CSCI 3210: Computational Game Theory

Market Equilibria: An Algorithmic Perspective

Ref: Ch 5 [AGT]
Ch 7 [Kleinberg-Tardos]

Mohammad T. Irfan

Email: mirfan@bowdoin.edu
Web: www.bowdoin.edu/~mirfan

Many of the slides are adapted from Vazirani's and Kleinberg-Tardos' textbooks.

1

Market

2

Study of markets

- **General equilibrium (GE) theory**
  - Seeks to explain the behavior of supply, demand and prices in an economy
  - Partial equilibrium vs GE

3

Competitive equilibrium (CE)

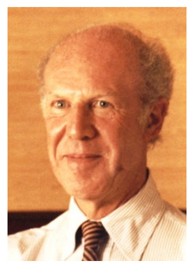
- AKA Walrasian equilibrium
  - Formal mathematical modeling of markets by Leon Walras (1874)
- CE consists of prices and allocations
- Equilibrium pricing: demand = supply
- GE $\Longrightarrow$ CE, but CE $\not\Rightarrow$ GE

4

Background

- Good news
  - CE **exists** in Walrasian economy
  - Proved by Arrow and Debreu (1954)
- Bad news
  - Existence proof is not algorithmic

Arrow



Debreu

5

Background

- 1st Welfare Theorem
  - Any **CE** (Walrasian equilibrium) leads to a "***pareto optimal***" allocation of resources

Nobody can be better off without making somebody else worse off

- Social justification
  - Let the competitive market do the work (everybody pursuing self-interest)
  - It will lead to pareto optimality (socially maximal benefit)

6

Timeline

- 1954 – 2001
  - We are happy. Equilibrium exists.
    Why bother about computation?
  - Sporadic computational results
    - Eisenberg-Gail convex program, 1959
    - Scarf's computation of approximate fixed point, 1973
    - Nenakov-Primak convex program, 1983

7

Today's markets

8

Electronic marketplaces

9

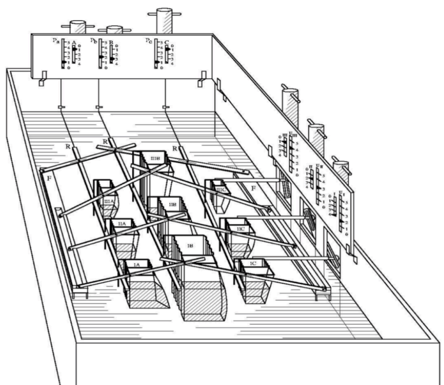
Need for algorithms

- New types of markets
  - The internet market
  - Massive computational power available
  - Need to "compute" equilibrium prices
- Effects of
  - Technological advances
  - New goods
  - Changes in the tax structure
- Deng, Papadimitriou and Safra (2002)- Complexity of finding an equilibrium; polynomial time algorithm for linear utility case
- Devanur, Papadimitriou, Saberi, Vazirani (2002) – polynomial time algorithm for Fisher's linear case

10

Fisher economy

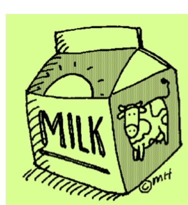
- Irving Fisher (1891)
  - Mathematical model of a market

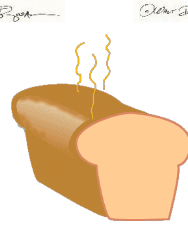


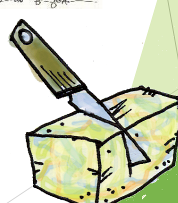
Fisher's apparatus to compute equilibrium prices

11

Fisher economy

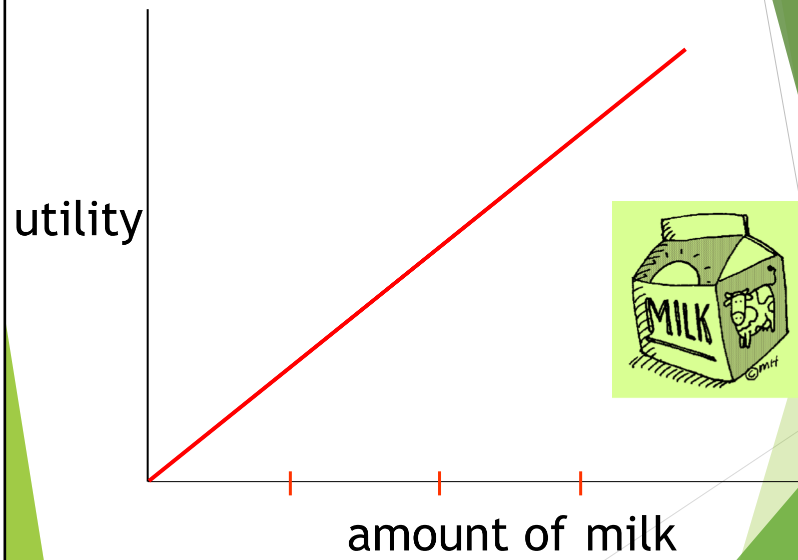






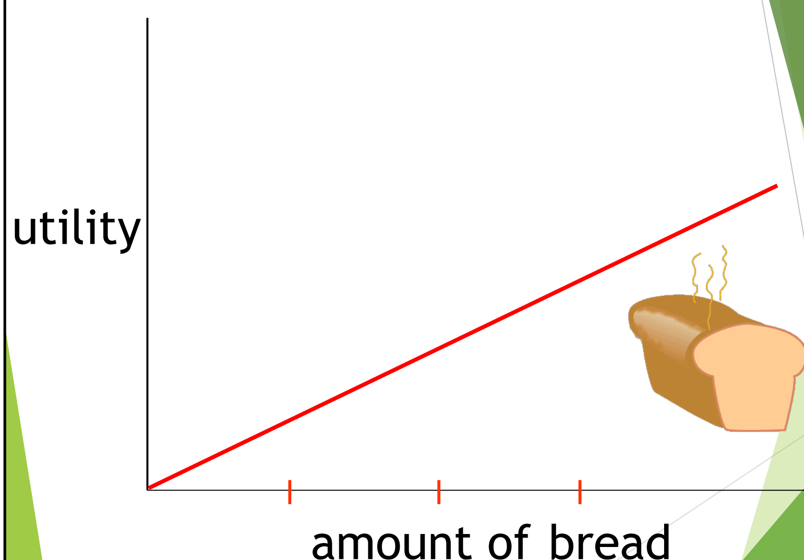
12

Utility function



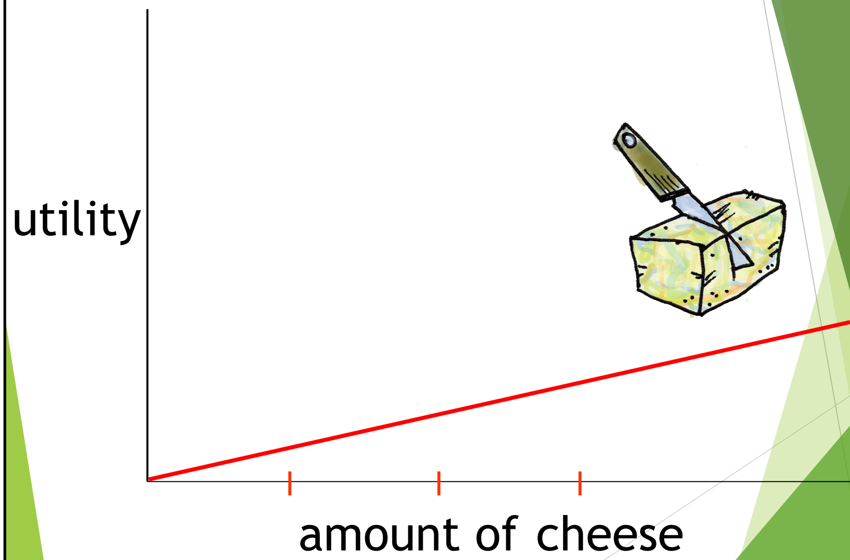
13

Utility function



14

Utility function



15

Total utility

- Total utility of a “bundle” of goods
  = Sum of the utilities of individual goods

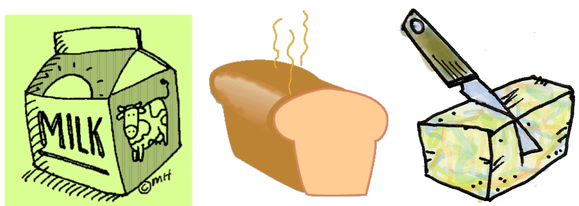
16

Easy problem

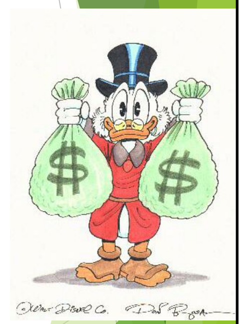
- Prices given
- What would be the optimal bundle of goods for a buyer?

Bang-per-buck (BPB)

Example: $u_2/p_2 > u_1/p_1 > u_3/p_3$



p_1 p_2 p_3



17

Fisher market – setup

- Multiple buyers, with individual budgets and utilities
- Multiple goods, fixed amount of each good
- Equilibrium/market-clearing prices
  - Each buyer maximizes utility at these prices
    - Buyers will exhaust their budgets
  - No excess demand or supply

18

Fisher's linear case

- Model parameters (what's given)
  - ***n divisible goods*** (1 unit each wlog) and ***n' buyers***
  - e_i = buyer *i*'s budget (integral wlog)
  - u_{ij} = buyer *i*'s utility per unit of good *j* (integral wlog)
  - *Linear* utility functions
- Want (not given): equilibrium allocations
  - x_{ij} = amount of good *j* that *i* buys to maximize

    his/her utility $u_i(\mathbf{x}) = \sum_{j=1}^{n} u_{ij} x_{ij}$

    - No excess demand or supply

19

Dual (proof later)

- Want (not given): equilibrium/market-clearing prices
- Prices: $p_1, p_2, \ldots, p_n$
  - After each buyer is assigned an optimal basket of goods (x_{ij}'s) w.r.t. these prices, there's no excess demand or supply
  - x_{ij}'s at these prices: equilibrium/market-clearing allocations

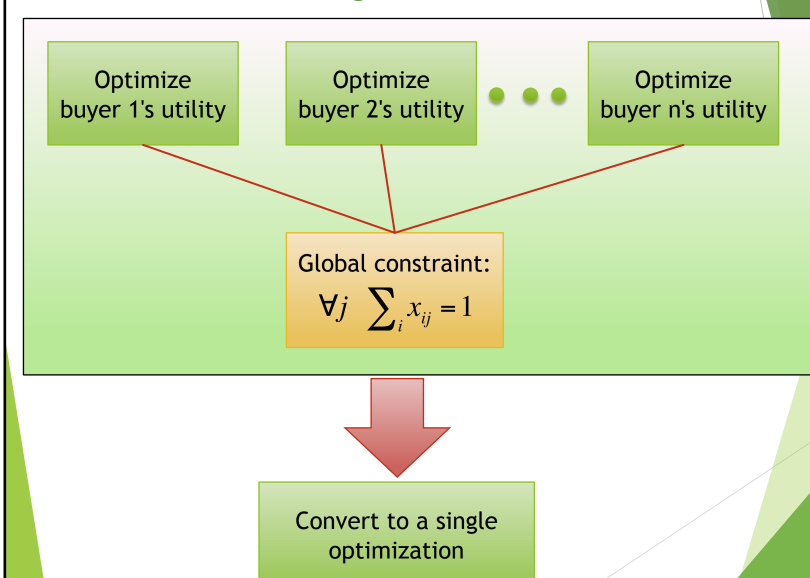
20

Can we formulate an optimization routine?

- Does LP work?
- Anything else?

21

Main challenge



26

Eisenberg-Gale Formulation of Fisher Market

27

How to devise duals of nonlinear programs?

Lagrange function

KKT conditions

28

Eisenberg-Gale convex program (1959)

- Equilibrium allocations captured as
  - Optimal solutions to the Eisenberg-Gale convex program
- Objective function
  - Money weighted geometric mean of buyers' utilities

$$\max\left(\prod_i u_i^{e_i}\right)^{1/\sum_i e_i} \Leftrightarrow \max\left(\prod_i u_i^{e_i}\right) \Leftrightarrow \max \sum_i e_i \log u_i$$

29

Eisenberg-Gale convex program

$$\max \sum_i e_i \log\left(\sum_j u_{ij}\, x_{ij}\right)$$

$$subject\ to$$

$$\sum_i x_{ij} \leq 1, \forall j$$

$$x_{ij} \geq 0, \forall i, j$$

- Lagrange function

$$L(x,\lambda,\mu) = -\sum_i e_i \log \sum_j u_{ij} x_{ij} + \sum_j \lambda_j \left(\sum_i x_{ij} - 1\right) + \sum_i \sum_j \mu_{ij}(-x_{ij})$$

30

KKT conditions

- Stationary condition

$$\frac{e_i u_{ij}}{\sum_j u_{ij} x_{ij}^*} = \lambda_j^* - \mu_{ij}^* \qquad (1)$$

$$\frac{u_{ij}}{\lambda_j^*} \leq \frac{\sum_j u_{ij} x_{ij}^*}{e_i}$$

- Primal feasibility

$$\sum_i x_{ij}^* \leq 1, \forall j$$
$$x_{ij}^* \geq 0, \forall i, j$$

- Dual feasibility

$$\lambda_i^*, \mu_{ij}^* \geq 0, \forall i, j$$

- Complementary slackness

$$\lambda_j^* \left(\sum_i x_{ij}^* - 1 \right) = 0 \quad \Leftrightarrow \quad \lambda_j^* > 0 \Rightarrow \sum_i x_{ij}^* = 1$$
$$\mu_{ij}^* (-x_{ij}^*) = 0 \quad \Leftrightarrow \quad x_{ij}^* > 0 \Rightarrow \mu_{ij}^* = 0$$

31

Does Eisenberg-Gail convex program work for Fisher market?

- Prove: There exist market-clearing prices **iff** each good has some interested buyer (someone who gets positive utility for that good)

32

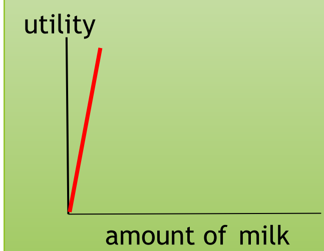
Example

- 2 buyers, 1 good (1 unit of milk)

Buyer 1

Budget, e_1 = \$100

u_{11} = 10/unit of milk

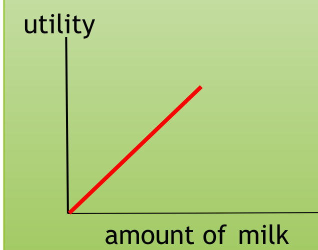


x^*_{11} = ?

Buyer 2

Budget, e_2 = \$50

u_{21} = 1/unit of milk



x^*_{21} = ?

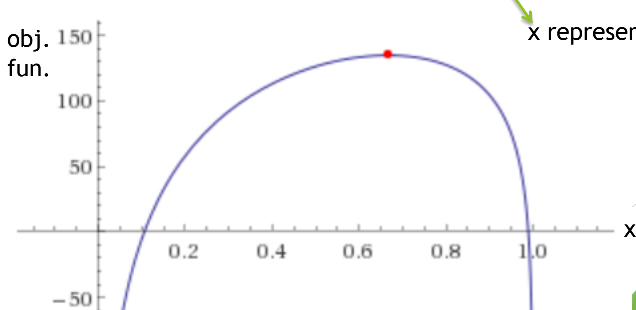
33

Solution

- $x_{11} = 2/3$, $x_{21} = 1/3$

maximize	function	$100 \log(10x) + 50 \log(1 - x)$
	domain	$0 \leq x \leq 1$

x represents x_{11}



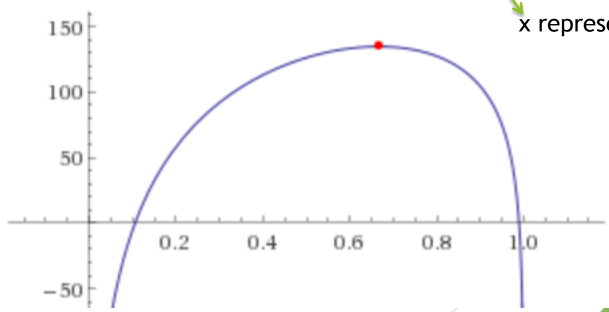
34

Solution

- Why $x_{11} = 2/3$, $x_{21} = 1/3$?
- Set price of milk = \$150/unit

maximize	function	$100 \log(10\,x) + 50 \log(1 - x)$
	domain	$0 \le x \le 1$

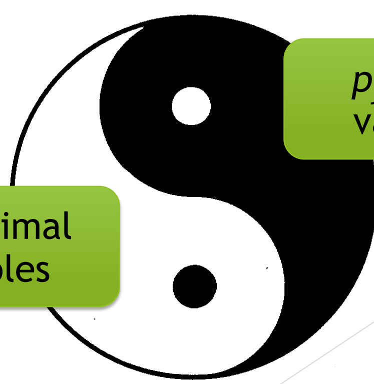
x represents x_{11}



35

Primal-dual

- p_j
  = The price of good j at an equilibrium
  = Dual variable corresponding to the primal constraint for good j: $\sum_i x_{ij} \le 1$



p_j's: dual variables

x_{ij}'s: primal variables

36

Interesting properties

- The set of equilibria is convex
- Equilibrium prices are unique!
- All entries rational => equilibrium allocations and prices rational

42

Flow

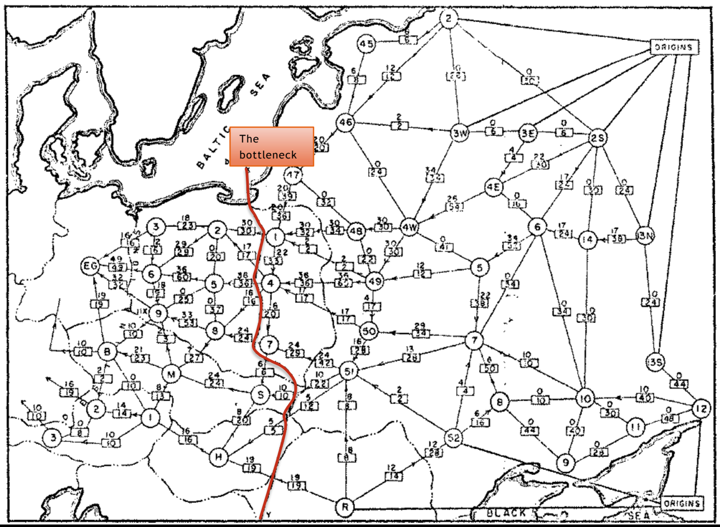
Max Flow & Min Cut
Ref: Ch 7 of Kleinberg-Tardos

Slides adapted from the Algorithm Design textbook slides
[Kleinberg, Tardos, K. Wayne, P. Kumar]

43

History: Schrijver (2002)

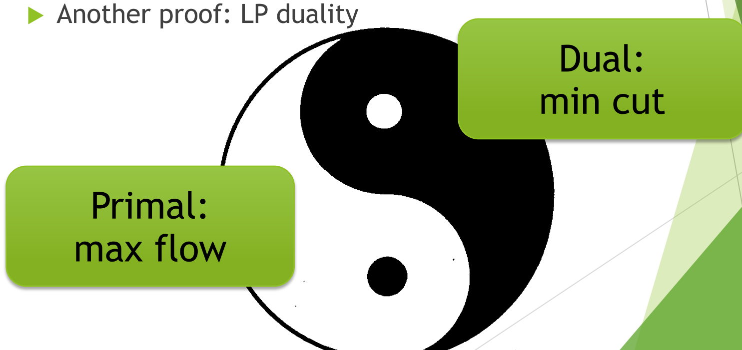
- http://homepages.cwi.nl/~lex/files/histtrpclean.pdf
- Soviet rail network: Harris and Ross [1955] (declassified 1999)



44

Big picture

- Tolstoi (1930): Find max flow
- Harris & Ross (1955): Find min cut
- Ford & Fulkerson (1956): They are the same
  - Their proof: combinatorial
  - Another proof: LP duality



45

Applications

Network	Nodes	Arcs	Flow
communication	telephone exchanges, computers, satellites	cables, fiber optics, microwave relays	voice, video, packets
circuits	gates, registers, processors	wires	current
mechanical	joints	rods, beams, springs	heat, energy
hydraulic	reservoirs, pumping stations, lakes	pipelines	fluid, oil
financial	stocks, currency	transactions	money
transportation	airports, rail yards, street intersections	highways, railbeds, airway routes	freight, vehicles, passengers
chemical	sites	bonds	energy

46

Applications

- **Fisher market**
- Network connectivity
- Bipartite matching
- Data mining
- Open-pit mining
- Airline scheduling
- Image processing
- Project selection
- Baseball elimination
- Network reliability
- Security of statistical data
- Distributed computing
- Egalitarian stable matching
- Distributed computing
- Many many more . . .

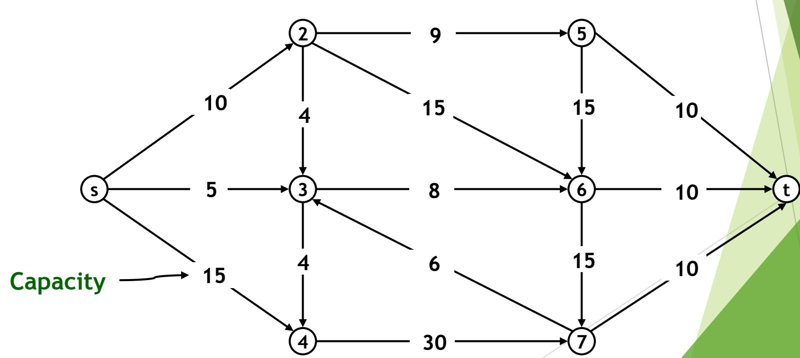
47

Max flow problem

48

Max flow network

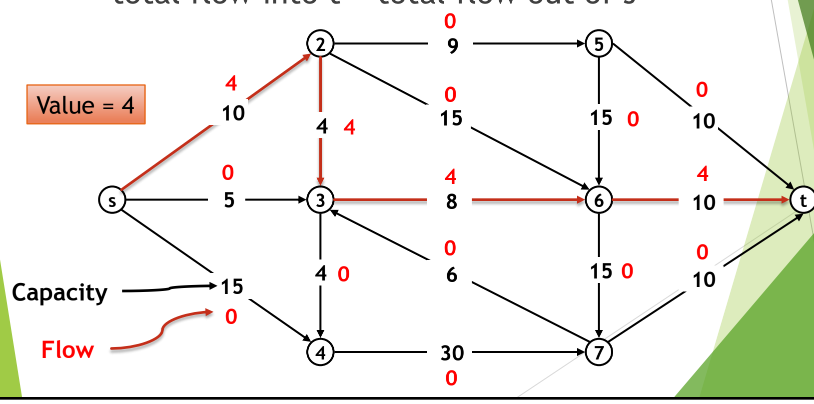
- Directed graph (may have cycles)
- Two distinguished nodes: s = source, t = sink
- c(e) = capacity of arc e (integer)



49

Definition: s-t flow

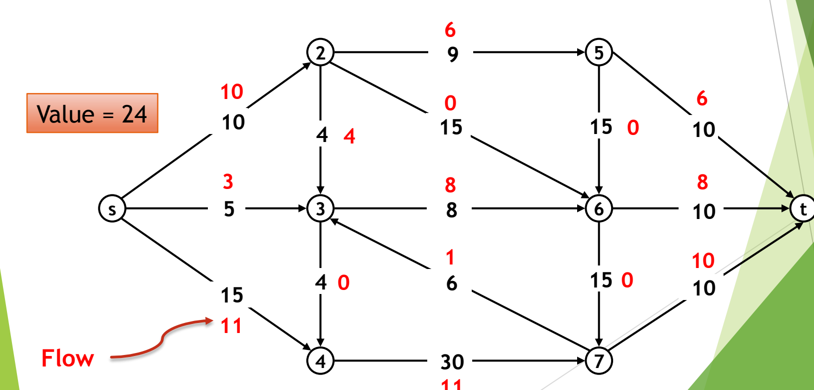
- Assignment of integer "flow" >= 0 on each arc:
  - (Capacity) Can't exceed arc's capacity
  - (Conservation) flow in = flow out at any node ≠ s, t
- Flow value
  = total flow into t = total flow out of s



50

Can we increase the flow value?

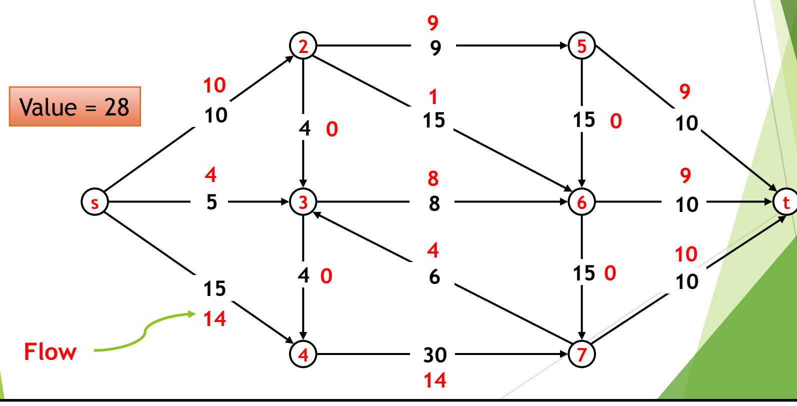
- Capacity and flow conservation constraints are satisfied
- Further increase in flow value?



51

Max flow problem

- Compute the maximum value of an s-t flow



52

Algorithms for max flow

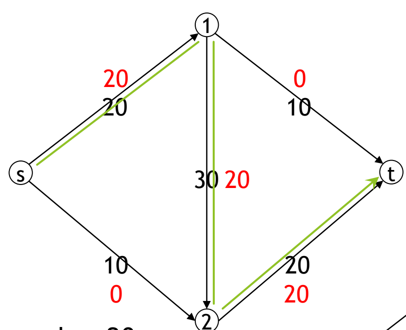
53

First try: greedy

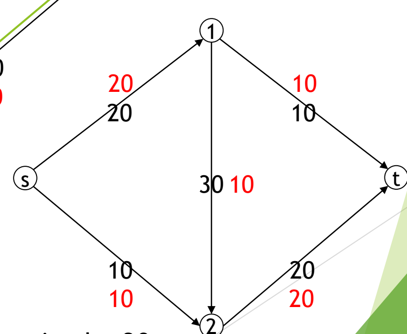
- Start with f(e) = 0 for all arcs e
- Repeat until stuck:
  - Find an s-t path where each edge has f(e) < c(e)
  - Push more flow along that path

54

Greedy doesn't work– why?



greedy = 20

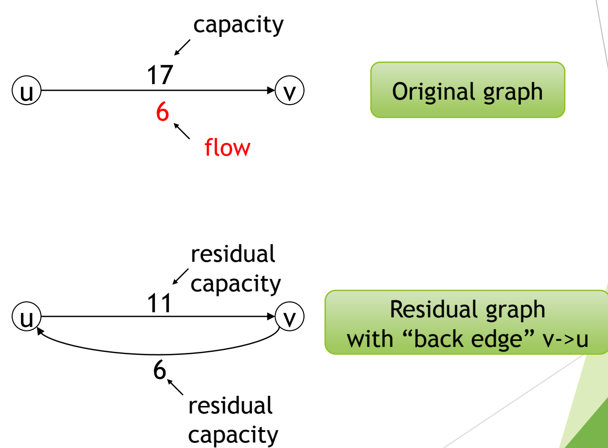


optimal = 30

55

Fix: residual graph

- A way of undoing previous flows



56

Ford-Fulkerson algorithm

- Iteratively find s-t paths that admit more flow in the residual graph
  - Such s-t paths: augmenting paths
- Push more flow along augmenting paths
- No further augmenting path?
  - Optimal solution!

57

Ford-Fulkerson Demo

58

Running time of Ford-Fulkerson

- At most nC iterations
- Total running time: O(mnC)
  - n = # of nodes
  - m = # of edges
  - C = max capacity of any edge
- Not strongly polynomial
  - There are strongly polynomial algorithms

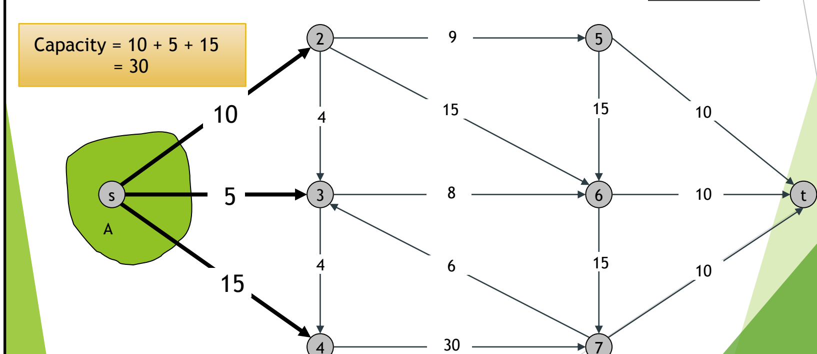
59

Min cut problem

60

s-t cut

- Partition the nodes into two sets A and B such that s is in A and t is in B
- (A, B) is called an s-t cut
- Capacity of s-t cut (A, B)
  cap(A, B) = sum of capacities of arcs out of A

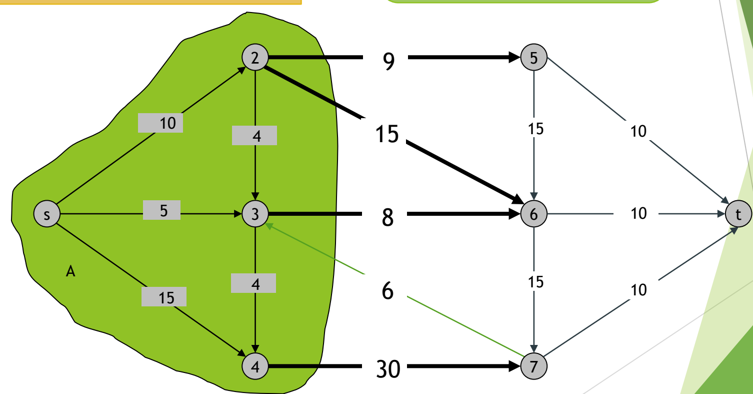


61

s-t cut: more example

Capacity = 9 + 15 + 8 + 30
= 62

Note: there's no flow here!

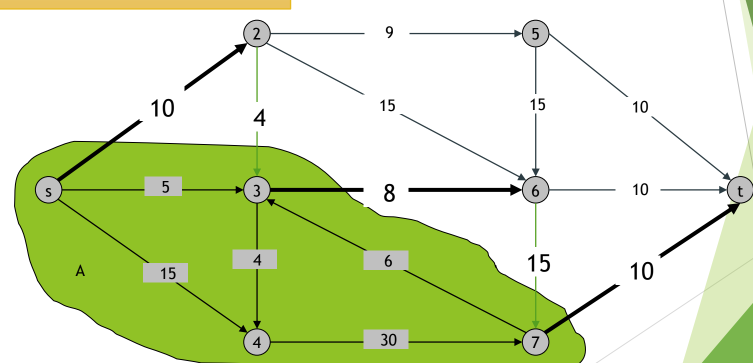


62

Min cut problem

- Find an s-t cut of minimum capacity

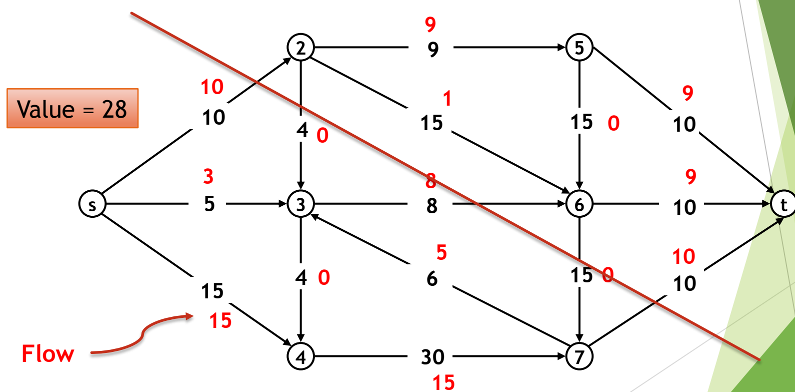
Capacity = 10 + 8 + 10
= 28



63

Max flow solution

- Max flow value is also 28!



64

Max flow vs. min cut

65

LP formulation: max flow

- Maximize $v(f) = \sum_{e \text{ out of } s} f(e)$ (flow)
- Subject to

$$0 \le f(e) \le c(e), \quad \forall e$$ (capacity)

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e), \quad \forall v \text{ except } s,t$$

Integrality theorem: if all capacities are integers, then there exists a max flow with all integer flows.

66

LP formulation: min cut

- Dual of max flow
- Weak duality: any flow <= any cut capacity
  - Proof (on board) without using LP duality
- Strong duality: max flow = min cut capacity
  - Ford-Fulkerson's proof without using LP duality

67

Max-flow min-cut theorem

- Ford & Fulkerson (1956)
- In any network, the value of the max flow is equal to the value of the min cut.

69

How to: max flow → min cut

- Want an s-t cut or partition (A, B)
- A = s and all nodes reachable from s in the final residual graph
- B = rest of the nodes

70

Algorithm for Fisher Market

Max Flow

71

Reminder: Fisher's linear case

- Model parameters (what's given)
  - *n* ***divisible goods*** (1 unit each wlog) and *n'* ***buyers***
  - e_i = buyer *i*'s budget (integral wlog)
  - u_{ij} = buyer *i*'s utility per unit of good *j* (integral wlog)
  - *Linear* utility functions
- Want (not given): equilibrium allocations
  - x_{ij} = amount of good *j* that *i* buys to maximize his/her utility $u_i(\mathbf{x}) = \sum_{j=1}^{n} u_{ij} x_{ij}$
- Want (not given): equilibrium prices $p_1, p_2, \ldots, p_n$
- No deficit or surplus of any good
- No deficit or surplus of buyers' budgets

72

Reminder: KKT conditions of Eisenberg-Gale convex program

- Optimal solutions x_{ij}'s and p_j's must satisfy:

1. $\forall j \in A: \; p_j \geq 0.$
2. $\forall j \in A: \; p_j > 0 \Rightarrow \sum_{i \in A} x_{ij} = 1.$
3. $\forall i \in B, \forall j \in A: \frac{u_{ij}}{p_j} \leq \frac{\sum_{j \in A} u_{ij} x_{ij}}{e_i}.$
4. $\forall i \in B, \forall j \in A: x_{ij} > 0 \Rightarrow \frac{u_{ij}}{p_j} = \frac{\sum_{j \in A} u_{ij} x_{ij}}{e_i}.$

- No deficit or surplus of goods
- Can show no deficit or surplus of buyers' budgets

73

Idea of the algorithm

- Look at individual optimization problem
- Buyer i's optimization program:

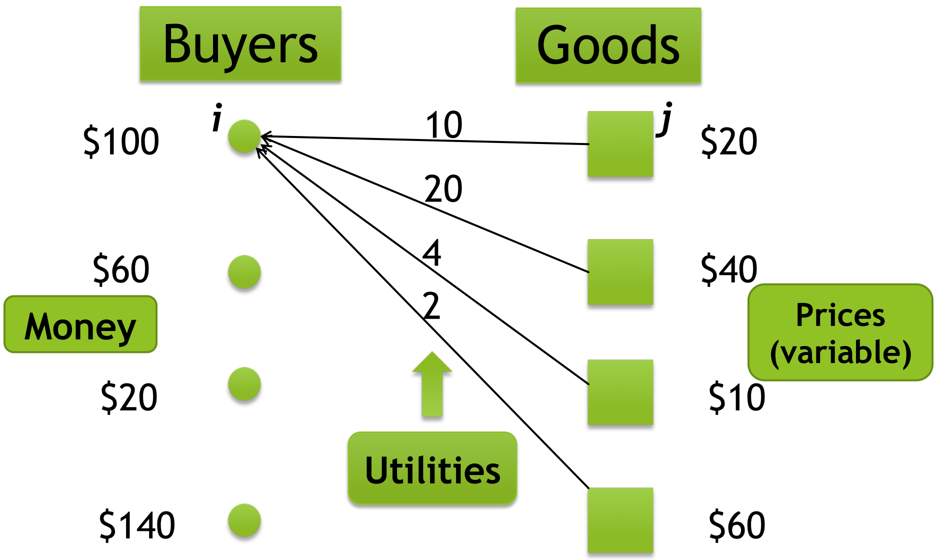
$$\max \sum_j u_{ij} x_{ij}$$
$$\text{s.t. } \sum_j p_j x_{ij} \leq e_i$$

- Global constraint:

$$\forall j \;\; \sum_i x_{ij} = 1$$

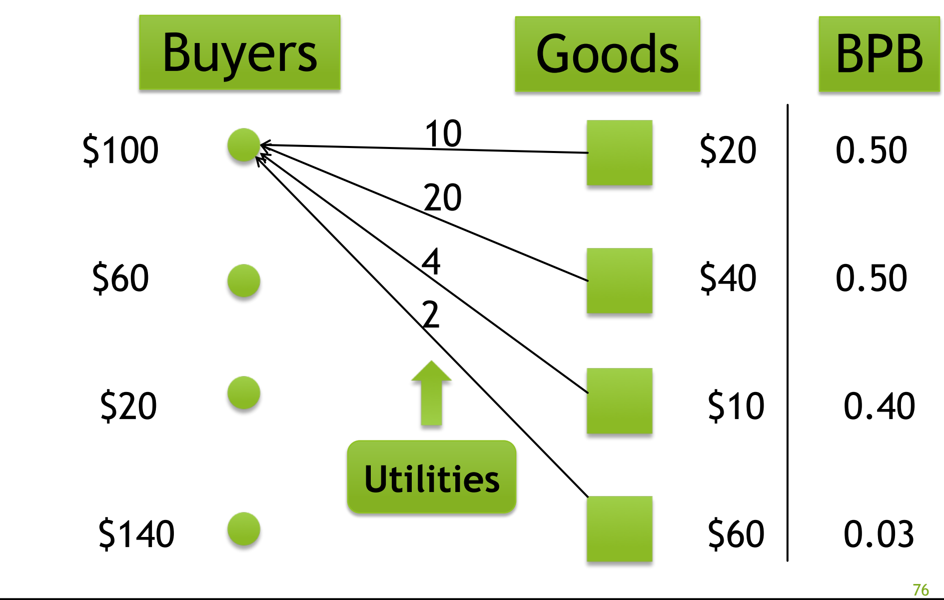
74

Example

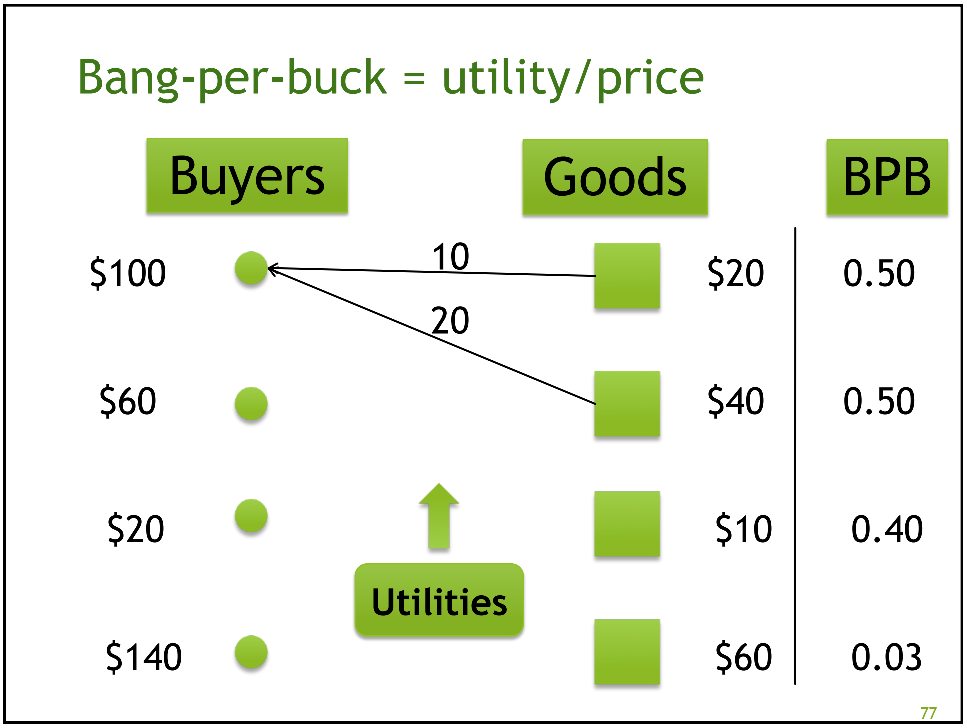


75

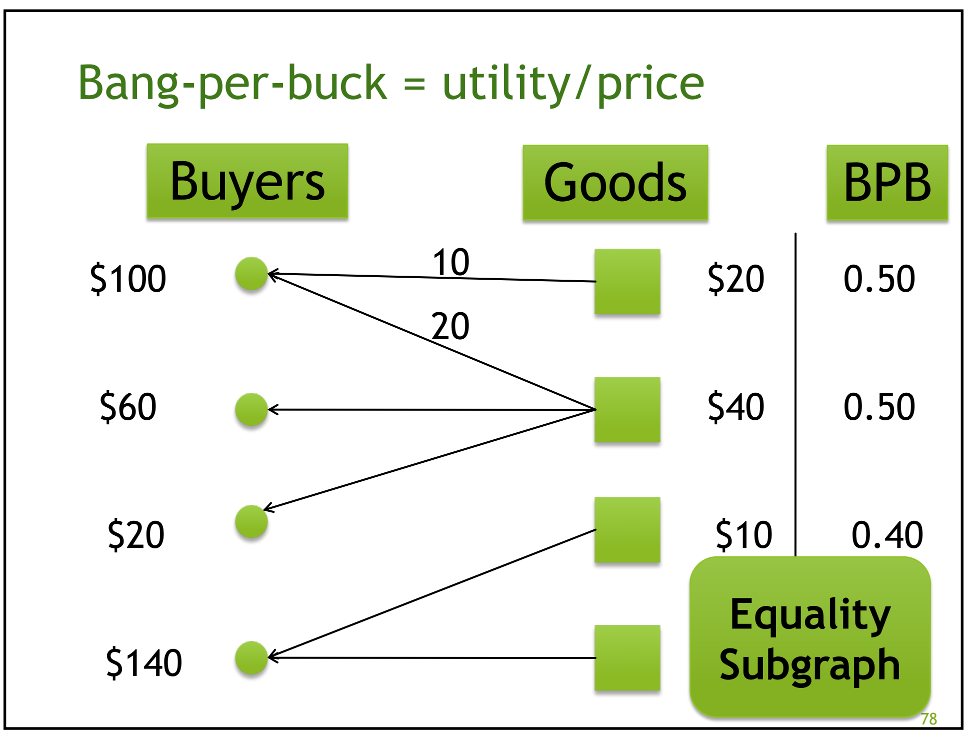
Bang-per-buck = utility/price



76


Bang-per-buck = utility/price
Buyers
Goods
BPB
$100
$60
$20
$140
10
20
Utilities
$20
$40
$10
$60
0.50
0.50
0.40
0.03
77

77


Bang-per-buck = utility/price
Buyers
Goods
BPB
$100
$60
$20
$140
10
20
$20
$40
$10
0.50
0.50
0.40
Equality Subgraph
78

78

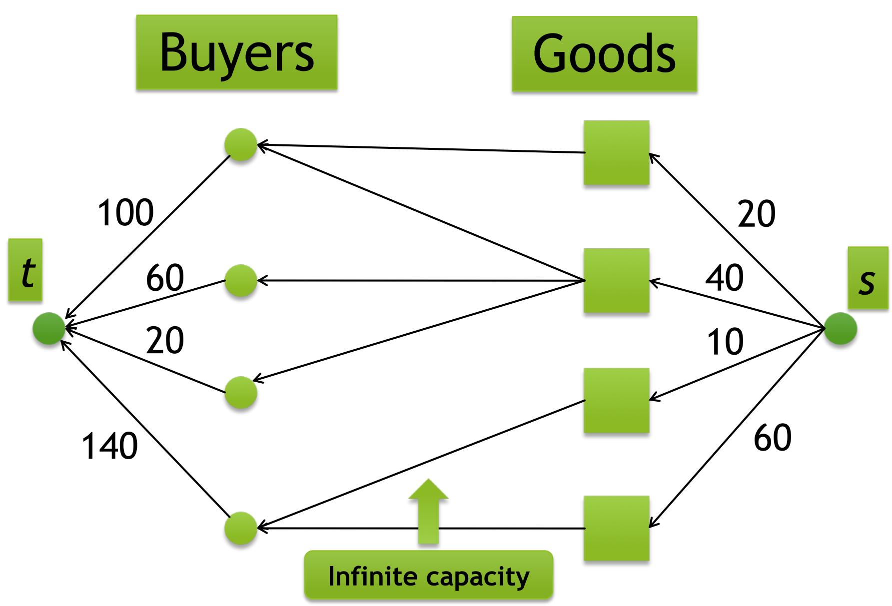
Equality subgraph

- Buyer is happiest when she can buy goods in equality subgraph
- How to maximize sales (market clearance) in the equality subgraph at a given price?

Use max flow!
("balanced flow" here)

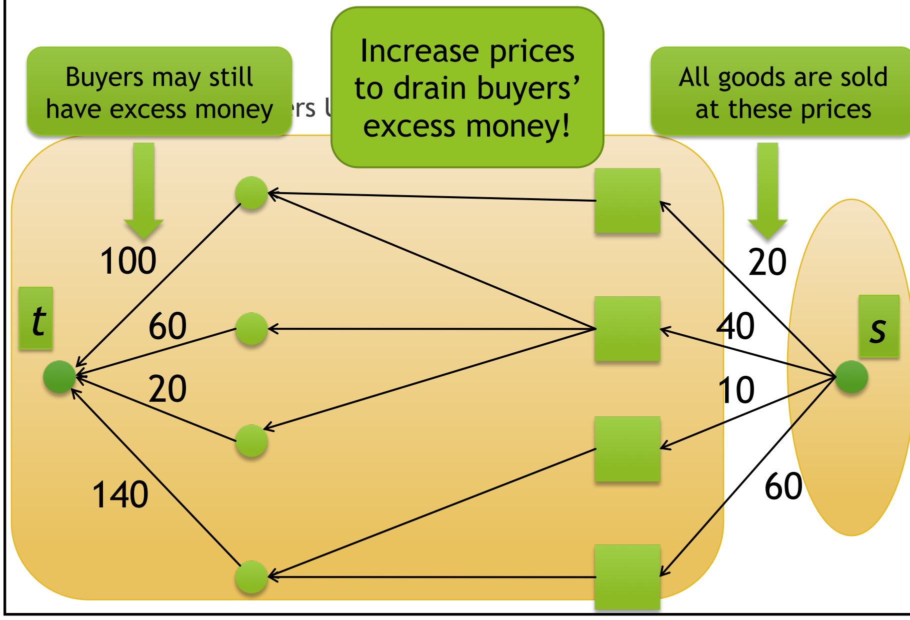
79

Max flow



80

Idea of the algorithm



81

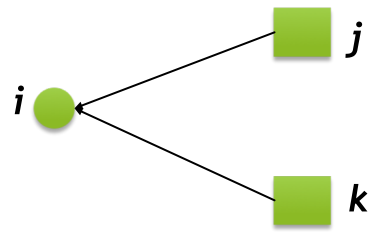
Initialize

- All prices = 1/n, n = # of goods
- Assume
  - Each buyer has integral amount of money

82

How to raise prices?

- We do not want to kill off any edge from the equality subgraph



$$\frac{u_{ij}}{p_j} = \frac{u_{ik}}{p_k}$$

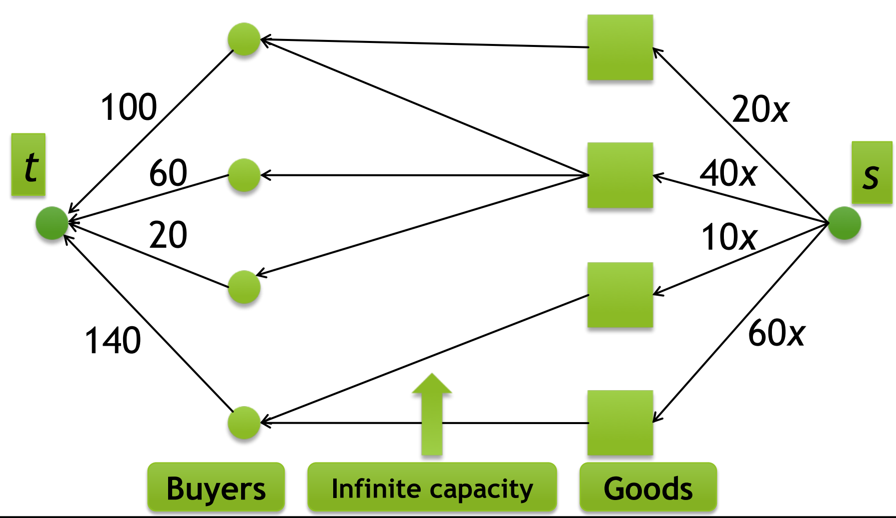
$$\Rightarrow \frac{p_k}{p_j} = \frac{u_{ik}}{u_{ij}}$$

- Multiply prices by the same number ***x***
  - Initially *x* = 1

83

Algorithm

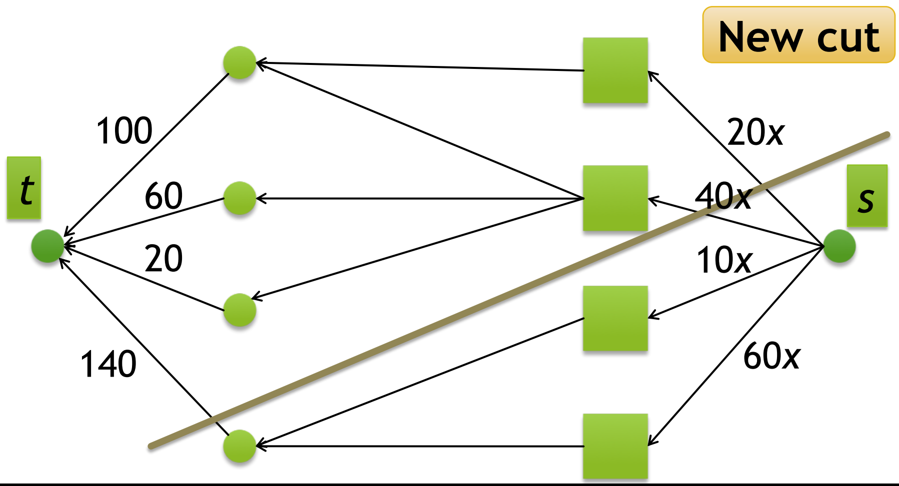
- Initially *x* = 1, then *x*↑
  - How to increase *x*? (Later)



84

Algorithm

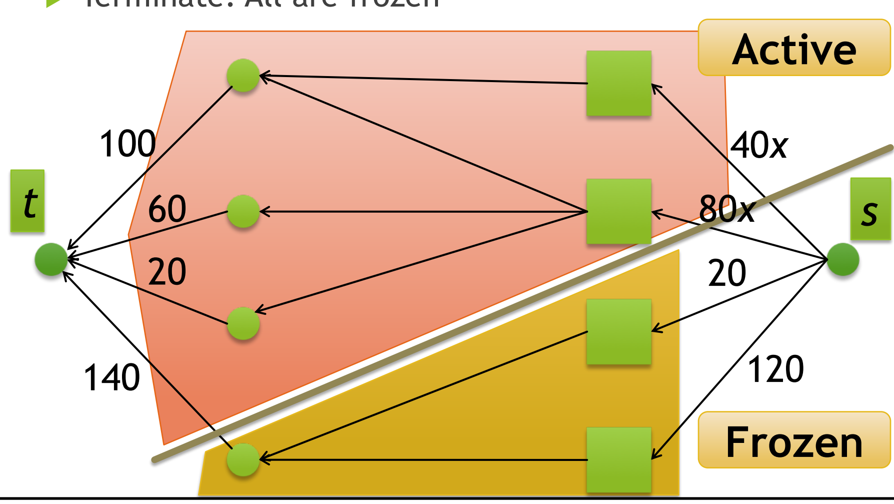
- x = 2: A buyer's surplus money = 0 (Event #1)



85

Algorithm

- Reinitialize x = 1, work the active part
- Terminate: All are frozen

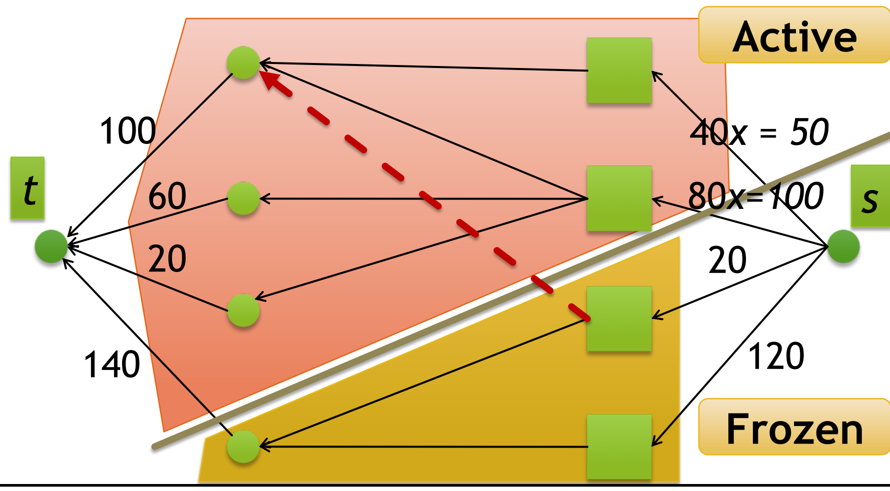


86

Algorithm

- x = 1.25 (Event #2)

Note: In the original algorithm, the active buyers are partitioned into sets: max-surplus buyers and the rest (not shown here for presentation, but x is relevant to the max-surplus buyers).



87

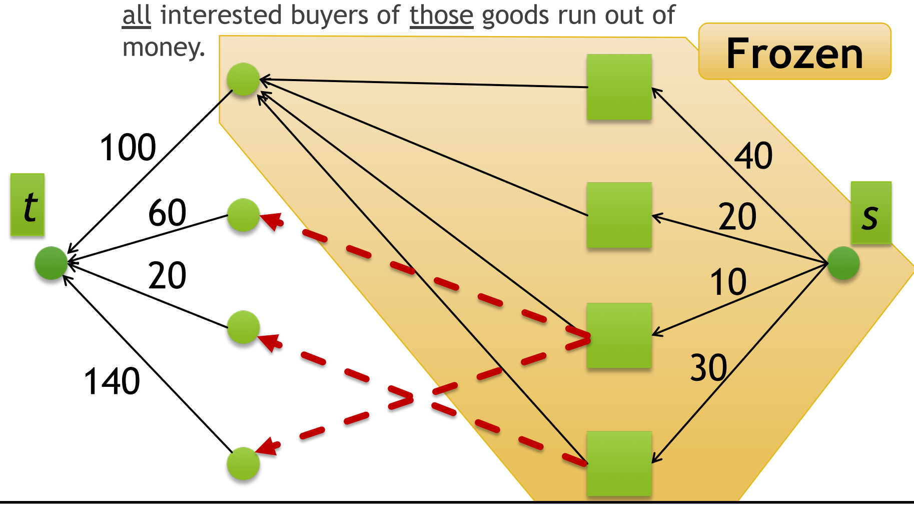
Algorithm

- In case of event #2:"Unfreeze" the relevant part and recurse

88

Question

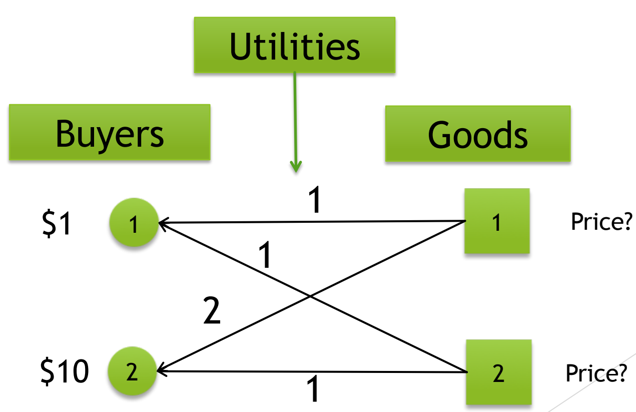
- Is this scenario possible?
  - No. Every buyer must have some goods that maximize BPB. Can't freeze a set of goods unless all interested buyers of those goods run out of money.



89

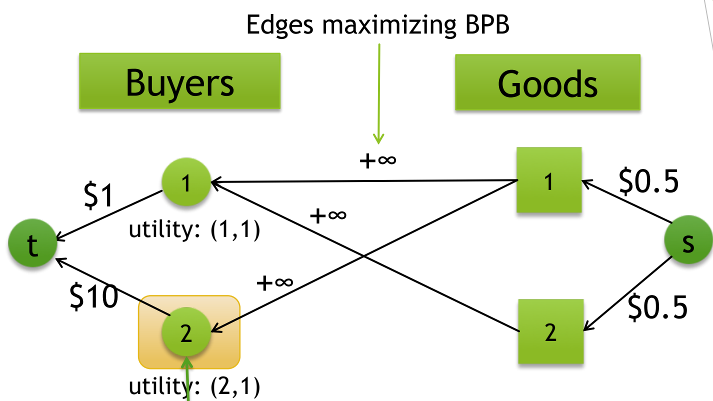
More example

- 2 buyers, 2 goods



90

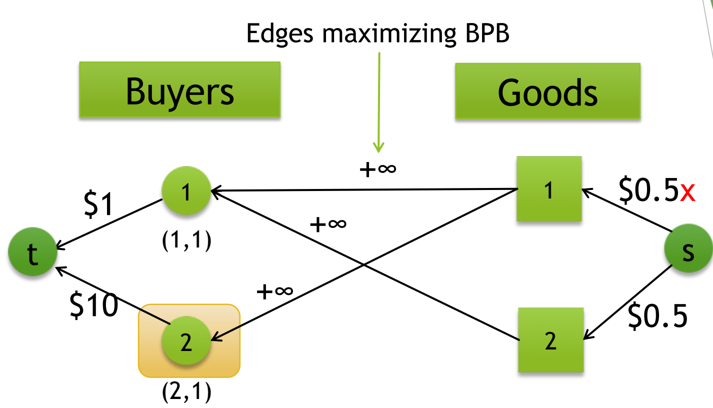
Example (continued)



- Initialize all prices to ½
- Compute a "balanced flow" (not shown here)
- Find the buyer(s) with the max surplus money
- Nothing frozen: No subset of goods for which all interested buyers run out of money

91

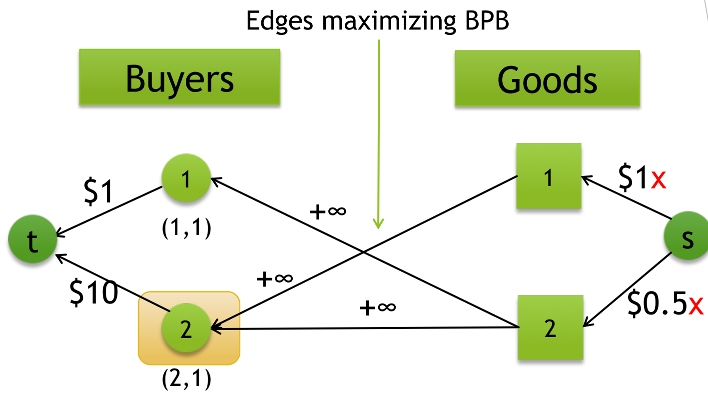
Example (continued)



- Increase prices of goods that the max-surplus buyers are interested in
- x > 0 ➔ edge from 1 to 1 disappears
- x = 2 ➔ New edge (event 2) from 2 to 2

92

Example (continued)



- Increase prices of goods that the max-surplus buyers are interested in
- x = 7⅓ ➔ All goods frozen ➔ equilibrium
  - Note: all buyers run out of money

93

Running time

- We can compute x values at event #1 and #2 efficiently
- Balanced flow is polynomial, but not strongly polynomial
  - Running time depends on the amount of money each buyer has
  - $O(n^4 (\log n + n\log U + \log M))$ applications of max-flow
    - n = # of goods
    - U = max u_{ij} for any i and j
    - M = total amount of money of all buyers

94