CSL 356: Analysis and Design of Algorithms

Ragesh Jaiswal

CSE, IIT Delhi

Network Flow

- For an $s - t$ flow f and a positive integer Δ, let $G_f(\Delta)$ denote a subset of the residual graph G_f consisting only of edges with residual capacity of at least Δ.
- <u>Idea</u>: Instead of finding augmenting paths in G_f, we will find augmenting paths in $G_f(\Delta)$ for smaller and smaller values of Δ.

Scaling-Max-Flow

- Start with an $s - t$ flow such that for all e, $f(e) = 0$
- Δ=largest power of 2 smaller than C
- while $\Delta \geq 1$
    - while there is an $s - t$ path P in $G_f(\Delta)$
        - Execute the augmenting path algorithm to obtain f'
        - Update f to f' and $G_f(\Delta)$ to $G_{f'}(\Delta)$
    - $\Delta = \Delta/2$
- return f

Network Flow

- Claim 1: The algorithm returns max flow on termination.

Scaling-Max-Flow

- Start with an $s-t$ flow such that for all e, $f(e) = 0$
- Δ=largest power of 2 smaller than C
- while $\Delta \geq 1$
  - while there is an $s-t$ path P in $G_f(\Delta)$
    - Execute the augmenting path algorithm to obtain f'
    - Update f to f' and $G_f(\Delta)$ to $G_{f'}(\Delta)$
  - $\Delta = \Delta/2$
- return f

Network Flow

- <u>Claim 1</u>: The algorithm returns max flow on termination.
- <u>Claim 2</u>: The while loop runs for at most $(1 + \lceil \log(C) \rceil)$ steps.

Scaling-Max-Flow

- Start with an $s - t$ flow such that for all $e, f(e) = 0$
- Δ=largest power of 2 smaller than C
- while $\Delta \geq 1$
    - while there is an $s - t$ path P in $G_f(\Delta)$
        - Execute the augmenting path algorithm to obtain f'
        - Update f to f' and $G_f(\Delta)$ to $G_{f'}(\Delta)$
    - $\Delta = \Delta/2$
- return f

Network Flow

- Claim 1: The algorithm returns max flow on termination.
- Claim 2: The while loop runs for at most $(1 + \lceil \log(C) \rceil)$ steps.
- Claim 3: Each augmentation increases the flow by at least Δ.

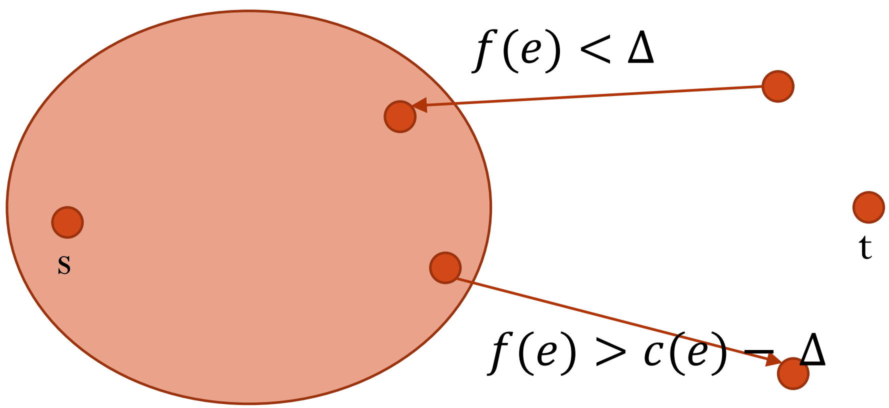
Scaling-Max-Flow

- Start with an $s-t$ flow such that for all e, $f(e) = 0$
- Δ=largest power of 2 smaller than C
- while $\Delta \geq 1$
  - while there is an $s-t$ path P in $G_f(\Delta)$
    - Execute the augmenting path algorithm to obtain f'
    - Update f to f' and $G_f(\Delta)$ to $G_{f'}(\Delta)$
  - $\Delta = \Delta/2$
- return f

Network Flow

- Claim 1: The algorithm returns max flow on termination.
- Claim 2: The while loop runs for at most $(1 + \lceil \log(C) \rceil)$ steps.
- Claim 3: Each augmentation increases the flow by at least Δ.
- Claim 4: Let f be the flow at the end of a Δ-scaling phase. Then there is an $s - t$ cut (A, B) such that $c(A, B) \leq v(f) + m \cdot \Delta$.
  - Corollary: The max flow in the graph has value at most $(v(f) + m \cdot \Delta)$.

$$
\begin{aligned}
v(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e) \\
&\geq \sum_{e \text{ out of } A} (c(e) - \Delta) - \sum_{e \text{ into } A} \Delta \\
&\geq c(A, B) - m \cdot \Delta
\end{aligned}
$$



A (all vertices reachable from s in $G_f(\Delta)$.

Network Flow

- Claim 1: The algorithm returns max flow on termination.
- Claim 2: The while loop runs for at most $(1 + \lceil \log(C) \rceil)$ steps.
- Claim 3: Each augmentation increases the flow by at least Δ.
- Claim 4: Let f be the flow at the end of a Δ-scaling phase. Then there is an $s - t$ cut (A, B) such that $c(A, B) \leq v(f) + m \cdot \Delta$.
  - Corollary: The max flow in the graph has value at most $(v(f) + m \cdot \Delta)$.
- Claim 5: The total number of iterations of the inner while loop is at most $2m$.
- Claim 6: The running time of Scaling-max-flow algorithm is $O(m^2 \cdot \log(C))$.

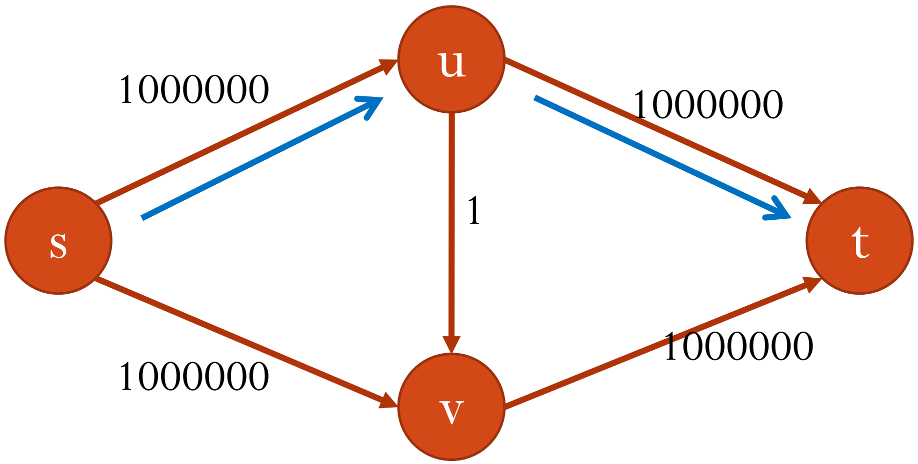
Network Flow

Strongly polynomial time algorithm for max-flow

Network Flow: Edmonds-Karp

Max-Flow *//Edmonds-Karp algorithm*

- Start with a flow f such that $f(e) = 0$
- while there is an $s - t$ path P in G_f
  - Find an $s - t$ path with **least hop-length**
    - Execute the augmenting path algorithm to obtain f'
    - Update f to f' and G_f to $G_{f'}$
- return f



Network Flow: Edmonds-Karp

- Let $d_f(s, v)$ denote the hop-length of the shortest path from s to v in G_f.
- Claim 1: For all $v \neq s, t$, $d_f(s, v)$ either remains same or increases with each flow augmentation.

Network Flow: Edmonds-Karp

- Let $d_f(s, v)$ denote the hop-length of the shortest path from s to v in G_f.
- Claim 1: For all $v \neq s, t$, $d_f(s, v)$ either remains same or increases with each flow augmentation.
- Proof:
  - Let f be the flow just before the first augmentation that decreases the shortest distance of some vertex. Let f' be the flow after this augmentation.
  - Let v be the vertex with minimum value of $d_{f'}(s, v)$ whose shortest path length was reduced.
  - Let u be the vertex just before **v** in the shortest path from **s** to v in $G_{f'}$.

Network Flow: Edmonds-Karp

- <u>Claim 1</u>: For all $v \neq s, t$, $d_f(s, v)$ either remains same or increases with each flow augmentation.
- Proof:
  - Let f be the flow just before the first augmentation that decreases the shortest distance of some vertex. Let f' be the flow after this augmentation.
  - Let v be the vertex with minimum value of $d_{f'}(s, v)$ whose shortest path length was reduced.
  - Let u be the vertex just before v in the shortest path from s to v in $G_{f'}$.
  - We have:
    - $d_{f'}(s, u) = d_{f'}(s, v) - 1$
    - $d_{f'}(s, u) \geq d_f(s, u)$
  - <u>Claim</u>: (u, v) is not present in G_f.
  - <u>Proof</u>: Since otherwise,

    $$d_f(s, v) \leq d_f(s, u) + 1 \leq d_{f'}(s, u) + 1 = d_{f'}(s, v).$$

Network Flow: Edmonds-Karp

- Claim 1: For all $v \neq s, t$, $d_f(s, v)$ either remains same or increases with each flow augmentation.
- Proof:
  - Let f be the flow just before the first augmentation that decreases the shortest distance of some vertex. Let f' be the flow after this augmentation.
  - Let v be the vertex with minimum value of $d_{f'}(s, v)$ whose shortest path length was reduced.
  - Let u be the vertex just before v in the shortest path from s to v in $G_{f'}$.
  - We have:
    - $d_{f'}(s, u) = d_{f'}(s, v) - 1$
    - $d_{f'}(s, u) \geq d_f(s, u)$
  - Claim: (u, v) is not present in G_f.
  - Proof: Since otherwise,

    $$d_f(s, v) \leq d_f(s, u) + 1 \leq d_{f'}(s, u) + 1 = d_{f'}(s, v).$$
  - This means that (v, u) was in the augmenting path. This means: $d_f(s, v) = d_f(s, u) - 1 \leq d_{f'}(s, u) - 1 \leq d_{f'}(s, v) - 2$

Network Flow: Edmonds-Karp

- Claim 2: The total number of flow augmentations in the Edmonds–Karp algorithm is $O(nm)$.
- Proof:
  - An edge is said to be critical while augmentation if it is the bottleneck edge.
  - Claim: Any edge can become critical at most $(n/2)$ times.
  - Proof:
    - Consider any edge (u, v). Let f be the flow just before (u, v) becomes critical. The we have
$$d_f(s, v) = d_f(s, u) + 1 \qquad (1)$$
    - After this the edge (u, v) disappears. Let f' be the flow just before the augmentation that brings back edge (u, v). Then we have
$$d_{f'}(s, u) = d_{f'}(s, v) + 1 \qquad (2)$$

Network Flow: Edmonds-Karp

- <u>Claim 2</u>: The total number of flow augmentations in the Edmonds–Karp algorithm is $O(nm)$.
- Proof:
  - An edge is said to be critical while augmentation if it is the bottleneck edge.
  - <u>Claim</u>: Any edge can become critical at most $(n/2)$ times.
  - <u>Proof</u>:
    - Consider any edge (u, v). Let f be the flow just before (u, v) becomes critical. The we have
    $$d_f(s, v) = d_f(s, u) + 1 \qquad (1)$$
    - After this the edge (u, v) disappears. Let f' be the flow just before the augmentation that brings back edge (u, v). Then we have
    $$d_{f'}(s, u) = d_{f'}(s, v) + 1 \qquad (2)$$
    - Using (1) and (2) we get:
    $$d_{f'}(s, u) = d_{f'}(s, v) + 1 \geq d_f(s, v) + 1 = d_f(s, u) + 2$$
    - The shortest distance has increased by 2 between the instances when (u, v) becomes critical.

Network Flow: Edmonds-Karp

- <u>Claim 2</u>: The total number of flow augmentations in the Edmonds–Karp algorithm is $O(nm)$.

- <u>Theorem</u>: The running time of Edmonds-Karp algorithm is $O(nm^2)$.

End