Advanced Algorithms
Lecture 7: Maximum Flow II

Antoine Vigneron

Ulsan National Institute of Science and Technology

March 23, 2021

Max-Flow Min-Cut Theorem

Theorem (Max-flow min-cut theorem)

In a flow network, the maximum value of a flow is equal to the minimum capacity of a cut.

Proof: follows from the Lemma below.

Lemma

If f is a flow in a network G, then the following three conditions are equivalent:

1. *f is a maximum flow in G.*
2. *The residual network G_f admits no augmenting path.*
3. *The value $|f|$ of f is equal to the capacity $c(S,T)$ of a cut (S,T).*

Proof of the Lemma: $(1) \Rightarrow (2) \Rightarrow (3) \Rightarrow (1)$.

The Basic Ford-Fulkerson Algorithm

Ford-Fulkerson

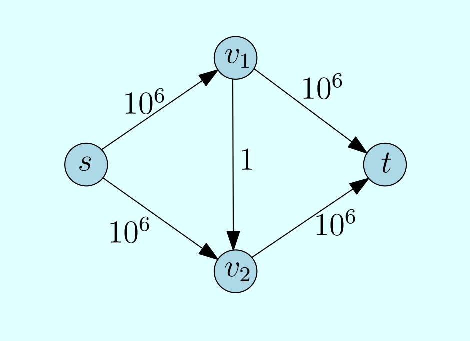
1: **for** each edge $(u, v) \in E$ **do**
2: $\quad f[u, v] \leftarrow 0$
3: $\quad f[v, u] \leftarrow 0$
4: **while** $\exists$ simple path $p : s \rightsquigarrow t$ in G_f **do**
5: $\quad c_f(p) \leftarrow \min\{c_f(u, v) \mid (u, v) \text{ is in } p\}$
6: $\quad$ **for** each edge (u, v) in p **do**
7: $\qquad f[u, v] \leftarrow f[u, v] + c_f(p)$
8: $\qquad f[v, u] \leftarrow -f[u, v]$
9: **return** f

At Line 4, the path p is found by depth-first search or breadth-first search.

Analysis

- We assume integral capacities: $c(u, v) \in \mathbb{N}$ for each u, v.
- Denote by $|f^*|$ the value of an optimal flow.
- Lines 1–3: $O(|E|)$.
- The While loop is iterated at most $|f^*|$ times.
- At each iteration:
  - Line 4: graph search (DFS or BFS) can be done in $O(|E| + |V|)$ time.
    - This is $O(|E|)$ in our case because the graph is connected, hence $|E| \geqslant |V| - 1$.
  - Lines 5–8: $O(|E|)$.
- Overall running time: $O(|E| \times |f^*|)$.

Bad Case



- $|f^*| = 2.10^6$.
- In this example, in the worst case, the while loop is iterated $|f^*|$ time.

The Edmonds-Karp Algorithm

The *Edmonds-Karp* algorithm is the basic Ford-Fulkerson method with breadth-first search.

- In particular, we take an augmenting path with as few edges as possible.

Theorem

The Edmonds-Karp algorithm computes a maximum flow in $O(|V| \cdot |E|^2)$ time.

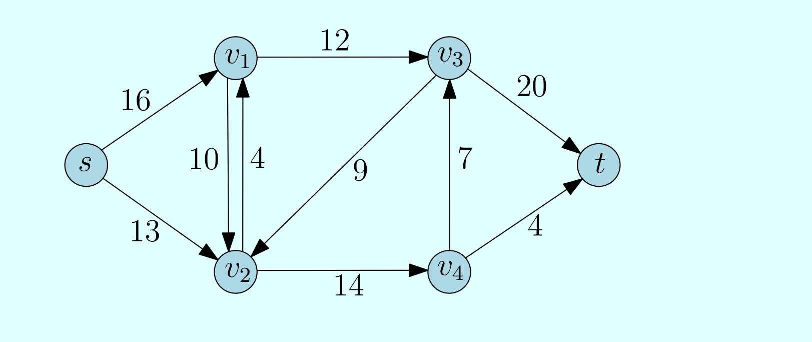
We denote by $\delta_f(u, v)$ the shortest path distance from u to v in G_f, where each edge has unit distance.

Lemma

For each vertex v, the shortest path distance $\delta_f(s, v)$ never decreases during the course of the Edmonds-Karp algorithm.

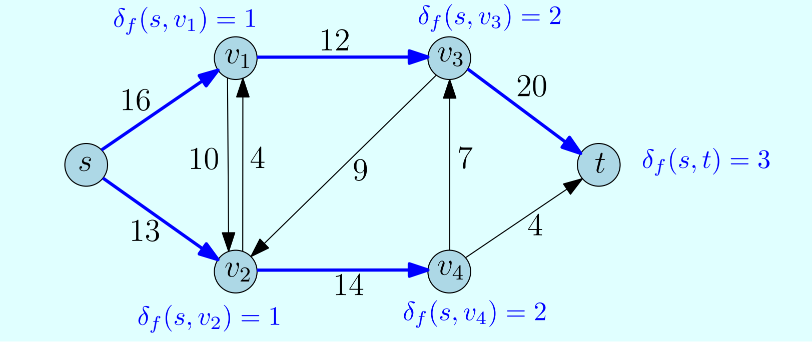
See next slides for the proofs of the lemma and the theorem.

The Edmonds-Karp Algorithm: Example



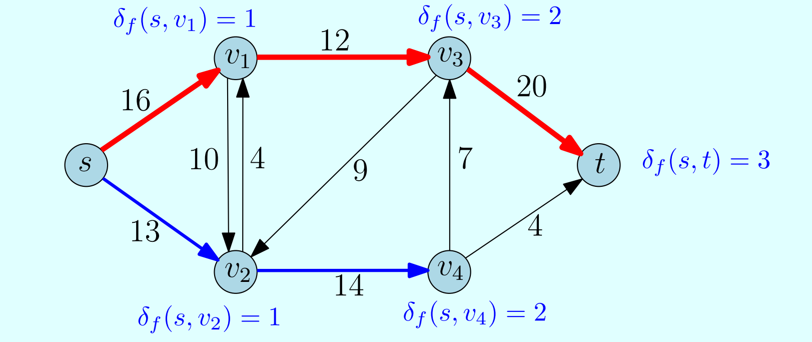
The flow network G.

The Edmonds-Karp Algorithm: Example



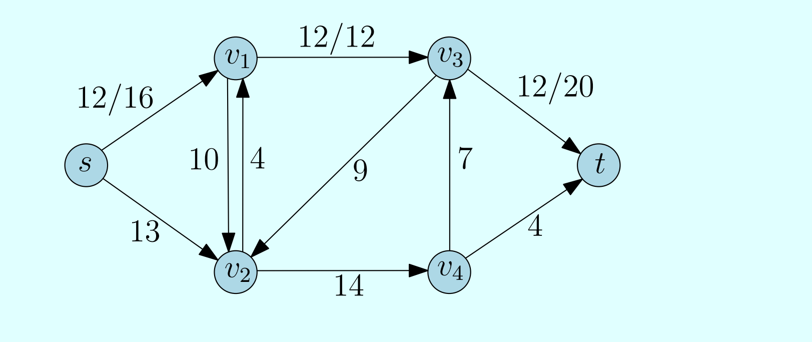
Breadth-first search in G_f for $f = 0$.

The Edmonds-Karp Algorithm: Example



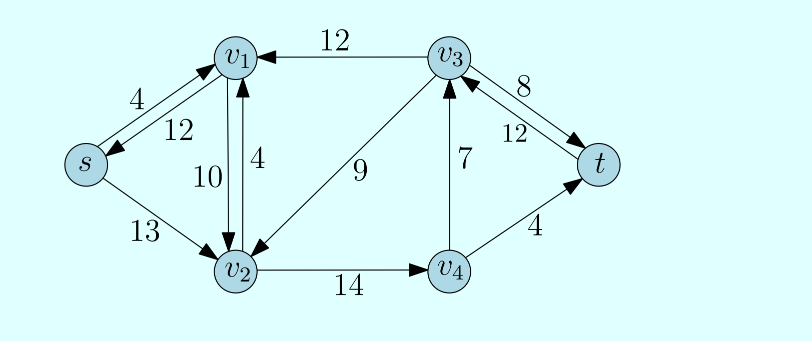
The augmenting path p, with residual capacity $c_f(p) = 12$.

The Edmonds-Karp Algorithm: Example



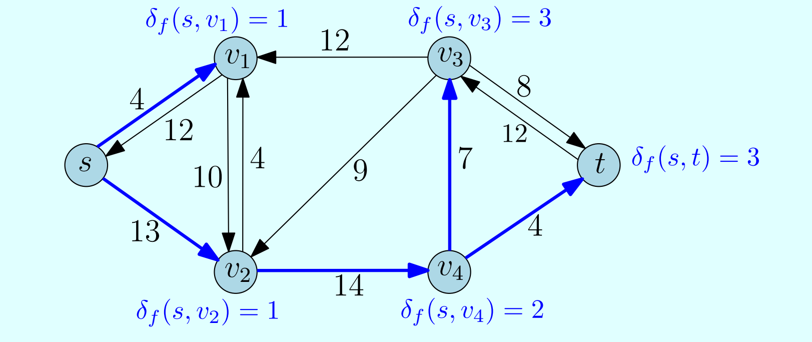
The flow f after pushing 12 units through p.

The Edmonds-Karp Algorithm: Example



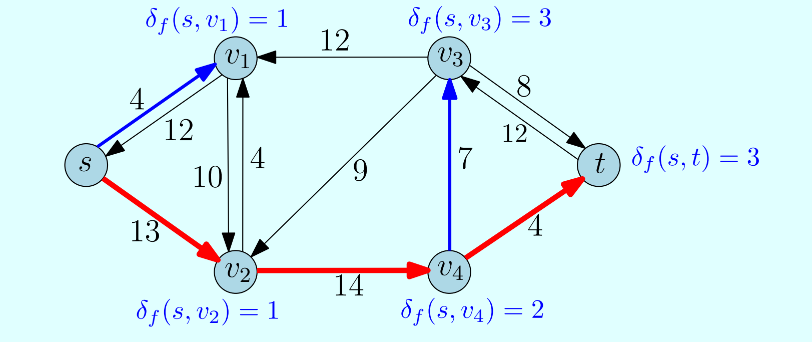
The residual network G_f.

The Edmonds-Karp Algorithm: Example



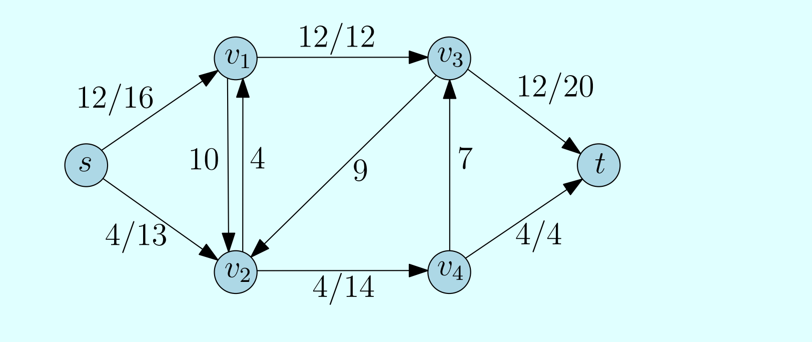
Breadth-first search in G_f.

The Edmonds-Karp Algorithm: Example



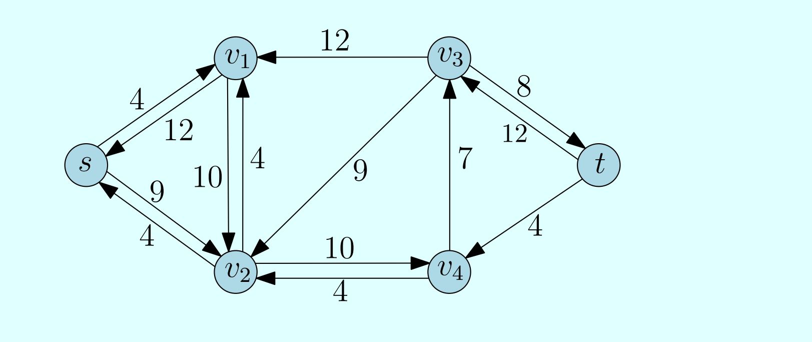
The augmenting path p, with residual capacity $c_f(p) = 4$.

The Edmonds-Karp Algorithm: Example



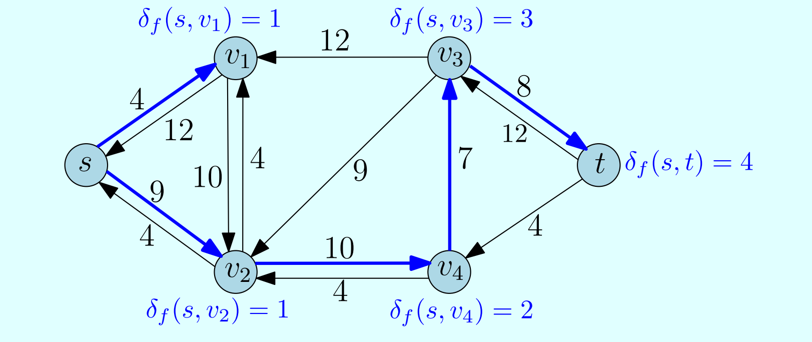
The flow f after pushing 4 units through p.

The Edmonds-Karp Algorithm: Example



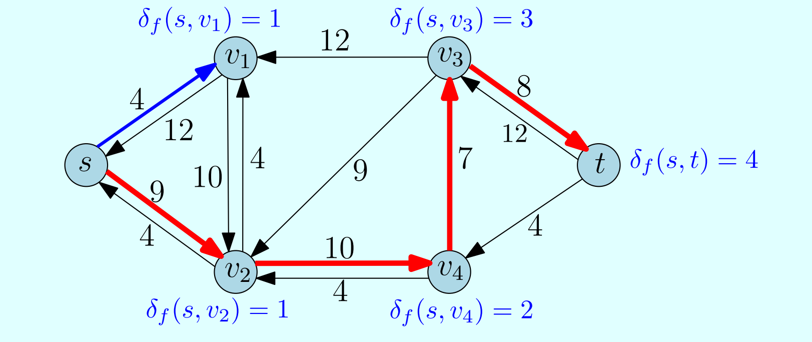
The residual network G_f.

The Edmonds-Karp Algorithm: Example



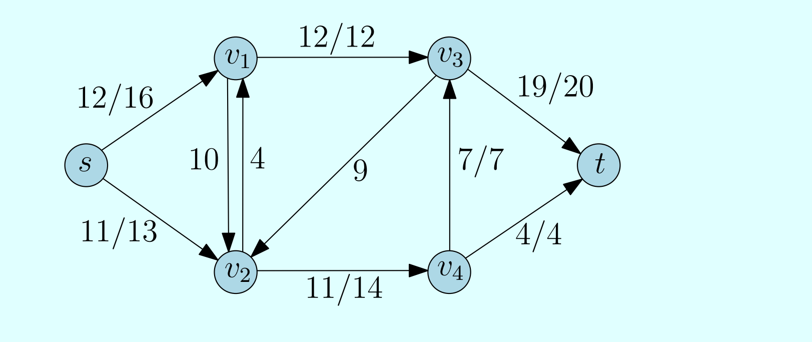
Breadth-first search in G_f.

The Edmonds-Karp Algorithm: Example



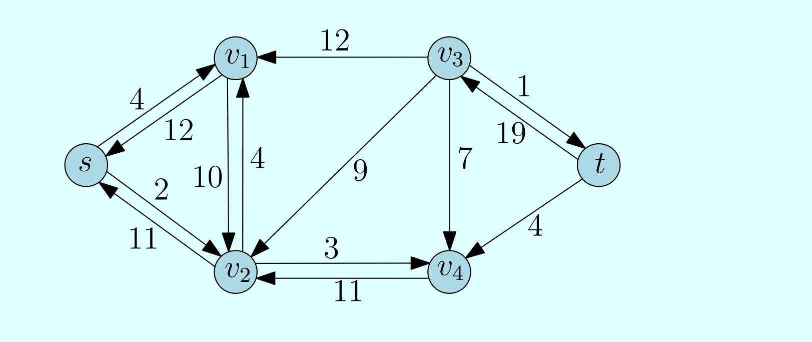
The augmenting path p, with residual capacity $c_f(p) = 7$.

The Edmonds-Karp Algorithm: Example



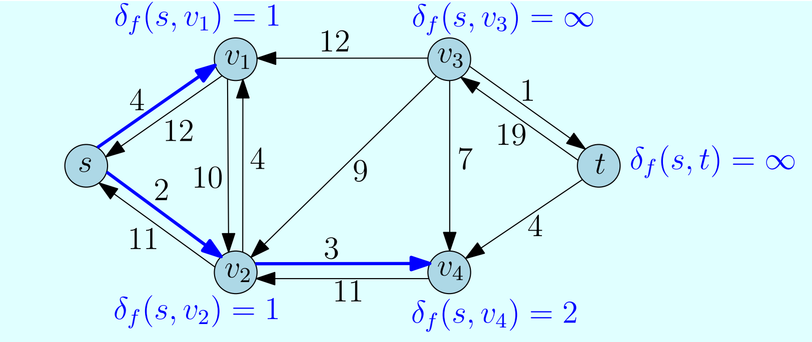
The flow f after pushing 7 units through p.

The Edmonds-Karp Algorithm: Example



The residual network G_f.

The Edmonds-Karp Algorithm: Example



Breadth-first search in G_f.

The sink t is unreachable, so the algorithm terminates.

The Edmonds-Karp Algorithm: Proof of the Lemma

We want to prove that for each v, the distance $\delta_f(s, v)$ never decreases during the course of the Edmonds-Karp algorithm.

- When the flow is augmented along p, some edges are created or deleted in G_f, which may affect δ_f.
- We will first apply the insertions, and then the deletions, and see how δ_f is affected.
- So we first consider the new edges.
  - An edge (v, u) may be created if $(u, v) \in p$.
  - But then, before the edge is introduced, $\delta_f(s, v) = \delta_f(s, u) + 1$.
  - So in the resulting graph, a shortest path to u cannot go through v.
  - Therefore, the insertion of edge (v, u) does not affect δ_f.
- So after we insert all the new edges, δ_f is unchanged.
- Then we delete some edges.
  - When we delete an edge, δ_f cannot decrease.
- So overall, $\delta_f(s, v)$ cannot decrease for any vertex v.

The Edmonds-Karp Algorithm: Proof of the Theorem

It suffices to prove that there are $O(|V| \cdot |E|)$ flow augmentations.
Proof done in class.

Integer Values

- If all the capacities $c(u, v)$ are integers, then the Ford-Fulkerson algorithm (both the basic version and the Edmonds-Karp algorithm) never introduces any number that is not an integer. It follows that:

Theorem (integrality theorem)

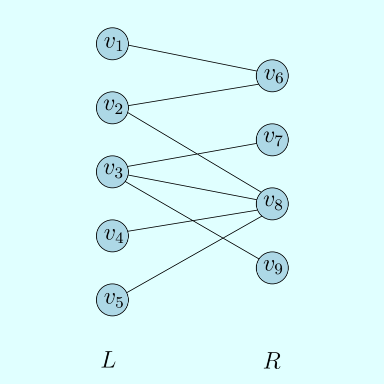
If the capacity function c takes only integral values, then the maximum flow f^* produced by the Ford-Fulkerson method is such that for all u, v, the value $f^*(u, v)$ is an integer. Thus, the value $|f^*|$ of a maximum flow is an integer.

Maximum Bipartite Matching

Definition

A graph $G = (V, E)$ is *bipartite* if its vertex set V can be partitioned into two sets L, R such that $E \subseteq L \times R$.

Example:

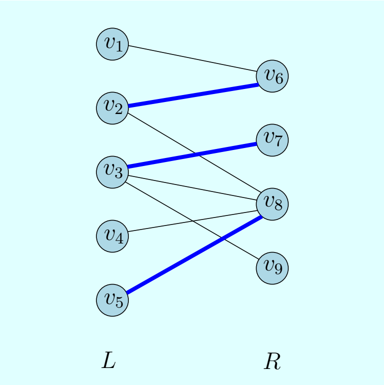


Maximum Bipartite Matching

Definition

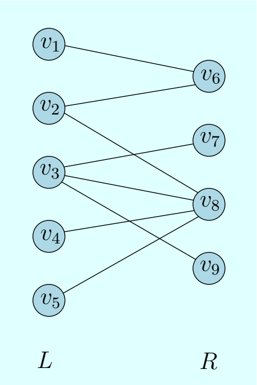
A *maximum bipartite matching* of a bipartite graph G is a matching in G with maximum cardinality.

Example:

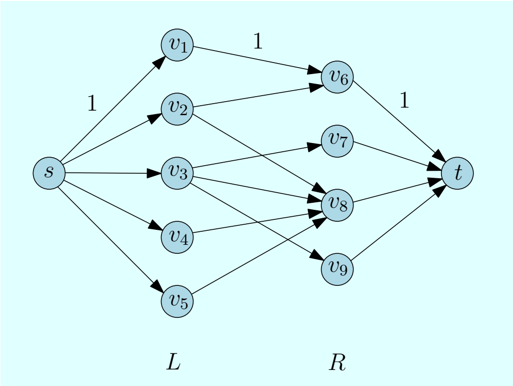


Maximum Bipartite Matching and Maximum Flow

The problem of computing a maximum bipartite matching reduces to computing a maximum flow.

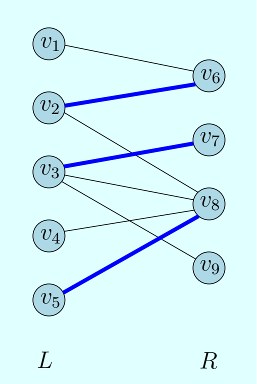


Instance G of maximum bipartite matching.

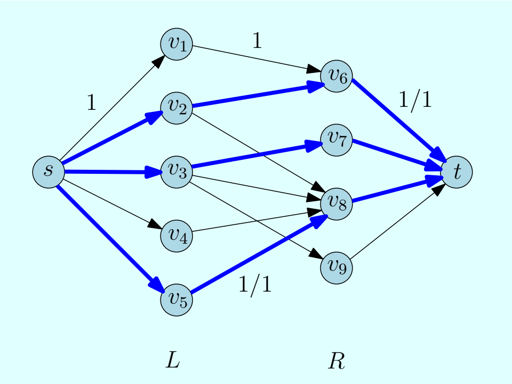


The corresponding flow network G'. All capacities $c(u, v)$ are set to 1.

Maximum Bipartite Matching and Maximum Flow



A maximum bipartite matching in G.



The corresponding maximum flow in G'.

Maximum Bipartite Matching and Maximum Flow

Let $G = (V, E)$ be an instance of maximum bipartite matching, with V partitioned into L, R, and all edges in $L \times R$.

As above, we transform it into a flow network $G'(V', E')$ such that:

- $V' = V \cup \{s, t\}$.
- $E' = E \cup (\{s\} \times L) \cup (R \times \{t\})$.
- $c(u, v) = 1$ for all $(u, v) \in E'$.

We say that a flow f is *integer-valued* if $f(u, v)$ is an integer for all (u, v).

Lemma

If M is a matching in G, then there is an integer-valued flow f in G' with value $|f| = |M|$. Conversely, if f is an integer-valued flow in G', then there is a matching M in G with cardinality $|M| = |f|$.

Proof done in class.

Maximum Bipartite Matching and Maximum Flow

So it follows from the integrality theorem that:

Corollary

The cardinality of a maximum matching M^* in a bipartite graph G is the value $|f^*|$ of a maximum flow in the corresponding flow network G'.

Thus, using the Edmonds-Karp algorithm:

Corollary

We can compute a maximum matching in a bipartite graph $G(V, E)$ in time $O(|V| \cdot |E|^2)$.