Algorithmen für Min Cut und Max Flow in ungerichteten planaren Graphen

Robin Münstermann

18. Juni 2018

Inhaltsübersicht

Motivation

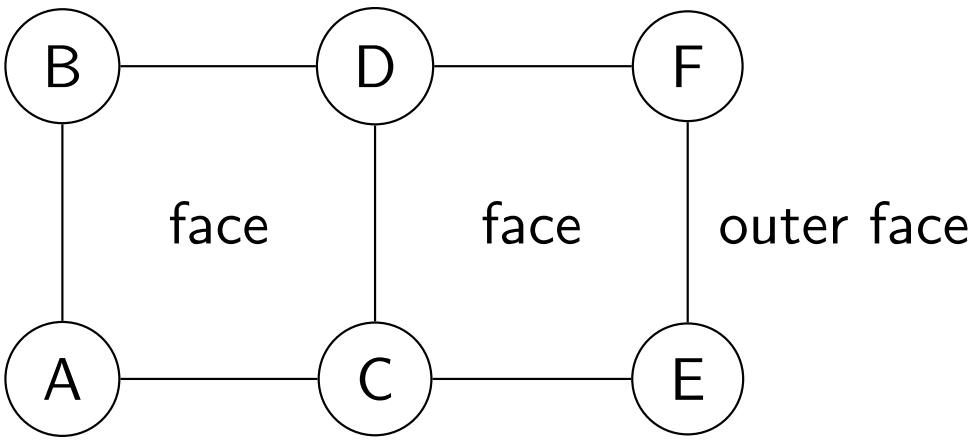
Problem: Bei gegebenem ungerichtetem Graph Max Flow und Min Cut bestimmen.

- Verbesserung von Reif's Algorithmus für Min-Cut von $O(n \log^2 n)$ zu $O(n \log \log n)$.
- Aus dem Min Cut einen Max Flow bestimmen.

Einführende Begriffe

face

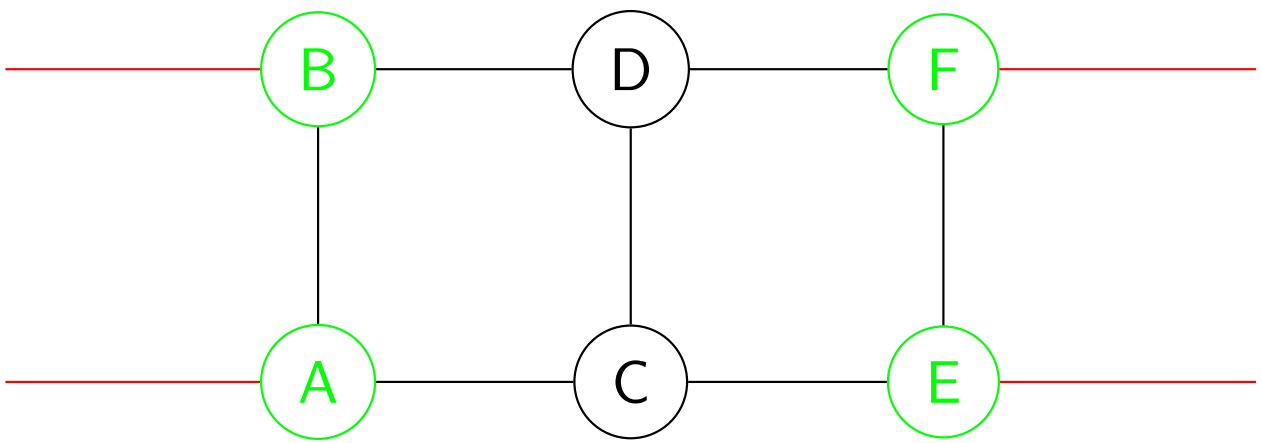
face (Fläche) bezeichnet eine Region, die sich durch angrenzenden Kanten und Knoten ergibt.



Einführende Begriffe

piece

Ein piece (Teilgraph) P auf einem Graphen G wird durch eine Kantenmenge $E \subseteq G$ gegeben. Dabei werden die Knoten, die inzident zu Knoten nicht in P sind, als boundary vertices (Grenzknoten) bezeichnet. Die übrigen inneren Knoten werden interior vertices genannt.



r-Division

r-Division

G wird in $O(n/r)$ pieces unterteilt, wobei jedes piece $O(r)$ Knoten hat und $O(\sqrt{r})$ boundary vertices.

Holes

Holes (Löcher) von einem piece P sind faces auf P, die nicht faces von G sind und keine outer faces sind.

Im Algorithmus wird r passend gewählt.

Theorem 1

Theorem 1

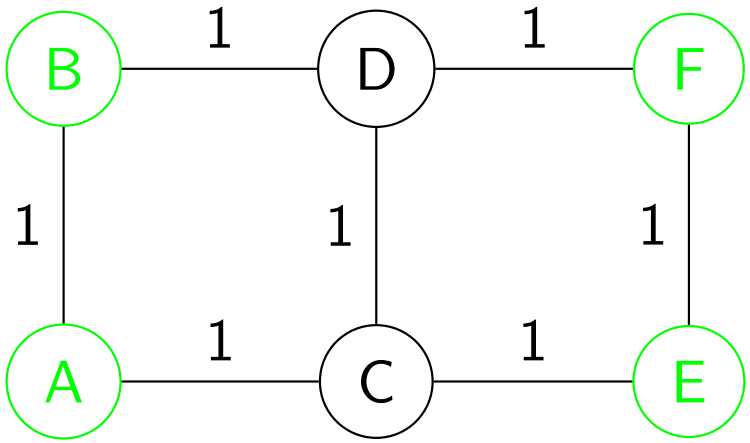
Für einen planaren Graph $G = (V, E)$ kann eine r-division in $O(n \log r + (n/\sqrt{r}) \log n)$ so gefunden werden, dass jeder Teilgraph $O(1)$ holes hat.

Dense Distance Graph

Dense Distance Graph

Der Dense Distance Graph von einem piece P ist der vollständige Graph auf den boundary vertices von P. Die Kantengewichte sind die kürzesten Weg-Distanzen von zwei Knoten (u, v).

Seien A, B, F, G boundary vertices

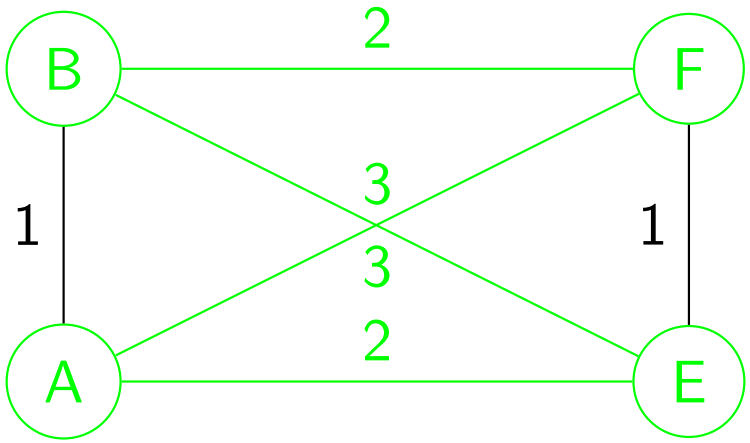


Dense Distance Graph

Dense Distance Graph

Der Dense Distance Graph von einem piece P ist der vollständige Graph auf den boundary vertices von P. Die Kantengewichte sind die kürzesten Weg-Distanzen von zwei Knoten (u, v).

Seien A, B, F, G boundary vertices



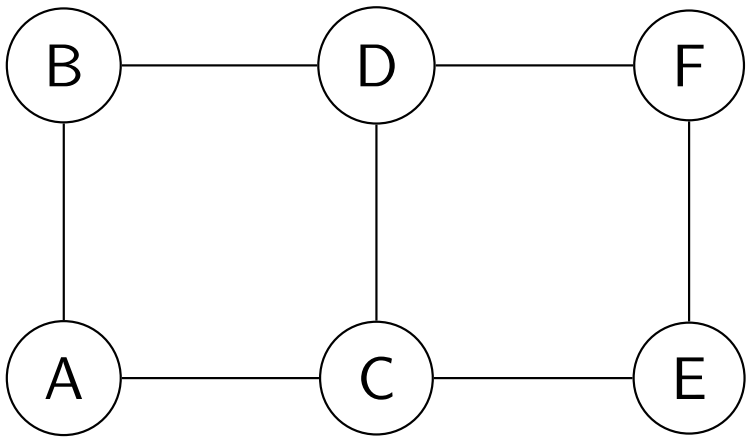
Laufzeit von Klein's Algorithmus

- Durch die r-Division: $O(n/r)$ pieces
- Laufzeit für ein piece P: $O(r \log(r))$.
- Das Berechnen der Dense Distance Graphen aller pieces: $O(n/r) * O(r \log(r)) = O(n \log(r))$ bei gegebener r-Division.

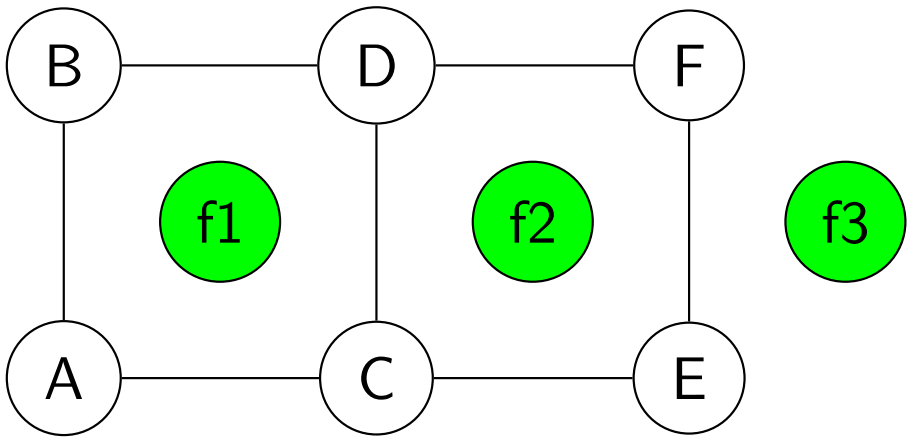
Fast Dijkstra

- Benutze Dense Distance Graph, um Dijkstra schneller auszuführen
- Sei b die gesamte Anzahl von boundary vertices (bei uns $O(\sqrt{r})$)
- Laufzeit: $O(b \log^2 n) = O(\sqrt{r} \log^2 n)$

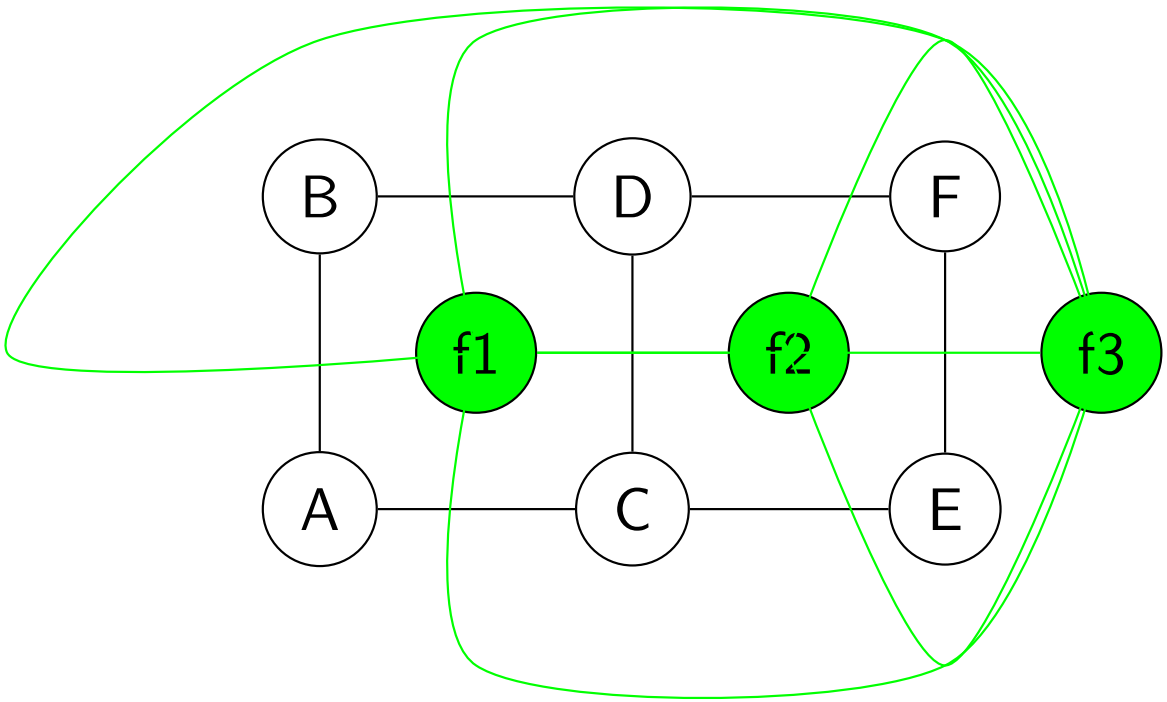
Dualer Graph



Dualer Graph



Dualer Graph



Gewichte der Kanten im primären entsprechen den Gewichten der Kanten des dualen Graphs.

Min st-seperating cycle

Lemma 4. Min st-seperating cycle

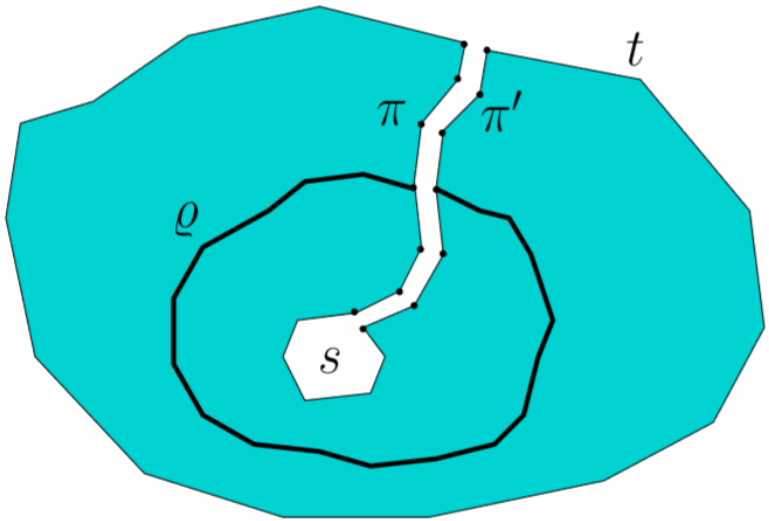
Ein min st-seperating cycle im dualen Graph G^* definiert einen min st-cut im primären Graph G.

Dadurch folgt die Idee von Reif's Algorithmus:
Berechne den min st-cut über den min st-seperating cycle des dualen Graphs.

Reif's Algorithmus

Reif's Algorithmus

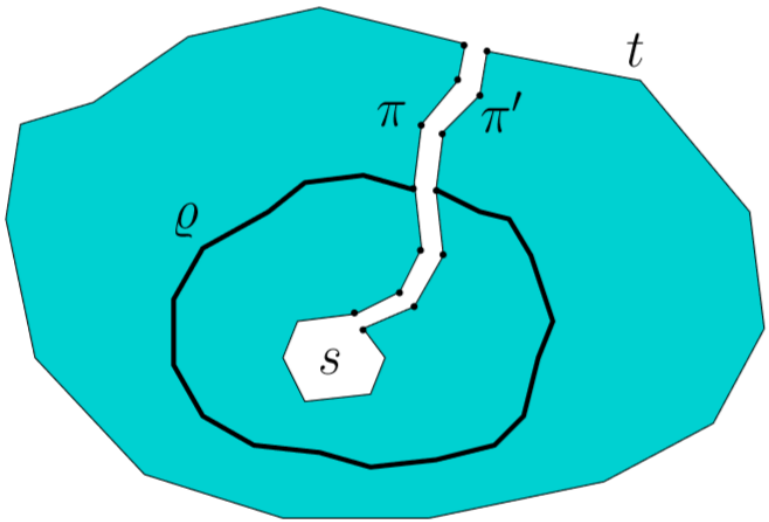
1. Berechne kürzesten Weg π von einem Knoten p_1 von face s zu einem Knoten $p_{|\pi|}$ von face t.
2. Incision entlang π.
   1. Lösche alle Kanten E_r, die rechts von π ausgehen.
   2. Kopie π' von π.
   3. Füge E_r ein, nur statt $p_i \in \pi$ setze diesen Knoten als $p_i' \in \pi'$.
3. Berechne kürzesten Pfad ϱ von $p_{|\pi|/2}$ nach $p_{|\pi|'/2}$.



Reif's Algorithmus

Reif's Algorithmus

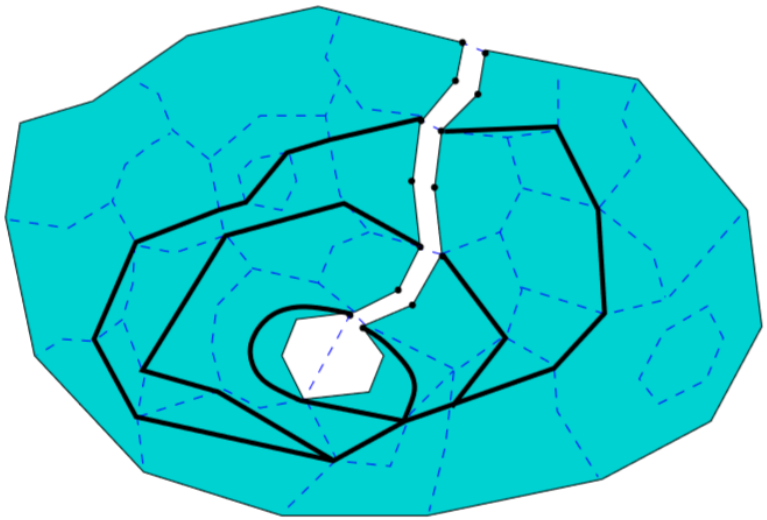
4. Rufe Schritt 3 rekursiv auf den Teilgraphen auf, die durch ϱ entstanden sind.
5. Von allen $p_i \in \pi$ sind dann die min cycle berechnet. Ein ϱ_i ist der min st-seperating cycle.



Laufzeit mit normalen Dijkstra $O(n \log^2 n)$.
Mit schnellerer kürzester Pfadsuche wie zum Beispiel mit Frederickson's Algorithmus kann dieser Algorithmus in $O(n \log n)$ ausgeführt werden.

Faster Min st-Cut Algorithmus

- Idee: Reifs Algorithmus mit Fast Dijkstra.
- π finden: $O(n)$
- Aufteilen in zwei Phasen:
  1. Reif wird nur mit den boundary vertices ausgeführt.
  2. Verfeinerte Version von Reif's Algorithmus anwenden auf die Subgraphen, um die exakten Pfade zu finden.
- Laufzeit $O(n \log \log n)$



Faster Min st-Cut Algorithmus

Ziel: Laufzeit in $O(n \log \log n)$.

Finden des min st-seperating cycle bei unterteiltem Graphen:

Lemma 5, Lösung eines Subproblems

Seien faces s und t, sowie ein Pfad π zwischen diesen gegeben. Ein Subproblem auf einem Subgraphen H und einem Subpfad π' von π mit $\pi' = O(log^c n)$ und c konstant kann in $O(|H| \log \log n)$ gelöst werden.

Beweis

Lemma 5, Lösung eines Subproblems

Seien faces s und t, sowie ein Pfad π zwischen diesen gegeben. Ein Subproblem auf einem Subgraphen H und einem Subpfad π' von π mit $\pi' = O(log^c n)$ und c konstant kann in $O(|H| \log\log n)$ gelöst werden.

Beweis:

- Rekursionstiefe in H bei Reif's Algorithmus ist $O(\log(\log^c n)) = O(\log\log n)$. Maximallänge eines Weges durch Dense Distance Graph: $\log^c n$ mit $r = \log^c n$
- Dann kann man das Subbroblem $O(|H| \log\log n)$ mithilfe eines kürzesten Pfad Algorithmus lösen.

Erste Phase

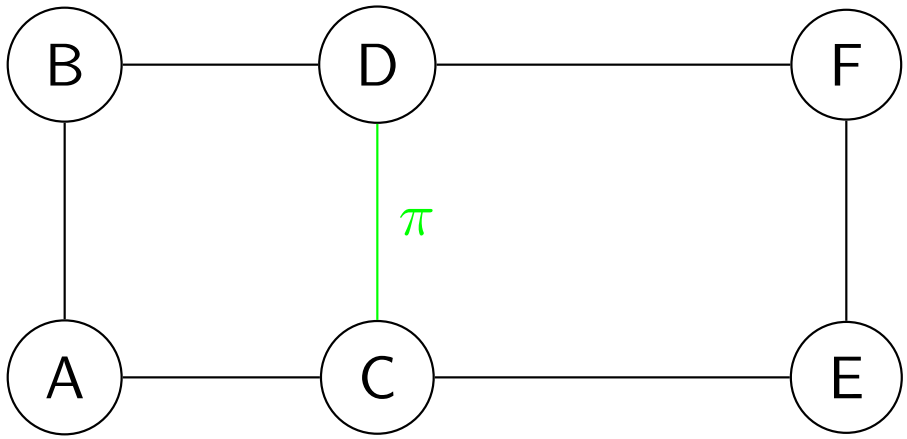
r-Division

$r = \log^6 n$
Nach Theorem 1 $O(n \log r + (n/\sqrt{r}) \log n) =$
$O(6n \log \log n + (n/log^6 n) \log n) =$
$O(n \log \log n)$

Cutting Pieces Open und Problem Overlapping Subgraphs

- Incision durch kürzesten Pfad π (wie bei Reif)
- Dadurch entsteht Incision in Pieces
- Knoten am Pfad werden zwar verdoppelt, aber insgesamt gehören die boundary vertices zum gleichen Piece.
- Gleichbleibend: $O(\sqrt{r})$ boundary vertices und konstante Anzahl von holes.

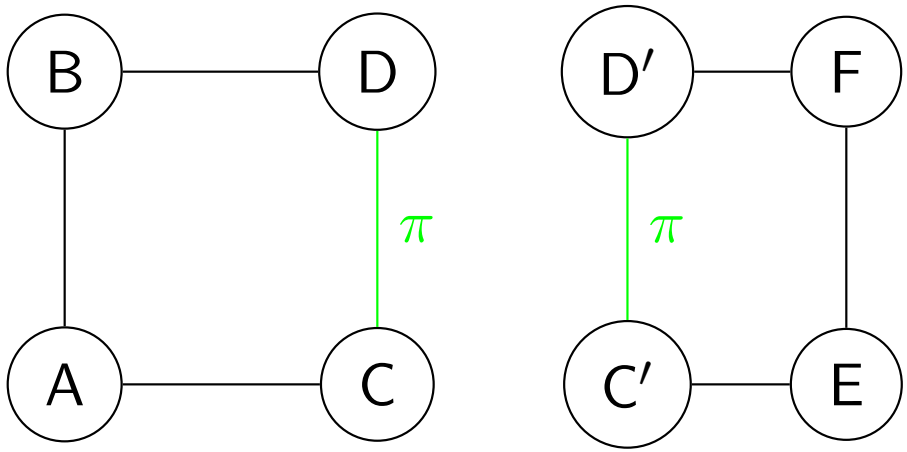
Dense Distance Graph eines pieces:



Cutting Pieces Open und Problem Overlapping Subgraphs

- Incision durch kürzesten Pfad π (wie bei Reif)
- Dadurch entsteht Incision in Pieces
- Knoten am Pfad werden zwar verdoppelt, aber insgesamt gehören die boundary vertices zum gleichen Piece.
- Gleichbleibend: $O(\sqrt{r})$ boundary vertices und konstante Anzahl von holes.

Dense Distance Graph eines pieces:



Alle boundary vertices gehören immer noch zu einem piece

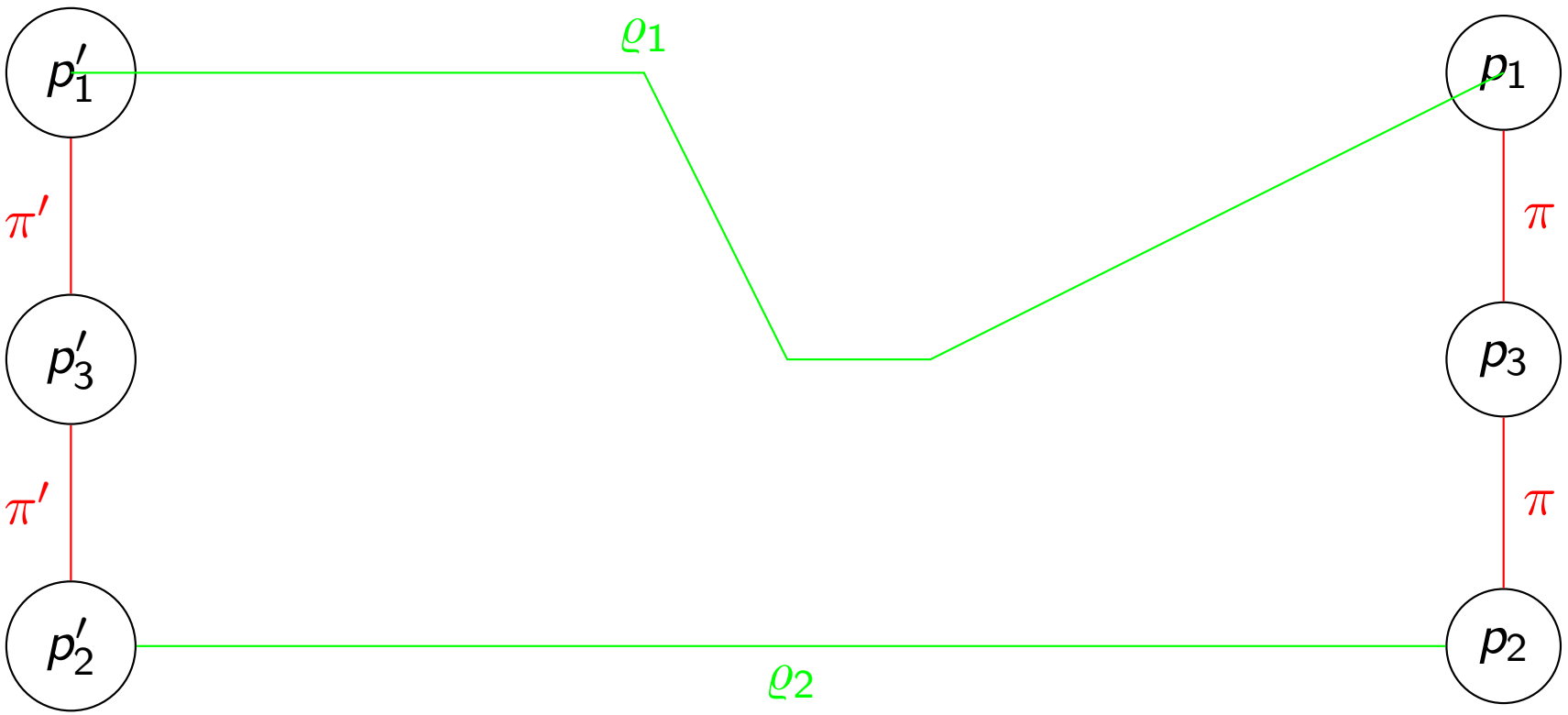
Dense Distance Graph

- Die Dense Distance Graphen können in $O(n \log r)$ $= O(n \log \log n)$ (Klein's Algorithmus) berechnet werden.
- $O(n/\sqrt{r})$ boundary vertices insgesamt
- Mit Fast Dijkstra kürzeste Wege von boundary vertice zu boundary vertice Berechnung: $O((n/\sqrt{r}) \log^2 n) = O(n/\log n)$
- Implizite Darstellung der kürzesten Wege

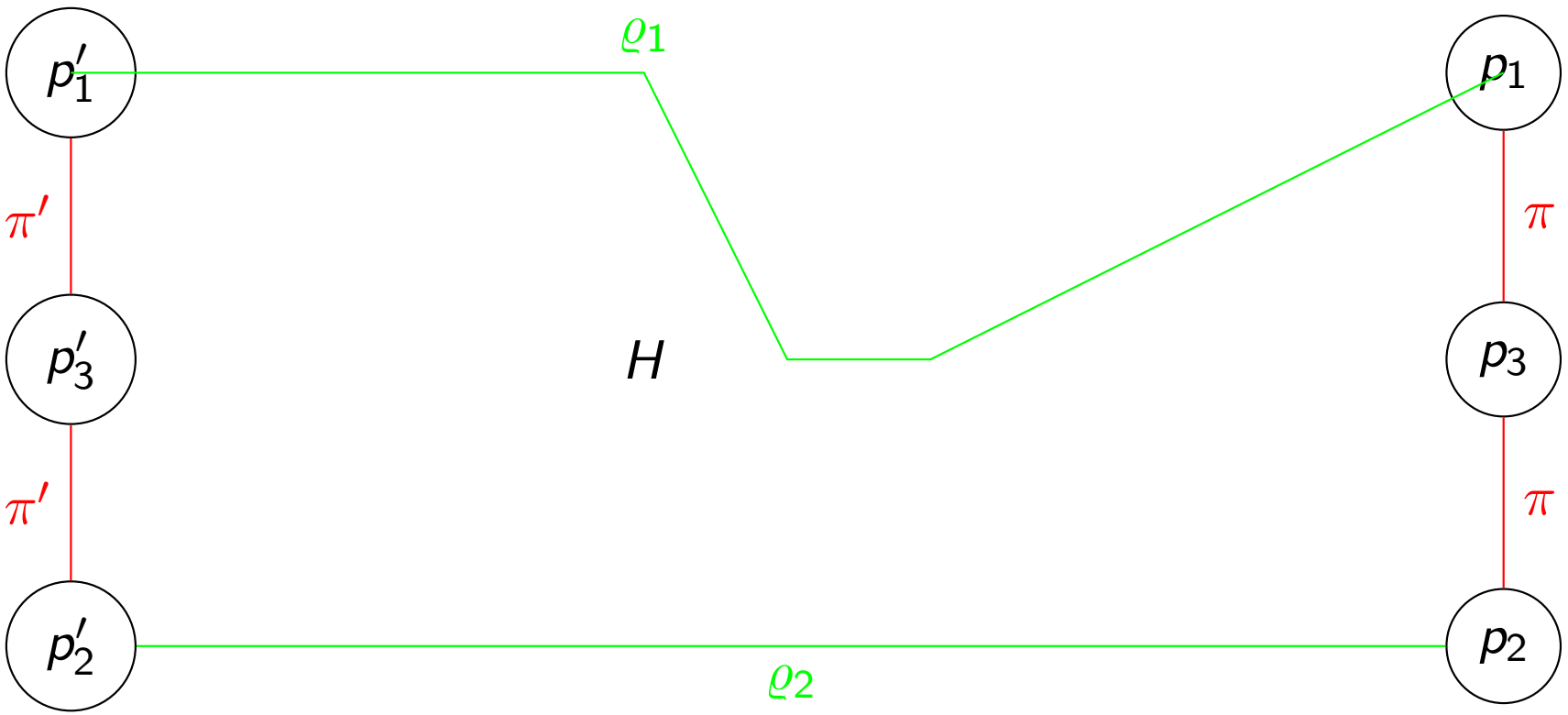
Grobe Version von Reif Laufzeit

- Von boundary vertex zu boundary vertex: $O(n/\log n)$ (durch Dense Distance Graph)
- Rekursions Tiefe ist $O(\log n)$
- Laufzeit: $O(\log n(n/\log n)) = O(n)$
- Achtung wir müssen sicherstellen, dass wir insgesamt nicht mehr als $O(n/\sqrt{r})$ boundary vertices betrachten.

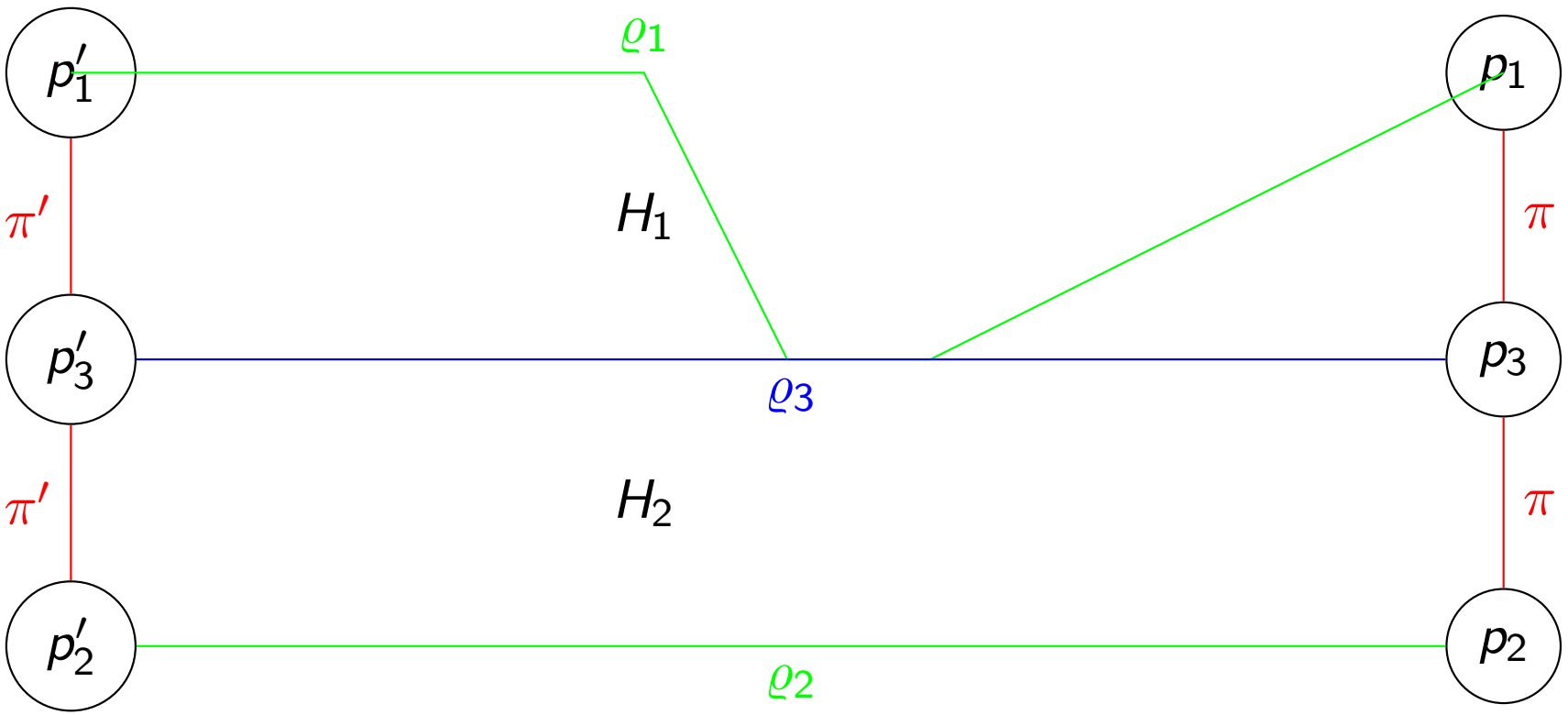
Erste Phase



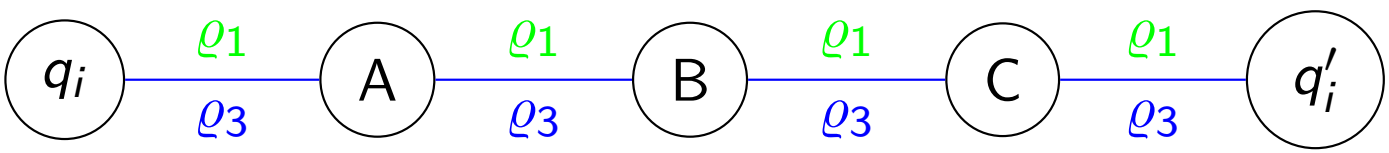
Erste Phase



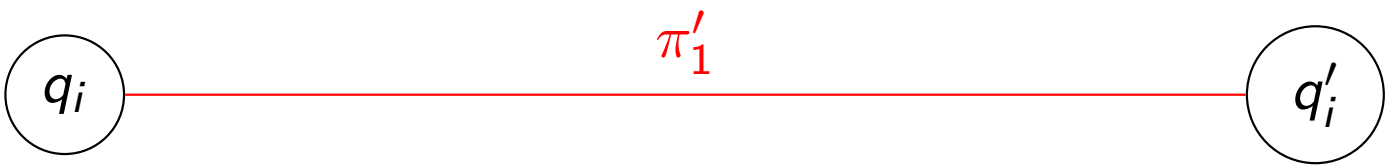
Erste Phase



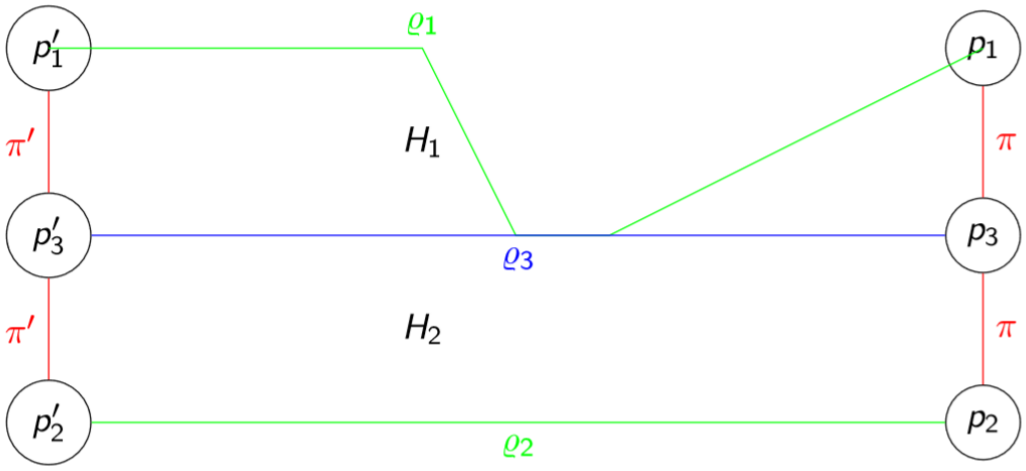
Erste Phase


q_i
ϱ_1
ϱ_3
A
ϱ_1
ϱ_3
B
ϱ_1
ϱ_3
C
ϱ_1
ϱ_3
q_i'

Erste Phase



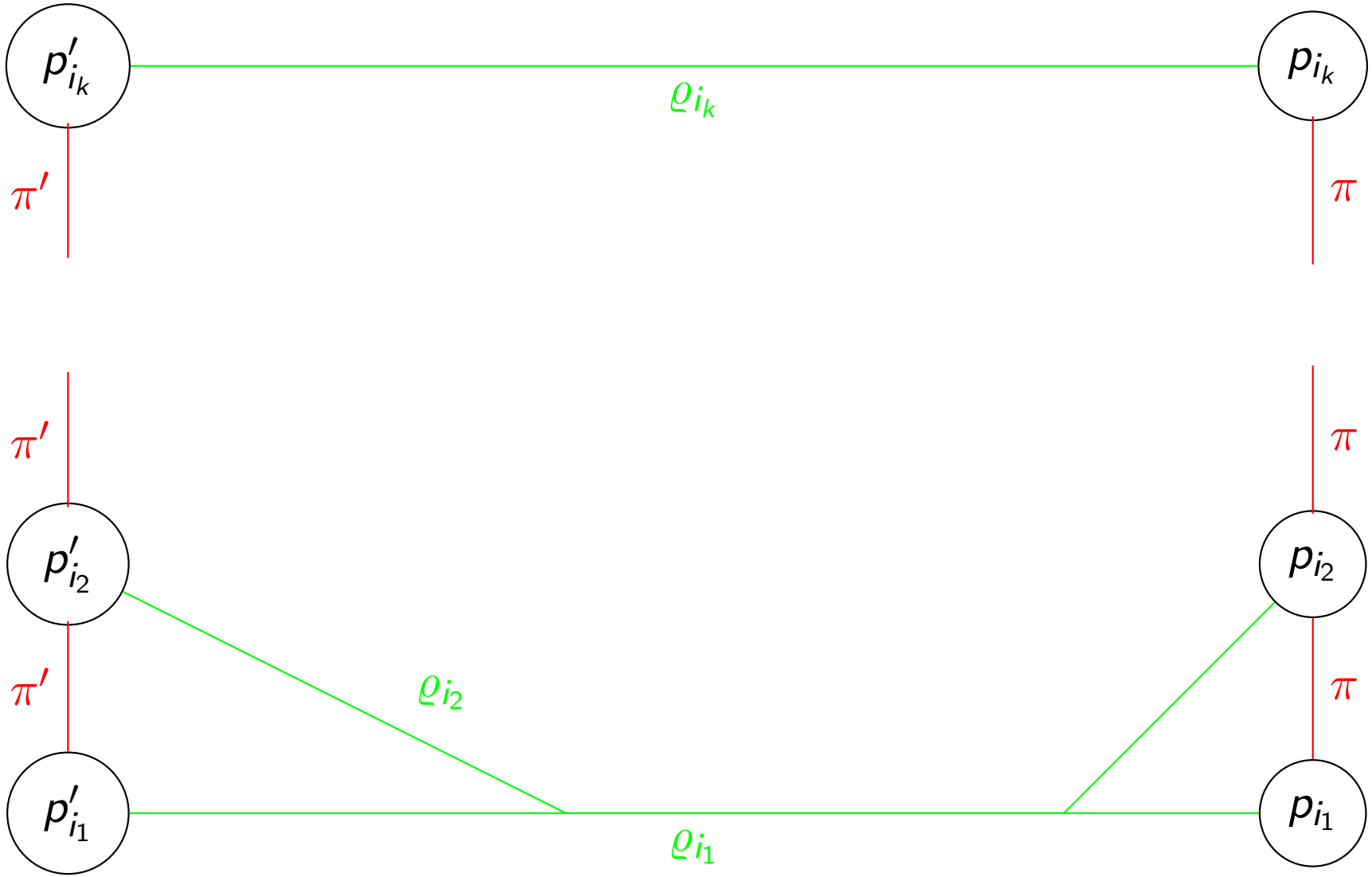
Erste Phase

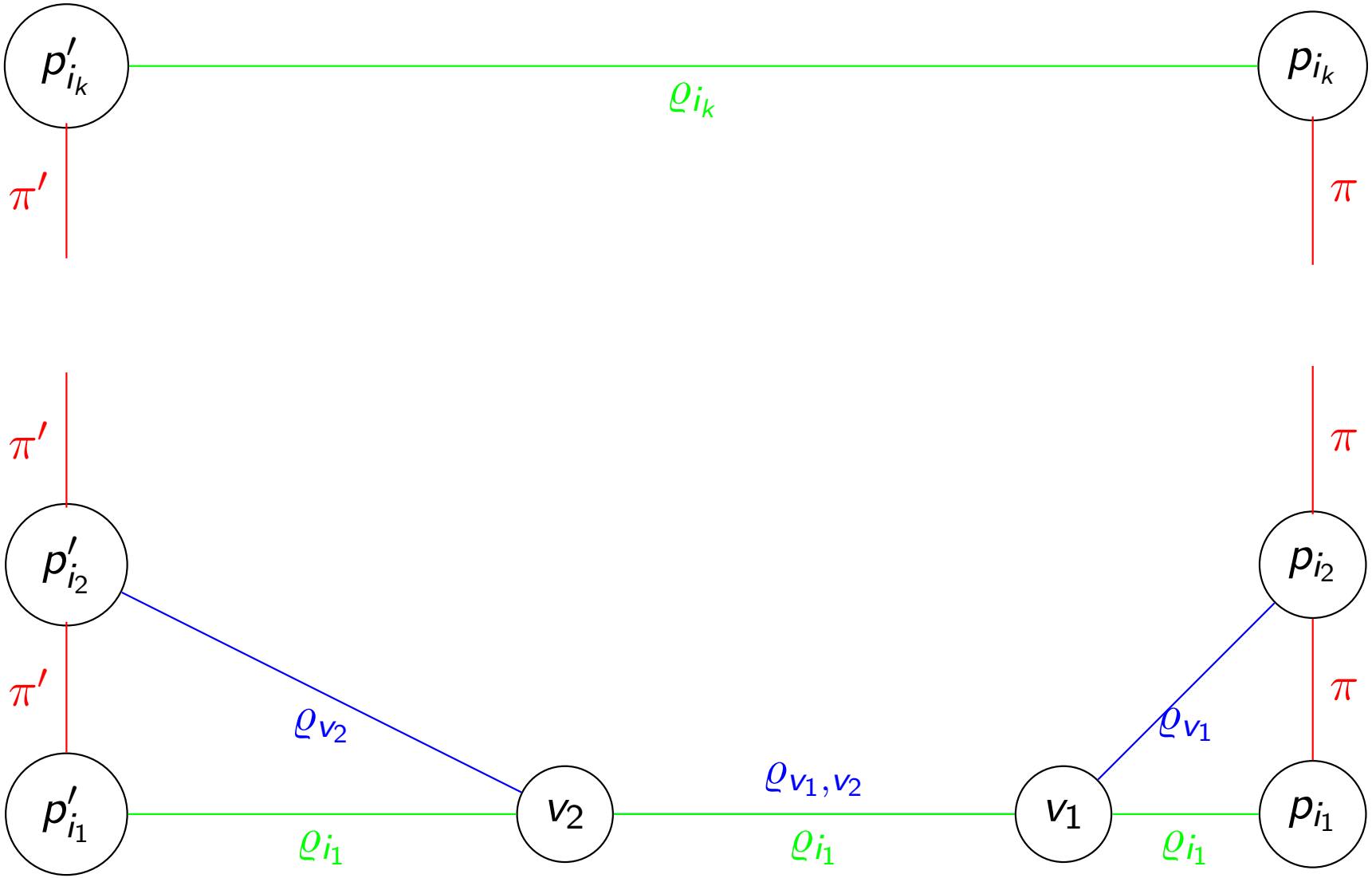


- Maximal 4 Endknoten (π'_1 und π'_2).
- Alle Knoten von $\varrho_3 \setminus (\pi'_1 \cup \pi'_2)$ sind in H_1 und H_2 und in keinen anderen Subgraphen.
- Es folgt: Die Größe der Subgraphen ist begrenzt durch ein Vielfaches der Anzahl von boundary vertices plus die Anzahl wie oft der Graph unterteilt wurde.

Zweite Phase

- Dadurch, dass G konstanten Grad hat, kann die implizite Darstellung in linearer Zeit, abhängig von der Pfadlänge, explizit dargestellt werden.
- Die resultierende Größe könnte aber zu groß sein, wenn Pfade zu viele Knoten teilen.
- Zu zeigen: Berechnung einer impliziten Darstellung in $O(n)$.


p'_{i_k}
ϱ_{i_k}
p_{i_k}
π'
π
π'
π
p'_{i_2}
p_{i_2}
π'
ϱ_{i_2}
π
p'_{i_1}
p_{i_1}
ϱ_{i_1}



Laufzeit ist $O(n)$

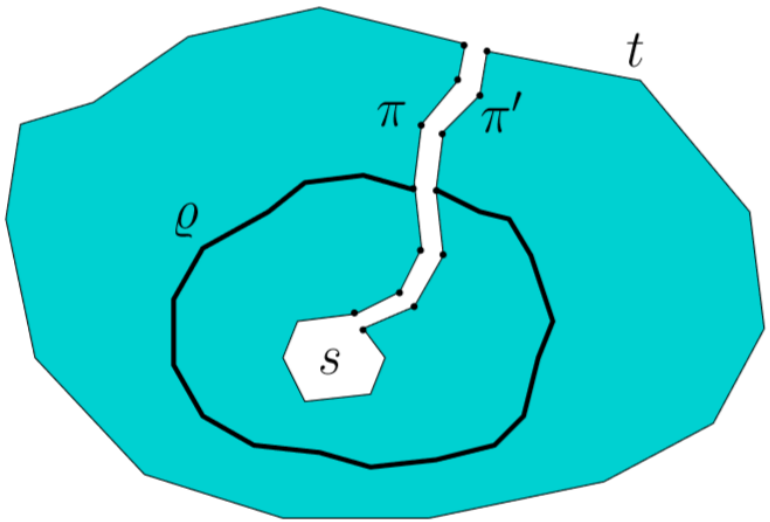
Faster Max st-Flow

- min st-Cut kann in $O(n \log \log n)$ gefunden werden.
- Mithilfe des Wertes des Min Cuts, kann nun in $O(n)$ der Max st-Flow berechnet werden.
- Dies geschieht mit einem Algorithmus von Hassin und Johnson

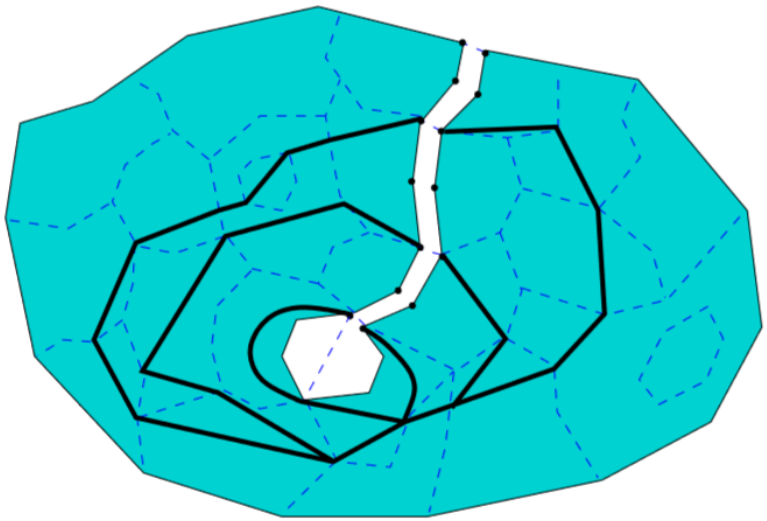
Max st-flow

Ein max st-flow eines ungerichteten planaren Graphen kann in $O(n \log \log n)$ berechnet werden.

Zusammenfassung



Reif's Algorithmus



Faster min st-cut Algorithmus

st-Flow in gerichteten Graphen

- Benutzt auch die Idee, kürzeste Pfade im dualen Graphen zu finden.
- Danach saturiert der Algorithmus die Pfade im Residualgraph mit höchster Kapazität.
- Laufzeit $O(n \log n)$

Multiple Source, Multiple Sink in gerichteten Graphen

- Superquelle und Supersenke kann man nicht einfügen, wenn der Graph planar bleiben soll.
- Komplex, benutzt aber unter anderem
  - Divide und Conquer
  - Dense Distance Graphs
- Laufzeit $O(n \log^3 n)$

Quellen


Glencora Borradaile and Philip Klein. "An O(N Log N) Algorithm for Maximum St-flow in a Directed Planar Graph". In: *J. ACM* 56.2 (Apr. 2009), 9:1–9:30. ISSN: 0004-5411. DOI: 10.1145/1502793.1502798. URL: http://doi.acm.org/10.1145/1502793.1502798.


Glencora Borradaile et al. "Multiple-Source Multiple-Sink Maximum Flow in Directed Planar Graphs in Near-Linear Time". In: ().


Giuseppe F. Italiano et al. "Improved Algorithms for Min Cut and Max Flow in Undirected Planar Graphs". In: STOC '11 (2011), pp. 313–322. DOI: 10.1145/1993636.1993679. URL: http://doi.acm.org/10.1145/1993636.1993679.

Vielen Dank für die Aufmerksamkeit

- Fragen/Anmerkungen?