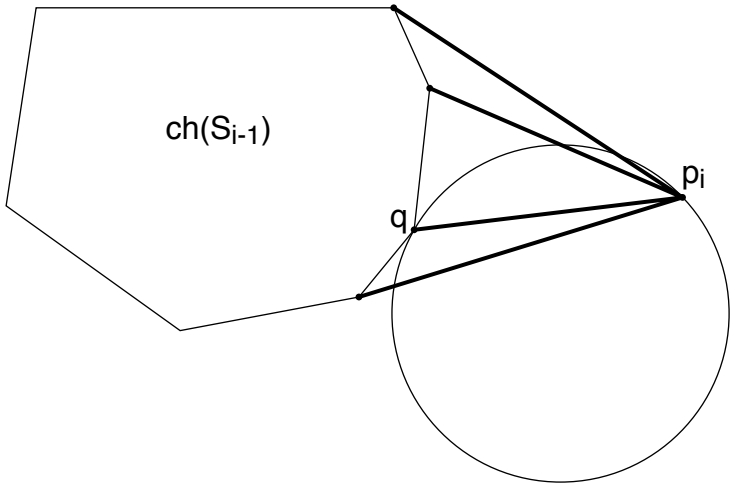
Zusammenfassung Delaunay Triangulation Inkrementelle Konstruktion

Elmar Langetepe
University of Bonn

Inkrementelle Berechnung: Delaunay $p_i \notin$ ch(S_{i-1})

- Alle sichtbaren Kanten sind Delaunay!
- Danach genauso wie vorher, Stern von p_i
- Unendliche Dreicke: UK$(p, q, \infty) = H(p, q)$



Beweis: Kreise durch p_i und q die leer sind! Konstruktiv, Tafel!

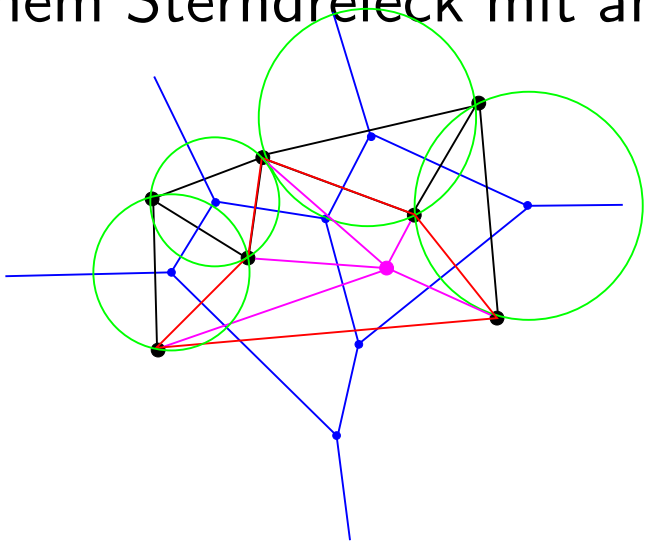
Inkrementelle Berechnung: Terminierung!

Übungsaufgabe 6.3: Falls für alle Umkreise am Rand des Sterns von p_i gilt, dass sie p_i nicht enthalten, kann das auch kein anderer Umkreis.

Beweis: Sterndreiecke(rot/magenta), alle Kanten gehören zu DT_i

Kein Konflikt von p_i mit weiterem Dreieck möglich!

Kein Konflikt von einem Sterndreieck mit anderem Punkt möglich!



Inkrementelle Berechnung: Algorithmus!

Sukzessive p_i in DT_{i-1} einfügen

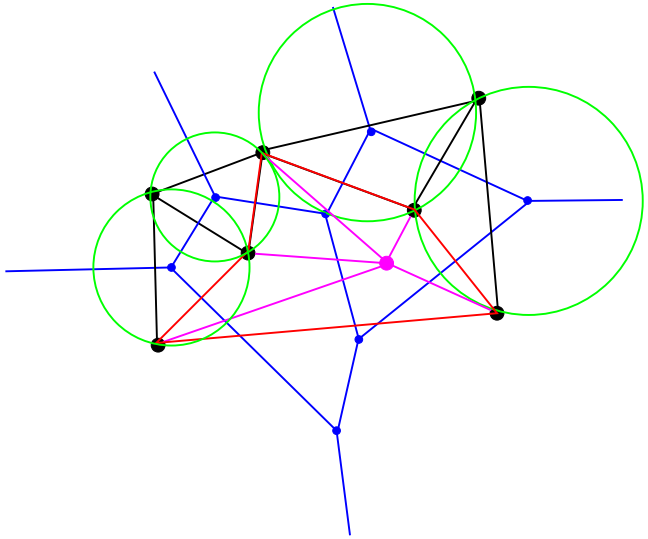
1. Lokalisiere p_i in DT_{i-1}
2. Bestimme Ausgangs-Stern von p_i
3. Sukzessive Edge-Flips durchführen

- Frage: Lokalisation?
- Frage: Wieviele Edge-Flips gibt es insgesamt bei allen Einfügeoperationen?
- Abschätzen durch Anzahl Konfliktdreiecke in DT_{i-1} mit p_i

Inkrementelle Berechnung: Abschätzen!

Lemma 6.3 Sei k_i die Anzahl der Dreiecke von DT_{i-1}, die mit p_i in Konflikt stehen. Sei d_i der Grad von p_i in DT_i. Dann gilt:
$k_i + 1 \leq d_i \leq k_i + 2$.

- $p_i \notin \mathsf{ch}(S_{i-1})$: $k_i + 2 = d_i$
- $p_i \in \mathsf{ch}(S_{i-1})$: $k_i + 1 = d_i$



Inkrementelle Berechnung: Abschätzen!

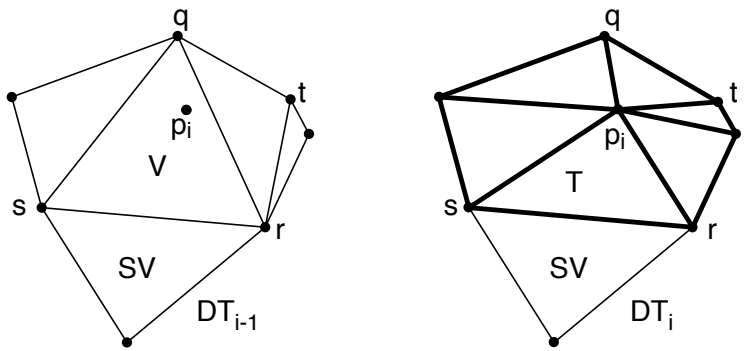
Lemma 6.4 Falls das Dreieck T aus DT_{i-1}, das p_i enthält, bekannt ist, kann DT_i in Zeit $O(d_i)$ berechnet werden.

1. Lokalisation, alle Dreiecke testen: $O(i)$
2. Danach Edge Flips: $O(d_i)$ viele, liegt in $O(i)$
   DCEL Nachbarn in $O(1)$

Korollar 6.5 $\mathsf{DT}(S)$ für $|S| = n$ kann in $O(n^2)$ berechnet werden.

Inkrementelle Berechnung: Verbesserung

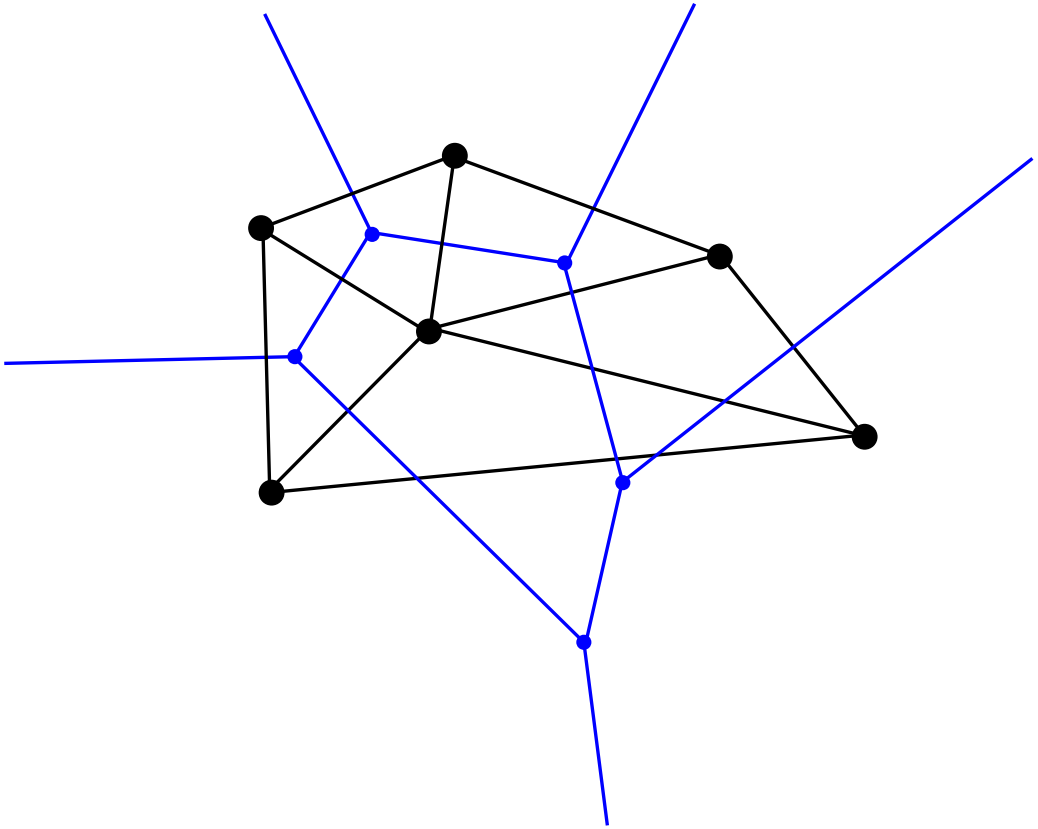
- Lokalisation: Finde p_i in DT_{i-1}
- Verbessern durch Datenstruktur: Delaunay-DAG
- Historie aller Dreiecke, die entstanden sind
- Beispiel: Vater/Stiefvater Beziehung, entlang Kante sr
- T neu DT_i, V Vater verantwortlich, SV Stiefvater angrenzend (Stern)



Weiteres Beispiel: Vater/Stiefvater nach Flip

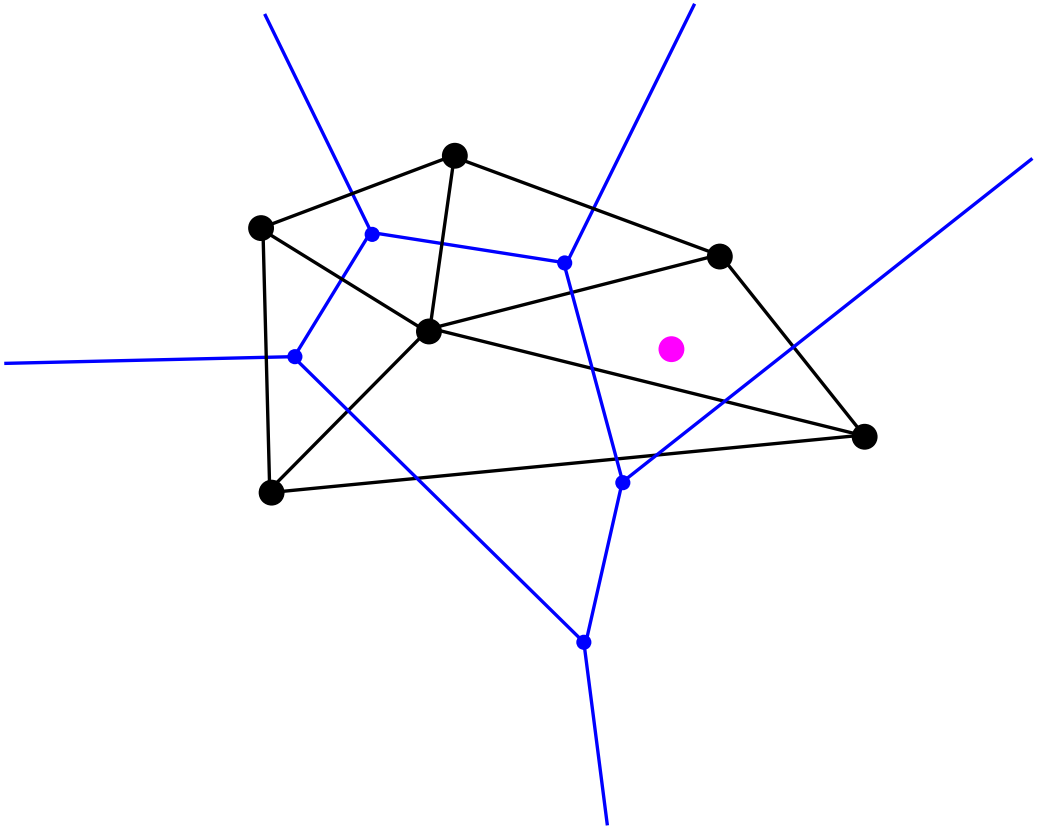
$V(T)$ ist für T verantwortlich, $SV(T)$ grenzt an (Stern)!

Weiteres Beispiel: Vater/Stiefvater nach Flip



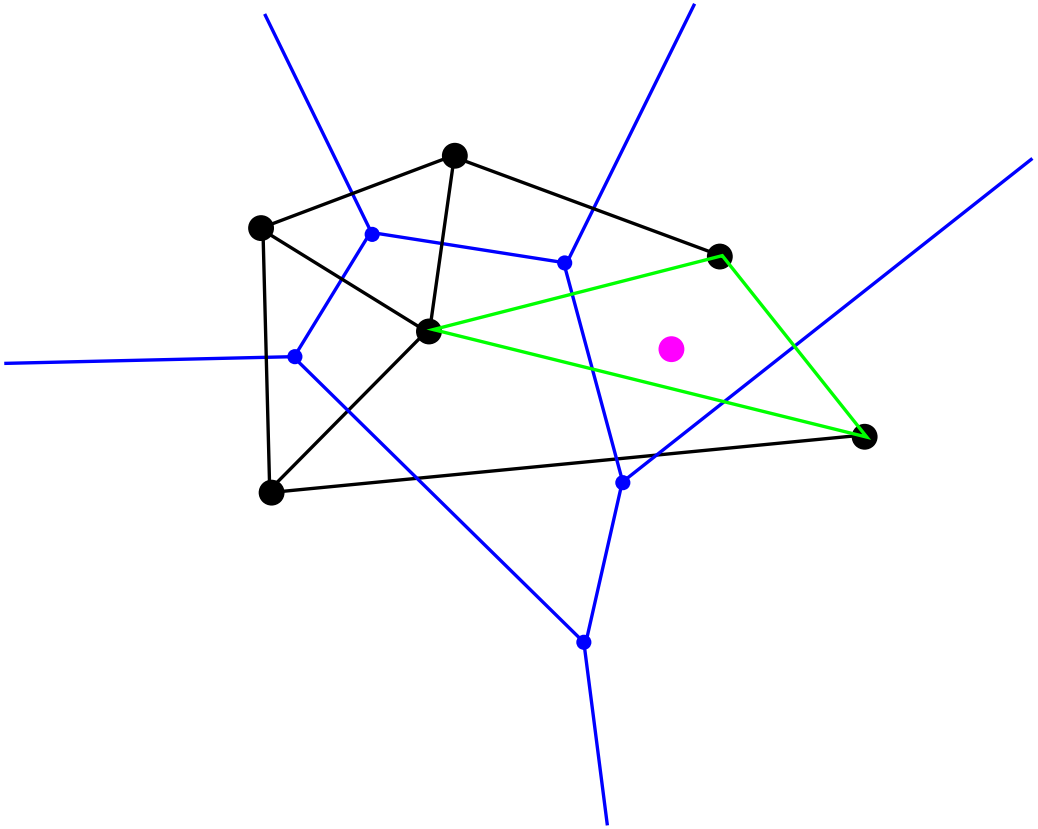
$V(T)$ ist für T verantwortlich, $SV(T)$ grenzt an (Stern)!

Weiteres Beispiel: Vater/Stiefvater nach Flip



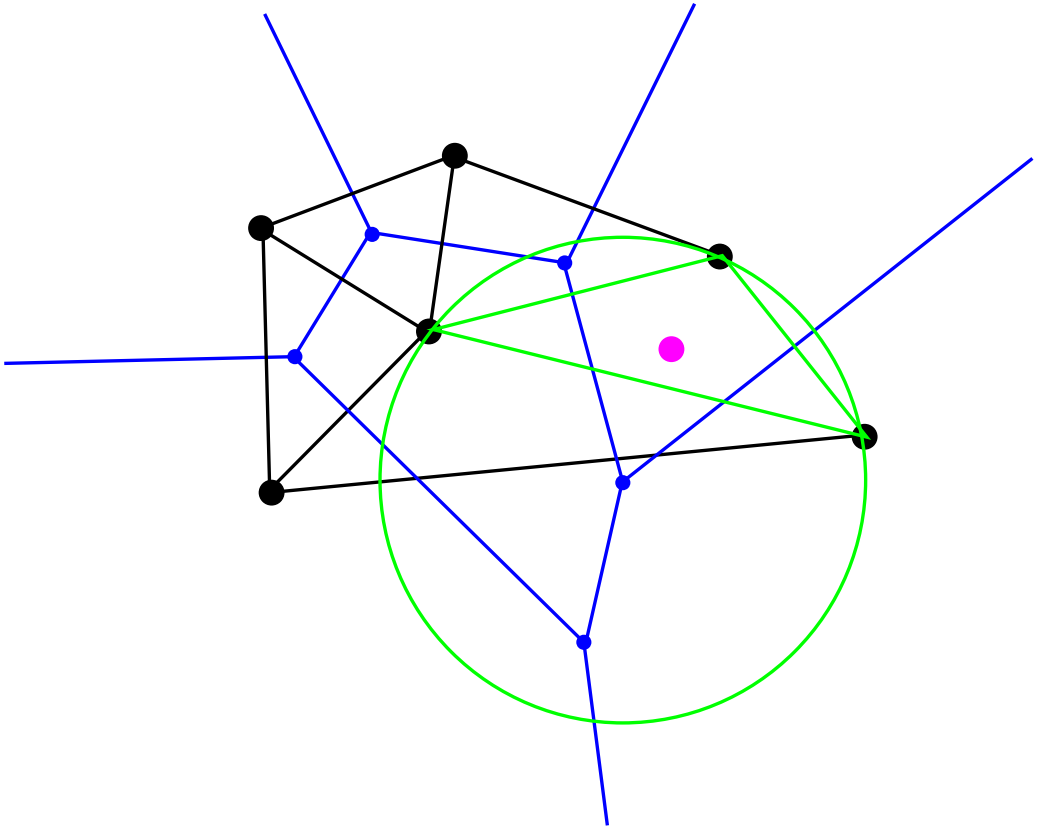
$V(T)$ ist für T verantwortlich, $SV(T)$ grenzt an (Stern)!

Weiteres Beispiel: Vater/Stiefvater nach Flip



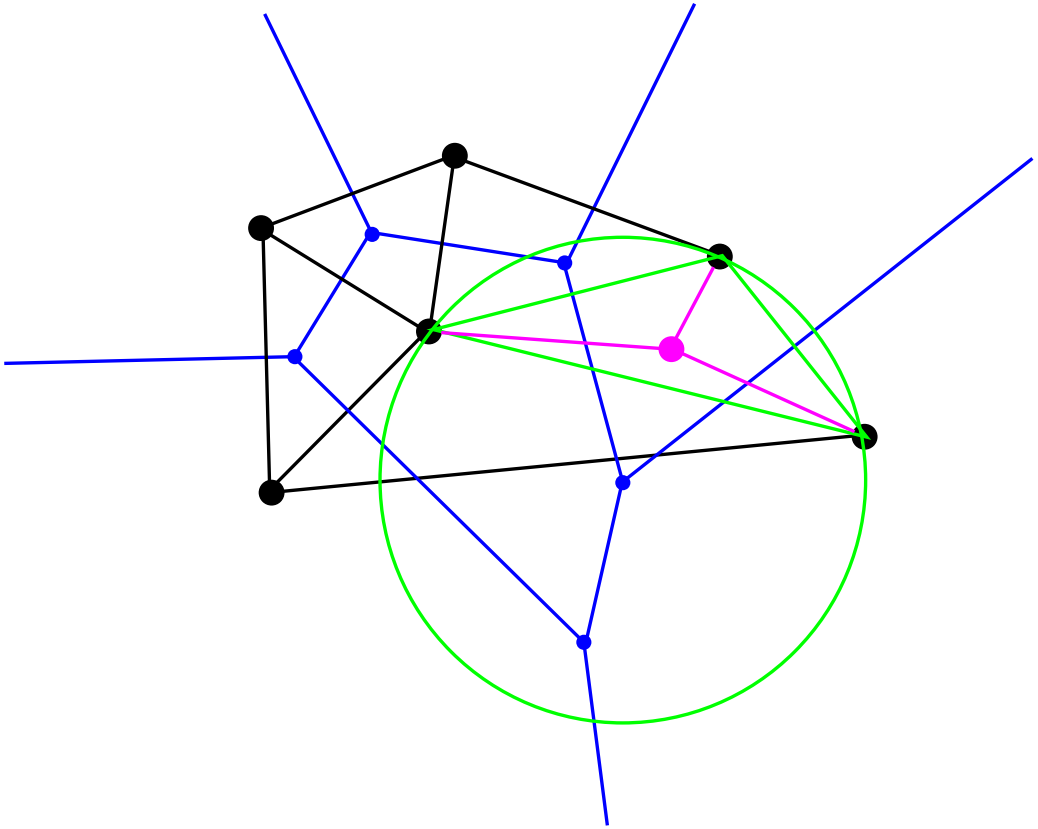
$V(T)$ ist für T verantwortlich, $SV(T)$ grenzt an (Stern)!

Weiteres Beispiel: Vater/Stiefvater nach Flip



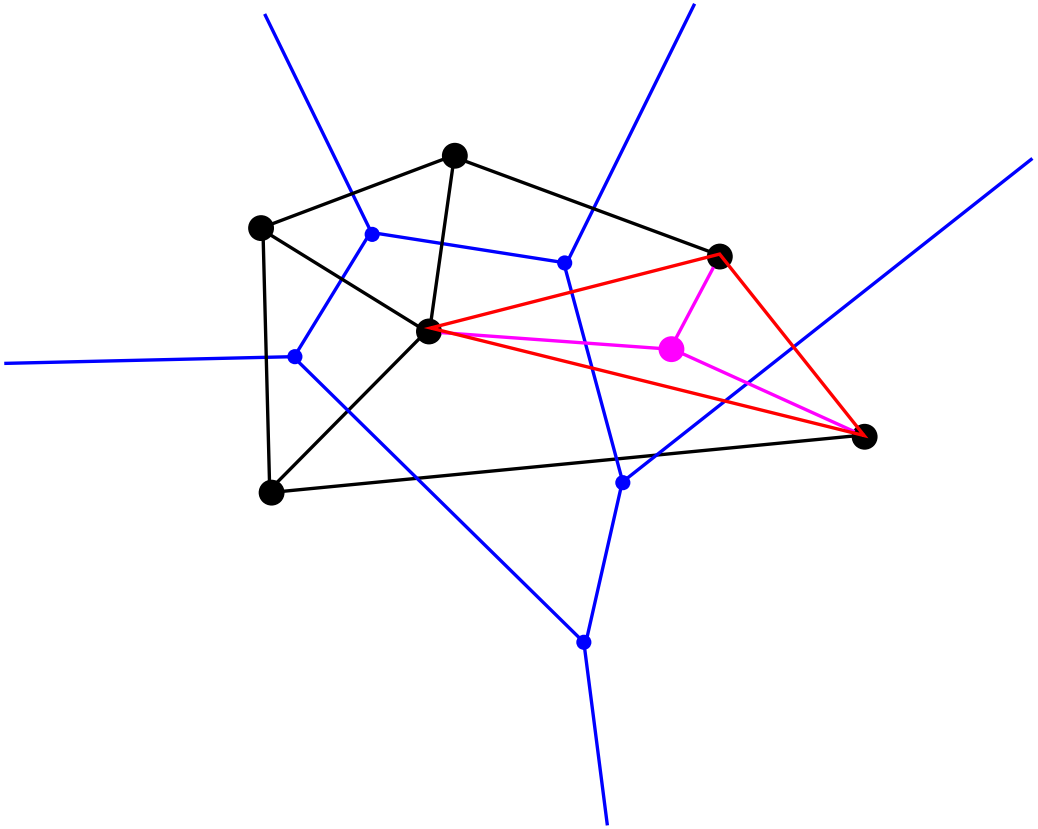
$V(T)$ ist für T verantwortlich, $SV(T)$ grenzt an (Stern)!

Weiteres Beispiel: Vater/Stiefvater nach Flip



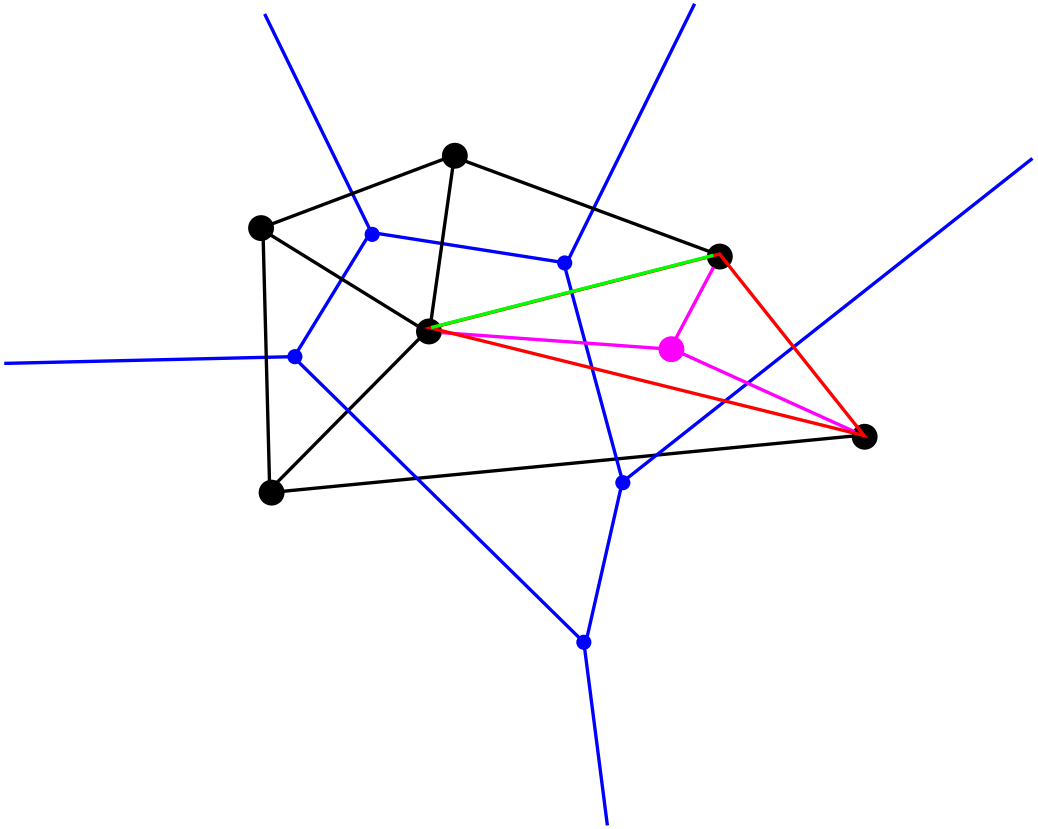
$V(T)$ ist für T verantwortlich, $SV(T)$ grenzt an (Stern)!

Weiteres Beispiel: Vater/Stiefvater nach Flip



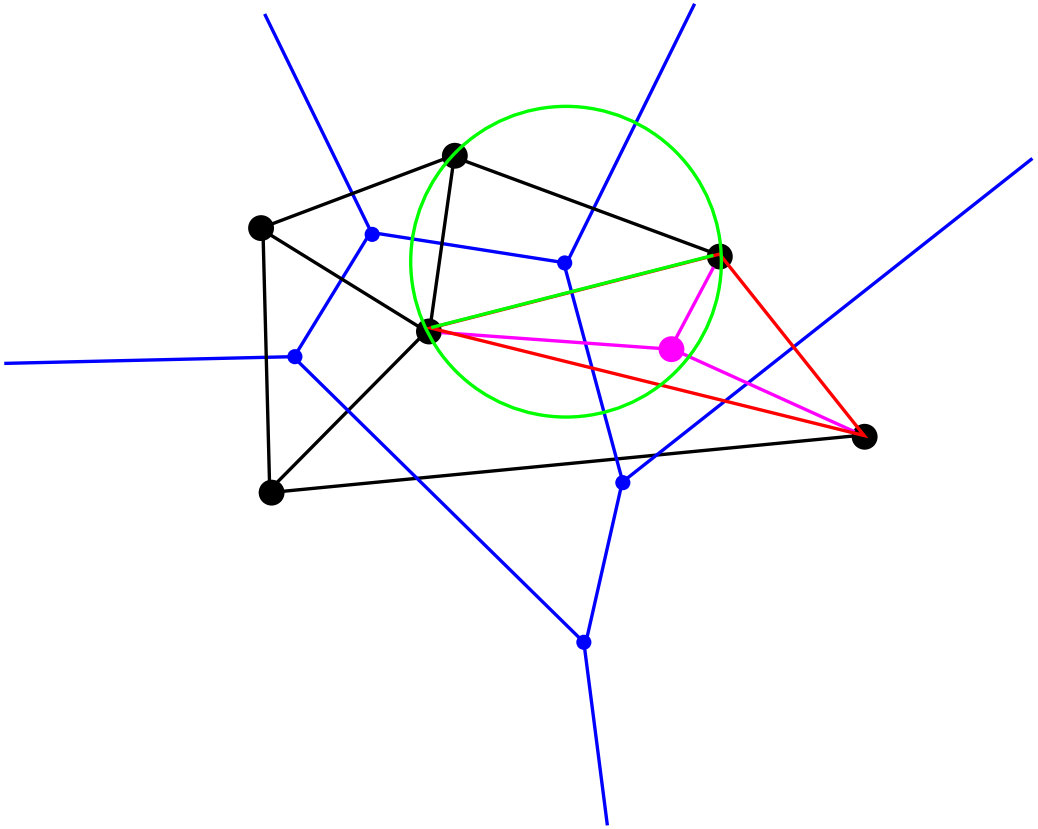
$V(T)$ ist für T verantwortlich, $SV(T)$ grenzt an (Stern)!

Weiteres Beispiel: Vater/Stiefvater nach Flip



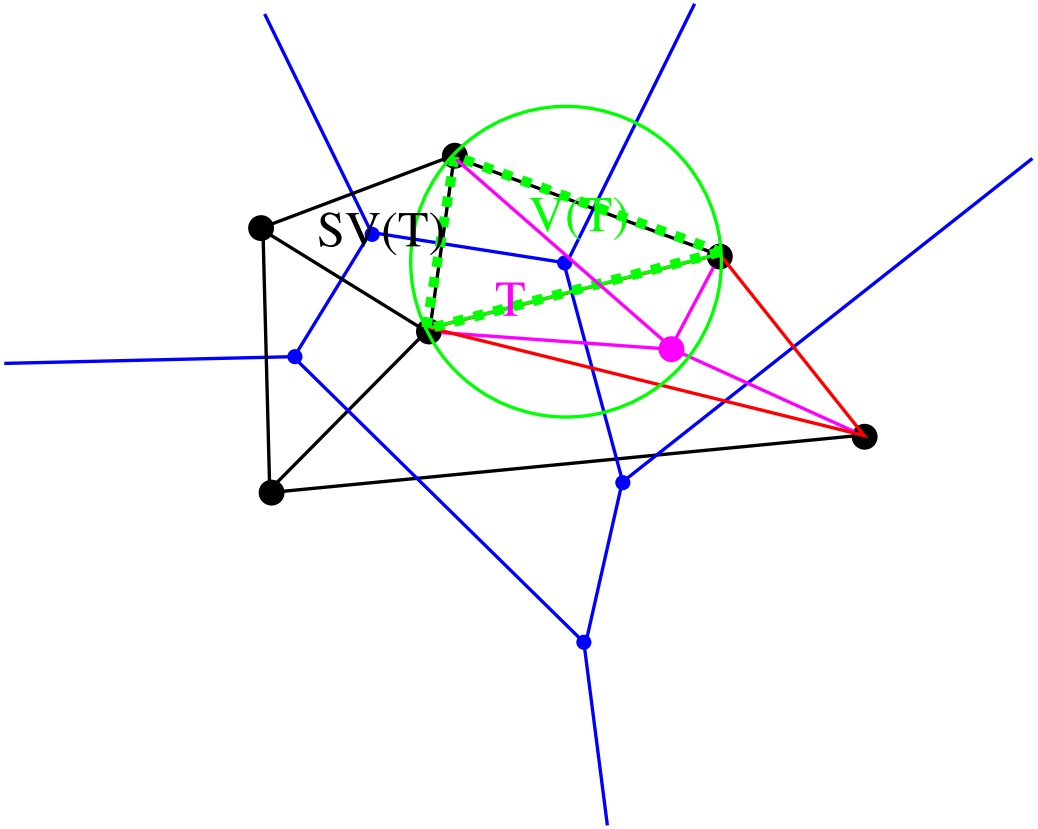
$V(T)$ ist für T verantwortlich, $SV(T)$ grenzt an (Stern)!

Weiteres Beispiel: Vater/Stiefvater nach Flip



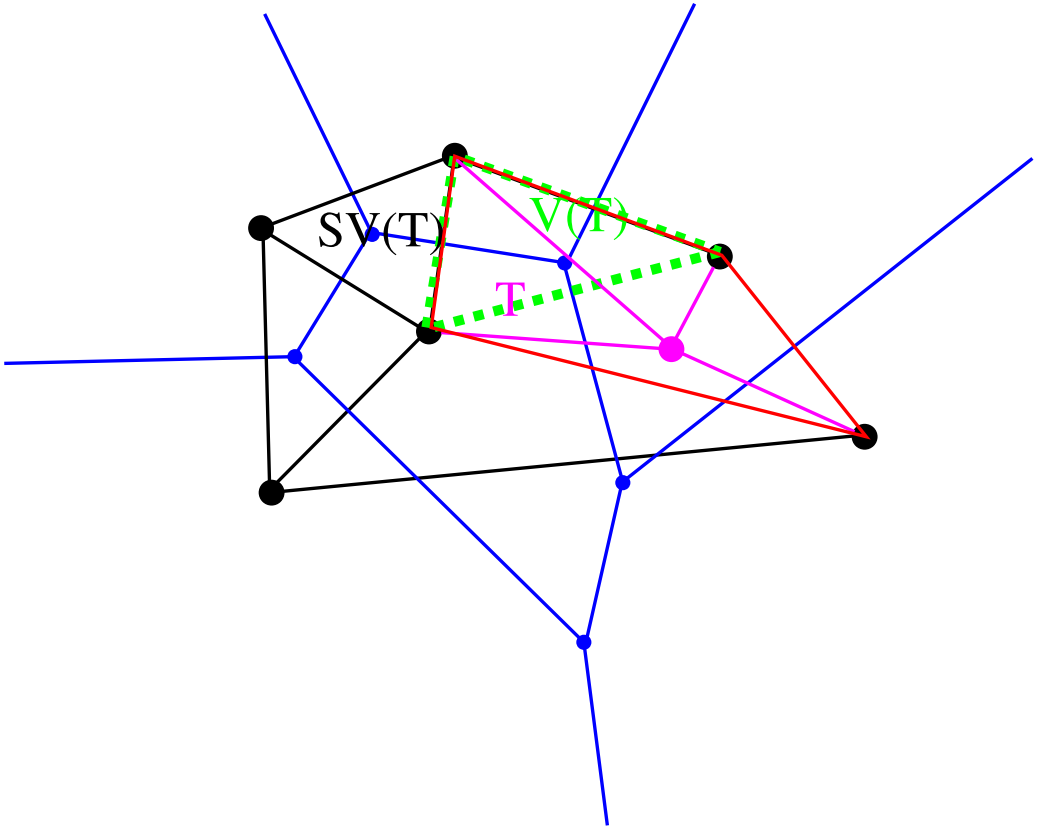
$V(T)$ ist für T verantwortlich, $SV(T)$ grenzt an (Stern)!

Weiteres Beispiel: Vater/Stiefvater nach Flip



$V(T)$ ist für T verantwortlich, $SV(T)$ grenzt an (Stern)!

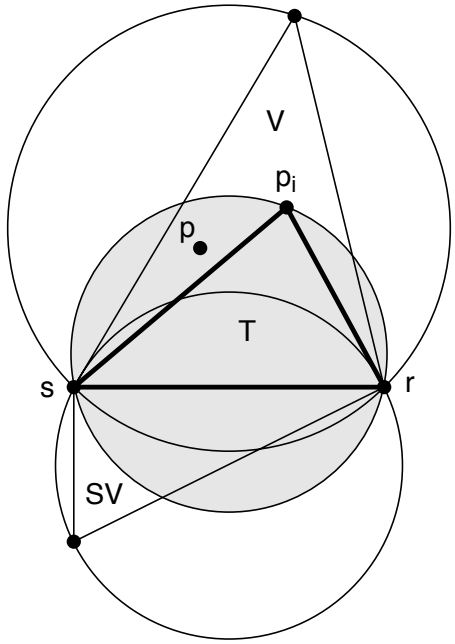
Weiteres Beispiel: Vater/Stiefvater nach Flip



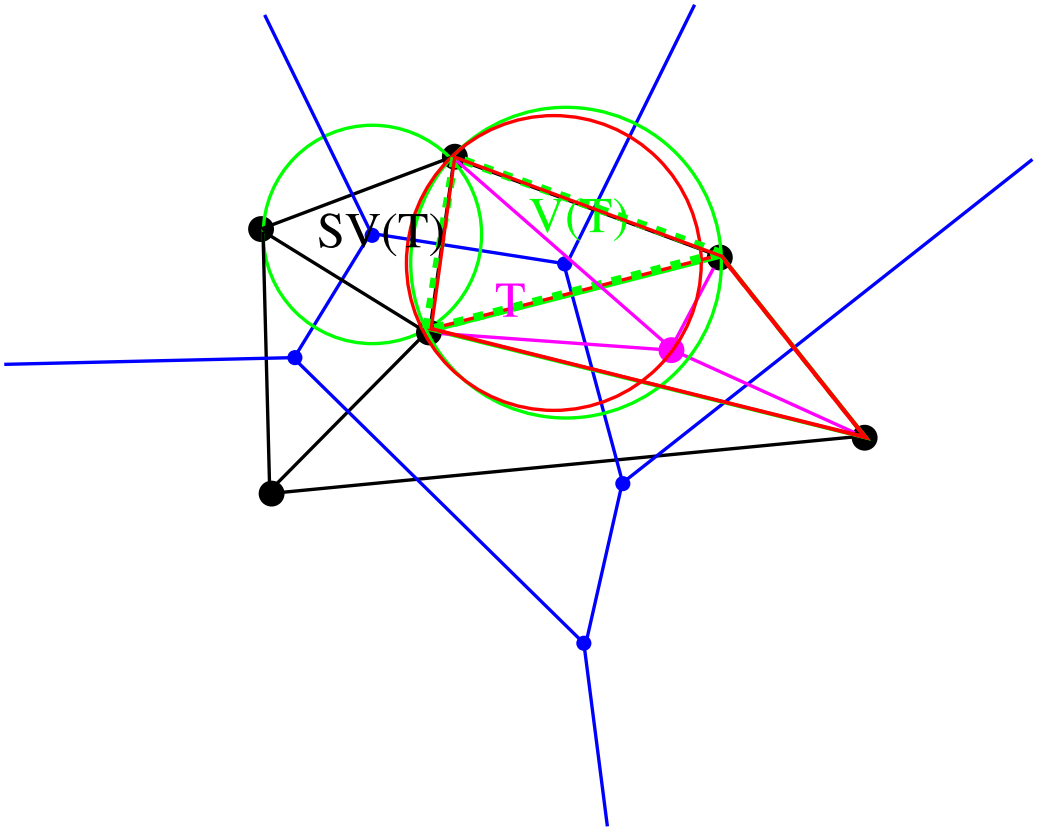
$V(T)$ ist für T verantwortlich, $SV(T)$ grenzt an (Stern)!

Konflikte werden vererbt

Lemma 6.6 Jeder Punkt $p \notin S_i$, der mit dem Dreieck tria(s, r, p_i) in Konflikt steht, steht entweder mit dem Vater oder dem Stiefvater in Konflikt.



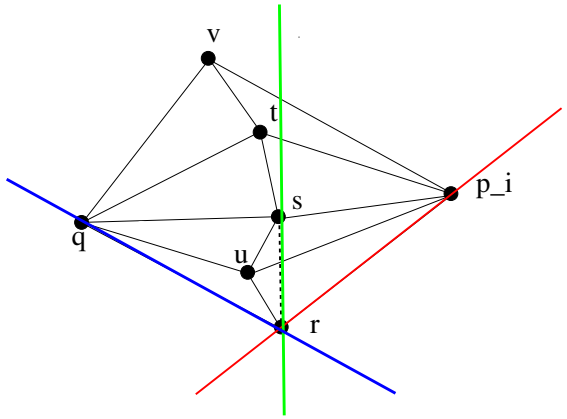
Weiteres Beispiel: Vater/Stiefvater



Umkreis von T liegt in Vereinigung der Umkreise von $V(T)$ und $SV(T)$

Delaunay DAG: Beziehungen-Beweis

Läßt sich auch auf *Unendliche* Dreiecke erweitern!



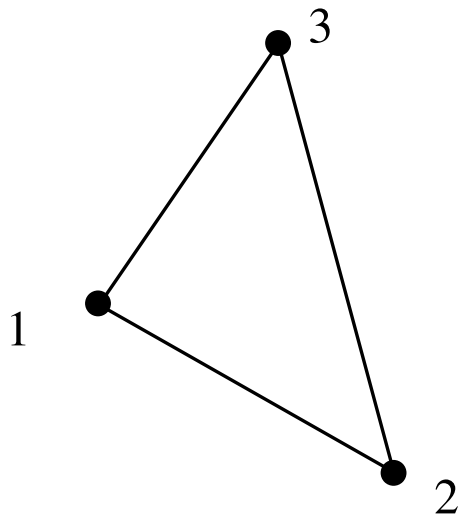
- $H(p_i, r)$ neu: $H(r, s)$ Vater, $H(q, r)$ Stiefvater (altes *Dreieck*)
- $H(p_i, r)$ ist in der Vereinigung von $H(q, r)$ und $H(r, s)$
- Beispiel: tria(p_i, s, t), Vater $H(s, t)$, Stiefvater tria(q, s, t)

Delaunay DAG$_i$: Schichten

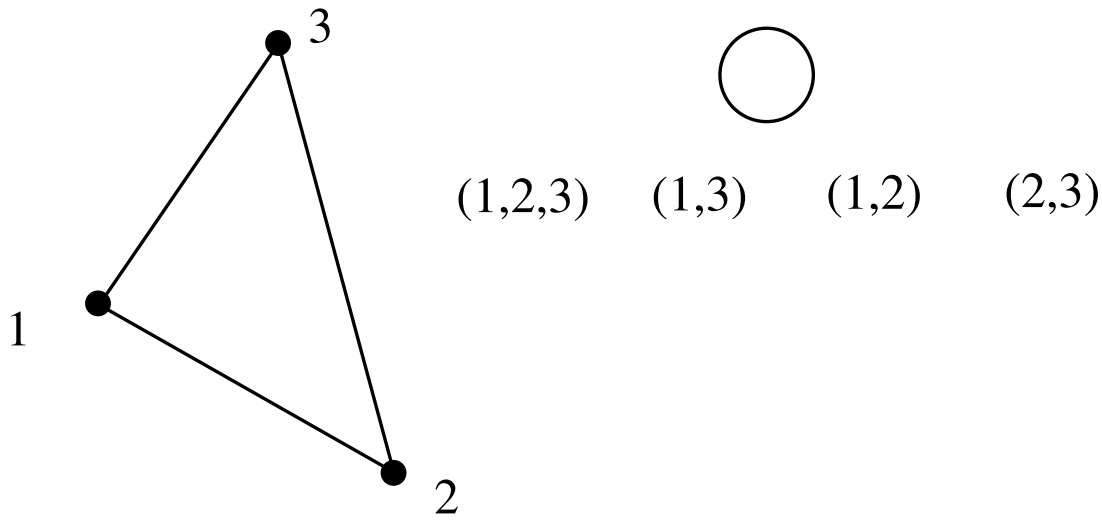
- DAG$_3$: Wurzel, erste Schicht, 4 Dreiecke
- DAG$_j$: In Schicht $(j-2)$ Dreiecke aus DT$_j$, die nicht in DT$_{j-1}$ sind
- Alle diese haben p_j als Endpunkt
- Ältere Dreiecke von DT$_{j-1}$ nicht auf unterster Schicht von DAG$_{j-1}$
- Kanten Vater/Stiefvater
- Beispiel! Sukzessiver Aufbau mit Lokalisation!

Delaunay DAG: Aufbau/Lokalisation

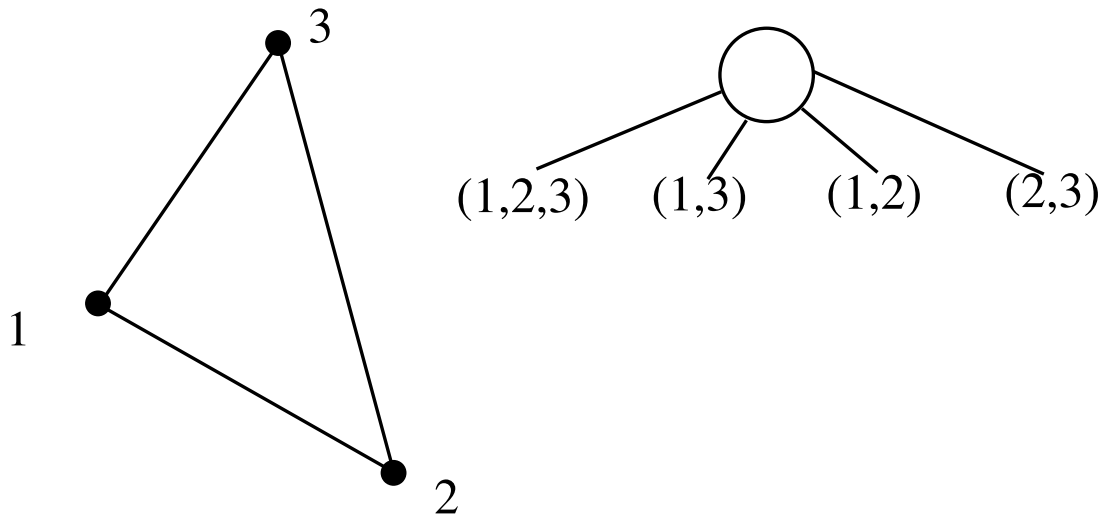
Delaunay DAG: Aufbau/Lokalisation



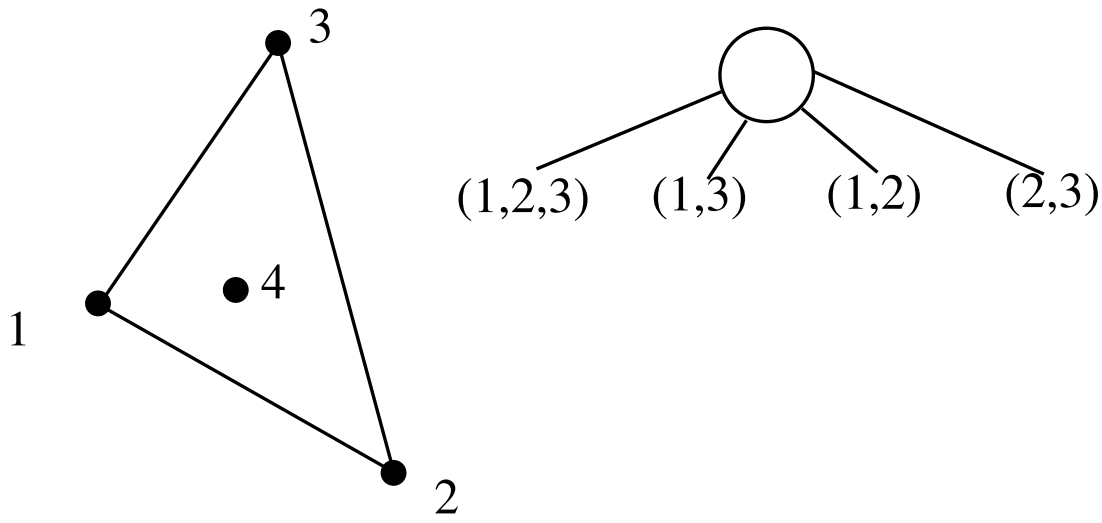
Delaunay DAG: Aufbau/Lokalisation



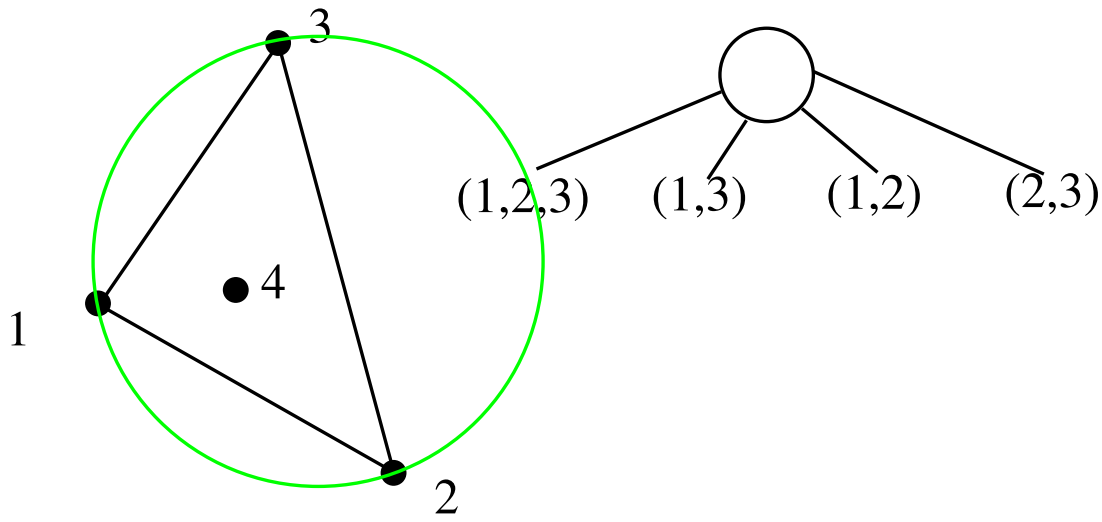
Delaunay DAG: Aufbau/Lokalisation



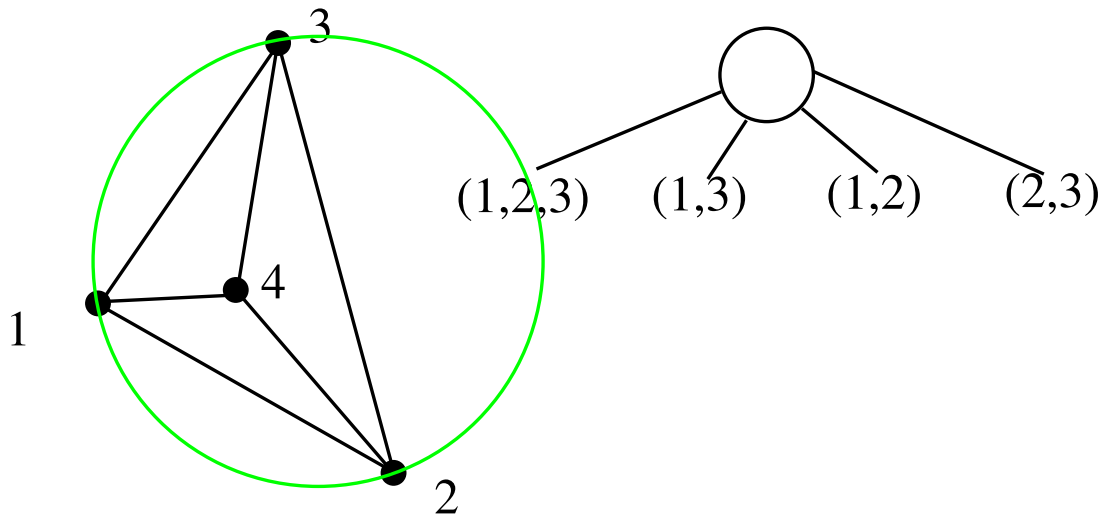
Delaunay DAG: Aufbau/Lokalisation



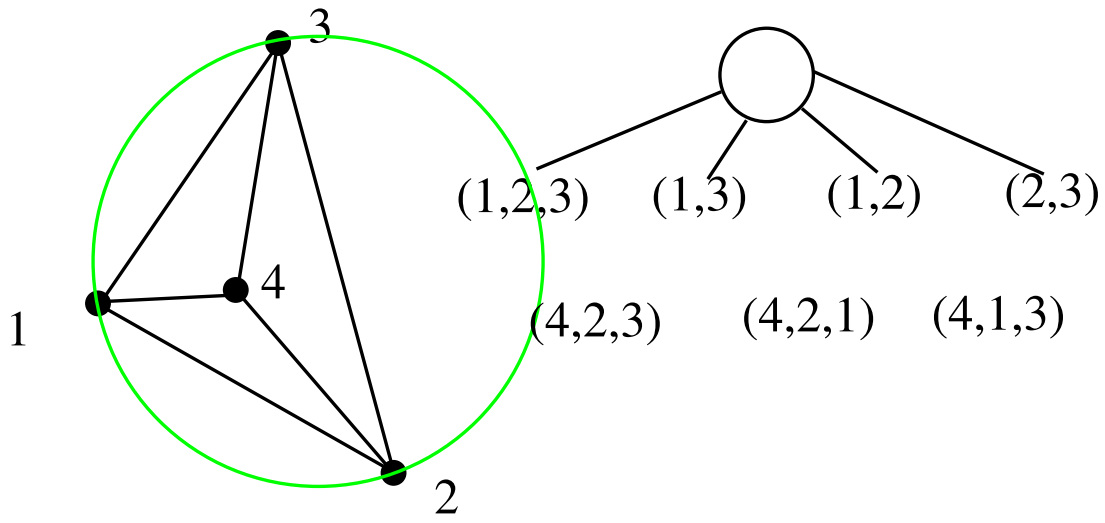
Delaunay DAG: Aufbau/Lokalisation



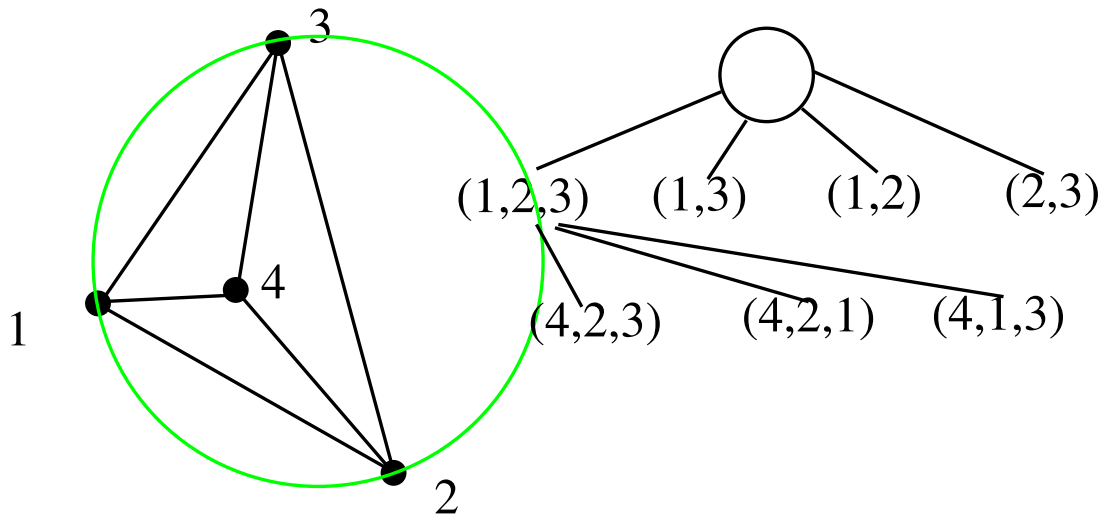
Delaunay DAG: Aufbau/Lokalisation



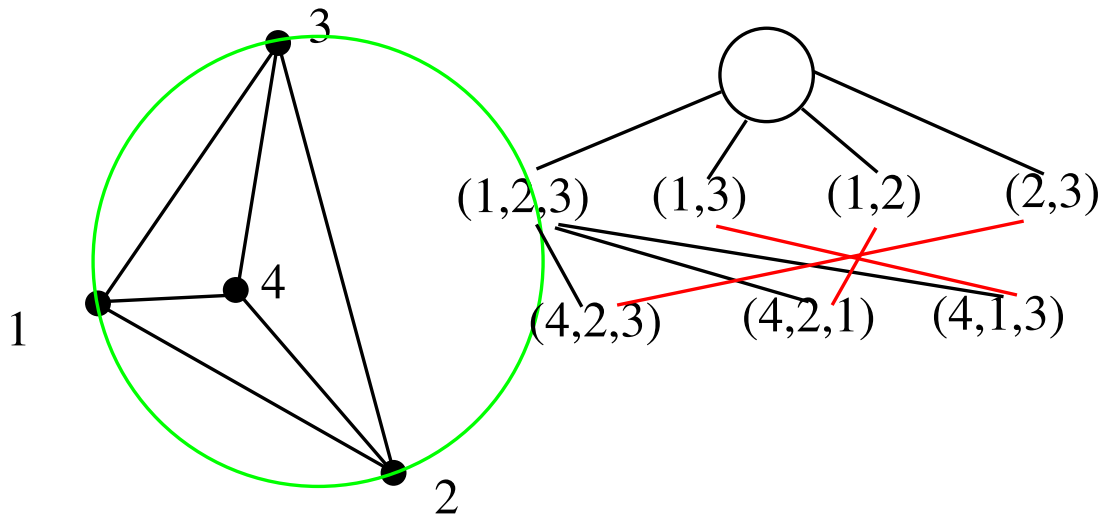
Delaunay DAG: Aufbau/Lokalisation



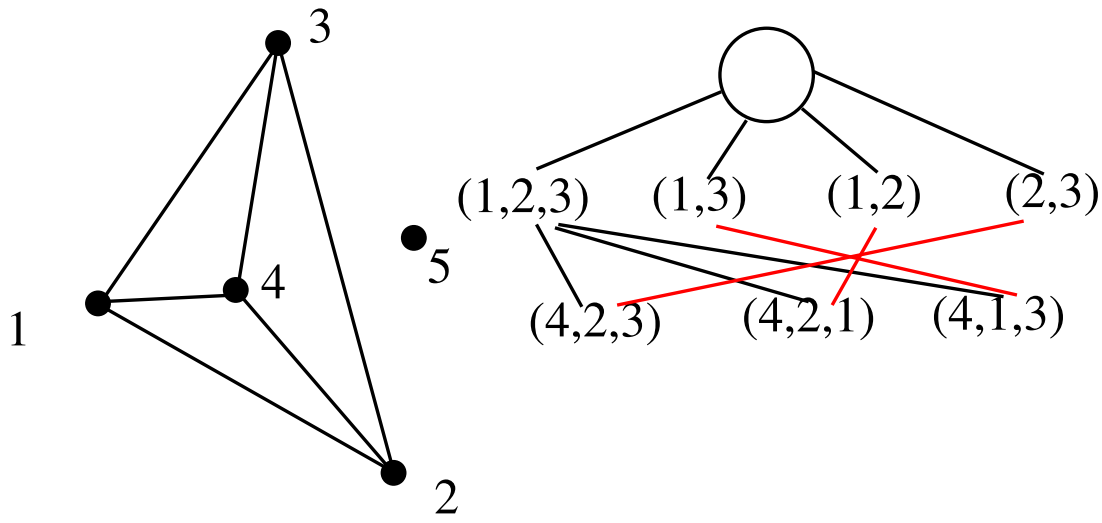
Delaunay DAG: Aufbau/Lokalisation



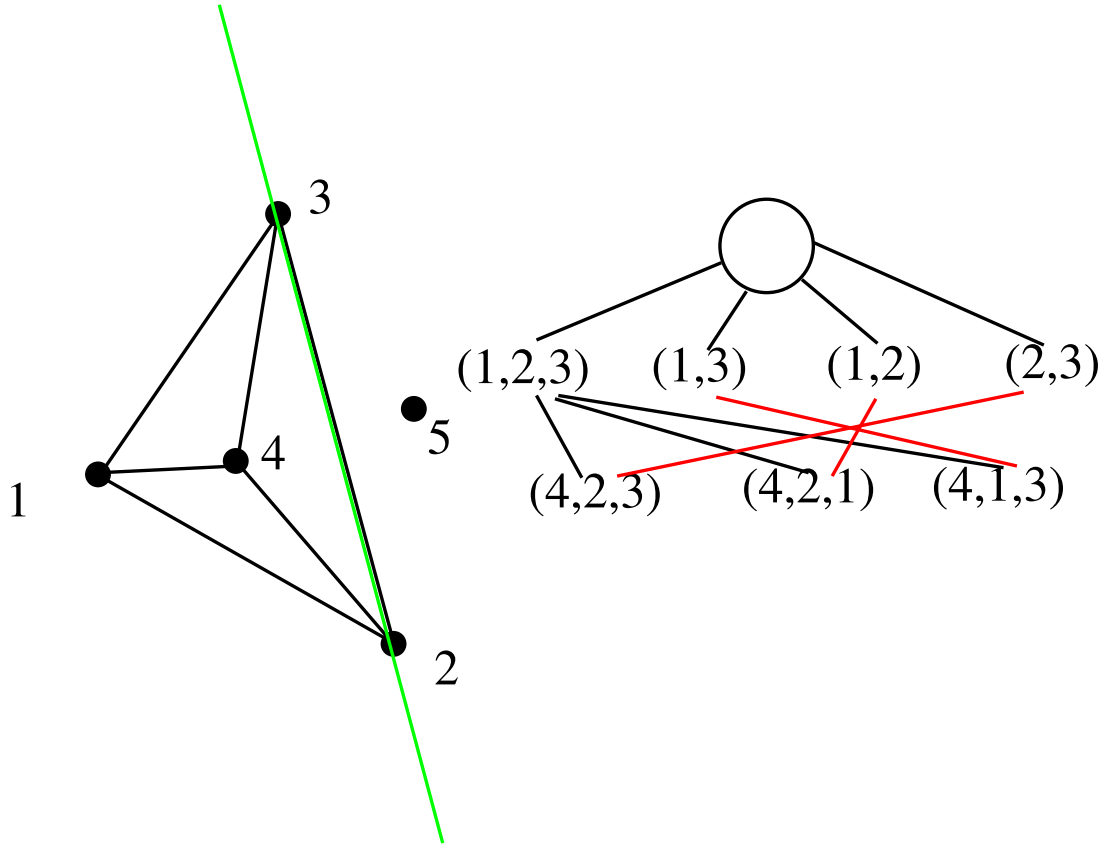
Delaunay DAG: Aufbau/Lokalisation



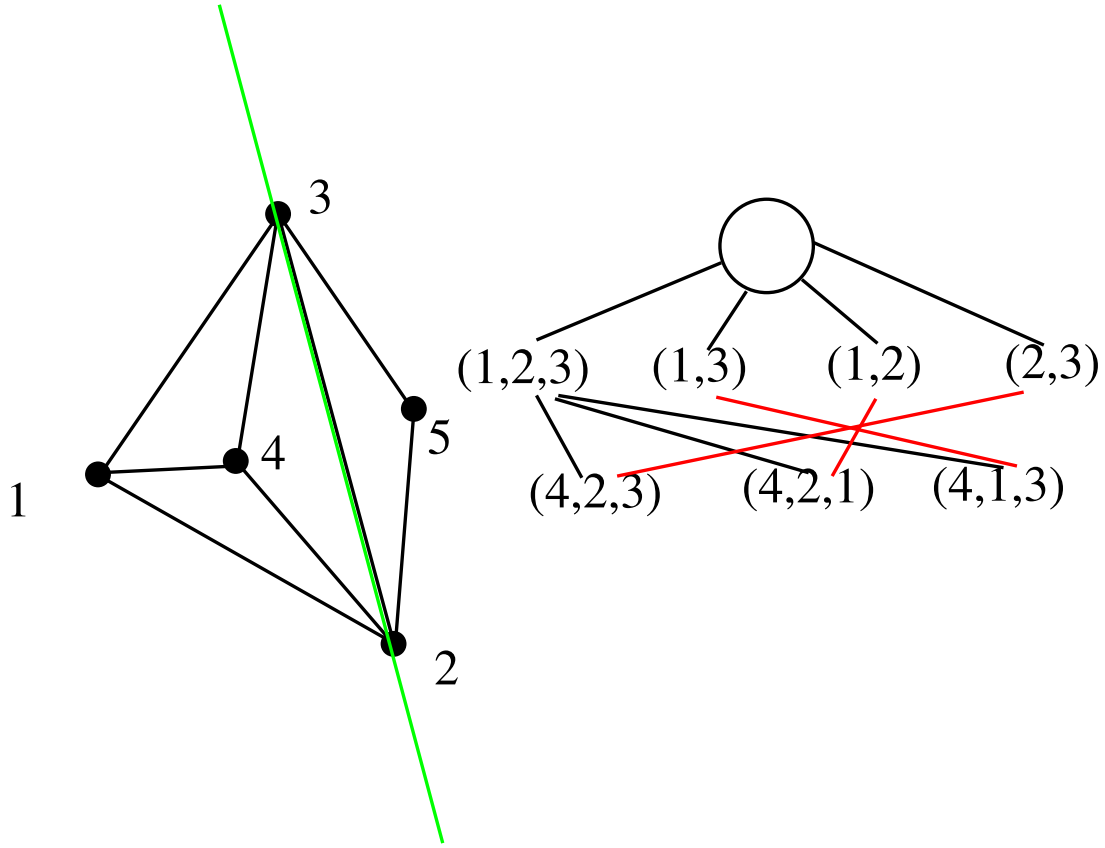
Delaunay DAG: Aufbau/Lokalisation



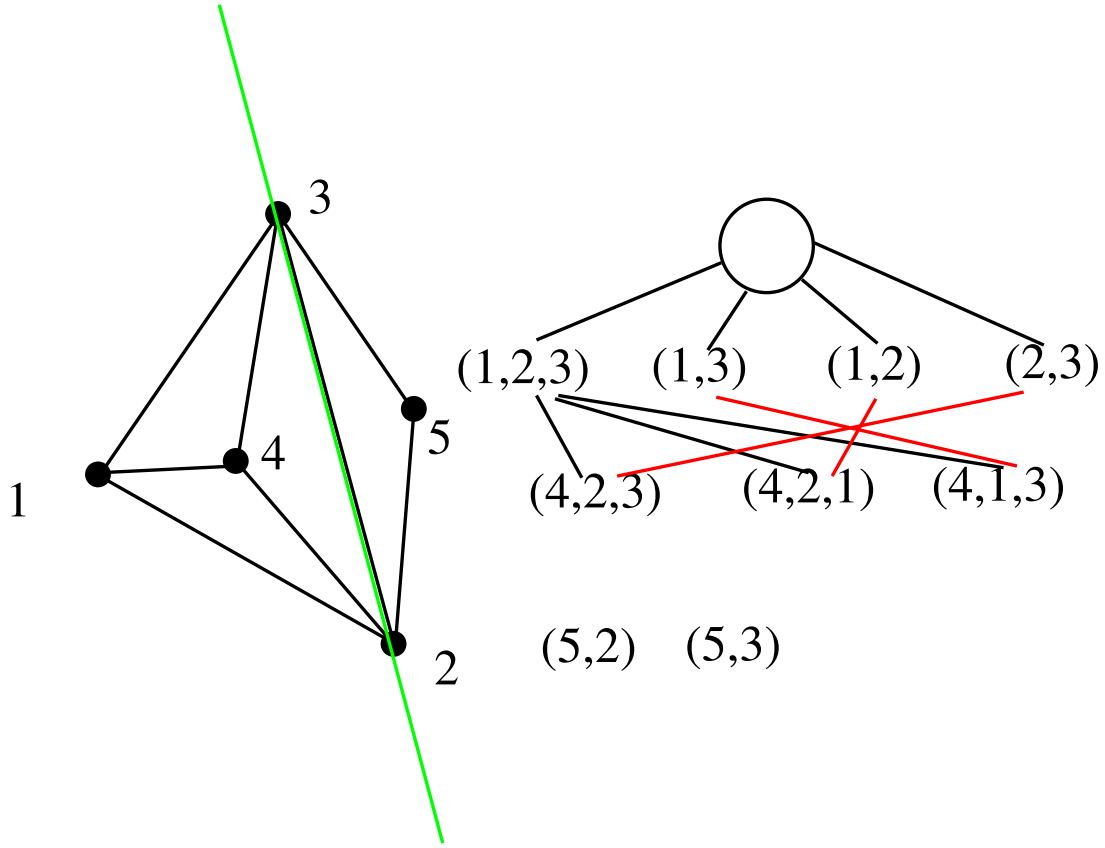
Delaunay DAG: Aufbau/Lokalisation



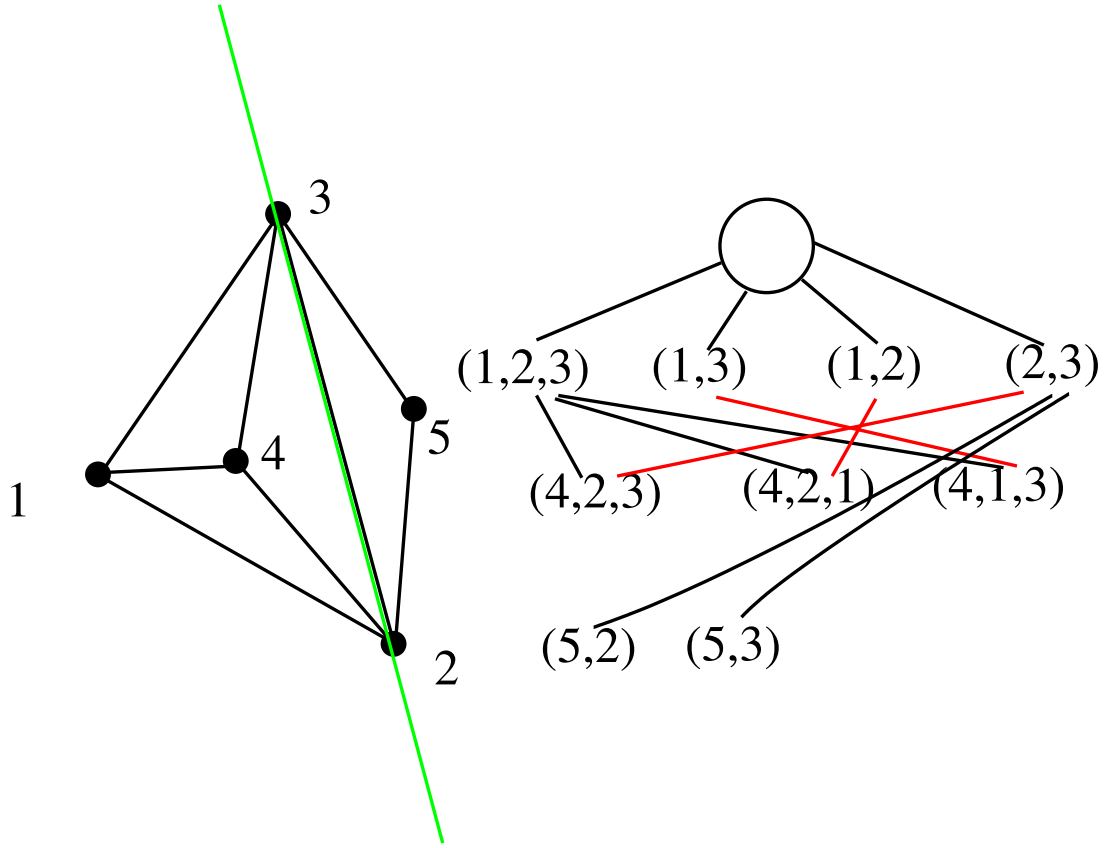
Delaunay DAG: Aufbau/Lokalisation



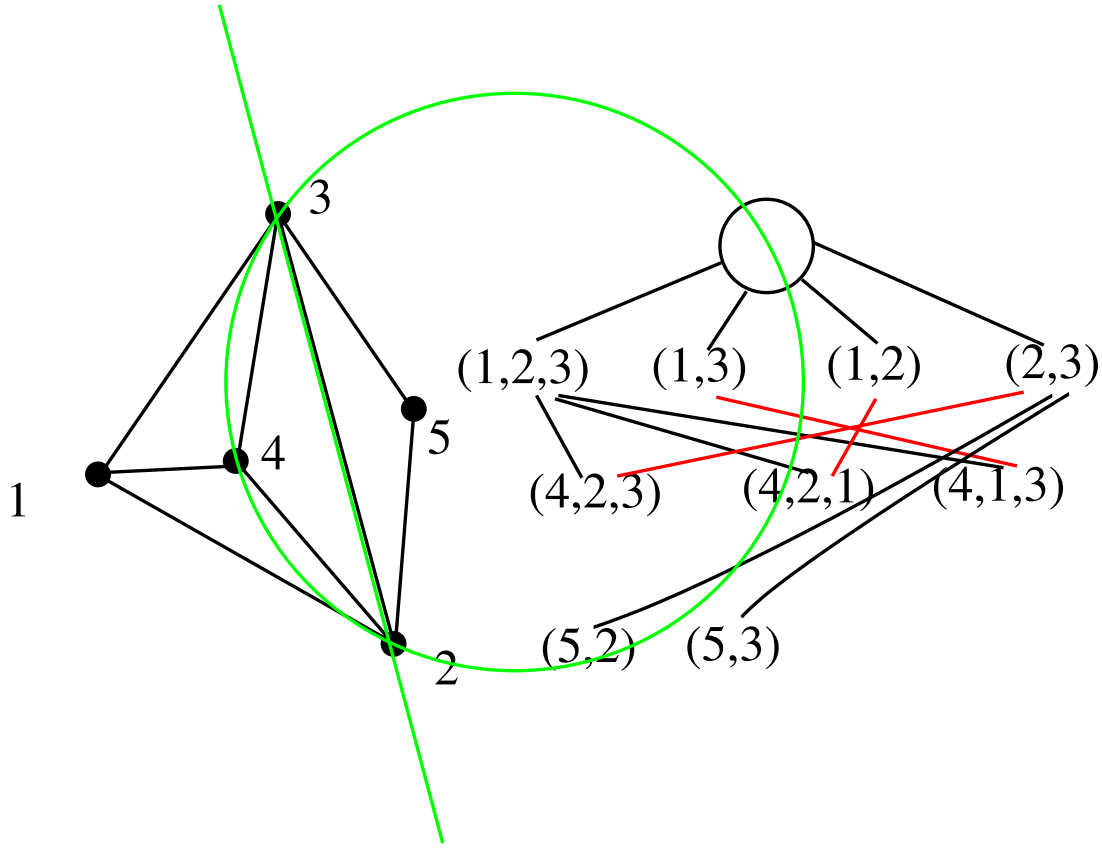
Delaunay DAG: Aufbau/Lokalisation



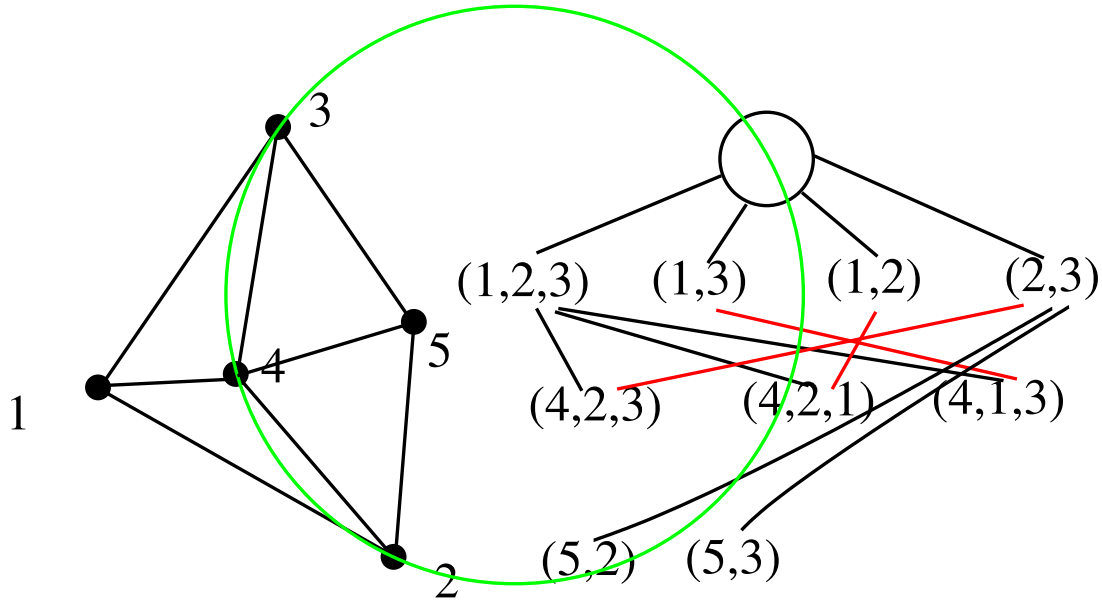
Delaunay DAG: Aufbau/Lokalisation



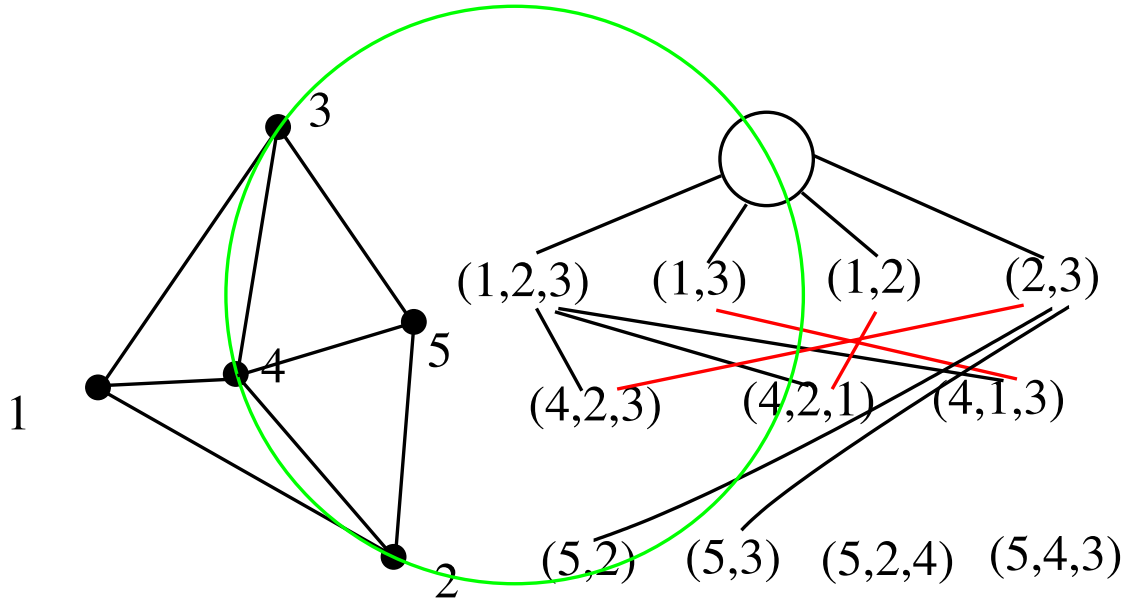
Delaunay DAG: Aufbau/Lokalisation



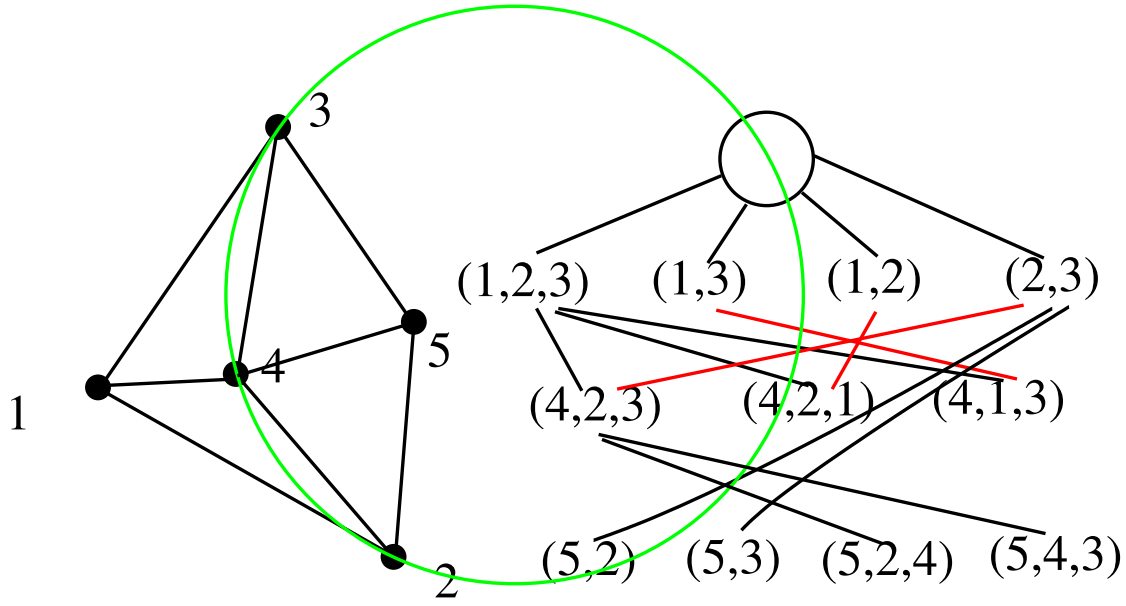
Delaunay DAG: Aufbau/Lokalisation



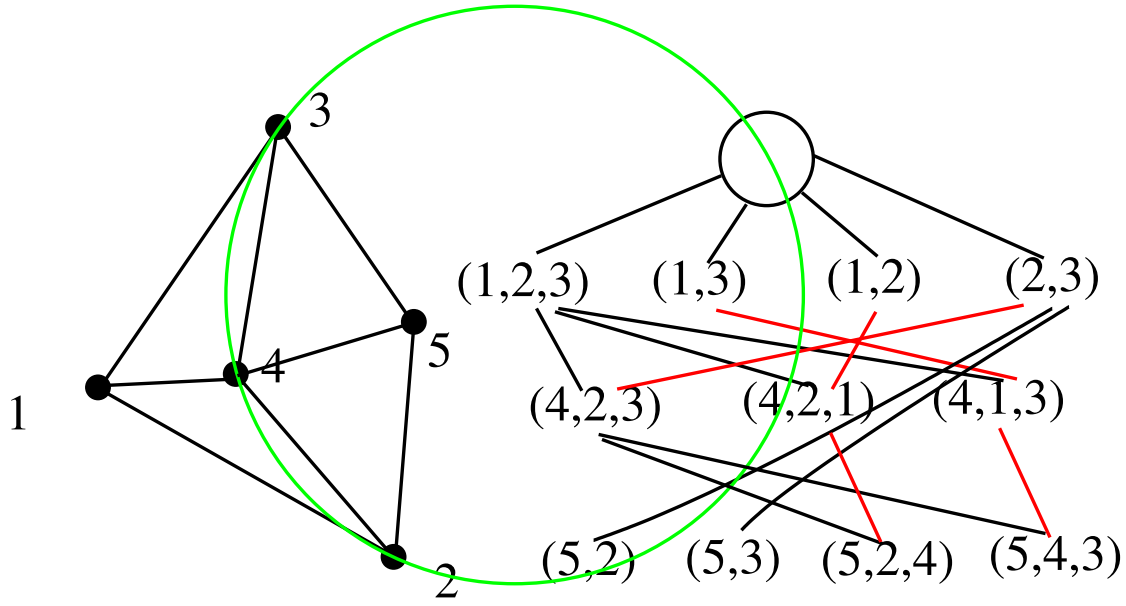
Delaunay DAG: Aufbau/Lokalisation



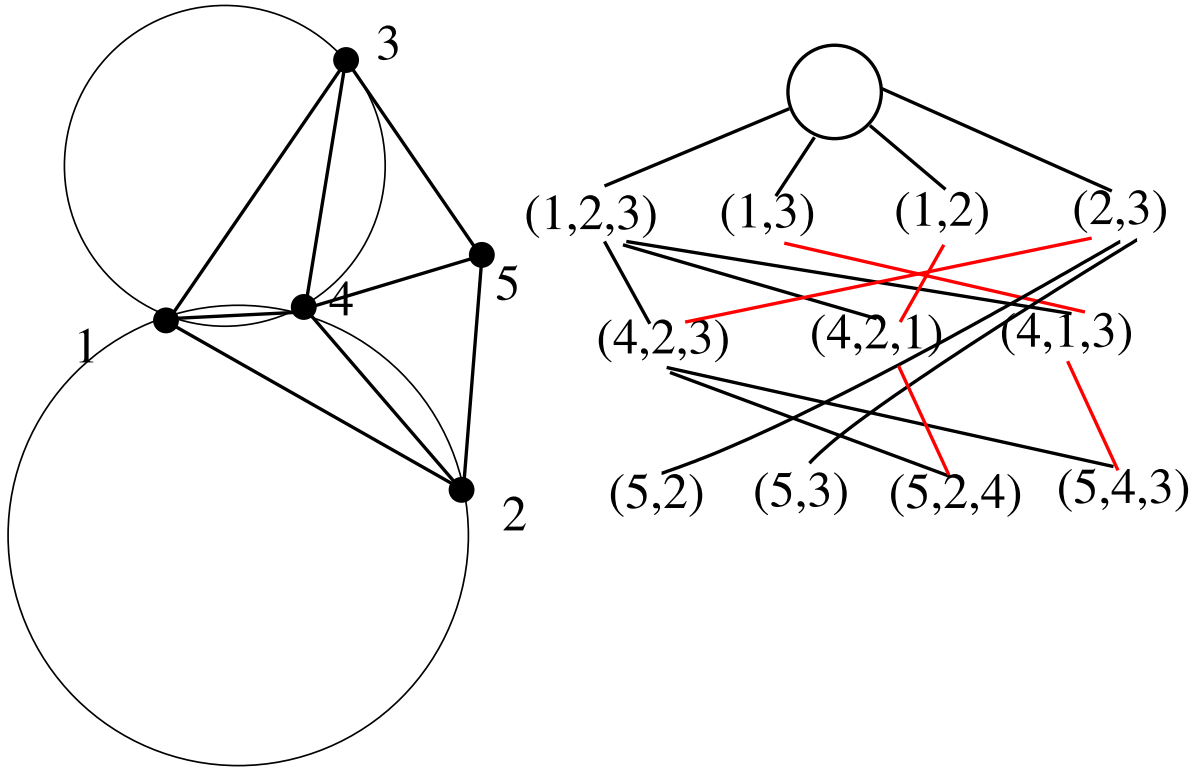
Delaunay DAG: Aufbau/Lokalisation



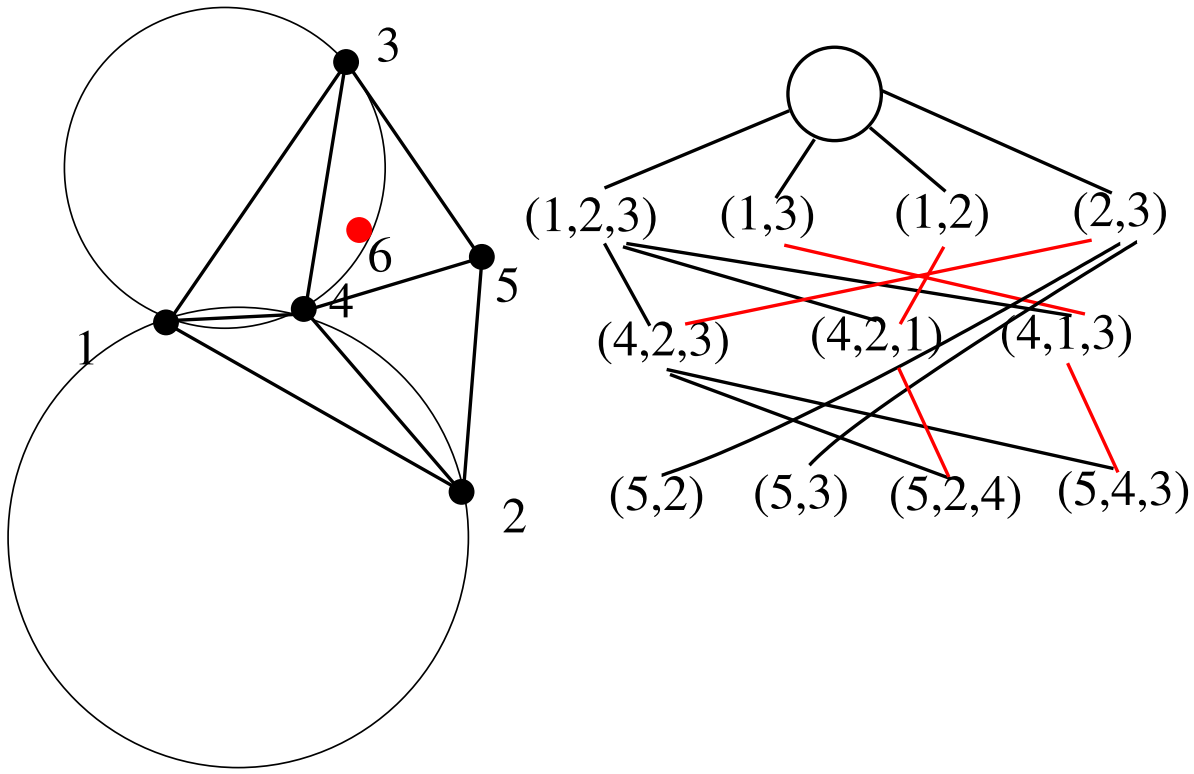
Delaunay DAG: Aufbau/Lokalisation



Delaunay DAG: Aufbau/Lokalisation



Delaunay DAG: Aufbau/Lokalisation



DAG: Lokalisieren von p_i (Einfügen)

- $\mathsf{DT}_3, \mathsf{DT}_3, \ldots, \mathsf{DT}_{i-1}$
- Neuer Punkt p_i, DAG_{i-1}

Korollar 6.7 Sei T ein Delaunay Dreieck in $\mathsf{DT}_3, \mathsf{DT}_3, \ldots, \mathsf{DT}_{i-1}$, das mit p_i in Konflikt steht. Dann gibt es in DAG_{i-1} einen gerichteten Weg von der Wurzel nach T, der nur Dreiecke enthält, die mit p_i in Konflikt stehen.

Beweis: Folgerung aus Lemma 6.6, Konflikt mit Vater oder Stiefvater, sukzessive zurückverfolgen

DAG: Lokalisieren von p_i (Einfügen)

- Tiefensuche in DAG_{i-1} von Wurzel aus
- Dreieck erreicht, ohne Konflikt mit p_i: Umkehren
- Alle Konflikt-Dreiecke werden erreicht, auch DT_{i-1}, das p_i enth.
- Def. m_i: Anzahl Dreiecke aus DT_{i-1}, die mit p_i in Konfl. stehen (auch alte, $m_i \geq d_i$!)
- Alle Konfliktdreiecke in $O(m_i)$
- Danach: Edge-Flips $O(k_i)$, Grad in DT, $k_i \leq m_i$
- DAG_{i-1} aktualisieren
- Neues Dreieck: Vater/Stiefvater stehen fest, neue Kanten
- $d_i \leq k_i + 2 \leq m_i + 2$: $O(m_i)$ neue Dreiecke und Kanten
- Gesamtlaufzeit: $O(m_i)$

Delaunay-DAG: Ergebnis

Delaunay-DAG: Ergebnis

Lemma 6.8 Die Verwendung eines Delaunay-DAG macht es möglich, einen neuen Punkt p_i in Zeit $O(m_i)$ in die Delaunay Triangulation DT_{i-1} einzufügen. Dabei bezeichnet m_i die Anzahl der Dreiecke, die in $DT_3, \ldots, DT_{i-1}$ vorkommen (einfach gezählt) und deren Vater oder Stiefvater mit p_i in Konflikt steht.

- Abschätzung vom m_i bei randomisierter Eingabefolge!
- Eingabereihenfolge gleich wahrscheinlich unter allen!
- Rückwärts: p_j mit WS $\frac{1}{j}$ aus $\{p_1, \ldots, p_j\}$

Gesamtergebnis: Die Delaunay Triangulation von n Punkten kann in erwarteter Laufzeit $O(n \log n)$ mit erwartetem linearen Speicherplatz konstruiert werden.

Delaunay-DAG: Ergebnis

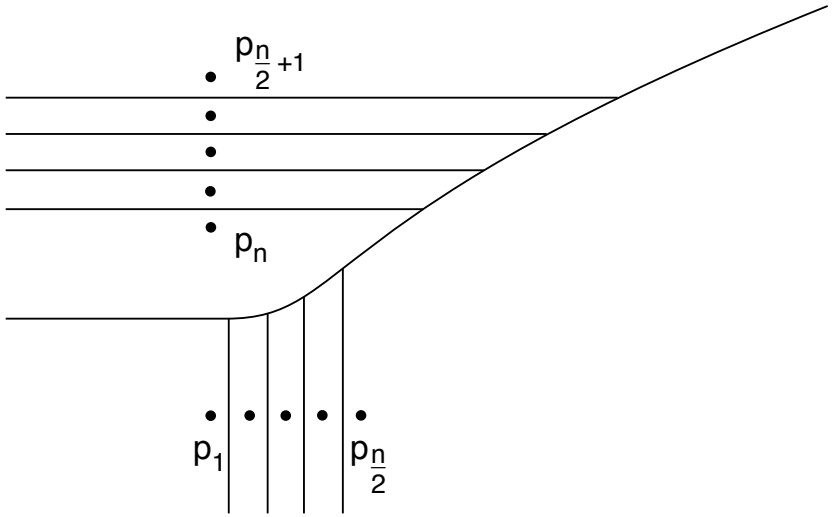
Theorem 6.9 Durch die Verwendung des Delaunay DAGs kann in mittlerer Zeit $O(\log i)$ einen Punkt p_i in DT_{i-1} einfügen, falls jede Reihenfolge von $p_1, p_2, \ldots, p_n$ gleich wahrscheinlich ist. Der Speicherplatzbedarf von DAG_i ist im Mittel linear.

- Speicherpl.: Anzahl Knoten aus DAG_i (nur zwei Kanten pro. Kn.)
- $b_j :=$ Anzahl Dreiecke in $\mathsf{DT}_j \setminus \mathsf{DT}_{j-1}$
- $b_j \leq \mathsf{grad}(p_j) + 1$ in DT_j, p_j zufällig gewählt
- Erwartungswert: $E(b_j) = E(\mathsf{grad}(p_j) + 1) = 6 + 1$, Region im Mittel 6 Kanten (Theorem 5.3)
- $\sum_{j=3}^{i} E(b_j) \in O(i)$
- $E(h_j) :=$ Erwartete Anzahl Dreiecke in $\mathsf{DT}_j \setminus \mathsf{DT}_{j-1}$ die mit p_i in Konflikt sind, $E(h_i) = \frac{6 \cdot 3}{j}$, $\sum_{j=3}^{i-1} E(h_j) \in O(\log i)$

Worst-Case Reihenfolge: $\Omega(n^2)$ Aufwand

Worst-Case Reihenfolge: $\Omega(n^2)$ Aufwand

- X-Achse $p_1, \ldots, p_{\frac{n}{2}}$, Y-Achse $p_{\frac{n}{2}+1}, \ldots p_n$
- Von oben nach unten: Mit jedem unteren Punkt eine neue Delaunay Kante
- Gilt für Delaunay und VD: Applet



Kapitel Buch

Kapitel 6 Seite 274 unten – S. 285 unten