Graph Algorithms

CptS 223 – Advanced Data Structures

Larry Holder
School of Electrical Engineering and Computer Science
Washington State University
Network Flow

How much freight can flow from Seattle to Pullman?

From Seattle to Chicago?

From Washington to Illinois?
Network Flow Approach

- Let the waypoints (cities, distribution centers) be vertices in a graph.
- Let the routes (roads, rails, waterways, airways) between waypoints be directed edges in the graph.
- Each edge has a weight representing the route’s capacity.
- Choose a starting point $s$ and an ending point $t$.
- Determine the maximum rate (e.g., tons/hour) at which we can flow freight from point $s$ to point $t$.
  - Without the freight piling up anywhere along the way.
Network Flow Applications

- Freight flow through shipping networks
- Fluid flow through a network of pipes
- Data flow (bandwidth) through computer networks
- Nutrient flow through food networks
- Electricity flow through power distribution systems
- People flow through mass transportation systems
- Traffic flow through a city
Network Flow Problem

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

Example

Network Graph

Initial Flow Graph

Residual Graph

Augmenting Path (s,b,d,t)
Maximum Flow Algorithm

- While an augmenting path exists in $G_r$
  - Choose one
  - Flow increase $F_I = \text{minimum residual capacity along augmenting path}$
  - Increase flows along augmenting path in flow graph $G_f$ by $F_I$
  - Update residual graph $G_r$
Maximum Flow Algorithm

- Example (cont.) after choosing \((s,b,d,t)\)
Maximum Flow Algorithm

- Example (cont.) after choosing \((s,a,c,t)\)
Maximum Flow Algorithm

- Example (cont.) after choosing \((s,a,d,t)\)

Network Graph

Flow Graph \((f=5)\)

Residual Graph

Terminates with maximum flow \(f = 5\).
Problem: Suppose we chose augmenting path \((s,a,d,t)\) first.

Terminates with flow \(f = 3\).
Maximum Flow Algorithm

- Solution
  - Indicate potential for back flow in residual graph
  - 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
Maximum Flow Algorithm

- Example (cont.) after choosing \((s,b,d,a,c,t)\)

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 $O(f \cdot |E|)$.
    - 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 $f$. 

Variants

- Always choose augmenting path allowing largest increase in flow
  - \( O(|E|) \) calls to \( O(|E| \log |V|) \) Dijkstra algorithm
  - Running time \( O(|E|^2 \log |V|) \)

- Always choose the augmenting path with the fewest edges (unweighted shortest path)
  - At most \( O(|E||V|) \) augmenting steps, each costing \( O(|E|) \) for call to BFS
  - Running time \( O(|E|^2|V|) \)
Network Flow

- **Variants**
  - **Multiple sources and sinks**
    - Create super-source with infinite-capacity links to each source
    - Create super-sink with infinite-capacity links from each sink
  - **Min-cost flow**
    - Each edge has capacity and cost
    - Find maximum flow with minimum cost
    - No known fast solution
Network Flow

- Important algorithm with numerous practical applications
- Running time depends on method for finding augmenting path
  - BFS: $O(|E|^2|V|)$
  - Dijkstra: $O(|E|^2 \log |V|)$