UMBC

AN HONORS UNIVERSITY IN MARYLAND

Computational Methods in IS Research

Graph Algorithms
Network Flow Problems

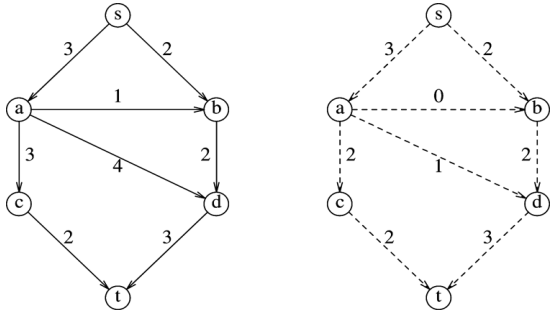
Nirmalya Roy

Department of Information Systems

University of Maryland Baltimore County

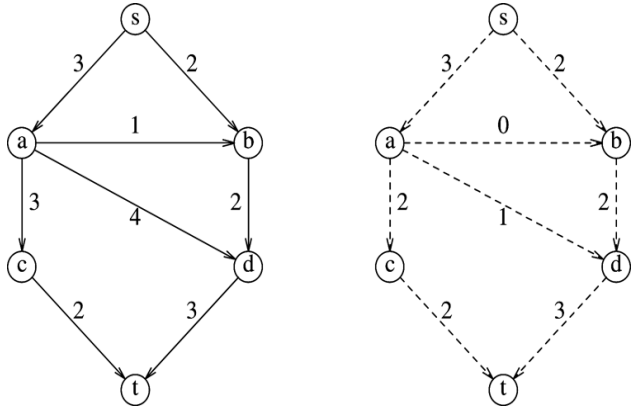
Network Flow Problems

- Given a directed graph G = (V, E) with **edge capacities** $c_{v, w}$
  - The capacities could represent the amount of water that could flow through a pipe or the amount of traffic that could flow on a street between two intersections
- Give two vertices `s`, called the **source**, and `t`, called the **sink**
- Through any edge, (v, w), at most $c_{v, w}$ units of "flow" may pass
- At any vertex `v`, that is not either `s` or `t`, the total flow coming in must equal the total flow coming out



Network Flow Problems (cont'd)

- The maximum flow problem is to determine the maximum amount of flow that can pass from s to t
- No edge carries more flow than its capacity
- A vertex can combine and distribute flow in any manner that it likes, as long as edge capacities are not violated as long as flow conservation is maintained (what does in must come out)

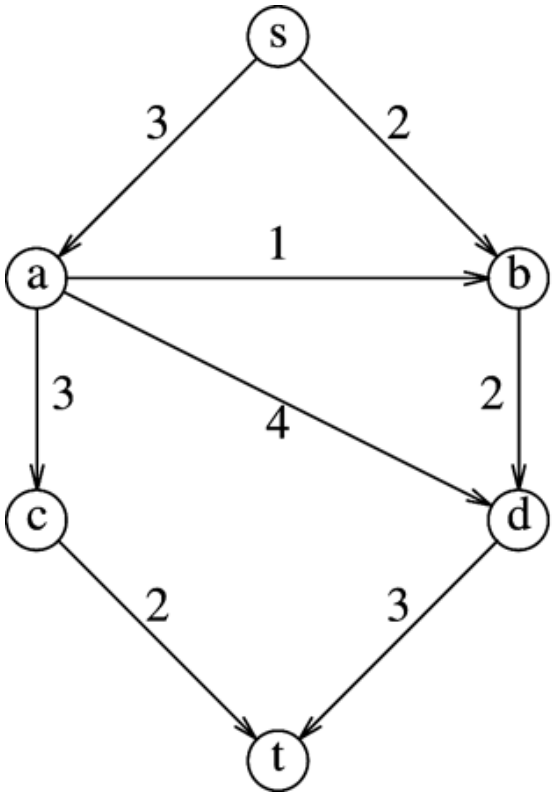


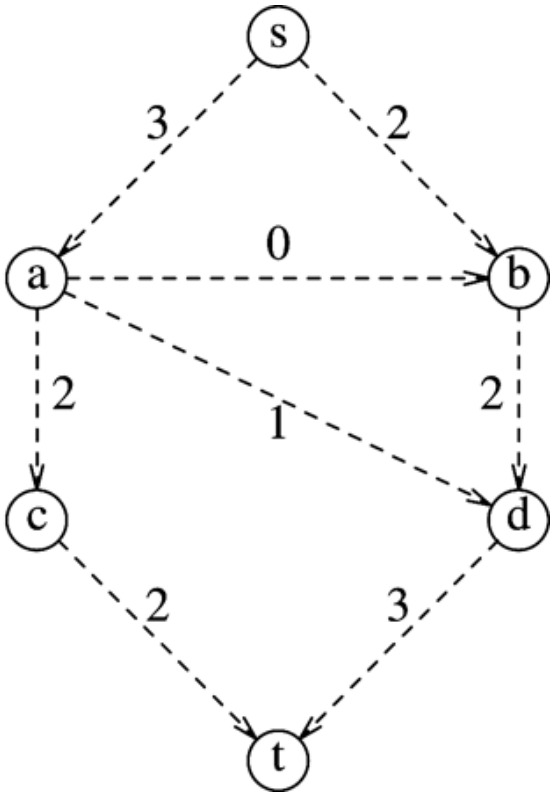
Network Flow Problem (cont'd)

- Given
  - Directed graph G = (V, E) with edge capacities $c_{v, w}$
  - Source vertex s
  - Sink vertex t
- Constraints
  - Flow along directed edge (v, w) cannot exceed capacity $c_{v, w}$
  - At every vertex (except s and t), the total flow coming in must equal the total flow going out
- Find
  - Maximum amount of flow from s to t

Network Flow

- Example network graph (left) and its maximum flow (right)



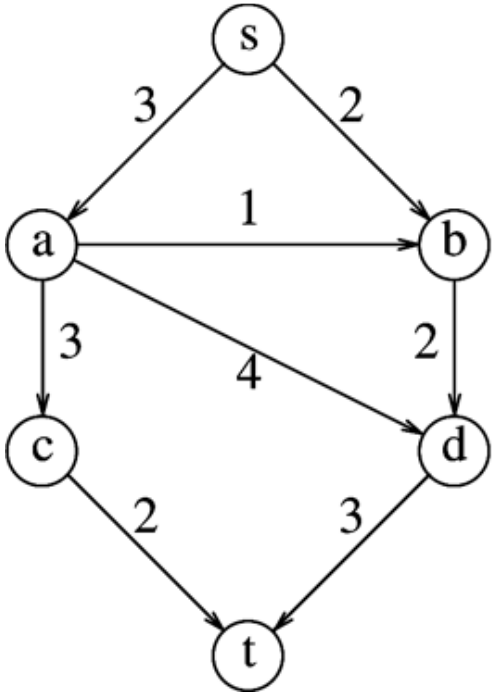


Maximum Flow Algorithm

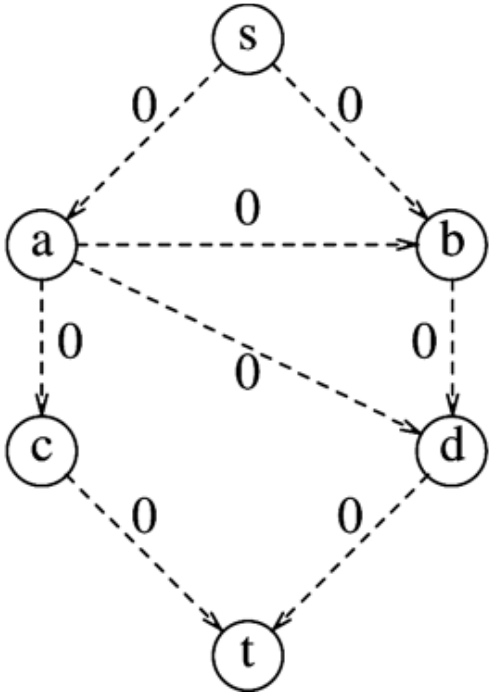
- Flow graph G_f
  - Indicates the amount of flow on each edge in the network
- Residual graph G_r
  - Indicates how much more flow can be added to each edge in the network
  - Residual capacity = capacity – current flow
  - Edges with zero residual capacity removed
- Augmenting path
  - Path from s to t in G_r
  - Edge with smallest residual capacity in the path indicates amount by which flow can increase along path

Maximum Flow Algorithm (cont'd)

- Example

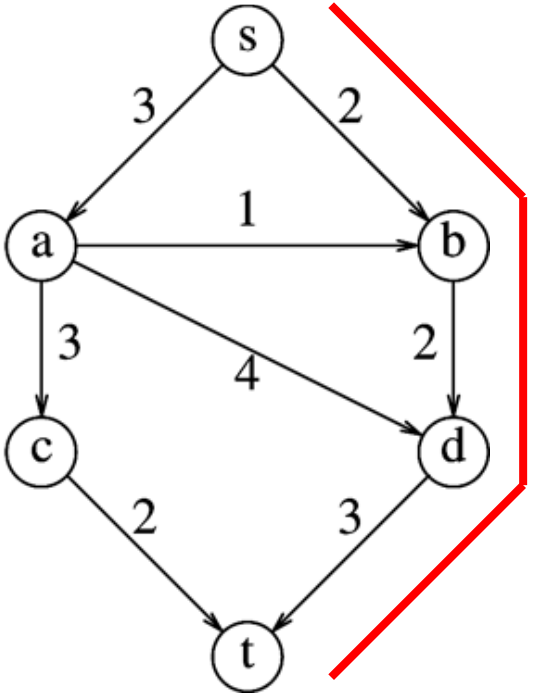


Network graph



Initial flow graph

Augmenting path (s, b, d, t)



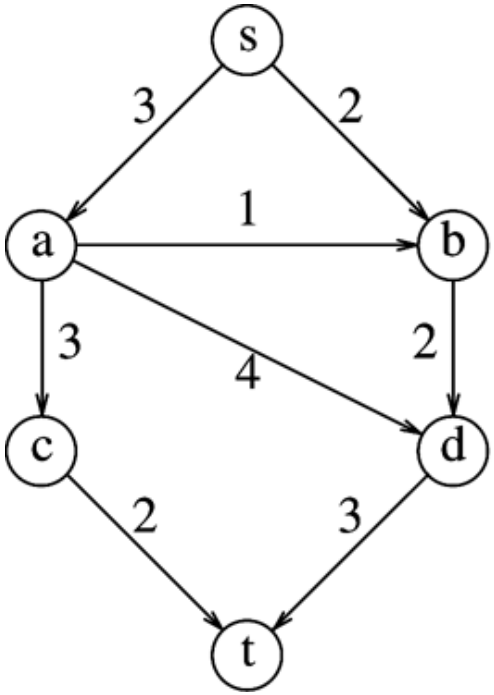
Residual graph

Maximum Flow Algorithm (cont'd)

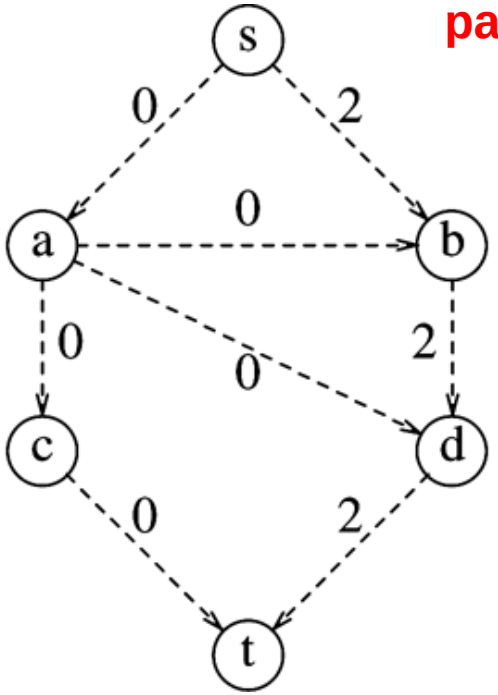
- While an augmenting path exists in G_r
  - Choose one
  - Flow increase FI = minimum residual capacity along augmenting path
  - Increase flows along augmenting path in flow graph G_f by FI
  - Update residual graph G_r

Maximum Flow Algorithm (cont'd)

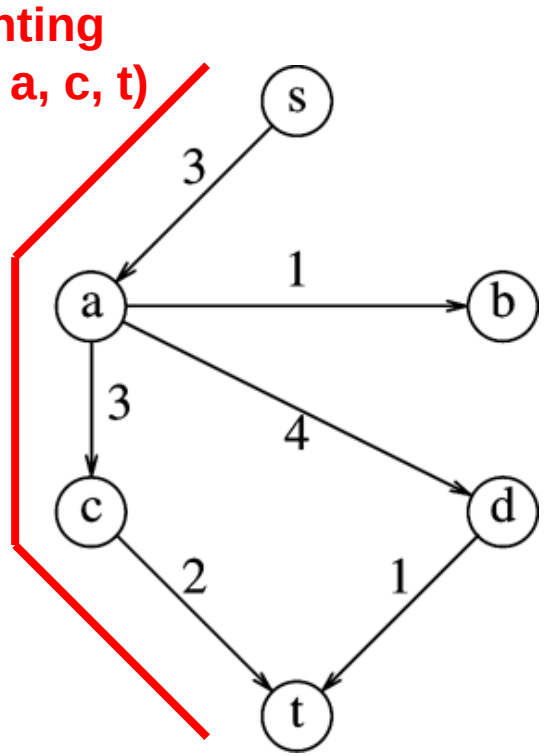
- Example (cont'd) after choosing (s, b, d, t)



Network graph



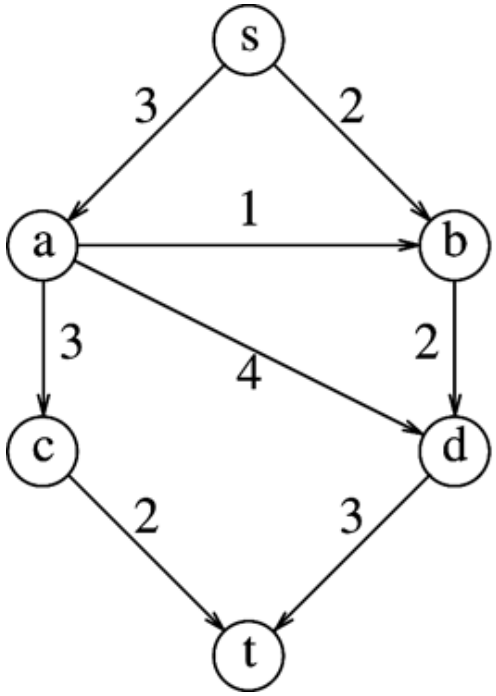
Flow graph (f = 2)



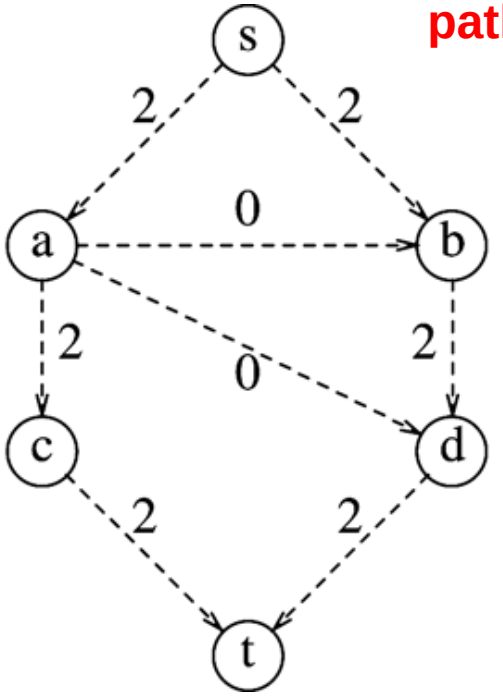
Residual graph

Maximum Flow Algorithm (cont'd)

- Example (cont'd) after choosing (s, a, c, t)

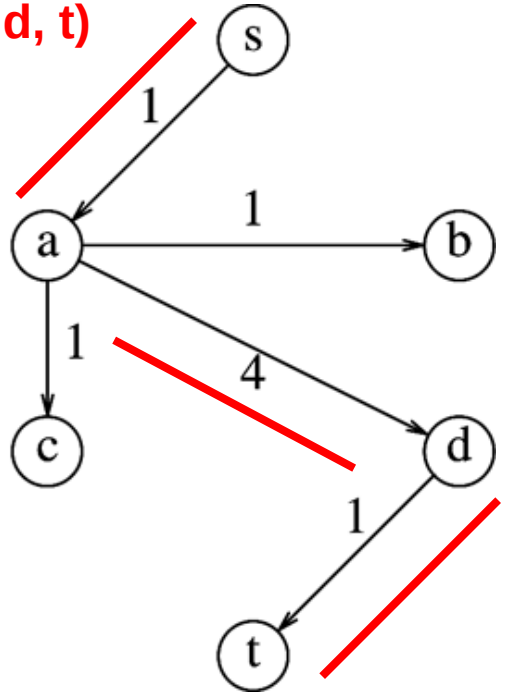


Network graph



Flow graph (f = 4)

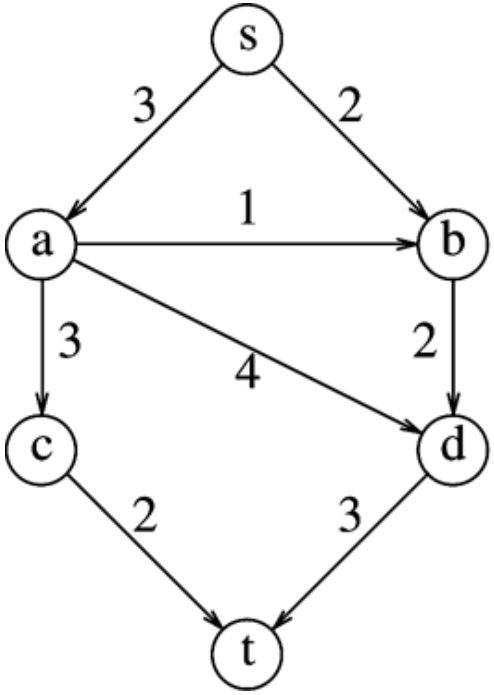
Augmenting path (s, a, d, t)



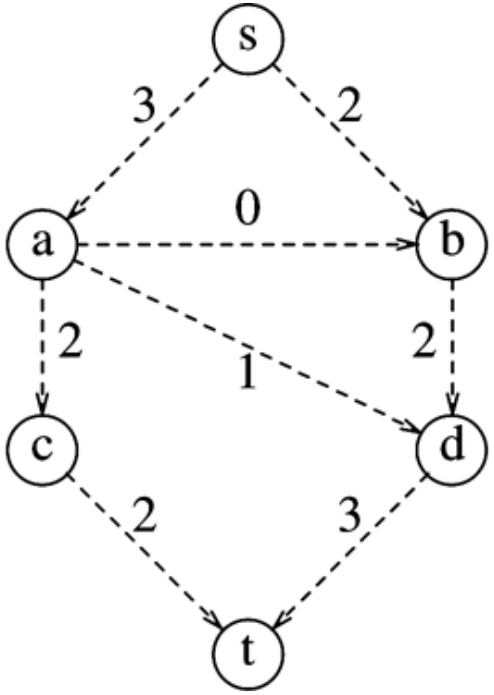
Residual graph

Maximum Flow Algorithm (cont'd)

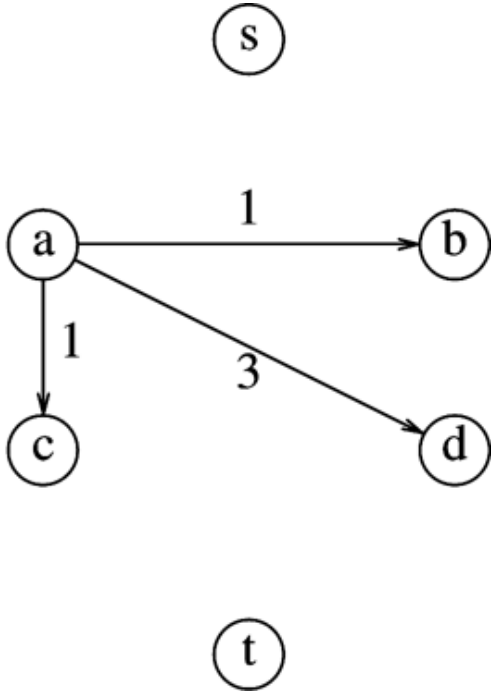
- Example (cont'd) after choosing (s, a, d, t)



Network graph



Flow graph (f = 5)

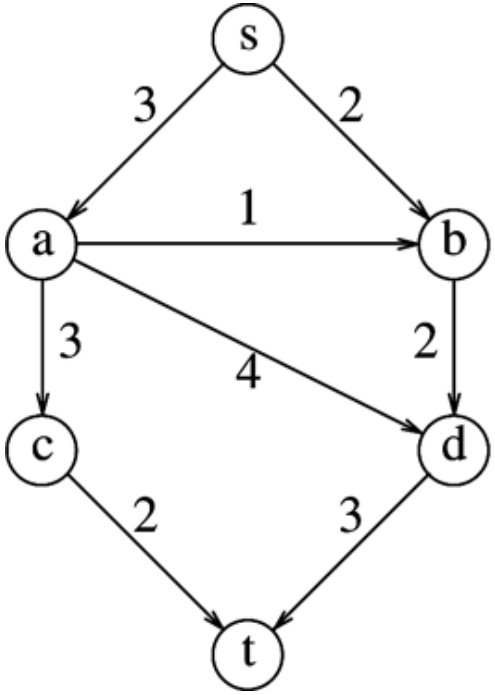


Residual graph

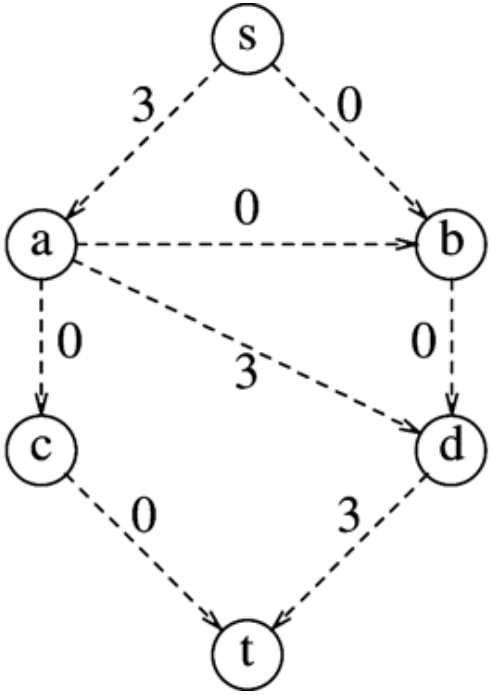
Terminates with maximum flow f = 5

Maximum Flow Algorithm (cont'd)

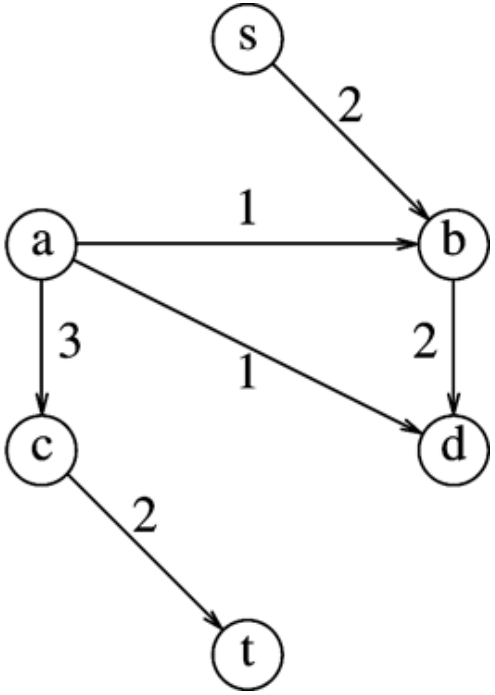
- Problem: Suppose we chose augmenting path (s, a, d, t) first



Network graph



Flow graph (f = 3)

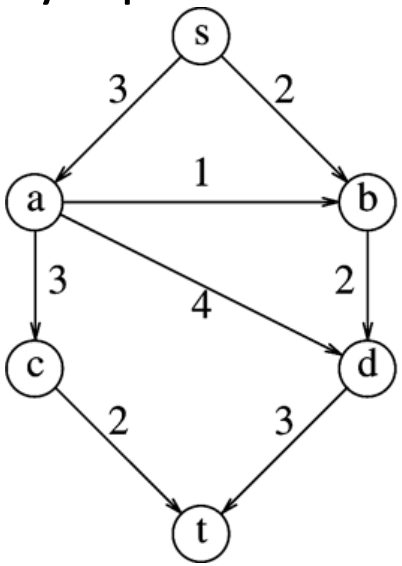


Residual graph

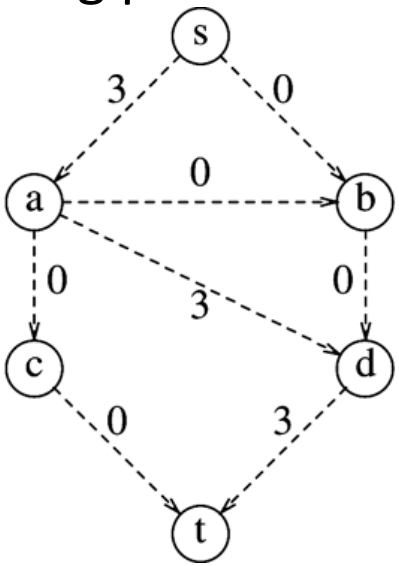
Terminates with maximum flow f = 3 (not optimal, suboptimal)

Maximum Flow Algorithm (cont'd)

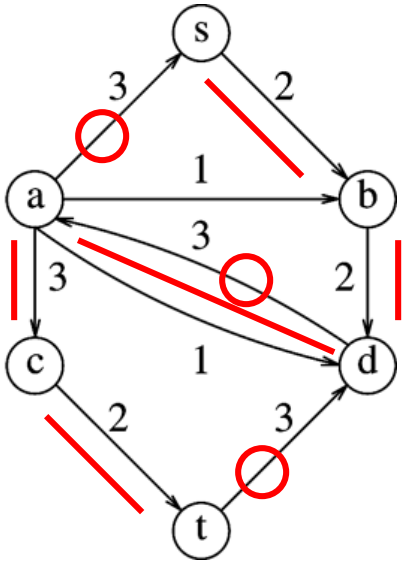
- Solution
  - Indicate potential for back flow in residual graph
    - For every edge (v,w) in flow graph with a flow, add an edge (w,v) in the residual graph with same capacity
  - i.e., allow another augmenting path to undo some of the flow used by a previous augmenting path



Network graph



Flow graph (f = 3)

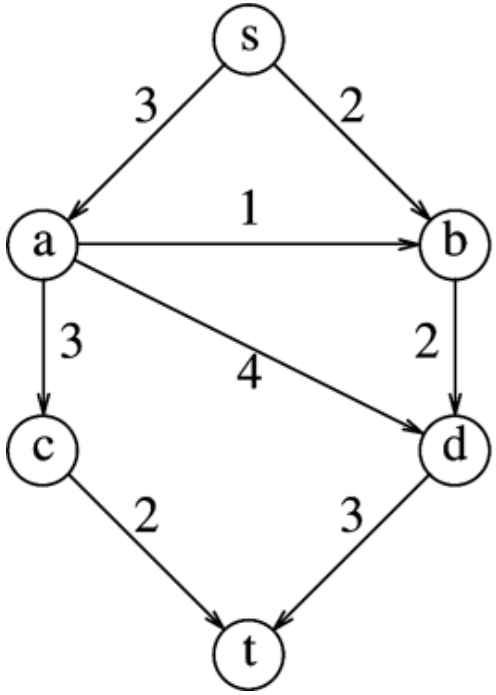


Residual graph

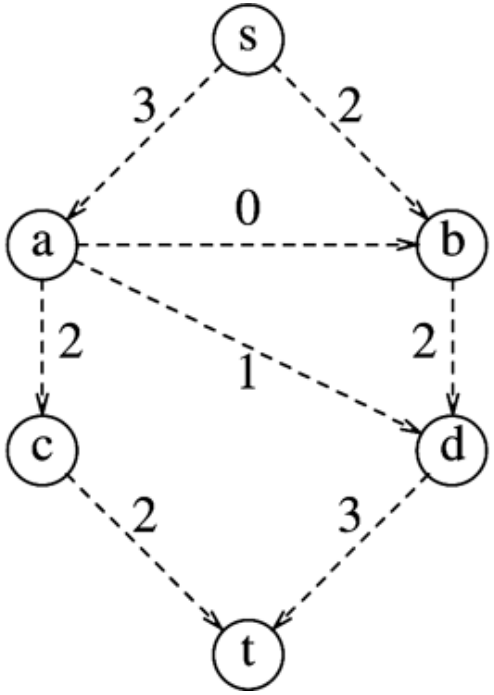
New Augmenting Path (s, b, d, a, c, t)

Maximum Flow Algorithm (cont'd)

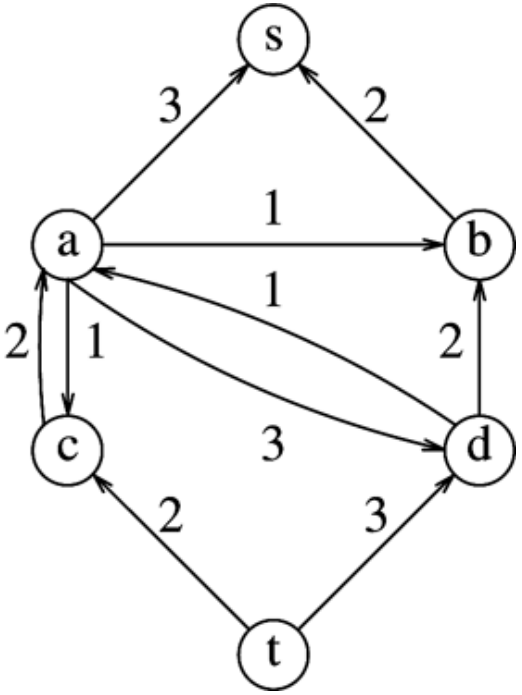
- Example (cont'd) after choosing (s, b, d, a, c, t)



Network graph



Flow graph (f = 5)



Residual graph

Terminates with maximum flow f = 5

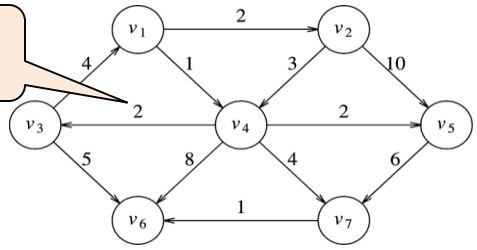
Maximum Flow Algorithm: Analysis

- If edge capacities are rational numbers, then this algorithm always terminates with maximum flow
- If capacities are integers and maximum flow is f, then running time is O(f x |E|)
  - Flow always increases by at least 1 with each augmenting path
  - Augmenting path can be found in O(|E|) time using unweighted shortest-path algorithm
  - Problem: Can be slow for large f

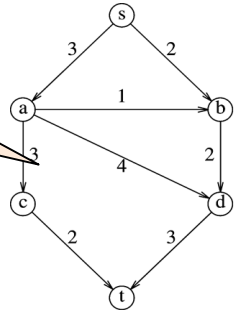
Maximum Flow Algorithm

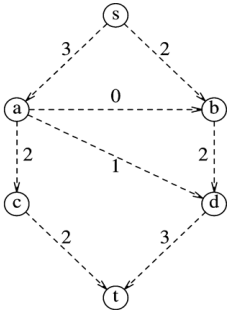
- Variants
  - Always choose augmenting path allowing largest increase in flow
    - Finding such a path is similar to solving weighted shortest path problem
    - O(|E|) calls to O(|E| log |V|) Dijkstra's algorithm
    - Running time O(|E|^2 log |V|)
  - Always choose the augmenting path with the fewest edges (unweighted shortest path)
    - At most O(|E| x |V|) augmenting steps, each costing O(|E|) for call to BFS
    - Running time O(|E|^2 |V|)

Dijkstra's weighted shortest/ minimum cost path problem



Maximum Network Flow Problem





Network Flow Problems

- Variants
  - Multiple sources and sinks
    - Create super-source with infinite-capacity links to each source
    - Create super-sink with infinite-capacity links to each sink
  - **Min-cost flow** problem
    - Each edge has not only a capacity but also a cost per unit of flow
    - Find maximum flow with minimum cost
    - No known fast algorithm

Summary

- Network flow is an important algorithm with numerous practical applications
- Running time depends on method for finding augmenting path
  - BFS: $O(|E|^2 |V|)$
  - Dijkstra's algorithm: $O(|E|^2 \log |V|)$