How to choose augmenting paths?

- We need to find paths efficiently.
How to choose augmenting paths?

- We need to find paths efficiently.
- We want to guarantee a small number of iterations.
How to choose augmenting paths?

▶ We need to find paths efficiently.
▶ We want to guarantee a small number of iterations.

Several possibilities:
How to choose augmenting paths?

▶ We need to find paths efficiently.
▶ We want to guarantee a small number of iterations.

Several possibilities:

▶ Choose path with maximum bottleneck capacity.
▶ Choose path with sufficiently large bottleneck capacity.
▶ Choose the shortest augmenting path.
Intuition:

▶ Choosing a path with the highest bottleneck increases the flow as much as possible in a single step.

▶ Don’t worry about finding the exact bottleneck.

▶ Maintain scaling parameter $\Delta$.

$G_f(\Delta)$ is a sub-graph of the residual graph $G_f$ that contains only edges with capacity at least $\Delta$. 
Capacity Scaling

Intuition:

- Choosing a path with the highest bottleneck increases the flow as much as possible in a single step.
Capacity Scaling

Intuition:

▶ Choosing a path with the highest bottleneck increases the flow as much as possible in a single step.
▶ Don’t worry about finding the exact bottleneck.
Capacity Scaling

Intuition:

- Choosing a path with the highest bottleneck increases the flow as much as possible in a single step.
- Don’t worry about finding the exact bottleneck.
- Maintain scaling parameter $\Delta$. 
Capacity Scaling

Intuition:

- Choosing a path with the highest bottleneck increases the flow as much as possible in a single step.
- Don’t worry about finding the exact bottleneck.
- Maintain scaling parameter $\Delta$.
- $G_f(\Delta)$ is a sub-graph of the residual graph $G_f$ that contains only edges with capacity at least $\Delta$. 
Capacity Scaling

Intuition:

▶ Choosing a path with the highest bottleneck increases the flow as much as possible in a single step.
▶ Don’t worry about finding the exact bottleneck.
▶ Maintain scaling parameter $\Delta$.
▶ $G_f(\Delta)$ is a sub-graph of the residual graph $G_f$ that contains only edges with capacity at least $\Delta$. 

\[ G_f(\Delta) \]

$G_f(99)$

$G_f$
Algorithm 2 maxflow($G, s, t, c$)

1: foreach $e \in E$ do $f_e \leftarrow 0$;
2: $\Delta \leftarrow 2^{\lceil \log_2 C \rceil}$
3: while $\Delta \geq 1$ do
4: $G_f(\Delta) \leftarrow \Delta$-residual graph
5: while there is augmenting path $P$ in $G_f(\Delta)$ do
6: $f \leftarrow$ augment($f, c, P$)
7: update($G_f(\Delta)$)
8: $\Delta \leftarrow \Delta/2$
9: return $f$
Capacity Scaling

Assumption: All capacities are integers between 1 and C.

Invariant: All flows and capacities are/remain integral throughout the algorithm.

Correctness: The algorithm computes a maxflow:

▶ because of integrality we have $G_f(1)$

▶ therefore after the last phase there are no augmenting paths anymore

▶ this means we have a maximum flow.
Capacity Scaling

Assumption:
All capacities are integers between 1 and $C$. 
Capacity Scaling

Assumption:
All capacities are integers between 1 and $C$.

Invariant:
All flows and capacities are/remain integral throughout the algorithm.
Capacity Scaling

Assumption:
All capacities are integers between 1 and $C$.

Invariant:
All flows and capacities are/remain integral throughout the algorithm.

Correctness:
The algorithm computes a maxflow:

- because of integrality we have $G_f(1) = G_f$
Capacity Scaling

Assumption:
All capacities are integers between 1 and \( C \).

Invariant:
All flows and capacities are/remain integral throughout the algorithm.

Correctness:
The algorithm computes a maxflow:
- because of integrality we have \( G_f(1) = G_f \)
- therefore after the last phase there are no augmenting paths anymore
Capacity Scaling

Assumption:
All capacities are integers between 1 and $C$.

Invariant:
All flows and capacities are/remain integral throughout the algorithm.

Correctness:
The algorithm computes a maxflow:
- because of integrality we have $G_f(1) = G_f$
- therefore after the last phase there are no augmenting paths anymore
- this means we have a maximum flow.
Lemma 1

There are $\lceil \log C \rceil + 1$ iterations over $\Delta$.

Proof: obvious.

Lemma 2

Let $f$ be the flow at the end of a $\Delta$-phase. Then the maximum flow is smaller than $\text{val}(f) + m \Delta$.

Proof: less obvious, but simple:

1. There must exist an $s$-$t$ cut in $G_f(\Delta)$ of zero capacity.
2. In $G_f$ this cut can have capacity at most $m \Delta$.
3. This gives me an upper bound on the flow that I can still add.

11.3 Capacity Scaling 11. Apr. 2018

Ernst Mayr, Harald Räcke
Lemma 1

There are $\lceil \log C \rceil + 1$ iterations over $\Delta$.

Proof: obvious.

Lemma 2

Let $f$ be the flow at the end of a $\Delta$-phase. Then the maximum flow is smaller than $val(f) + m\Delta$.

Proof: less obvious, but simple:

1. There must exist an $s$-$t$ cut in $G_f(\Delta)$ of zero capacity.
2. In $G_f$ this cut can have capacity at most $m\Delta$.
3. This gives me an upper bound on the flow that I can still add.
Capacity Scaling

Lemma 1
There are \( \lceil \log C \rceil + 1 \) iterations over \( \Delta \).

Proof: obvious.

Lemma 2
Let \( f \) be the flow at the end of a \( \Delta \)-phase. Then the maximum flow is smaller than \( \text{val}(f) + m\Delta \).

Proof: less obvious, but simple:
Lemma 1

There are ⌈log C⌉ + 1 iterations over Δ.

Proof: obvious.

Lemma 2

Let f be the flow at the end of a Δ-phase. Then the maximum flow is smaller than val(f) + mΔ.

Proof: less obvious, but simple:
- There must exist an s-t cut in G_f(Δ) of zero capacity.
Capacity Scaling

Lemma 1

There are \( \lceil \log C \rceil + 1 \) iterations over \( \Delta \).

Proof: obvious.

Lemma 2

Let \( f \) be the flow at the end of a \( \Delta \)-phase. Then the maximum flow is smaller than \( \text{val}(f) + m\Delta \).

Proof: less obvious, but simple:
- There must exist an \( s-t \) cut in \( G_f(\Delta) \) of zero capacity.
- In \( G_f \) this cut can have capacity at most \( m\Delta \).
Capacity Scaling

Lemma 1
There are $\lceil \log C \rceil + 1$ iterations over $\Delta$.

Proof: obvious.

Lemma 2
Let $f$ be the flow at the end of a $\Delta$-phase. Then the maximum flow is smaller than $\text{val}(f) + m\Delta$.

Proof: less obvious, but simple:
- There must exist an $s-t$ cut in $G_f(\Delta)$ of zero capacity.
- In $G_f$ this cut can have capacity at most $m\Delta$.
- This gives me an upper bound on the flow that I can still add.
Lemma 3

There are at most \( 2m \) augmentations per scaling-phase.

Proof:

Let \( f \) be the flow at the end of the previous phase.

\[
\text{val}(f^*) \leq \text{val}(f) + 2m \Delta
\]

Each augmentation increases flow by \( \Delta \).

Theorem 4

We need \( O(m \log C) \) augmentations. The algorithm can be implemented in time \( O(m^2 \log C) \).
Lemma 3

There are at most $2m$ augmentations per scaling-phase.
Capacity Scaling

Lemma 3
There are at most $2m$ augmentations per scaling-phase.

Proof:
- Let $f$ be the flow at the end of the previous phase.
Lemma 3

There are at most $2m$ augmentations per scaling-phase.

Proof:

- Let $f$ be the flow at the end of the previous phase.
- $\text{val}(f^*) \leq \text{val}(f) + 2m\Delta$
Lemma 3

There are at most $2m$ augmentations per scaling-phase.

Proof:

- Let $f$ be the flow at the end of the previous phase.
- $\text{val}(f^*) \leq \text{val}(f) + 2m\Delta$
- Each augmentation increases flow by $\Delta$. 
Lemma 3
There are at most $2m$ augmentations per scaling-phase.

Proof:
- Let $f$ be the flow at the end of the previous phase.
- $\text{val}(f^*) \leq \text{val}(f) + 2m\Delta$
- Each augmentation increases flow by $\Delta$.

Theorem 4
We need $\Theta(m \log C)$ augmentations. The algorithm can be implemented in time $\Theta(m^2 \log C)$.