

Algorithm Design and Analysis

CSE
565

LECTURES 27

Network Flow

- Choosing good augmenting paths
- Capacity scaling algorithm

Sofya Raskhodnikova

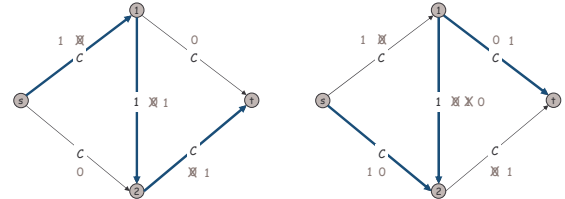
10/29/2007

S. Raskhodnikova; based on slides by K. Wayne

Ford-Fulkerson: Exponential Number of Augmentations

Q. Is generic Ford-Fulkerson algorithm polynomial in input size?
m, n, and log C

A. No. If max capacity is C, then algorithm can take C iterations.



Choosing Good Augmenting Paths

Use care when selecting augmenting paths.

- Some choices lead to exponential algorithms.
- Clever choices lead to polynomial algorithms.
- If capacities are irrational, algorithm not guaranteed to terminate!

Goal: choose augmenting paths so that:

- Can find augmenting paths efficiently.
- Few iterations.

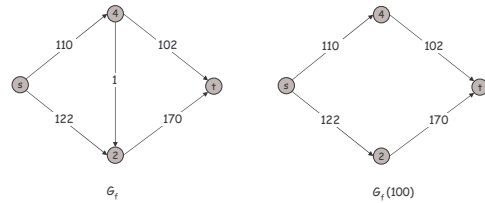
Choose augmenting paths with: [Edmonds-Karp 1972, Dinitz 1970]

- Max bottleneck capacity.
- Sufficiently large bottleneck capacity.
- Fewest number of edges.

Capacity Scaling

Intuition. Choosing path with highest bottleneck capacity increases flow by max possible amount.

- Don't worry about finding exact highest bottleneck path.
- Maintain scaling parameter Δ .
- Let $G_f(\Delta)$ be the subgraph of the residual graph consisting of only arcs with capacity at least Δ .



Capacity Scaling

```
Scaling-Max-Flow(G, s, t, c) {
  foreach e in E f(e) ← 0
  Δ ← smallest power of 2 greater than or equal to C
  Gz ← residual graph
  while (Δ ≥ 1) {
    Gz(Δ) ← Δ-residual graph
    while (there exists augmenting path P in Gz(Δ)) {
      f ← augment(f, c, P)
      update Gz(Δ)
    }
    Δ ← Δ / 2
  }
  return f
}
```

Capacity Scaling: Correctness

Assumption. All edge capacities are integers between 1 and C.

Integrality invariant. All flow and residual capacity values are integral.

Correctness. If the algorithm terminates, then f is a max flow.

Pf.

- By integrality invariant, when $\Delta = 1 \Rightarrow G_f(\Delta) = G_f$.
- Upon termination of $\Delta = 1$ phase, there are no augmenting paths. •

Capacity Scaling: Running Time

Lemma 1. The outer while loop repeats $1 + \lceil \log_2 C \rceil$ times.
 Pf. Initially $C \leq \Delta < 2C$. Δ decreases by a factor of 2 each iteration. •

Lemma 2. Let f be the flow at the end of a Δ -scaling phase. Then the value of the maximum flow f^* is at most $v(f) + m \Delta$. — proof on next slide

Lemma 3. There are at most $2m$ augmentations per scaling phase.
 • Let f be the flow at the end of the previous scaling phase.
 • Lemma 2 $\Rightarrow v(f^*) \leq v(f) + m(2\Delta)$.
 • Each augmentation in a Δ -phase increases $v(f)$ by at least Δ . •

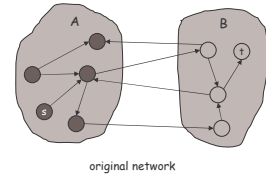
Theorem. The scaling max-flow algorithm finds a max flow in $O(m \log C)$ augmentations. It can be implemented to run in $O(m^2 \log C)$ time. •

Capacity Scaling: Running Time

Lemma 2. Let f be the flow at the end of a Δ -scaling phase. Then value of the maximum flow is at most $v(f) + m \Delta$.

Pf. (almost identical to proof of max-flow min-cut theorem)
 • We show that at the end of a Δ -phase, there exists a cut (A, B) such that $\text{cap}(A, B) \leq v(f) + m \Delta$.
 • Choose A to be the set of nodes reachable from s in $G_f(\Delta)$.
 • By definition of A , $s \in A$.
 • By definition of f , $t \in B$.

$$\begin{aligned} v(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \\ &\geq \sum_{e \text{ out of } A} (c(e) - \Delta) - \sum_{e \text{ in to } A} \Delta \\ &= \sum_{e \text{ out of } A} c(e) - \sum_{e \text{ out of } A} \Delta - \sum_{e \text{ in to } A} \Delta \\ &\geq \text{cap}(A, B) - m\Delta \quad \blacksquare \end{aligned}$$



Best Known Algorithms For Max Flow

Reminder: The scaling max-flow algorithm runs in $O(m^2 \log C)$ time.

Currently there are algorithms that run in time

- $O(mn \log n)$
- $O(n^3)$
- $O(\min(n^{2/3}, m^{1/2}) m \log n \log C)$