CSEP 521
Applied Algorithms

Richard Anderson

Lecture 9

Network Flow Applications

Announcements

- Reading for this week
  - 7.5-7.12. Network flow applications
  - Next week: Chapter 8. NP-Completeness
- Final exam, March 18, 6:30 pm. At UW.
  - 2 hours
  - In class (CSE 303 / CSE 305)
  - Comprehensive
    - 67% post midterm, 33% pre midterm

Network Flow

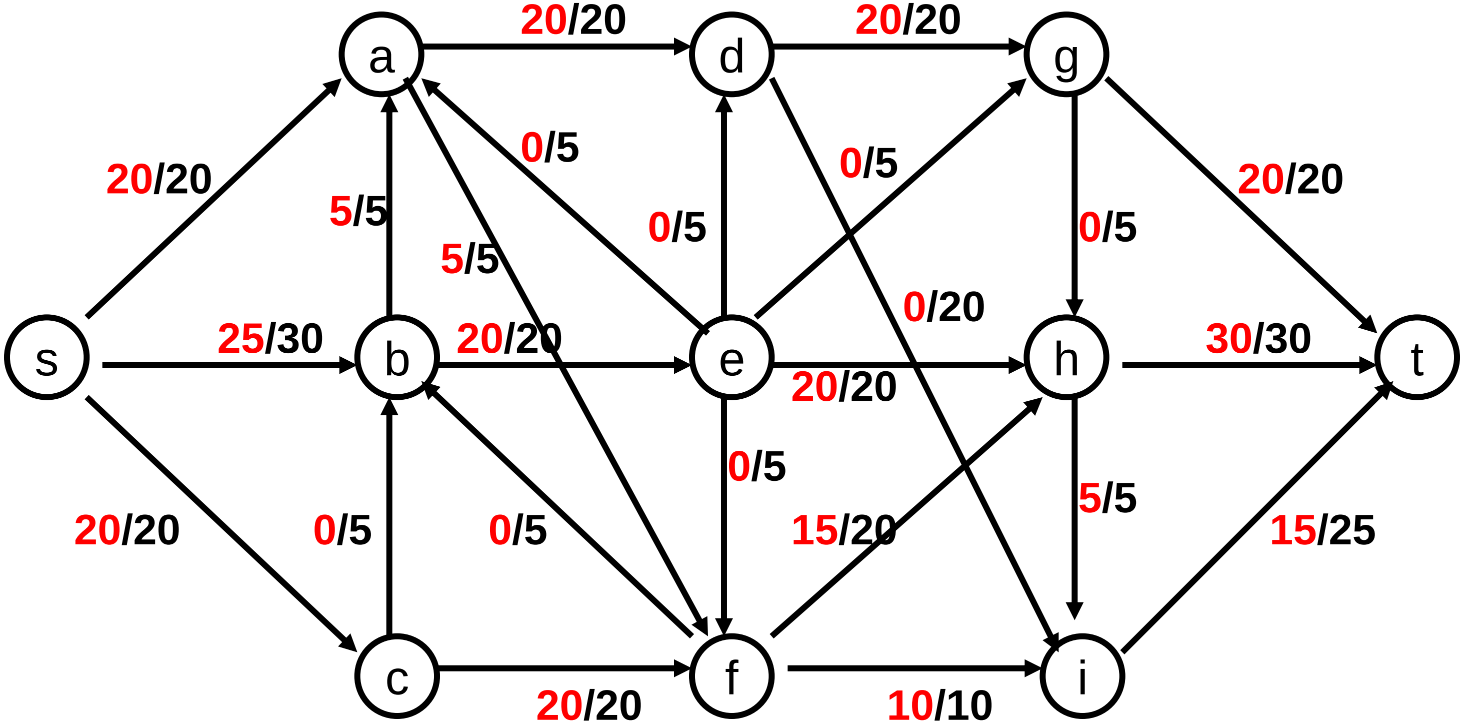
Review

- Network flow definitions
- Flow examples
- Augmenting Paths
- Residual Graph
- Ford Fulkerson Algorithm
- Cuts
- Maxflow-MinCut Theorem

Network Flow Definitions

- Flowgraph: Directed graph with distinguished vertices s (source) and t (sink)
- Capacities on the edges, c(e) >= 0
- Problem, assign flows f(e) to the edges such that:
  - 0 <= f(e) <= c(e)
  - Flow is conserved at vertices other than s and t
    - Flow conservation: flow going into a vertex equals the flow going out
  - The flow leaving the source is a large as possible

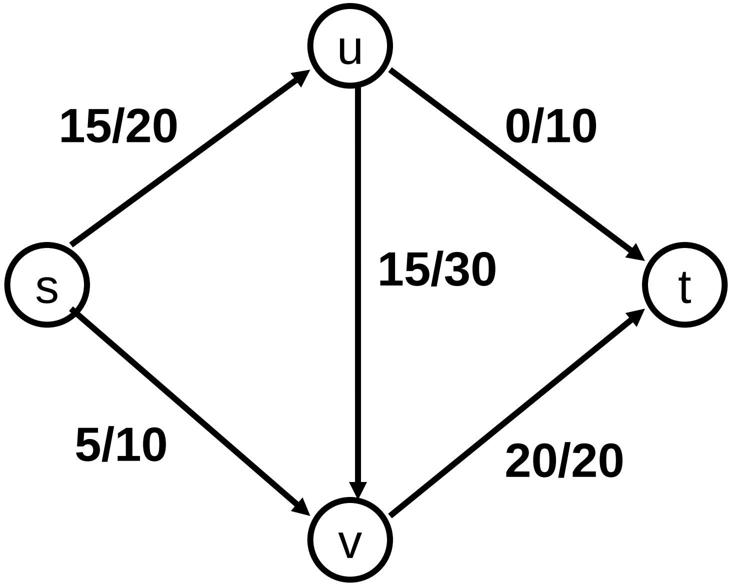
Find a maximum flow

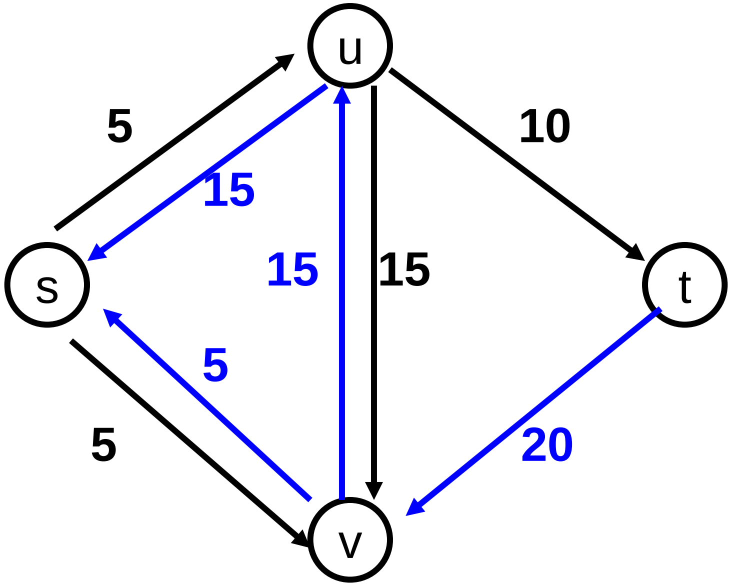


Residual Graph

- Flow graph showing the remaining capacity
- Flow graph G, Residual Graph G_R
  - – G: edge e from u to v with capacity c and flow f
  - – G_R: edge e' from u to v with capacity c – f
  - – G_R: edge e'' from v to u with capacity f

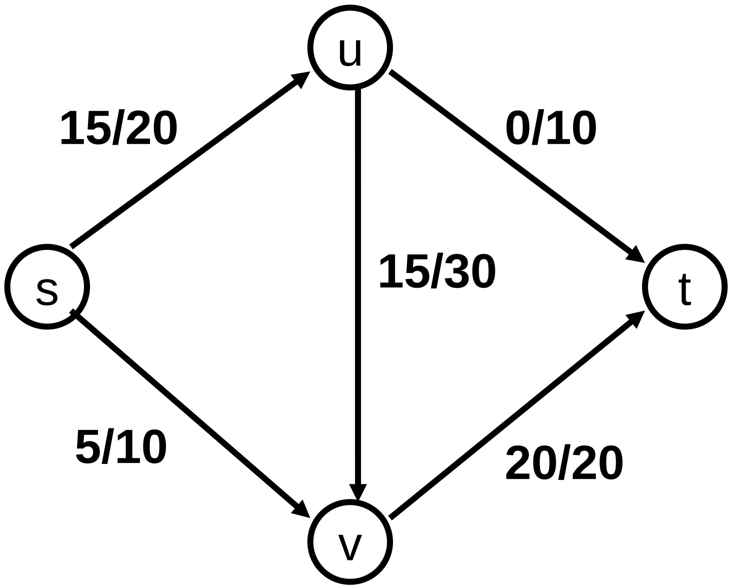
Residual Graph

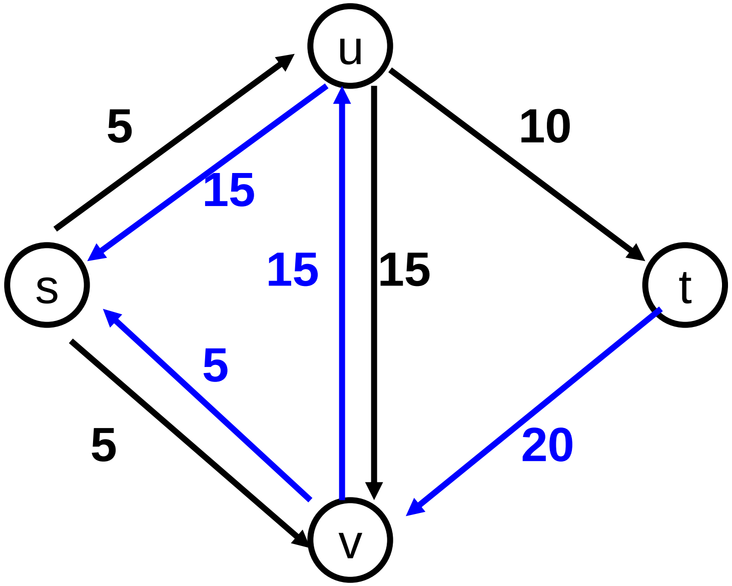




Augmenting Path Lemma

- Let P = $v_1, v_2, \ldots, v_k$ be a path from s to t with minimum capacity b in the residual graph.
- b units of flow can be added along the path P in the flow graph.





Ford-Fulkerson Algorithm (1956)

while not done

Construct residual graph G_R

Find an s-t path P in G_R with capacity $b > 0$

Add b units along in G

If the sum of the capacities of edges leaving S is at most C, then the algorithm takes at most C iterations

Cuts in a graph

- Cut: Partition of V into disjoint sets S, T with s in S and t in T.
- Cap(S,T): sum of the capacities of edges from S to T
- Flow(S,T): net flow out of S
  – Sum of flows out of S minus sum of flows into S

- Flow(S,T) <= Cap(S,T)

Ford Fulkerson MaxFlow – MinCut Theorem

- There exists a flow which has the same value of the minimum cut
  - Shows that a cut is the dual of the flow
  - Proves that the augmenting paths algorithm finds a maximum flow
  - Gives an algorithms for finding the minimum cut

Better methods of for constructing a network flow

- Improved methods for finding augmenting paths or blocking flows
- Goldberg's Preflow-Push algorithm
  - Text, section 7.4

Applications of Network Flow

Problem Reduction

- Reduce Problem A to Problem B
  - Convert an instance of Problem A to an instance of Problem B
  - Use a solution of Problem B to get a solution to Problem A
- Practical
  - Use a program for Problem B to solve Problem A
- Theoretical
  - Show that Problem B is at least as hard as Problem A

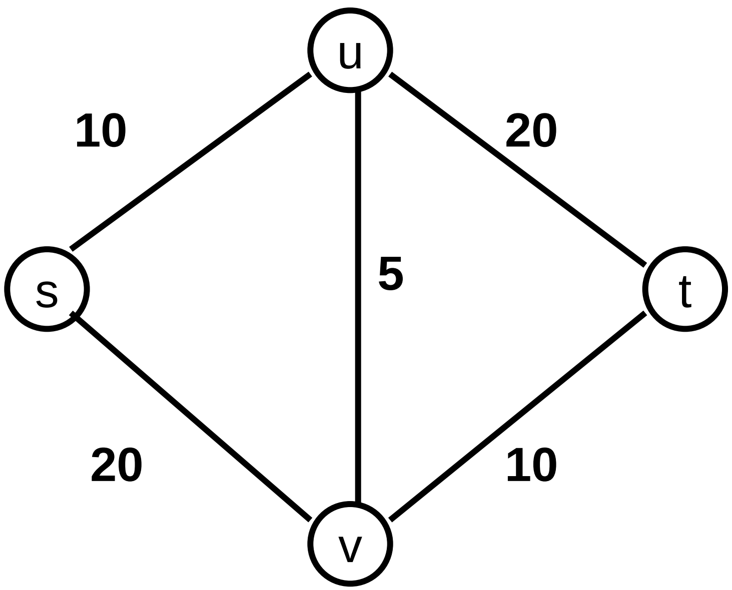
Problem Reduction Examples

- Reduce the problem of finding the path in a directed graph to the problem of finding a shortest path in a directed graph

Construct an equivalent minimization problem

Undirected Network Flow

- Undirected graph with edge capacities
- Flow may go either direction along the edges (subject to the capacity constraints)



Construct an equivalent flow problem

Multi-source network flow

- Multi-source network flow
  - Sources $s_1, s_2, . . ., s_k$
  - Sinks $t_1, t_2, . . . , t_j$
- Solve with Single source network flow

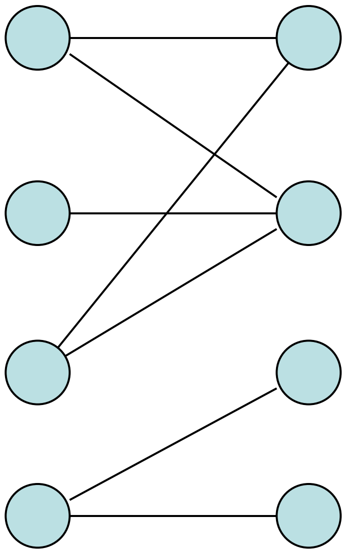
Bipartite Matching

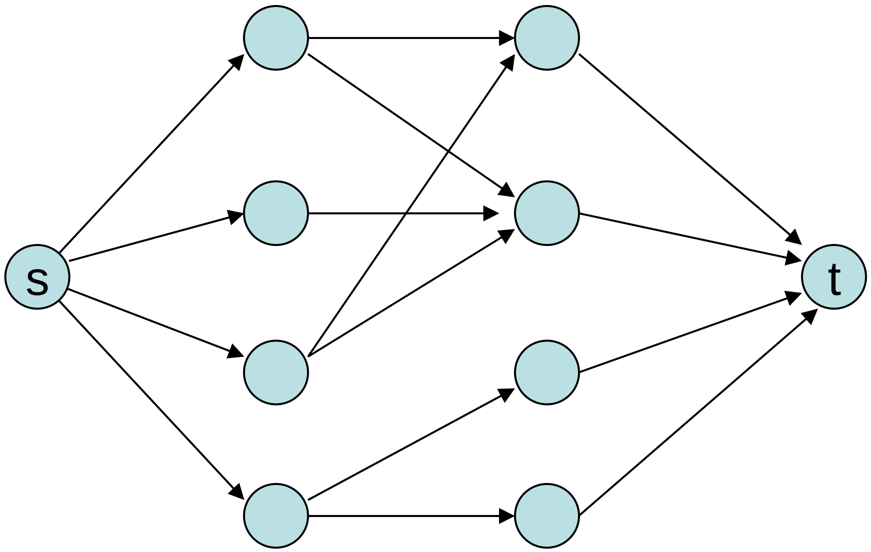
- A graph G=(V,E) is bipartite if the vertices can be partitioned into disjoints sets X,Y
- A matching M is a subset of the edges that does not share any vertices
- Find a matching as large as possible

Application

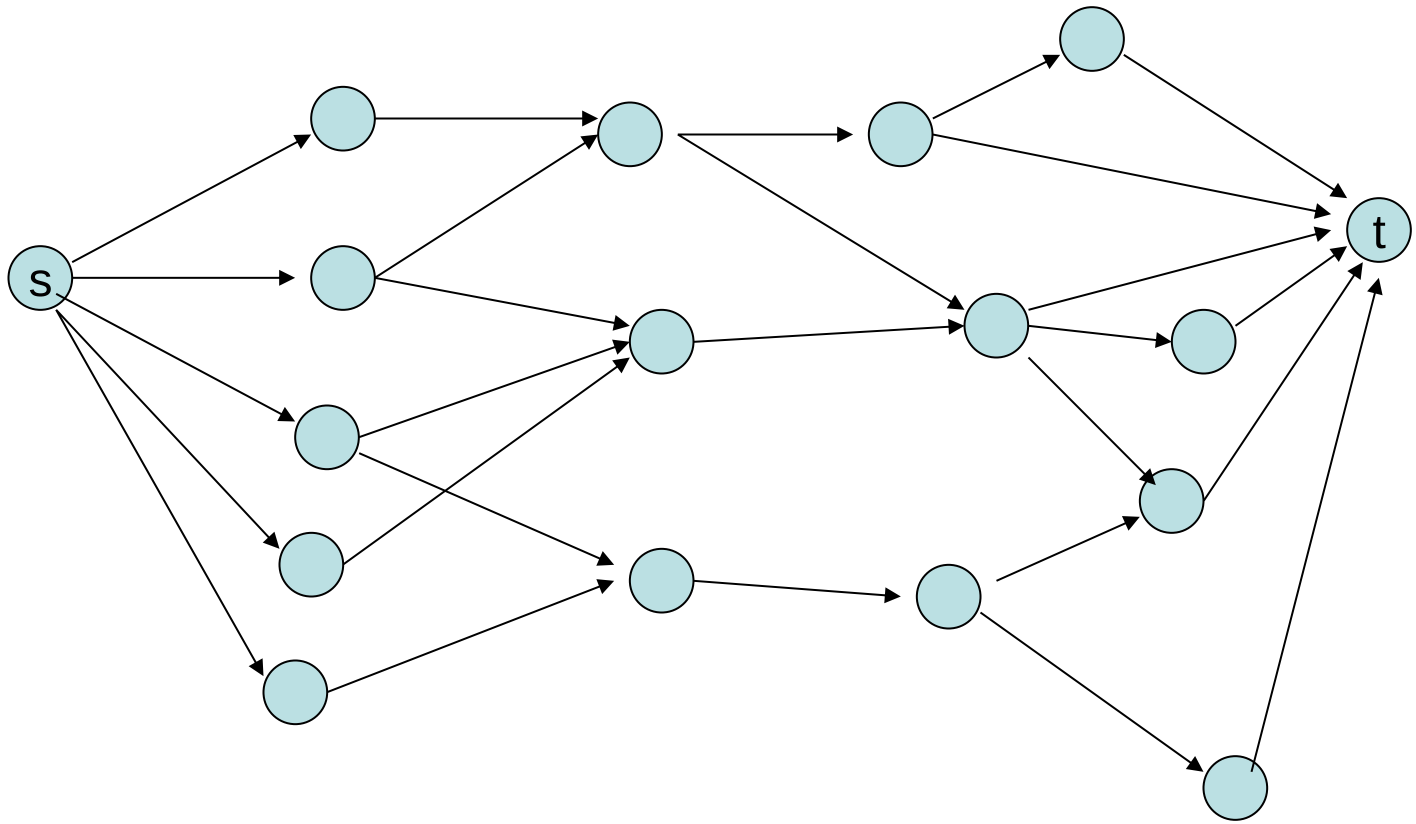
- A collection of teachers
- A collection of courses
- And a graph showing which teachers can teach which courses

Converting Matching to Network Flow





Finding edge disjoint paths

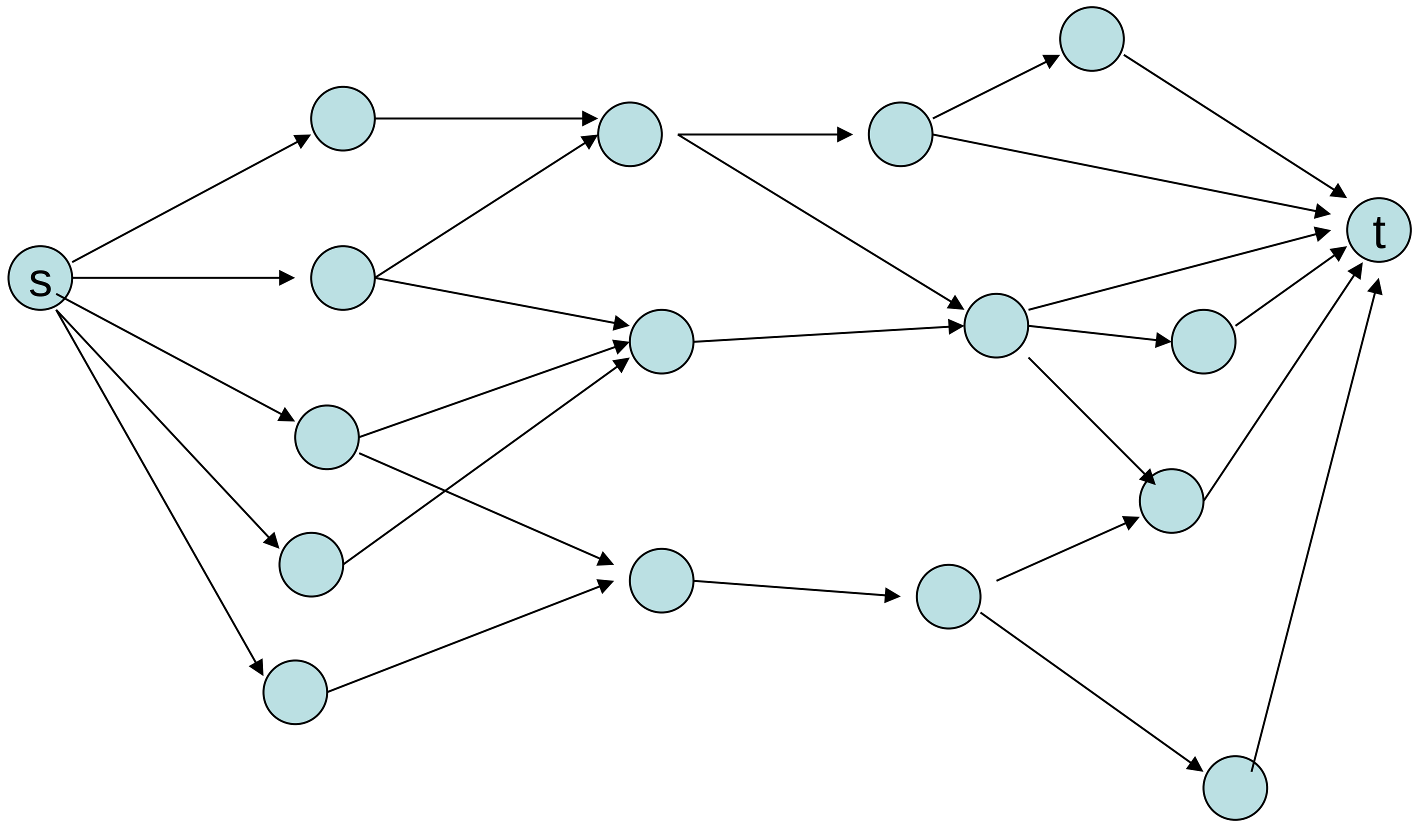


Construct a maximum cardinality set of edge disjoint paths

Theorem

- The maximum number of edge disjoint paths equals the minimum number of edges whose removal separates s from t

Finding vertex disjoint paths



Construct a maximum cardinality set of vertiex disjoint paths

Network flow with vertex capacities

Balanced allocation
Problem 9, Page 419

- To make a long story short:
  - N injured people
  - K hospitals
  - Assign each person to a hospital with 30 minutes drive
  - Assign N/K patients to each hospital

Baseball elimination

- Can the Dinosaurs win the league?
- Remaining games:
  - AB, AC, AD, AD, AD, BC, BC, BC, BD, CD

	W	L
Ants	4	2
Bees	4	2
Cockroaches	3	3
Dinosaurs	1	5

A team wins the league if it has strictly more wins than any other team at the end of the season
A team ties for first place if no team has more wins, and there is some other team with the same number of wins

Baseball elimination

- Can the Fruit Flies win or tie the league?
- Remaining games:
  - AC, AD, AD, AD, AF, BC, BC, BC, BC, BC, BD, BE, BE, BE, BE, BF, CE, CE, CE, CF, CF, DE, DF, EF, EF

	W	L
Ants	17	12
Bees	16	7
Cockroaches	16	7
Dinosaurs	14	13
Earthworms	14	10
Fruit Flies	12	15

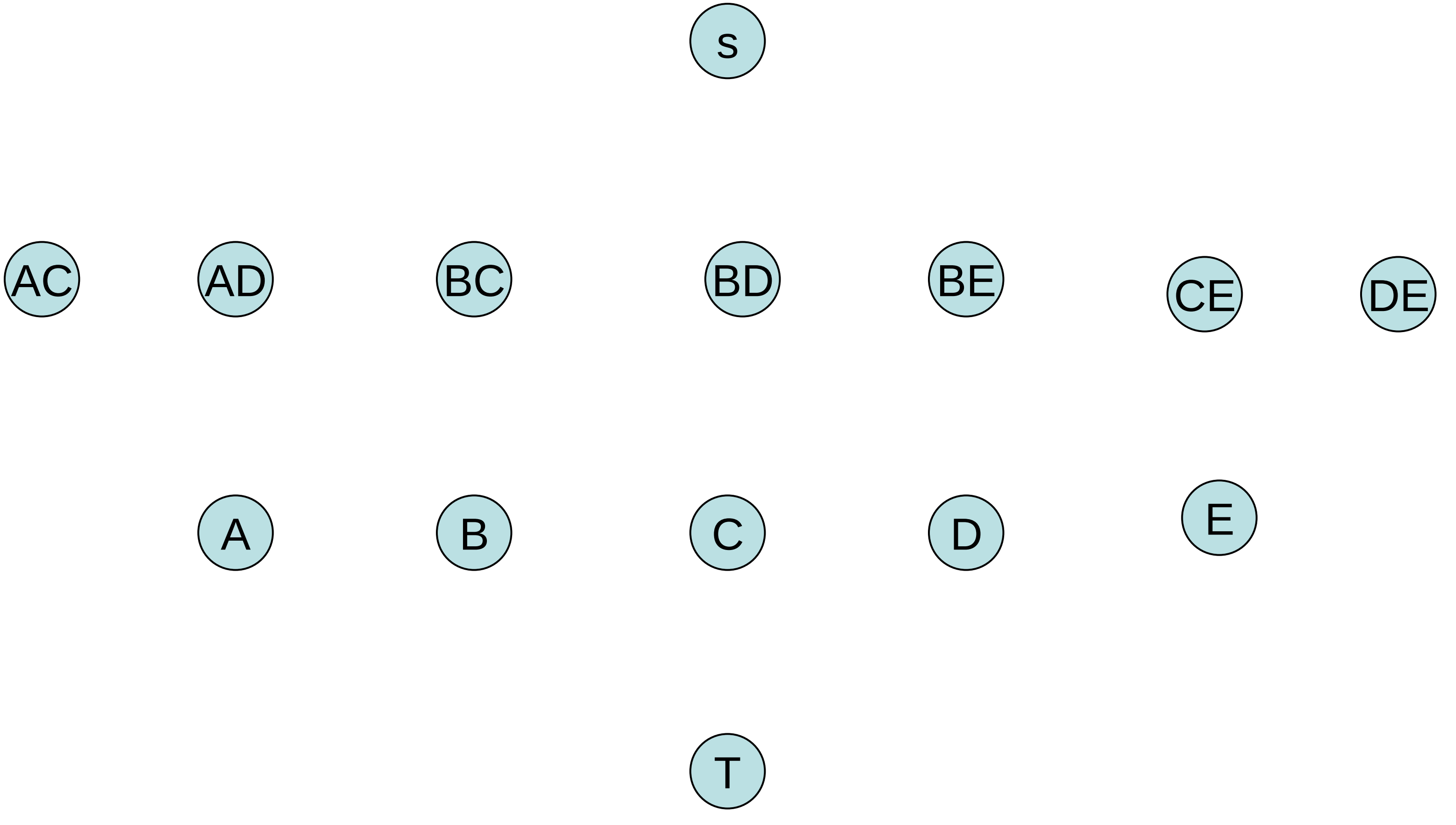
Assume Fruit Flies win remaining games

- Fruit Flies are tied for first place if no team wins more than 19 games
- Allowable wins
  - Ants (2)
  - Bees (3)
  - Cockroaches (3)
  - Dinosaurs (5)
  - Earthworms (5)
- 18 games to play
  - AC, AD, AD, AD, BC, BC, BC, BC, BC, BD, BE, BE, BE, BE, CE, CE, CE, DE

	W	L
Ants	17	13
Bees	16	8
Cockroaches	16	9
Dinosaurs	14	14
Earthworms	14	12
Fruit Flies	19	15

Remaining games

AC, AD, AD, AD, BC, BC, BC, BC, BC, BD, BE, BE, BE, BE, CE, CE, CE, DE



Solving problems with a minimum cut

- Image Segmentation
- Open Pit Mining / Task Selection Problem

S, T is a cut if S, T is a partition of the vertices with
s in S and t in T
The capacity of an S, T cut is the sum of the capacities of all edges going from S to T

Image Segmentation

- Separate foreground from background
- Reduction to min-cut problem

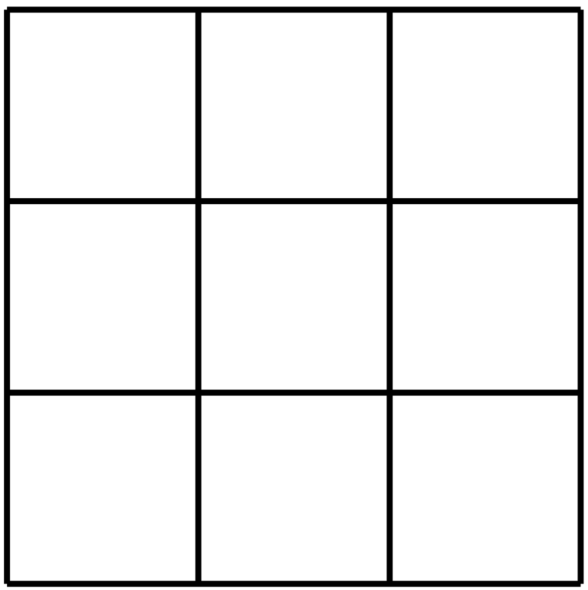
S, T is a cut if S, T is a partition of the vertices with
s in S and t in T

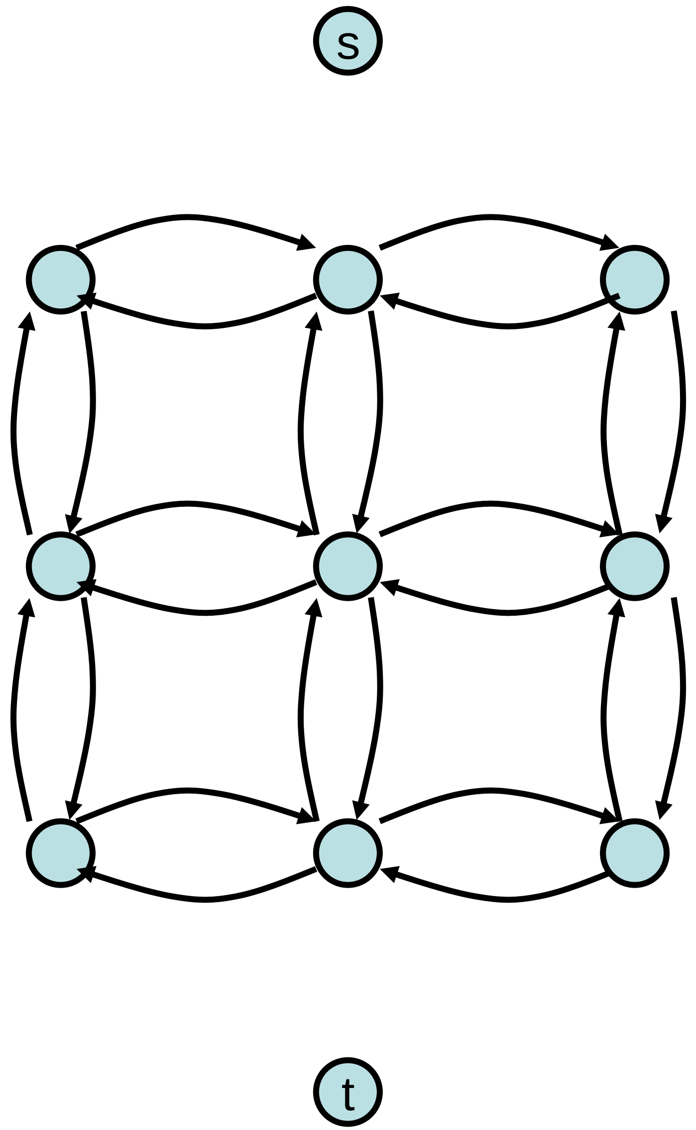
The capacity of an S, T cut is the sum of the capacities of all edges going from S to T

Image analysis

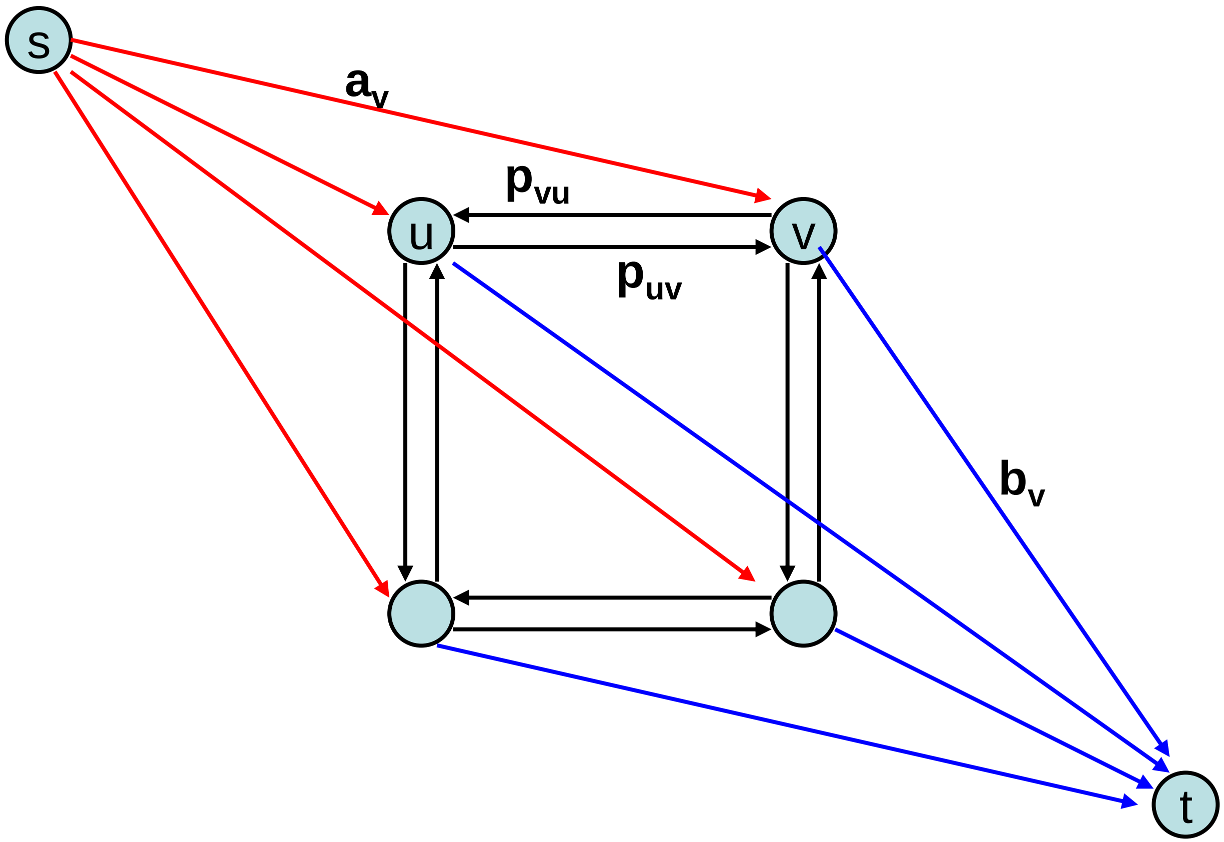
- a_i: value of assigning pixel i to the foreground
- b_i: value of assigning pixel i to the background
- p_{ij}: penalty for assigning i to the foreground, j to the background or vice versa
- A: foreground, B: background
- $Q(A,B) = \Sigma_{\{i \text{ in } A\}} a_i + \Sigma_{\{j \text{ in } B\}} b_j - \Sigma_{\{(i,j) \text{ in } E, \ i \text{ in } A, \ j \text{ in } B\}} p_{ij}$

Pixel graph to flow graph





Mincut Construction



Open Pit Mining

Application of Min-cut

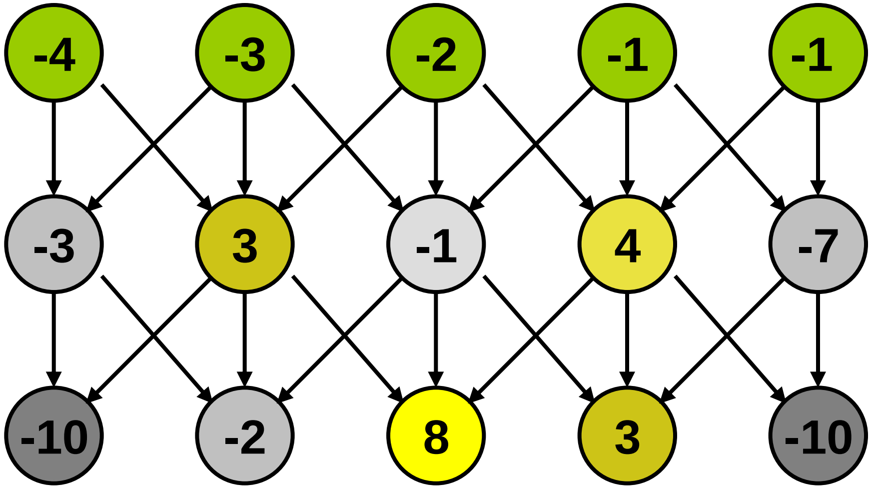
- Open Pit Mining Problem
- Task Selection Problem
- Reduction to Min Cut problem

S, T is a cut if S, T is a partition of the vertices with
s in S and t in T
The capacity of an S, T cut is the sum of the capacities of
all edges going from S to T

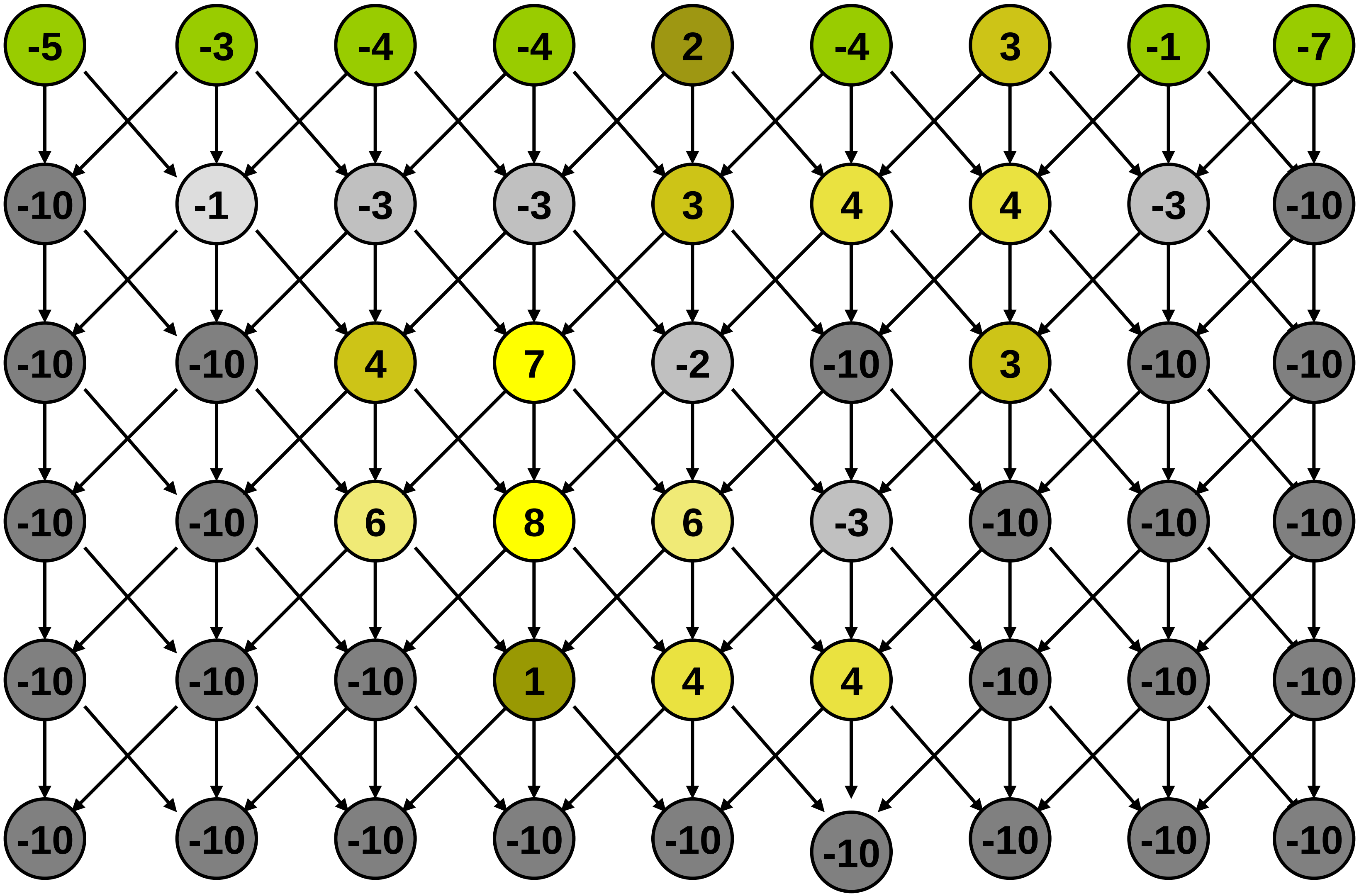
Open Pit Mining

- Each unit of earth has a profit (possibly negative)
- Getting to the ore below the surface requires removing the dirt above
- Test drilling gives reasonable estimates of costs
- Plan an optimal mining operation

Mine Graph

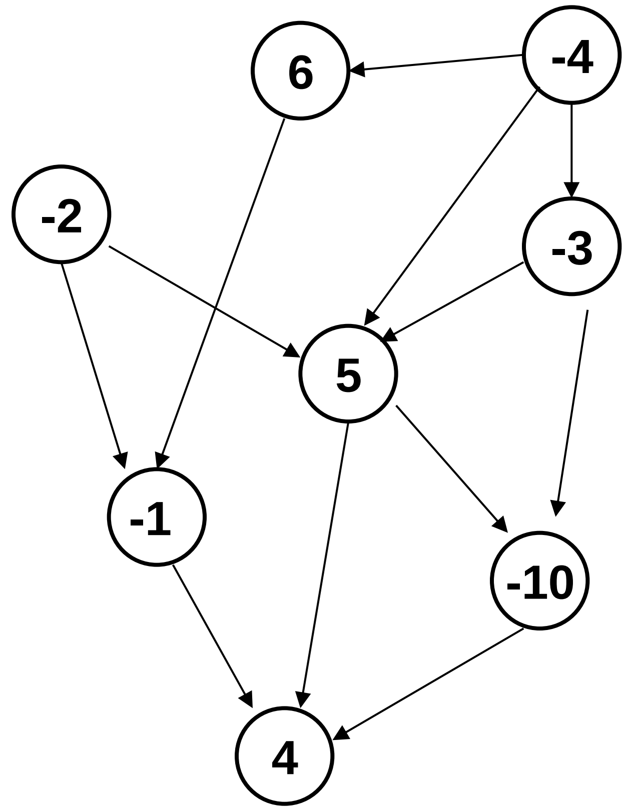


Determine an optimal mine



Generalization

- Precedence graph G=(V,E)
- Each v in V has a profit p(v)
- A set F if *feasible* if when w in F, and (v,w) in E, then v in F.
- Find a feasible set to maximize the profit

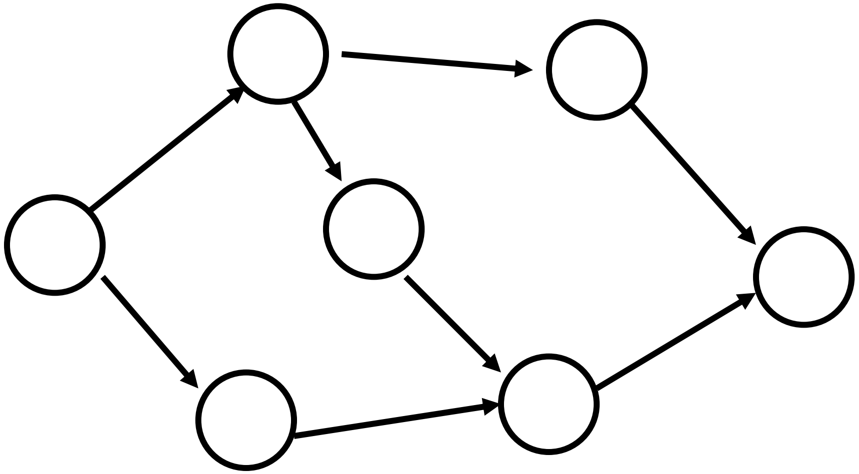


Min cut algorithm for profit maximization

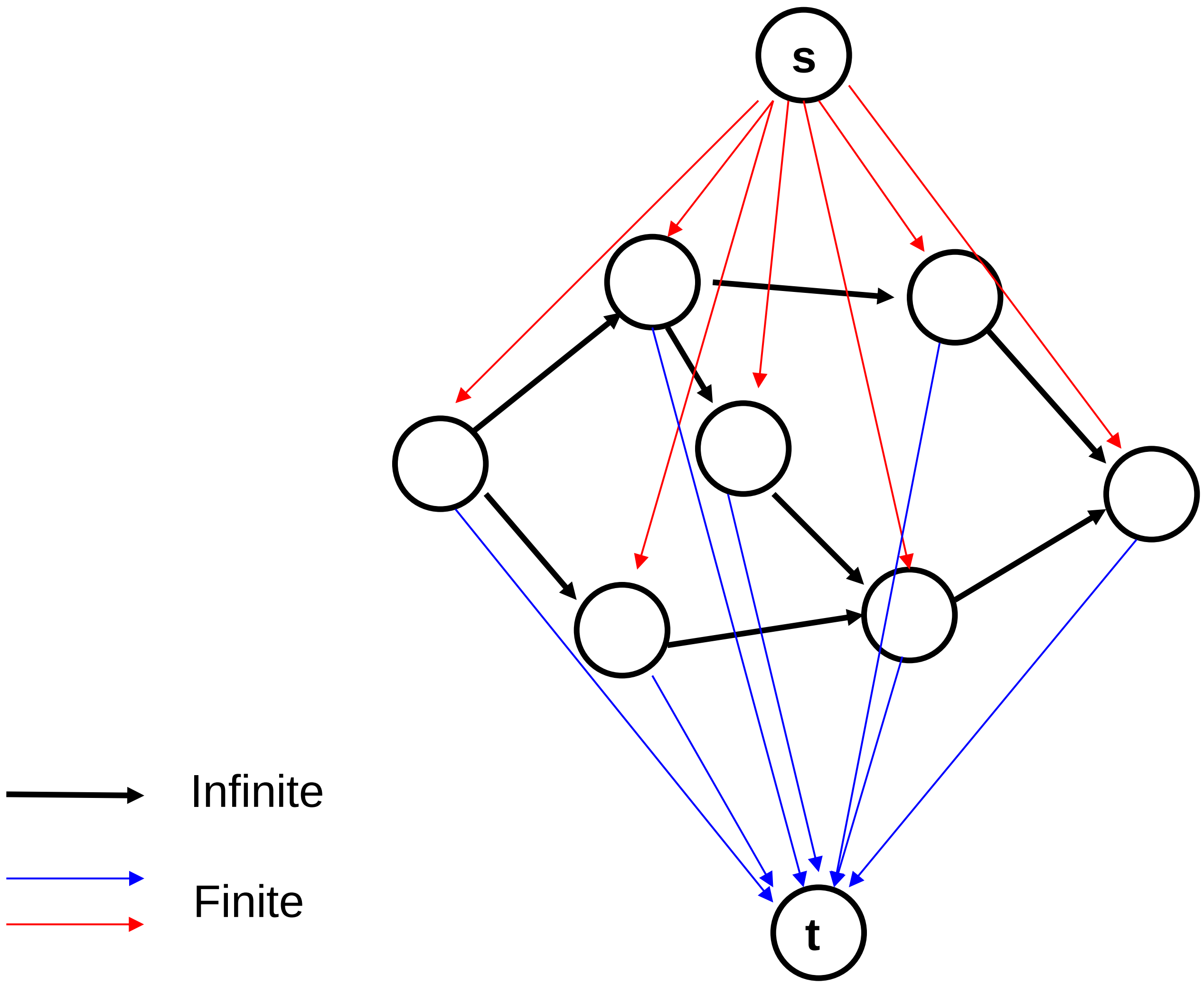
- Construct a flow graph where the minimum cut identifies a feasible set that maximizes profit

Precedence graph construction

- Precedence graph G=(V,E)
- Each edge in E has infinite capacity
- Add vertices s, t
- Each vertex in V is attached to s and t with finite capacity edges

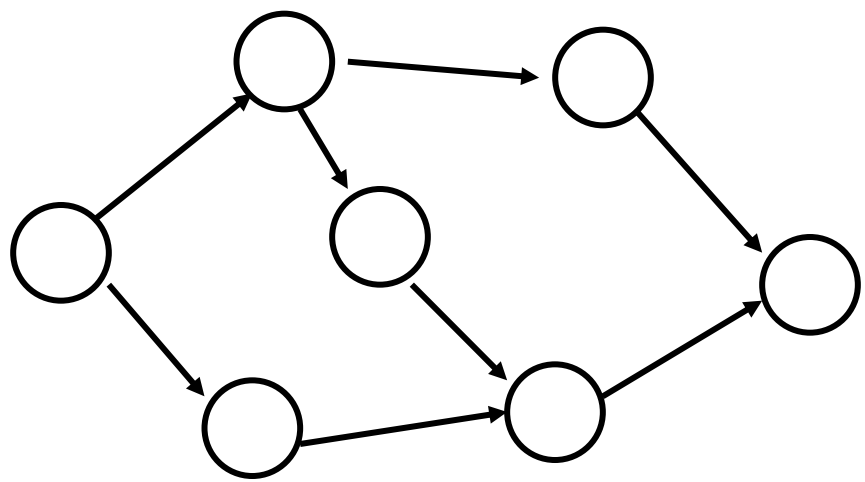


Show a finite value cut with at least two vertices on each side of the cut



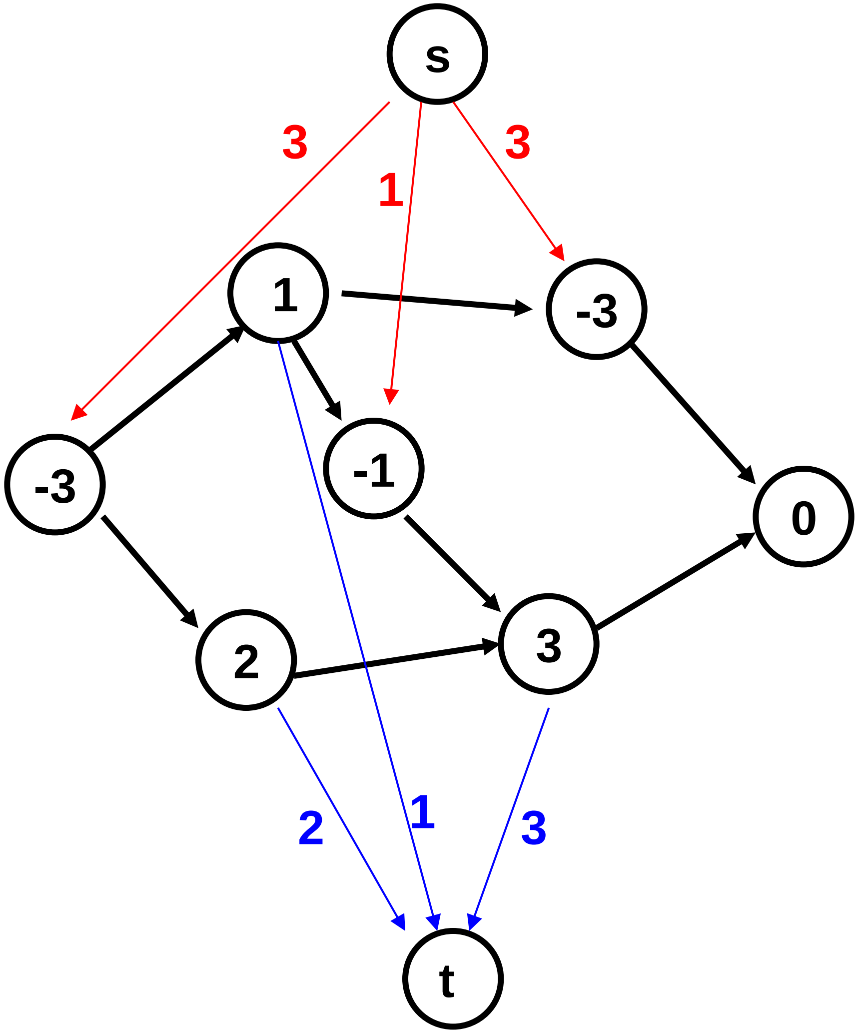
The sink side of a finite cut is a feasible set

- No edges permitted from S to T
- If a vertex is in T, all of its ancestors are in T

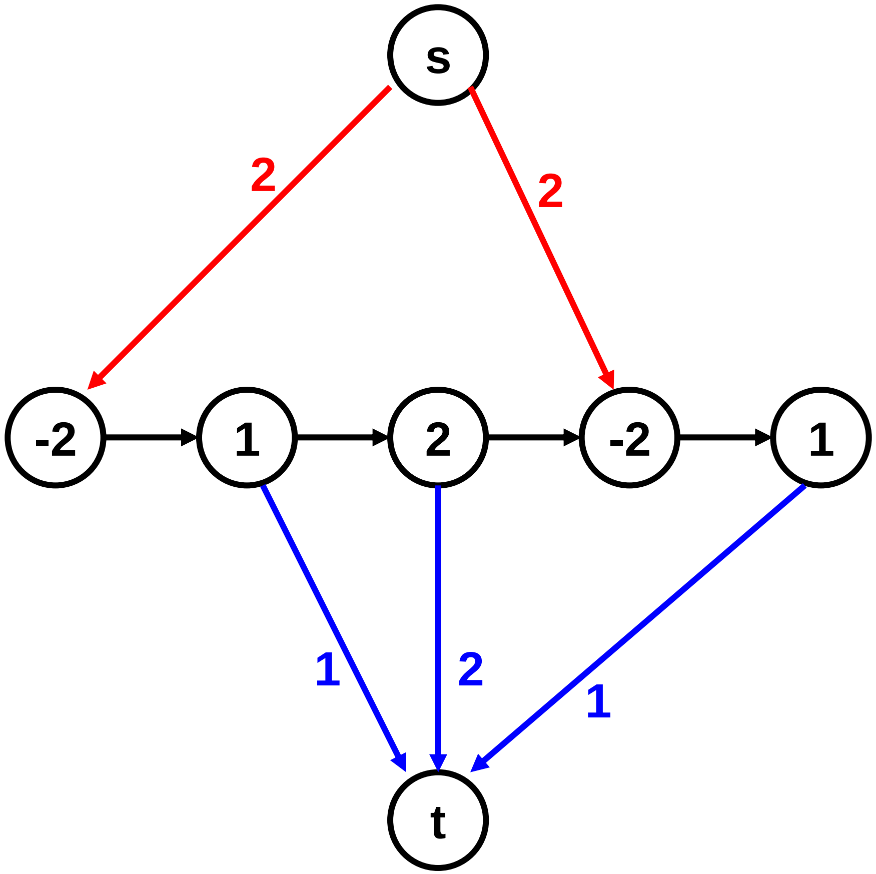


Setting the costs

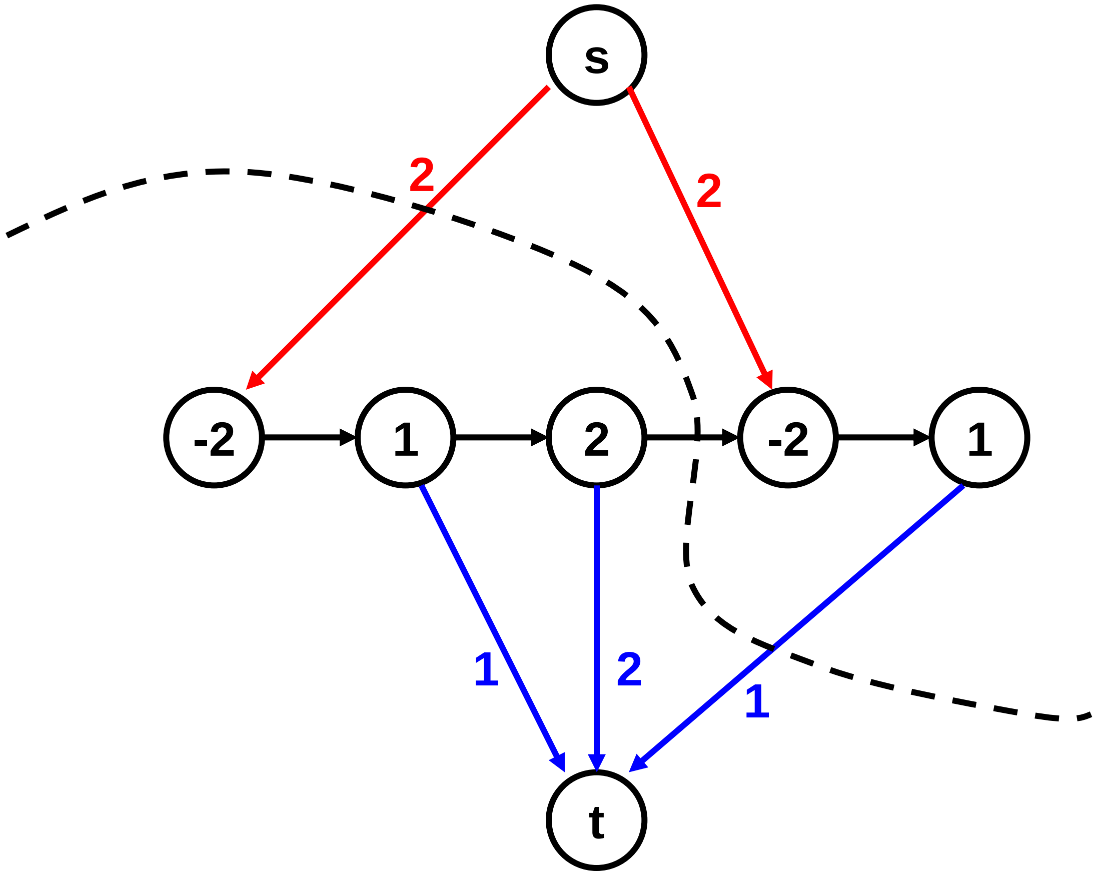
- If p(v) > 0,
  - cap(v,t) = p(v)
  - cap(s,v) = 0
- If p(v) < 0
  - cap(s,v) = -p(v)
  - cap(v,t) = 0
- If p(v) = 0
  - cap(s,v) = 0
  - cap(v,t) = 0



Enumerate all finite s,t cuts and show their capacities



Minimum cut gives optimal solution
Why?

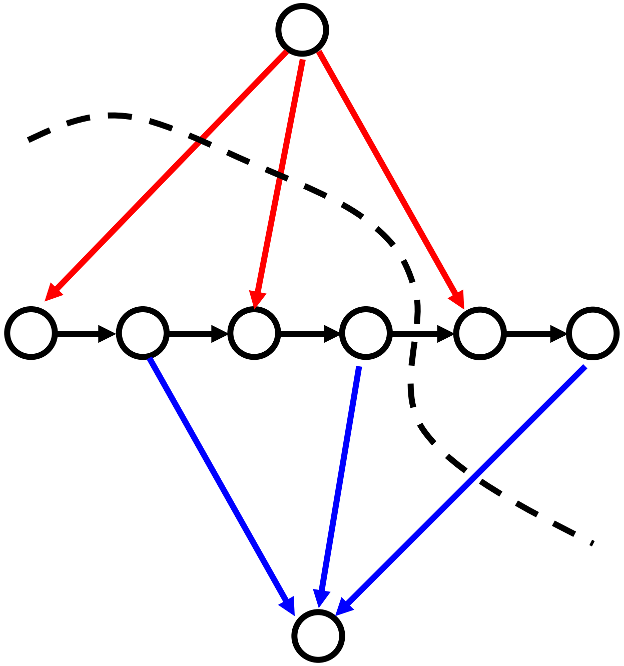


Computing the Profit

- $\mathrm{Cost}(W) = \Sigma_{\{w \text{ in } W;\ p(w) < 0\}} -p(w)$
- $\mathrm{Benefit}(W) = \Sigma_{\{w \text{ in } W;\ p(w) > 0\}}\ p(w)$
- Profit(W) = Benefit(W) – Cost(W)

- Maximum cost and benefit
  - C = Cost(V)
  - B = Benefit(V)

Express Cap(S,T) in terms of B, C, Cost(T), Benefit(T), and Profit(T)



Summary

- Construct flow graph
  - Infinite capacity for precedence edges
  - Capacities to source/sink based on cost/benefit
- Finite cut gives a feasible set of tasks
- Minimizing the cut corresponds to maximizing the profit
- Find minimum cut with a network flow algorithm