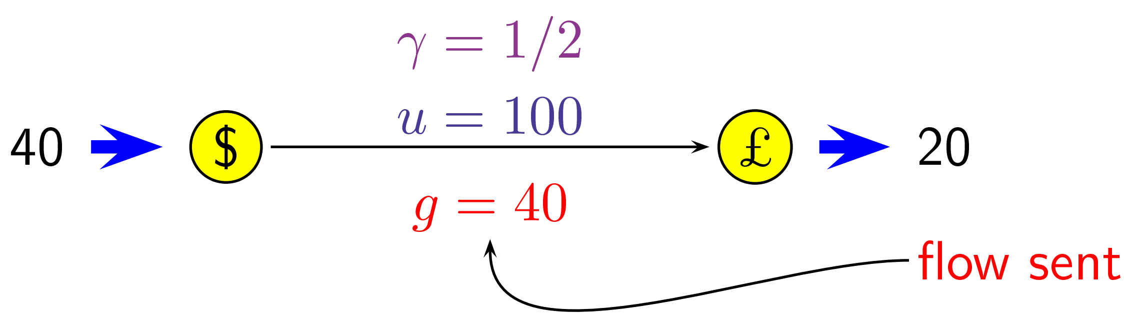
Organization of Talk

1. Applications
2. Previous work
3. Combinatorial structure and optimality conditions
4. Exponential-time augmenting path algorithm
5. Polynomial-time variant using gain-scaling

Residual Network

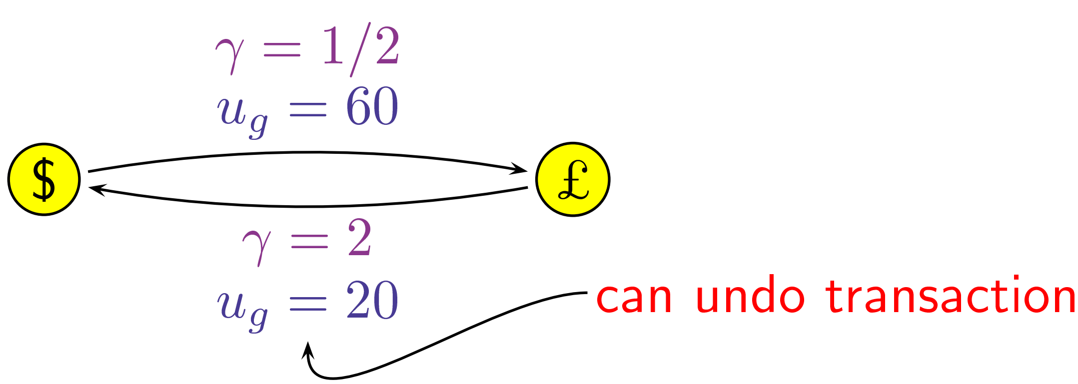
Produces equivalent but potentially simpler problem

Original network: $G = (V, E, u)$



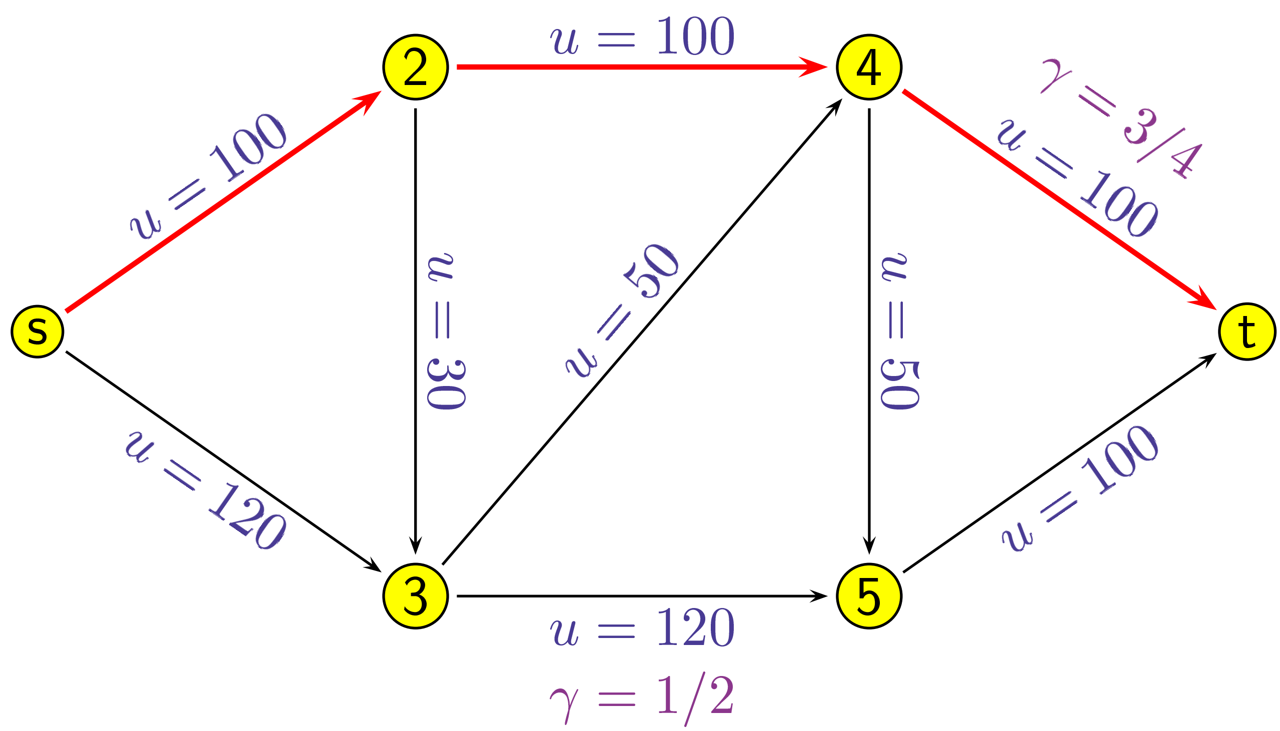
$u =$ capacity, $\gamma =$ gain

Residual network: $G_g = (V, E_g, u_g)$



Augmenting Paths

Residual network G_g:



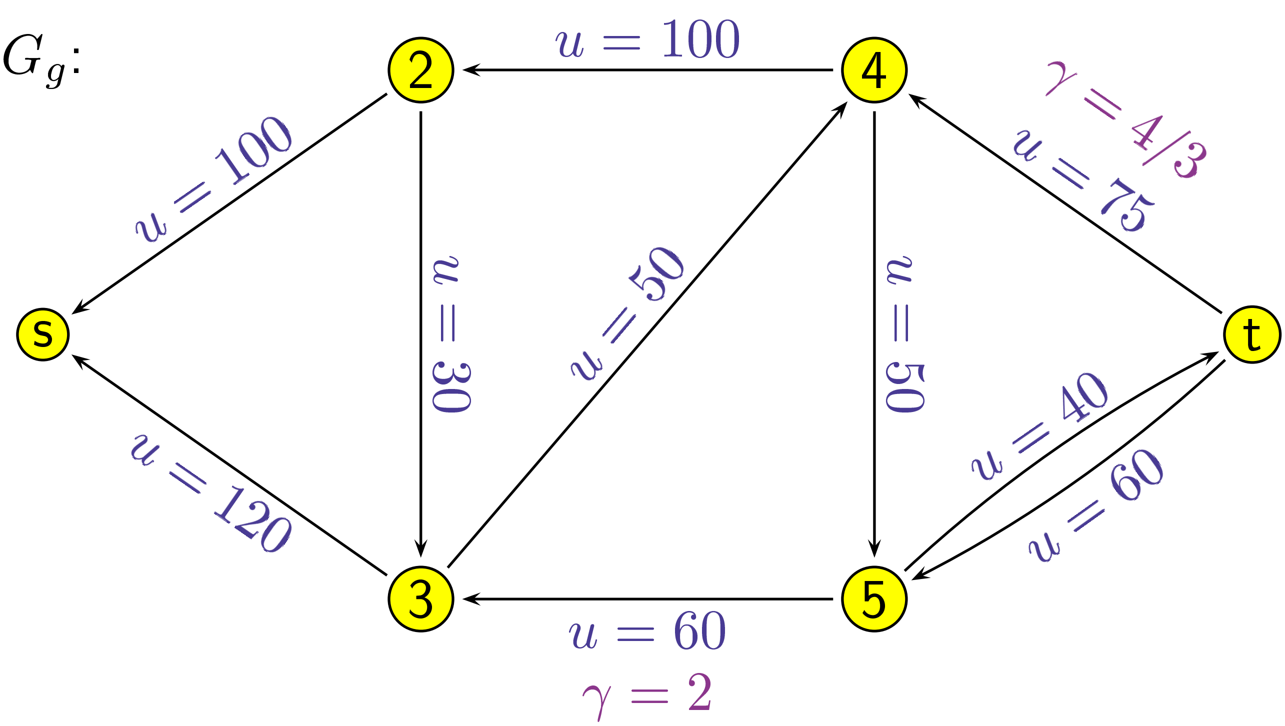
Augmenting path: residual path from s to t

Send 1 unit from s to t along path P then $\gamma(P) = \prod_{e \in P} \gamma(e)$ arrive at t

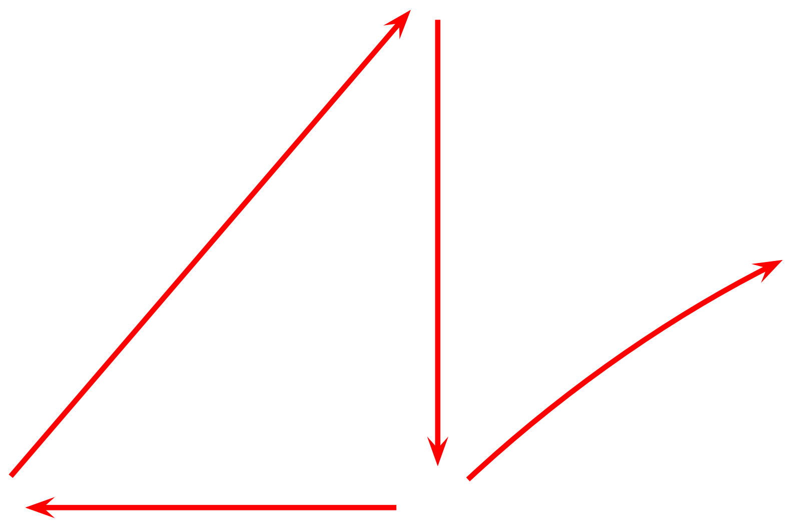
Generalized Augmenting Paths

optimality $\iff$ no augmenting paths ?

G_g:



No.

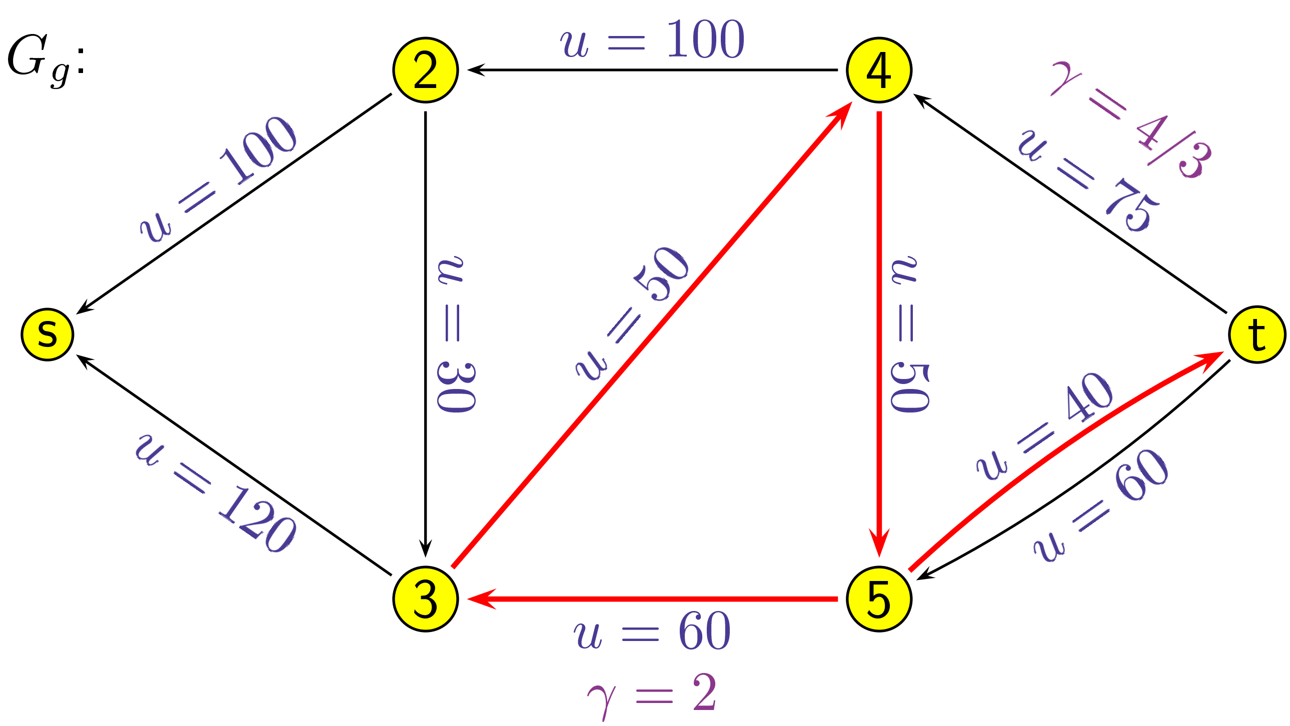


Flow-generating cycle: cycle Γ with $\gamma(\Gamma) > 1$
arbitrage

GAP: Residual flow-generating cycle + path to t

Generalized Augmenting Paths

optimality $\iff$ no augmenting paths ? No.



Flow-generating cycle: cycle Γ with $\gamma(\Gamma) > 1$
arbitrage

GAP: Residual flow-generating cycle + path to t

Optimality Conditions

For generalized max flow:

Theorem. [Onaga '66] *generalized flow g optimal iff no augmenting paths or GAPs in G_g.*

For min cost max-flow:

Theorem. [Negative Cost Cycle] *flow f optimal iff no augmenting paths or negative cost cycles in G_g.*

flow-generating cycle $\Longleftrightarrow$ negative cost cycle
using cost function $c(v, w) = -\log \gamma(v, w)$

Organization of Talk

1. Applications
2. Previous work
3. Combinatorial structure and optimality conditions
4. Exponential-time augmenting path algorithm
5. Polynomial-time variant using gain-scaling

Onaga's Algorithm '66

Analog to successive shortest path algorithm for min cost flows

- Assumes no arbitrage initially (i.e., no residual flow-generating cycles)
- Repeatedly augment flow along some highest-gain (most efficient) augmenting path
- Can find with shortest path computation using costs $c(v, w) = -\log \gamma(v, w)$

Correctness: Does not create flow-generating cycles if augmentations along highest gain path

Complexity: Very bad!

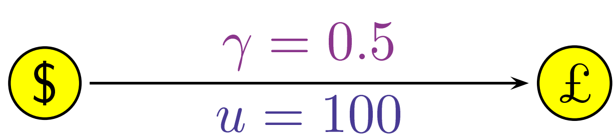
Relabeled Network

Node labels (dual variables): $\pi(v) \geq 0, \pi(t) = 1$
Changes local units in which flow is measured

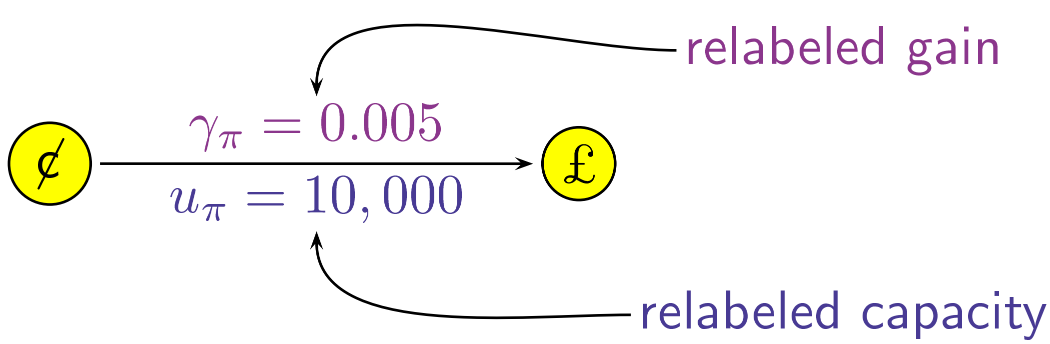
Example: Node v changed from dollars to pennies

$$\pi(v) = 100 = \#\text{ new units per old unit}$$

Original network: $G = (V, E, u, \pi)$

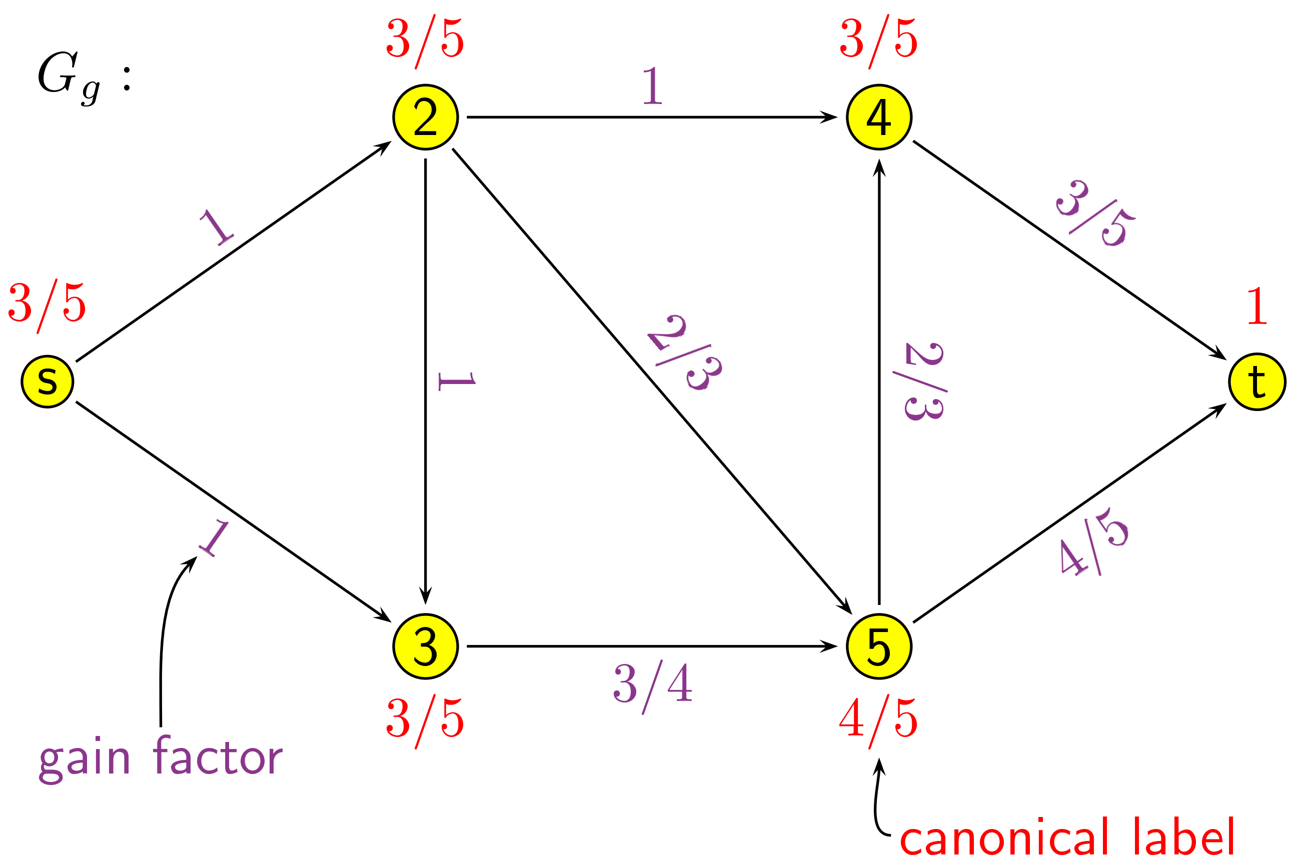


Relabeled network: $G_\pi = (V, E, u_\pi, \gamma_\pi)$



Canonical Labels

Canonical labels: $\pi(v)$ = gain of most efficient (highest-gain) residual v-t path



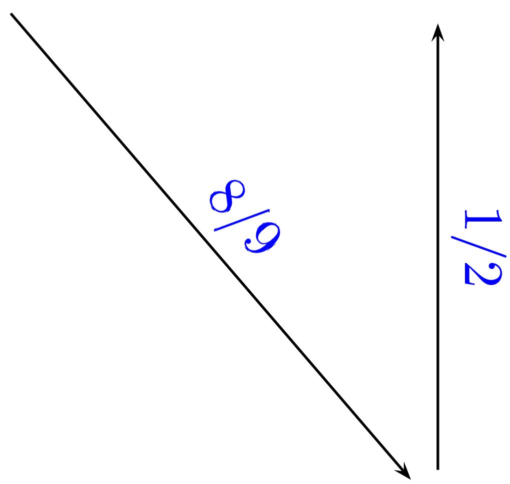
- Can compute if all gain factors ≤ 1 using cost function $c(v, w) = -\log \gamma(v, w)$

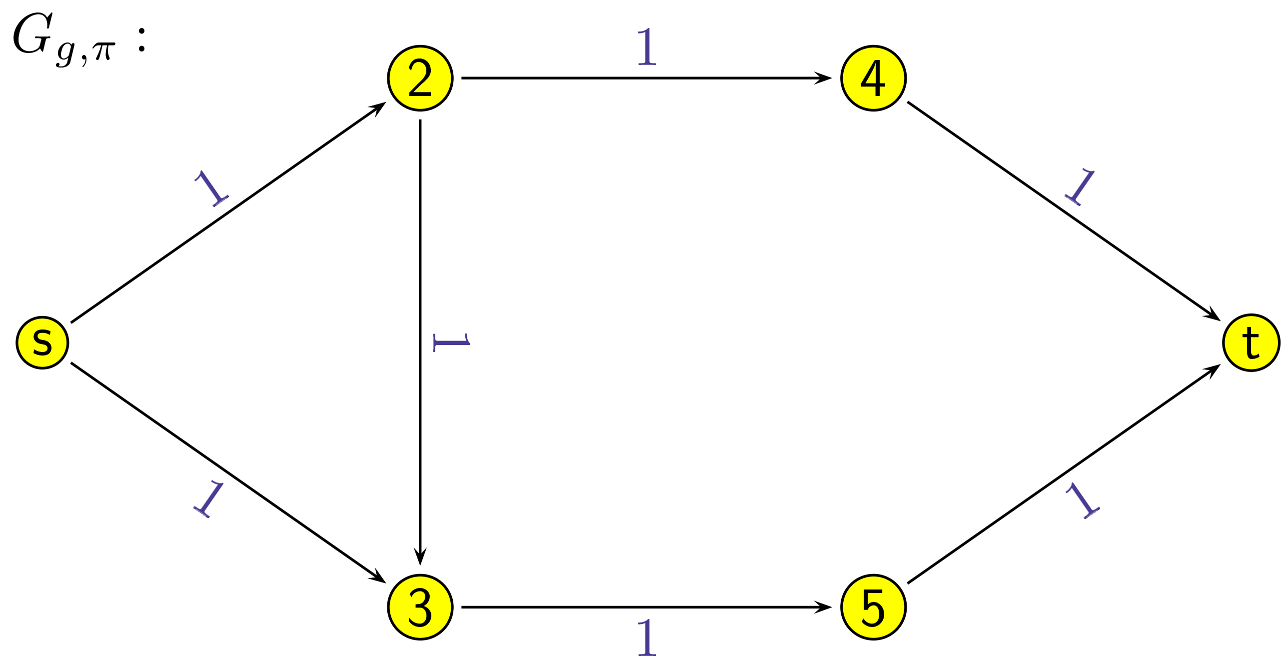
Canonical labels: $\pi(v)$ = gain of most efficient (highest-gain) residual v-t path

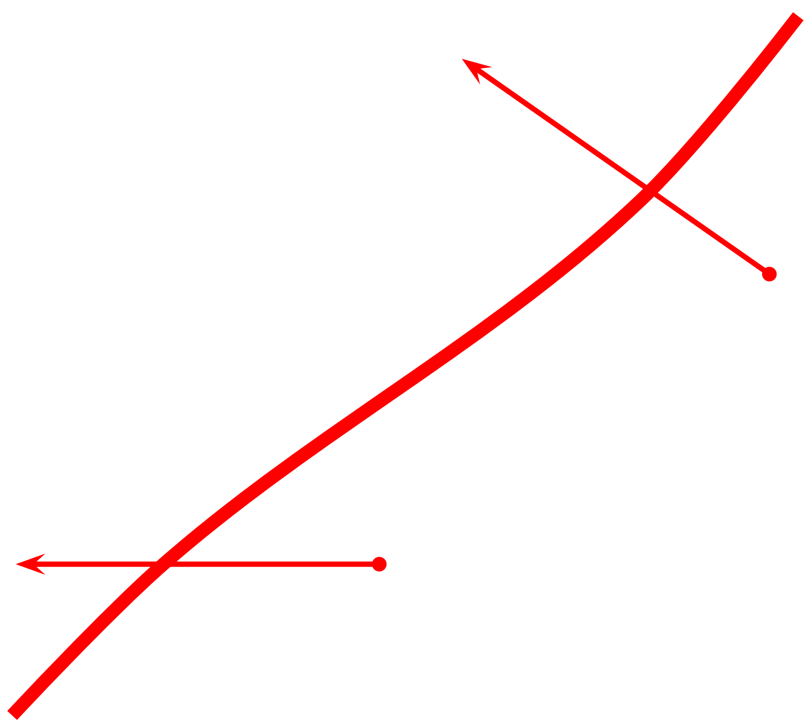
relabeled gain factor

- After canonical relabeling,
  - $\forall$ residual arcs (v, w): $\gamma_\pi(v, w) \le 1$
  - $\exists$ gain 1 relabeled residual s-t path

Canonically Relabeled Network




$G_{g,\pi}$:
s
2
3
4
5
t
1
1
1
1
1
1
1



Truemper's Algorithm '77

Analog of Ford and Fulkerson's primal-dual min cost flow algorithm

- Maintains flow g and canonical labels π such that $G_{g,\pi}$ has only lossy arcs (gain factor $\leq$ 1)

- Augment flow simultaneously along all highest-gain (most efficient) augmenting paths, i.e., all unit gain s-t paths in $G_{g,\pi}$

repeat
  $\pi \leftarrow$ canonical labels
  $f \leftarrow$ max flow from s to t in $G_{g,\pi}$ using
    only $\gamma_\pi = 1$ arcs
  $g(v,w) \leftarrow g(v,w) + \pi(v)f(v,w)$
until no augmenting paths

Correctness: as before

Truemper's Algorithm (cont.)

Complexity: After each max flow computation, the gain of most efficient augmenting path strictly decreases (optimal if no such paths)

$\#$ max flow iterations $\leq \#$ distinct gains of paths in G

If gain factors are powers of 2:

- Gains of arcs are between $\frac{1}{B}$ and 1
- Gains of residual paths are between $\frac{1}{B^n}$ and 1
- At most $\log_2 B^n$ distinct gains of paths
  $\Longrightarrow n \log_2 B$ max flow iterations

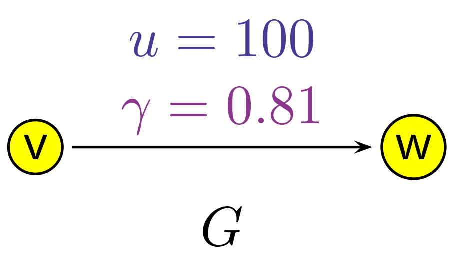
Organization of Talk

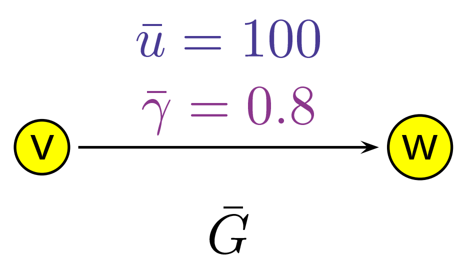
1. Applications
2. Previous work
3. Combinatorial structure and optimality conditions
4. Exponential-time augmenting path algorithm
5. Polynomial-time variant using gain-scaling

New Gain-Scaling Algorithm

Gain-scaling = rounding + recursion

- Applies if G has only lossy residual arcs (gain ≤ 1)
- Round gains down to powers of $b = (3/2)^{1/n}$



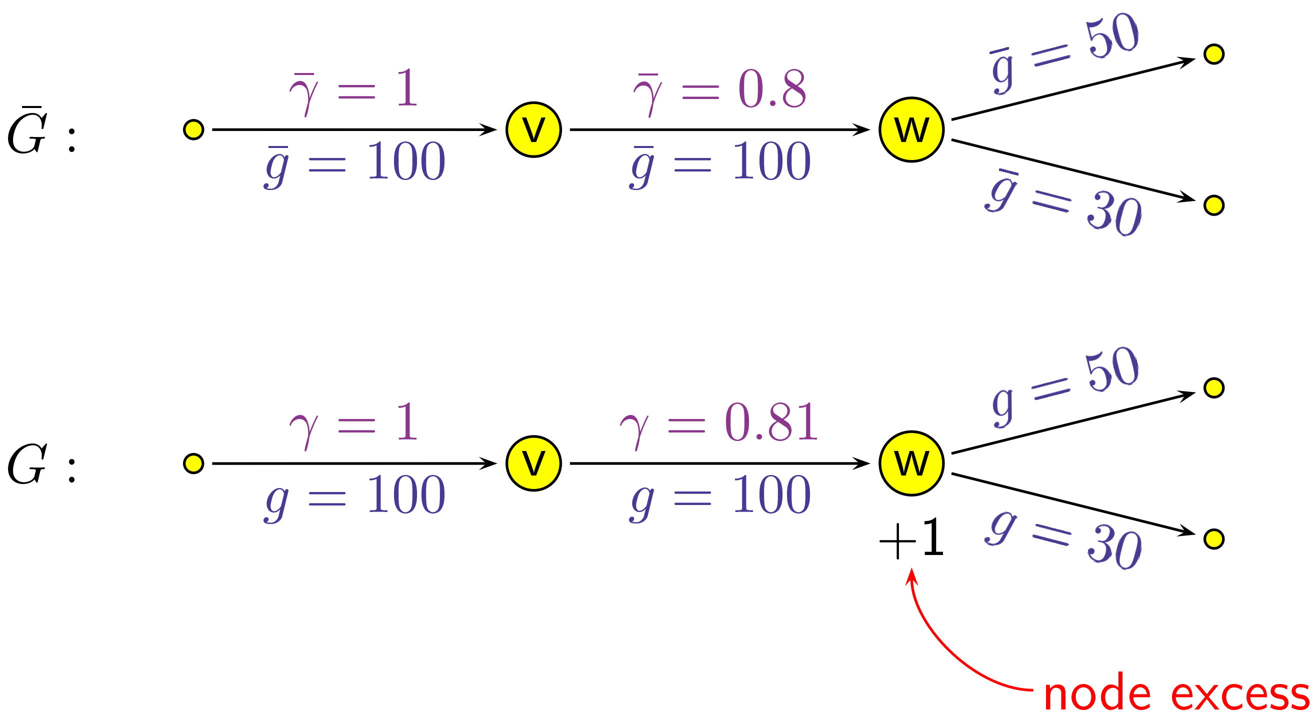


- Find optimal flow in rounded network $\bar{G}$ using Truemper's algorithm.

Complexity: $\log_b B^n = \mathcal{O}(n^2 \log B)$ max flows

New Gain-Scaling Algorithm

- Found optimal flow in rounded network $\bar{G}$
- Interpret flow in G



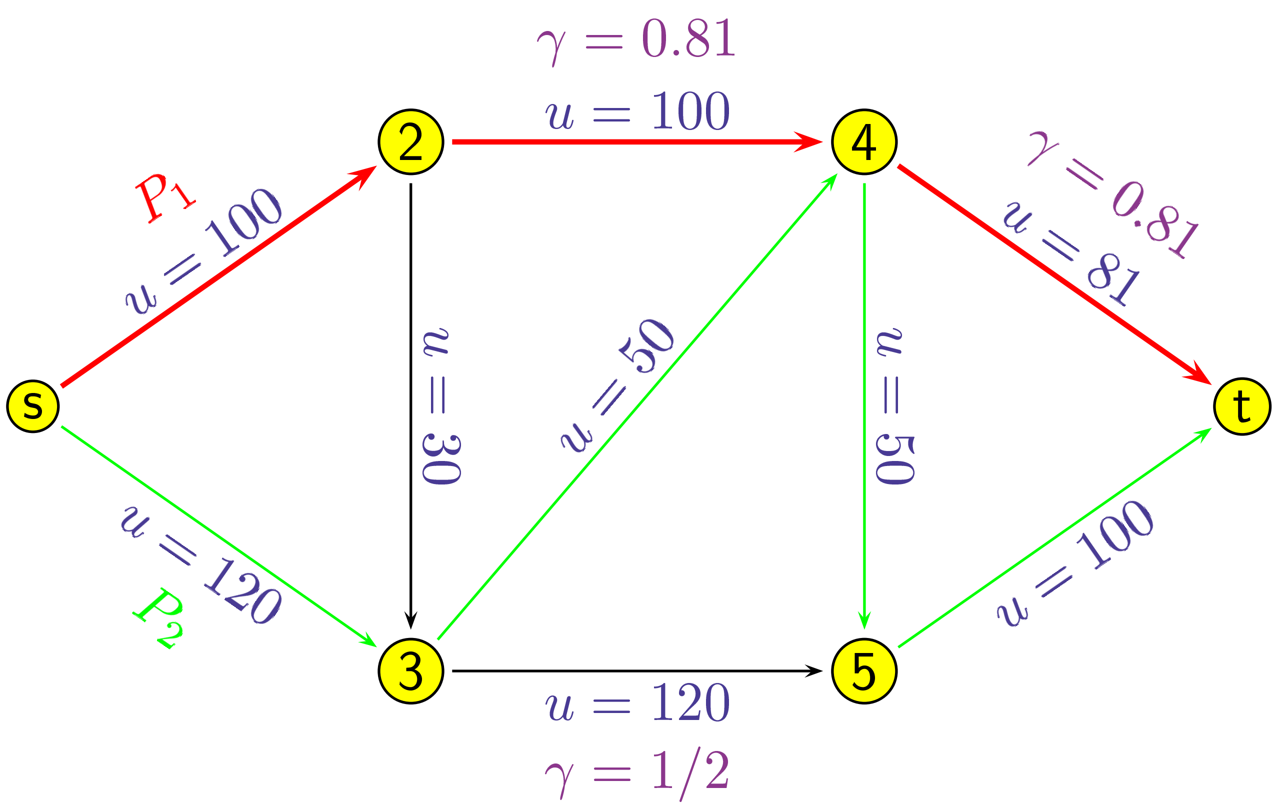
- Resulting flow in G
  - satisfies capacity constraints
  - is at least as good as flow in $\bar{G}$
  - may violate flow conservation constraints (but only in a good way!)

Rounded Network

Rounded network $\bar{G}$ close to original network G:

$$\tfrac{2}{3}\,\mathrm{OPT}(G) \;\le\; \mathrm{OPT}(\bar{G}) \le \mathrm{OPT}(G) \qquad \bar{\gamma} \le \gamma$$

Path flow formulation: x_j = flow sent on path P_j



Rounded Network

$\frac{2}{3}\,\mathrm{OPT}(G) \;\leq\; \mathrm{OPT}(\bar{G})$

let x^* be optimal path flow in G

$\Longrightarrow\;$ x^* feasible path flow in $\bar{G}$

$x_j^* = 100$

$$G: \quad s \xrightarrow[u\,=\,200]{\gamma\,=\,1} 2 \xrightarrow[u\,=\,200]{\gamma\,=\,0.81} 4 \xrightarrow[u\,=\,81]{\gamma\,=\,0.81} t \;\Rightarrow\; 65.61$$

$x_j^* = 100$

$$\bar{G}: \quad s \xrightarrow[\bar{u}\,=\,200]{\bar{\gamma}\,=\,1} 2 \xrightarrow[\bar{u}\,=\,200]{\bar{\gamma}\,=\,0.8} 4 \xrightarrow[\bar{u}\,=\,81]{\bar{\gamma}\,=\,0.8} t \;\Rightarrow\; 64$$

$$\bar{\gamma}(P) \geq \frac{\gamma(P)}{b^{|P|}} \geq \frac{\gamma(P)}{3/2} \quad\Longrightarrow\quad \mathrm{OPT}(\bar{G}) \geq \frac{2}{3}\,\mathrm{OPT}(G)$$

$\bar{\gamma}(e) \geq \frac{\gamma(e)}{b}$ $\qquad$ $b = (3/2)^{1/n}$

Geometric Improvement

Idea: Compute $1/3$-optimal flow. Recurse.

Initialize $g \leftarrow 0$
repeat
  $g' \leftarrow 1/3$-optimal flow in G_g
  $g \ \leftarrow g + g'$
until g is ϵ-optimal

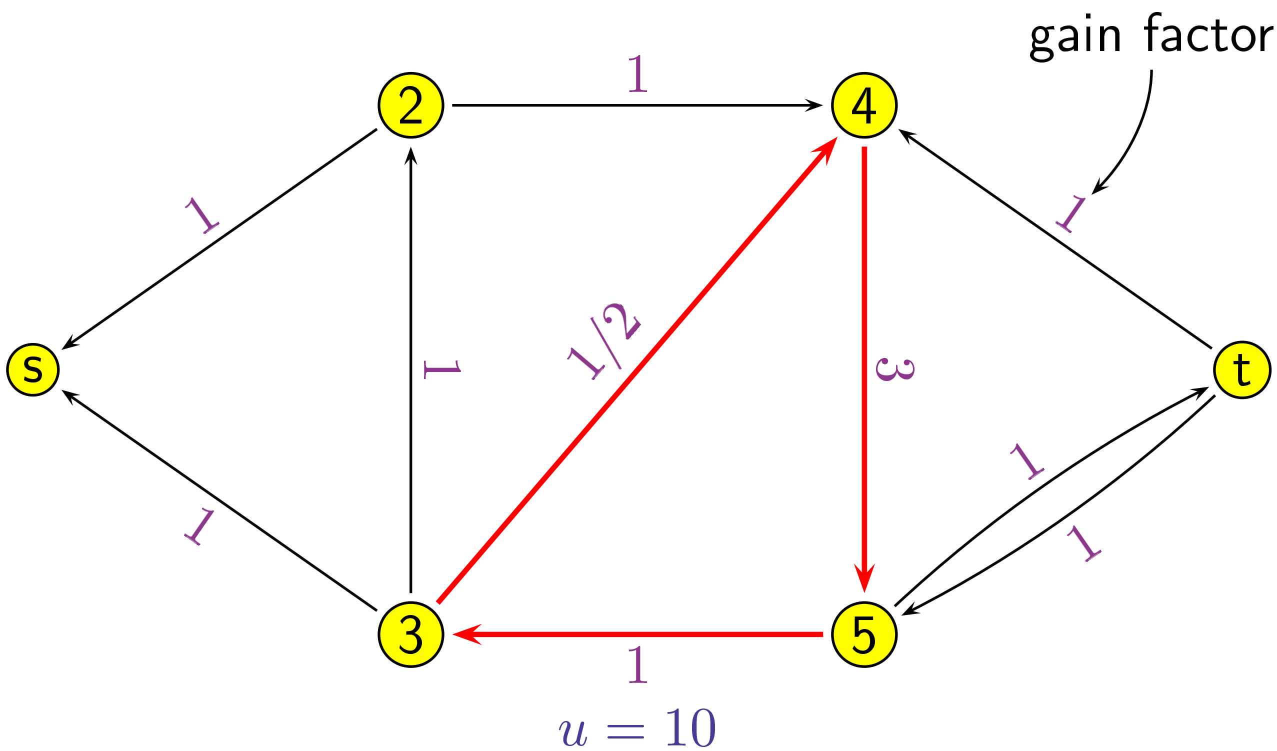
Analysis: Each iteration captures at least $2/3$ of remaining flow, so flow is ϵ-optimal in $\log(1/\epsilon)$ iterations

Our Result. *The algorithm computes an ϵ-optimal flow in $\mathcal{O}^*(mn^3 \log B) \log(1/\epsilon)$ time.*

Canceling Flow-Generating Cycles

Truemper's algorithm only works if no gain factor > 1

Can we relabel to eliminate gainy arcs?
Yes $\iff$ no residual flow-generating cycles



Cancel flow-generating cycles, creating only excess

Canceling Flow-Generating Cycles

Goal: Cancel (saturate) **all** flow-generating cycles, creating only excesses, so that network can be relabeled as a lossy network

Goldberg, Tarjan '88 mean cycle canceling
For min cost flows, repeatedly cancel residual cycle with most negative mean cost

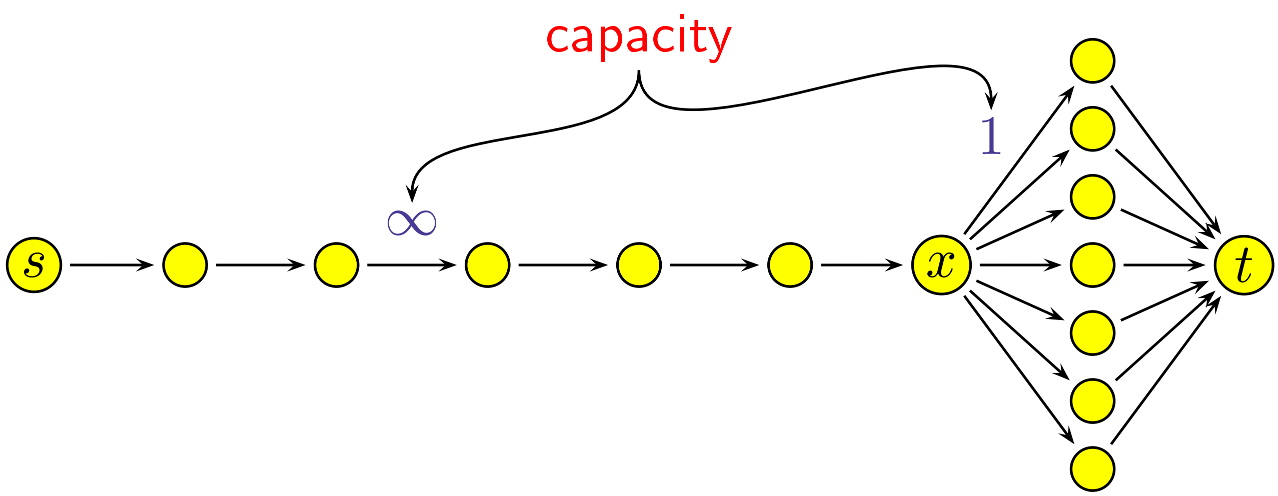
GPT '88 generalized flow analog

- Using cost function $c(v, w) = -\log \gamma(v, w)$, flow-generating cycle $\iff$ negative cost cycle
- Repeatedly cancel residual flow-generating cycle with maximum geometric-mean gain
- $\mathcal{O}^*(mn^2 \log B)$ running time

Our improved version in rounded networks

- $\mathcal{O}^*(mn \log \log B)$ running time

Preflow-Push



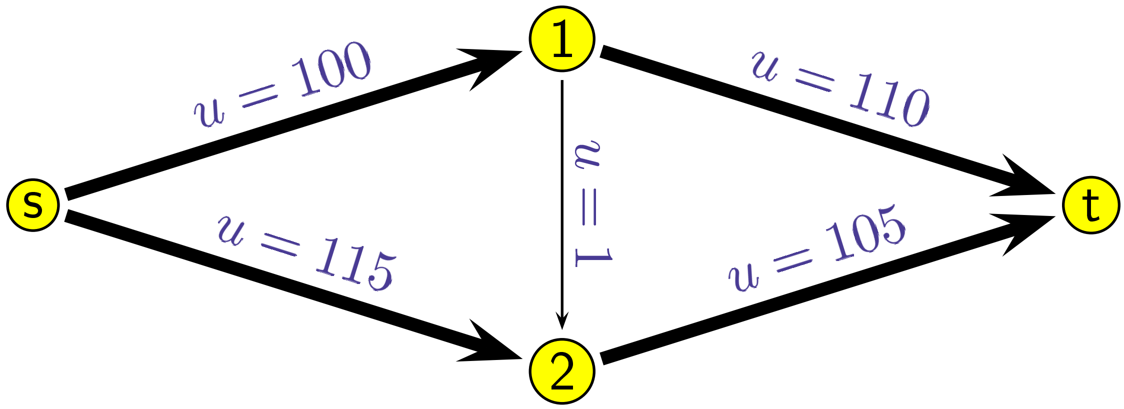
Goldberg, Tarjan '86 preflow-push

- Best algorithm in practice and theory for traditional max flows
- Each augmentation along an arc instead of whole path (only uses local information)

Our Result. *There exists a preflow-push algorithm for the generalized max flow problem that computes a* ϵ*-optimal flow in* $\mathcal{O}^*(mn^3 \log B) \log(1/\epsilon)$ *time.*

- practical implementation

Fat-Path



Goldberg, Plotkin, and Tardos '88
augment flow along "Fat-Paths"

Bottleneck canceling flow-generating cycles

- GPT - $\mathcal{O}^*(mn^2 \log B)$
- $\mathcal{O}^*(mn \log \log B)$ in our rounded networks

Theorem. [Radzik '93] Fat-Path variant computes an ϵ-optimal flow in $\mathcal{O}^*(m^2 + mn \log \log B) \log(1/\epsilon)$.

- cancel only highest gain flow-generating cycles
- very complicated

[Our Result.] *Much simpler version, same complexity.*

Ideas Needed to Improve Fat-Path

- Faster cycle-canceling
- More careful rounding
- Divide and conquer

Closing Remarks

Conclusions

- New gain-scaling technique
  - new intuitive combinatorial algorithms
  - matches best theoretical complexity
  - promising practical performance

Open Problems

- improve complexity
- faster implementation (no arbitrage assumption)
- generalized multicommodity flows
- generalized min cost flows

Linear Program

generalized maximum flow problem

$$\max_{e,\ g} e(t)$$

$$\sum_{w \in V} g(v,w) - \sum_{w \in V} \gamma(w,v) g(w,v) \quad = \quad \begin{cases} e(s) & v = s \\ 0 & v \neq s,t \\ -e(t) & v = t \end{cases}$$

$$0 \leq g(v,w) \leq u(v,w)$$

minimum cost flow problem

$$\min_{f} \sum_{(v,w) \in E} c(v,w) f(v,w)$$

$$\sum_{w \in V} f(v,w) - \sum_{w \in V} f(w,v) \quad = \quad \begin{cases} e(s) & v = s \\ 0 & v \neq s,t \\ -e(s) & v = t \end{cases}$$

$$0 \leq f(v,w) \leq u(v,w)$$

Linear Program

generalized maximum flow problem

$$\max_{e,\ g} e(t)$$

$$\sum_{w \in V} g(v,w) - \sum_{w \in V} \gamma(w,v) g(w,v) \quad = \quad \begin{cases} e(s) & v = s \\ 0 & v \neq s,t \\ -e(t) & v = t \end{cases}$$

$$0 \leq g(v,w) \leq u(v,w)$$

LP Dual

$$\min_{\pi} \sum_{(v,w) \in E} c_\pi(v,w) u(v,w)$$

$$c_\pi(v,w) = \max\{0, -\pi(v) + \gamma(v,w)\pi(w)\}$$

$$0 = \pi(s) \leq \pi(v) \leq \pi(t) = 1$$

Optimality Conditions

for generalized max flow:

Theorem. [complementary slackness] *generalized flow g optimal iff $\exists$ node labels $\pi(v) \geq 0$, $\pi(s) = 0$, and $\pi(t) = 1$ such that*

$$\forall\ (v, w) \in E_g: \quad \pi(v) - \gamma(v, w)\pi(w) \geq 0.$$

$\pi(v)$ = market price for commodity at node v
complementary slack $\iff$ no profitable residual arcs

for min cost flow:

Theorem. [complementary slackness] *flow f optimal iff $\exists$ node labels $p(v)$, $p(t) = 0$ s.t.*

$$\forall\ (v, w) \in E_g: \quad c(v, w) + p(v) - p(w) \geq 0.$$