Algorithmen im Chip-Entwurf 10

Globale Verdrahtung

Andreas Koch
FG Eingebettete Systeme
und ihre Anwendungen
TU Darmstadt

Überblick

- Globalverdrahtung
  - Problem
  - Modellierung
  - Vorgehensweisen
- Algorithmus
  - Für Standardzellen
  - Steiner-Bäume
    - Konstruktionsheuristik
    - Optimierung
- Ausblick
- Zusammenfassung

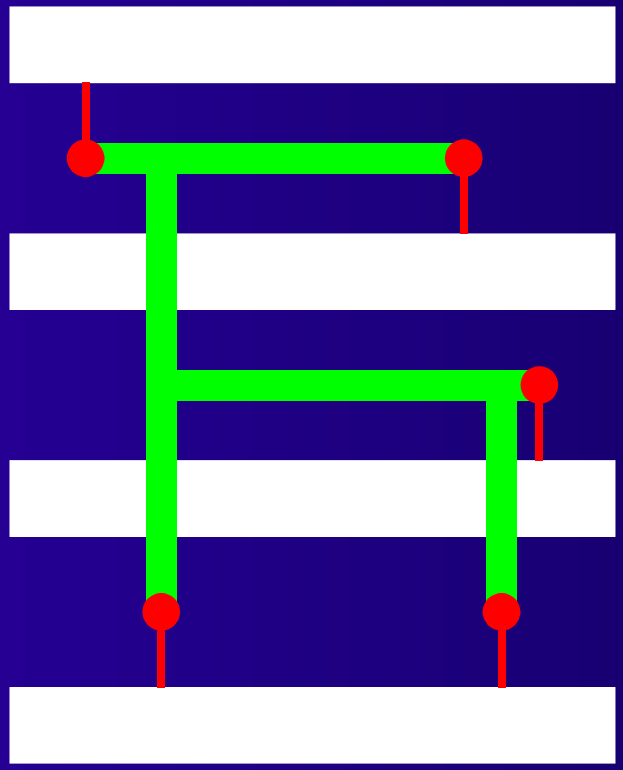
Globale Verdrahtung 1

- **Im Entwurfsfluß**
  - Nach Plazierung
  - Vor lokaler Verdrahtung
- **Verteilt Signale auf Kanäle**
  - Führung innerhalb der Kanäle bleibt offen
- **Optimiert auf**
  - Minimale Fläche
  - Einhalten der Zeitvorgaben
- **Hängt von Zieltechnologie ab**

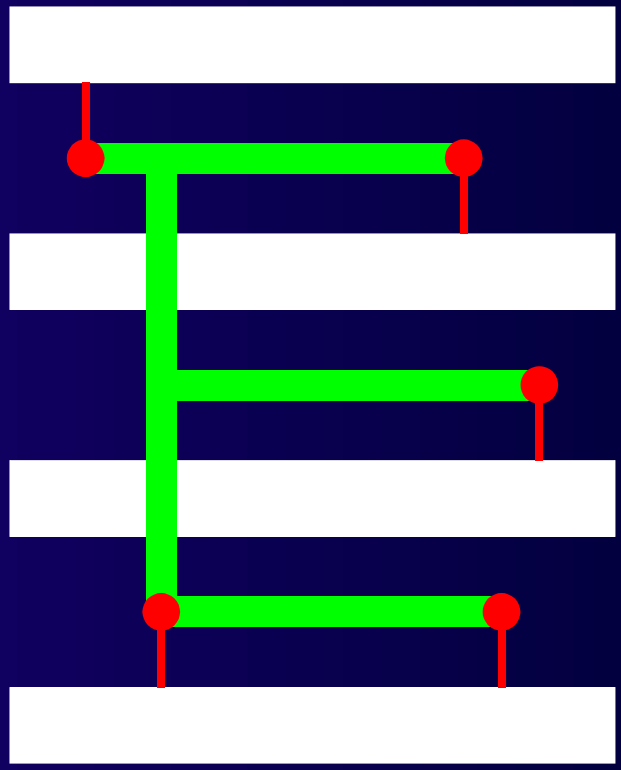
Globale Verdrahtung 2

- **Hier: Im Standardzellen-Entwurf**
- **Alle Terminals eines Netzes an einem Kanal?**
  - Falls ja: Nur lokale Verdrahtung erforderlich
- **Sonst: Globale Verdrahtung**
  - Trennt Netz auf einzelne Kanäle auf
  - Übergang zwischen Kanälen
    - Reservierte Verdrahtungsebenen
    - Feedthroughs einfügen (beeinflußt Plazierung)
    - Vorgegebene Feedthrough-Leitungen allozieren
  - Idee: Rechtwinkliger Minimaler Steiner Baum (RSMT)
    - Ggf. höhere Kosten für vertikale Segmente (feedthroughs)
    - Wenn begrenzte Ressource

Globale Verdrahtung 3



Rechtwinkliger Steiner-Baum
mit minimaler Länge



Rechtwinkliger Steiner-Baum
mit minimalen Übergängen

Globale Verdrahtung 4

- **RSMT nicht immer beste Lösung**
  - Neben Länge zu berücksichtigen:
    - Begrenzte Anzahl von Feedthroughs
    - Zeitvorgaben (timing-driven)
      - Kritische Netze kurz halten
  - Nur durch Gewichtung der Kosten möglich
    - Ungenau
- **Bessere Verzögerungsmodelle**
  - Nur Verdrahtungslänge ungenau
    - Widerstand und Kapazität zusammengeworfen
  - Besser
    - R, C getrennt für einzelne Segmente
    - Bewährt: Elmore-Modell
      - Auch in VPR verwendet
- **Andere Verfahren**
  - Multicommodity Flow, Pattern-based, Hierarchical

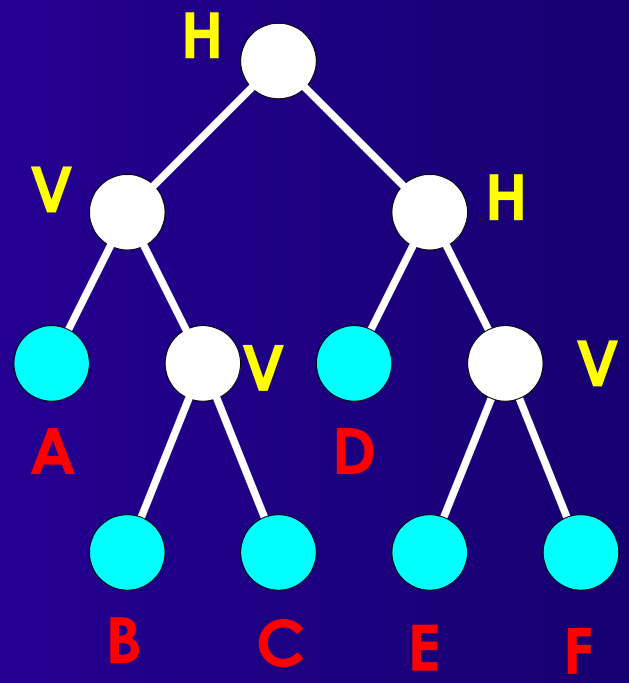
Globale Verdrahtung 5

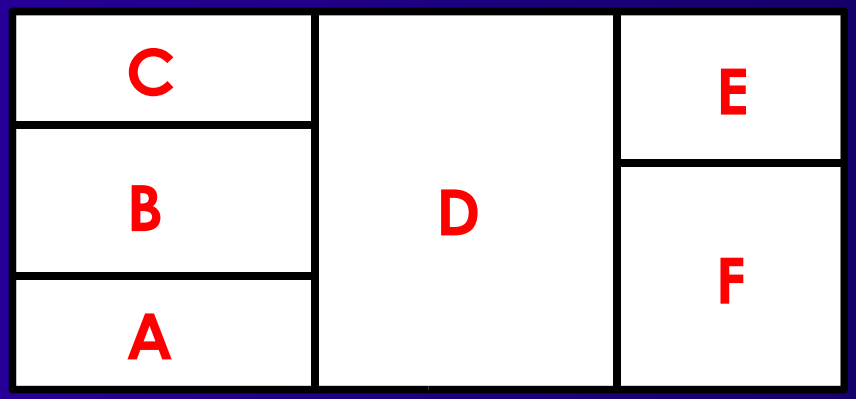
- **Bei unidirektionaler Sicht**
  - 1 Quelle / n Senken
- **Mögliche Teiloptimierungsziele**
  - Kurzer Weg zu kritischer Senke
  - Gleich lange Wege (kleiner skew)
    - Verdrahtung von Takt-Leitungen (H-Trees)
- **Gesamtziel**
  - Minimiere Verdrahtungsfläche
  - Schätze Kanalbreiten ab

Globale Verdrahtung 6

- Hier: Building-Block Layout
- Komplizierter
- Irreguläre Freiflächen zwischen Zellen
- Wie Flächen in Kanäle aufteilen?
  - Channel Definition Problem (CDP)
- Kanäle in welcher Reihenfolge verdrahten?
  - Channel Ordering Problem (COP)

Exkurs Slicing Floorplans



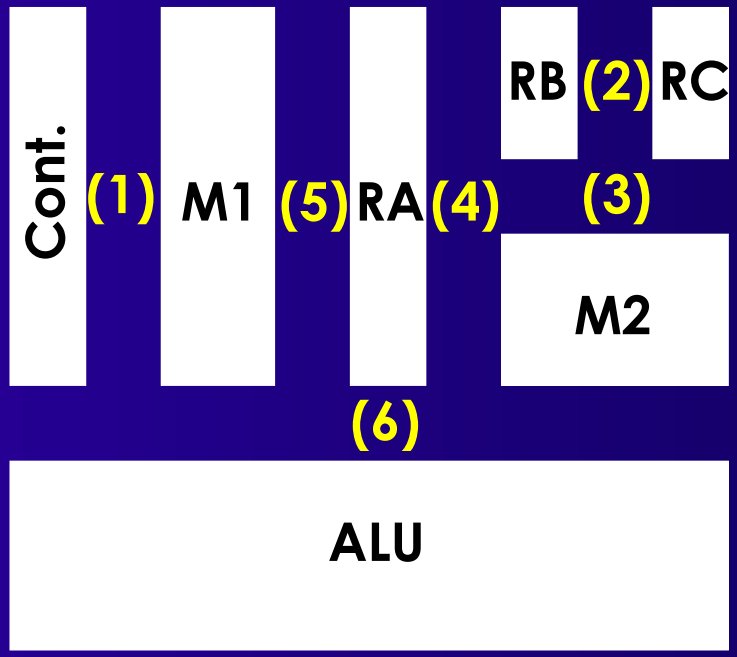


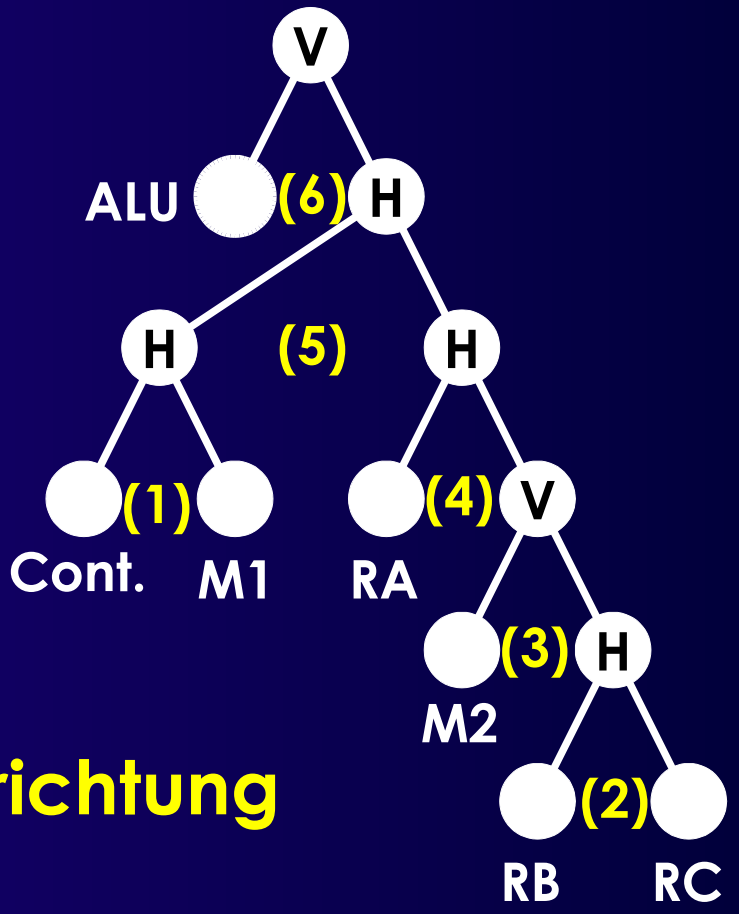
- **Darstellung durch Slicing Tree**
  - Knoten sind Schnitte oder Blattzellen
  - Schnitte nach Richtung getrennt
    - H: Linker Unterbaum *LINKS* von rechtem
    - V: Linker Unterbaum *UNTER* rechtem

Globale Verdrahtung 7

- **Für Slicing Floorplan: Einfach zu lösen**
- **CDP**
  - **Schnittlinien sind Kanäle**
  - **Kanalform abhängig von Reihenfolge**
- **COP**
  - **Grundlage ist Slicing Tree**
  - **DFS mit Post-Order Traversal**
    - **Numeriere bearbeitete Knoten aufsteigend**
    - **V-Schnitt: V-Kanal, L=Ober/Unterkante der Zellen**
    - **H-Schnitt: H-Kanal, L=linke/rechte Seite der Zellen**

Globale Verdrahtung 8





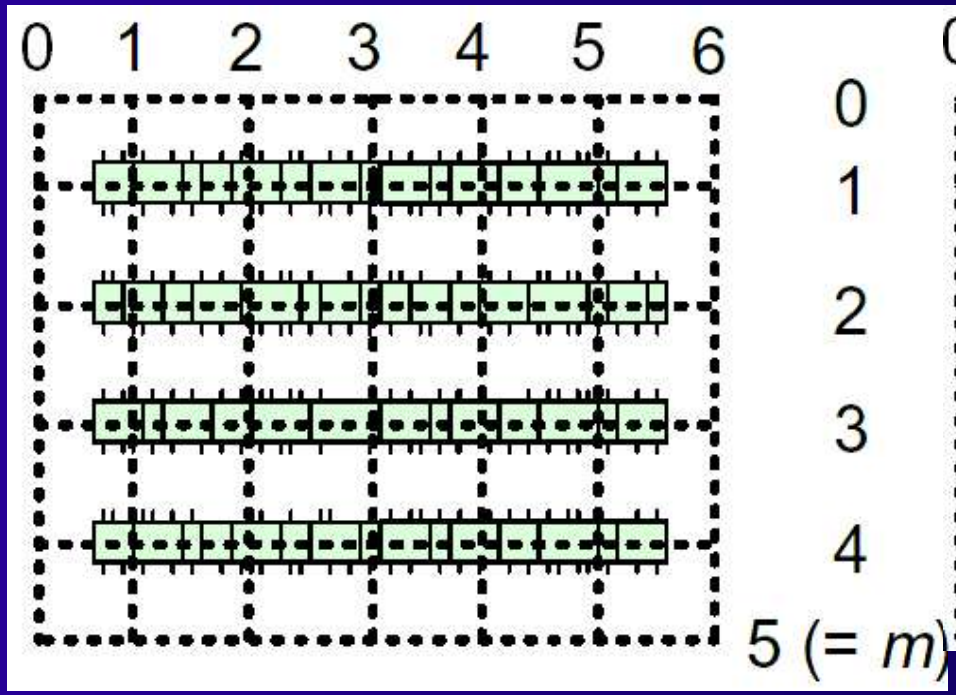
- CDP via Ordnung, Schnittrichtung
- COP via Slicing Tree
  - Post-Order DFS
  - Reihenfolge für Kanalverdrahtung

Globale Verdrahtung 9

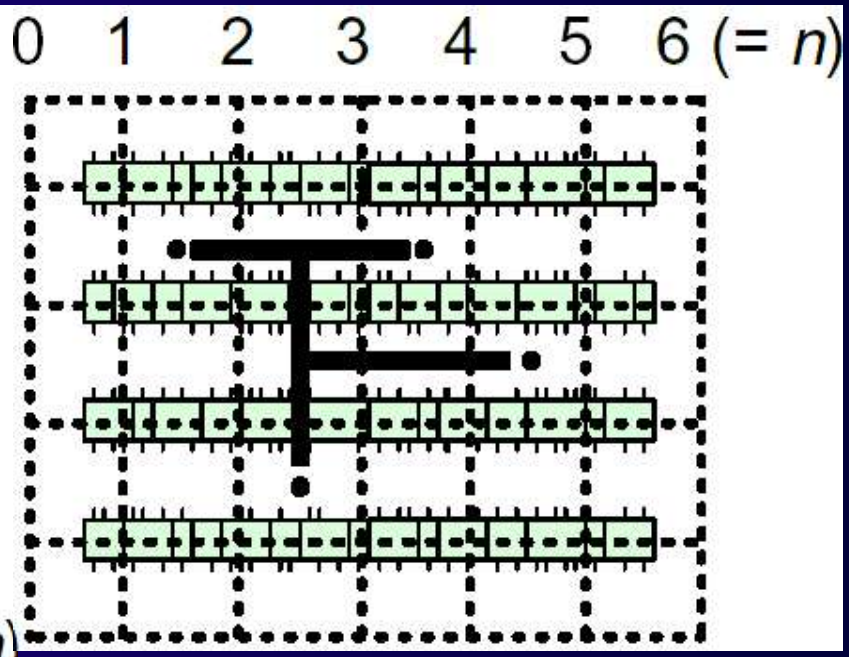
- **Bei Non-Slicing Floorplans**
  - Reine Kanalverdrahtung nicht ausreichend
  - Braucht
    - Switchbox Router
    - Dreiseitige Kanal-Router
      - Nur eine Kanalseite hat bewegliche Terminals
      - Verdrahtungsfläche ist fest (ähnl. Switchbox)
- **Nach Lösung des CDP: Steiner-Baum**
  - In der Regel keine Feedthroughs
  - Verdrahtung nur in den Kanälen
    - Sehe Kanäle als Kanten in Graph an
    - Löse Graphen-Version des minimalen Steiner-Baumes

Modellierung 1

- Für Standardzelltechnologie
- Modellierung der Baum-Geometrie



m x n Matrix
V-Abstand variabel



Eingebetteter Baum
Verschmolzene Terminals

Modellierung 2

- **Lokale vertikale Dichte $d_v(i,j)$**
  - Leitungen durch V-Segment *i*-1,*i* in Spalte *j*
- **Lokale horizontale Dichte $d_h(i,j)$**
  - Leitungen durch H-Segment *j*-1,*j* in Zeile *i*
- **Kanaldichte** $D_v(i) = \max_{j=1}^{n} d_v(i, j)$
- **Gesamtkanaldichte**

$$D_T = \sum_{i=1}^{m} D_v(i)$$

- **Ziel: Minimiere D_T mit $d_h(i,j) \leq M_{ij}$**
  - M_{ij}: Verfügbare vertikale Feedthroughs im H-Segment *j*-1,*j* in Zeile *i*

Mögliche Vorgehensweisen

1. **Variante von Lees Algorithmus**
   - Erhöhe Überquerungskosten je Segment
     - Nach jedem Netz
   - Probleme
     - Versagt bei Auswahl aus vielen gleich guten Routen
     - Qualität abhängig von Netzreihenfolge
2. **Sequentieller Aufbau von RSMT je Netz**
   - Bestimme Kantenkosten aus d_v, d_h
     - Gute einzelne Routing-Ergebnisse
   - Qualität noch abhängig von Reihenfolge
3. **Pseudo-simultanes Routing**
   - Konstruiere unabhängigen RSMT je Netz
     - Immer optimale Route, unabhängig von Reihenfolge
   - Korrigiere Verstopfung (congestion) später

Variante

- Hierarchische Vorgehensweise
- Beginne mit 2x2 Raster über gesamten Chip
- Löse globales Verdrahtungsproblem
- Für jeden der Quadranten
  - Unteraufteilung in eigenes 2x2 Raster
  - Löse globales Verdrahtungsproblem erneut
- Divide-and-Conquer Vorgehen
- Im Extremfall: Bis hin zu einzelnen Terminals
  - Erledigt komplette Verdrahtung
  - Inklusive Kanalverdrahtung
- Optimalitätsprinzip gilt aber nicht!
  - Leitungen aus Partition hinaus beeinflussen Unterentscheidungen

RSMT Problem

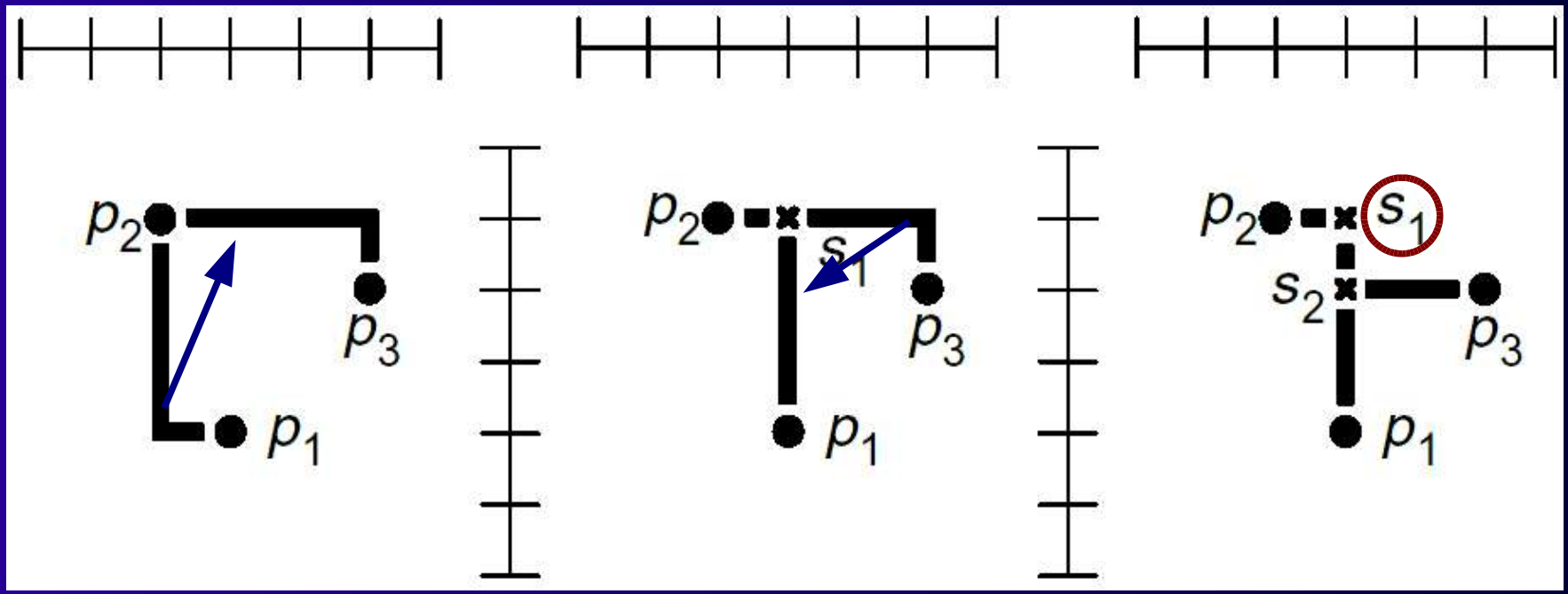
- **Rechtwinklige minimale Steiner-Bäume**
  - Nützlich zur Lösung von glob. Verdrahtungsproblemen
- **Gegeben**
  - P = {p1, p2, ...}: Punktmenge in der Ebene (2-D)
  - Distanzmetrik: $|x_i - x_j| + |y_i - y_j|$
- **Gesucht**
  - Finde verbindenden Baum für Punkte in P
    - Mit minimaler Gesamlänge!
  - Erlaube zusätzliche Punkte im Baum
    - Wenn sie zu kürzerer Gesamtlänge führen
    - Sogenannte „Steiner-Punkte“
- **Hier vernachlässigt**
  - Timing
  - Übersprechen

Lösung

- **Exakt: NP-vollständig**
- **Approximieren durch MRST**
  - Minimaler rechtwinkliger aufspannender Baum
  - Prims Algorithmus: $O(n^2)$
    - Maximal 1.5x länger als echter Steiner-Baum
  - Idee: Hinterher Ergebnis verbessern
- **Ausblick: Neue Heuristiken**
  - Verbesserter MRST max. 11/8x länger als RSMT
    - Fössmeier et al. 1997

MRST Optimierung

- **Beispiel: Lokales Umlegen von L-Stücken**
  - Führt zu Steiner Punkten
  - Ziel: Verschmelzen von Segmenten
    - Reduktion der Gesamtlänge



- **Steiner-Punkte haben Grad ≥ 3**
  - s_1 verschwindet (kein Steiner-Punkt mehr)

Besser: MRST-Erweiterung

- **Vorteil: Nicht schlechter als 4/3x RSMT**
  - Auch wenn MRST schlechtestes Ergebnis liefert
    - Wenn MRST = 1.5x RSMT, verbesserter MRST $\leq$ 1.33x RSMT
- **Beginnt mit MRST nach Prim**
- **Verfeinert dann schrittweise**
  - Nimmt jeweils einzelnen Punkt *s* zu *P* hinzu
    - *s* ist also Steiner-Punkt
  - Wähle *s* so, dass MRST $P \cup \{s\}$ minimal
  - Wird „1-Steiner-Baum-Problem“ genannt
- **Wiederhole!**
- **Liefert beweisbar gute Ergebnisse**
  - Kann aber keine optimale Lösung garantieren

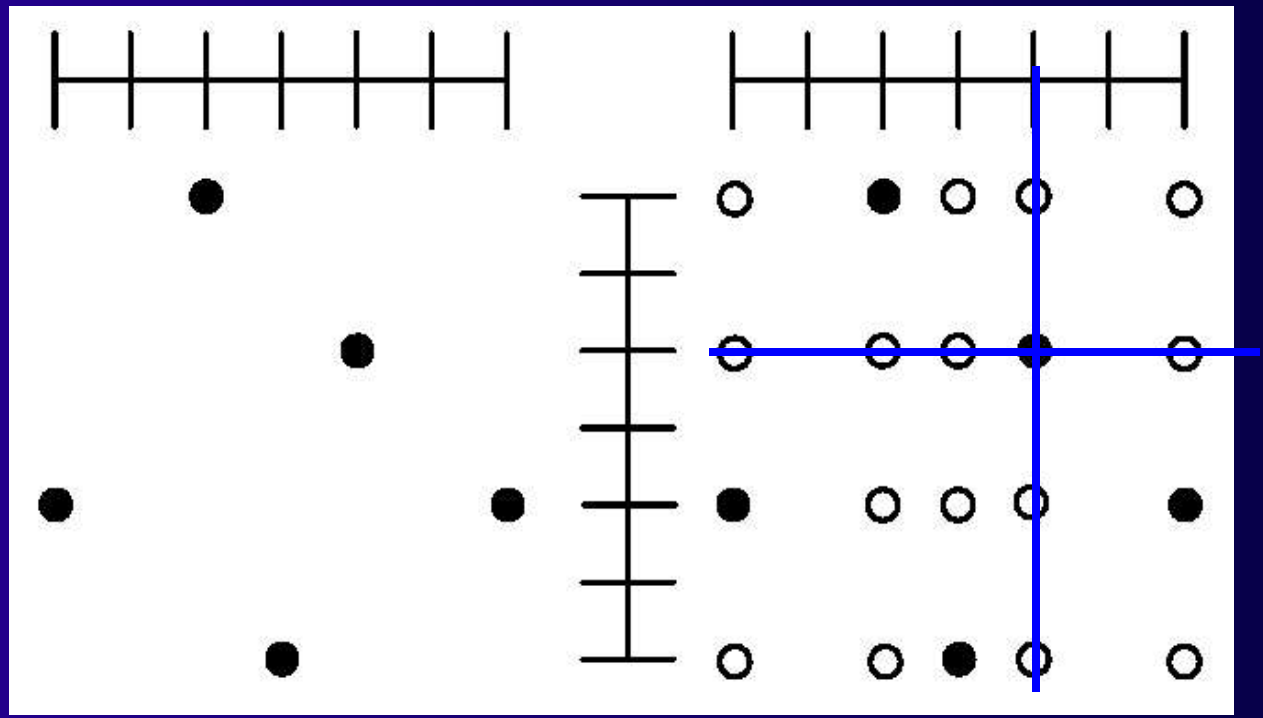
Algorithmus steiner

```
pair<set<vertex>,set<edge>>
steiner(set<vertex> P) {
        set<vertex>     T;
        set<edge>       E, F;
        int gain;   // Längenverkürzung

        E = P.primMRST();
        (T,F,gain) = oneSteiner(P, E);
        while (gain > 0) {
                P = T;
                E = F;
                (T,F,gain) = oneSteiner(P, E);
        }
        return (P,E);
}
```

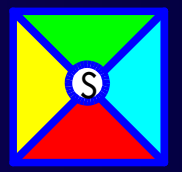
1-Steiner-Baum Konstruktion 1

- **Wie den Punkt *s* bestimmen?**
  - Alle Punkte ausserhalb von *P* ausprobieren
  - ... geht aber besser!
- **Auf Hanan-Punkte beschränken (1966)**
  - Erlaubt trotzdem Finden des Optimums



1-Steiner-Baum Konstruktion 2

- **Für Auswahl des besten Punktes *s***
  - Immer wieder MRST $P \cup \{s\}$ via Prim bestimmen
    - Punkt mit kürzestem Baum wird genommen
  - Geht auch besser ...
- **Inkrementelle Berechnung des MRST**
  - Aus MRST für *P* hin zu MRST für $P \cup \{s\}$
    - In linearer Zeit O(*n*)
- **Idee**
  - Punkte im Baum haben max. Grad 4
  - *s* muss an Baum für *P* angeschlossen werden
  - Lage des *s* nächsten Punktes im Baum für *P*
    - In einer der Regionen N,E,S,W um *s*
      - *N,S:* $|d_x| \leq |d_y|$, *E,W:* $|d_y| \leq |d_x|$



Algorithmus oneSteiner

```
triple<set<vertex>,set<edge>,int>
oneSteiner(set<vertex> V, set<edge> E) {
        int maxgain;        vertex maxpoint;
        int gain;
        set<vertex> W;      set<edge> F;

        maxgain = 0;
        foreach s ∈ „Hanan-Punkte von V“ do {
                (W,F,gain) = spanningUpdate(V,E,s);
                if (gain > maxgain) {
                        maxgain     = gain;
                        maxpoint    = s;
                }
        }
        if (maxgain > 0) {
                (W,F,gain) = spanningUpdate(V,E,maxpoint);
                return (W,F,gain);
        } else
                return (V,E,0);
}
```

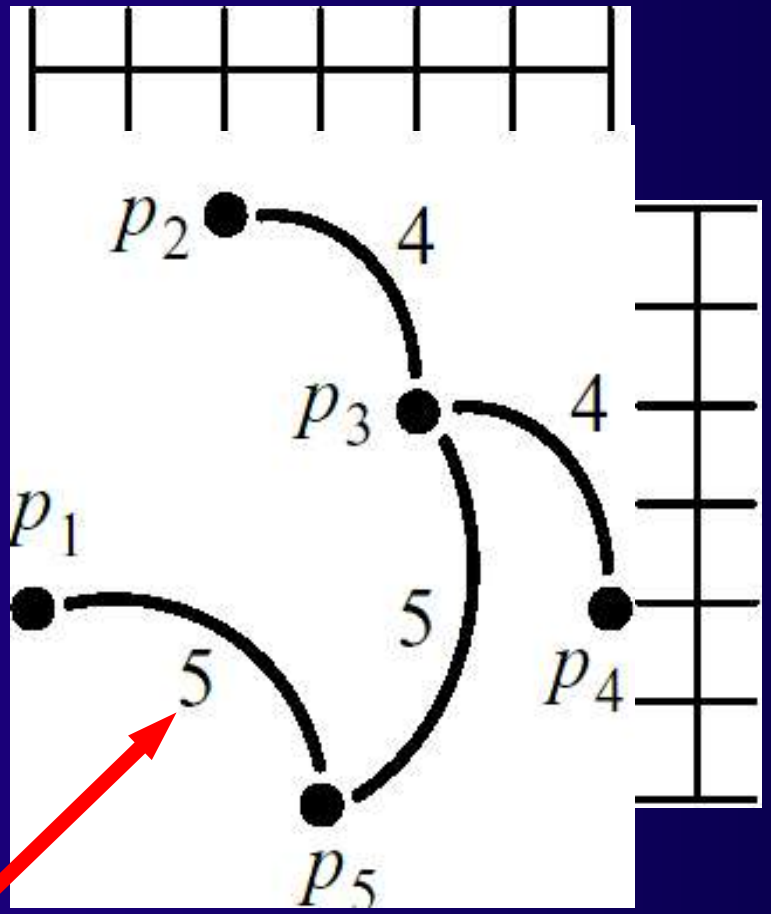
Algorithmus spanningUpdate

```
triple<set<vertex>,set<edge>,int>
spanningUpdate(set<vertex> V, set<edge> E, vertex s) {
        int             delta;          // Längenverkürzung
        vertex          u, v, w;

        delta = 0;
        V = V ∪ {s};
        foreach d ∈ {NORTH, EAST, SOUTH, WEST} do {
                u = s.closestPointInTree(V, d);
                E = E ∪ {(s,u)};        // s an alle Partner anschliessen
                delta = delta – distance(s,u);
                if (hasCycle(V, E)) {
                        (v,w) = findLongestCycleSegment(V, E);
                        E = E \ {(v,w)};
                        delta = delta + distance(v,w);
                }
        }
        return (V, E, delta);
}
```

Beispiel Schritt 1

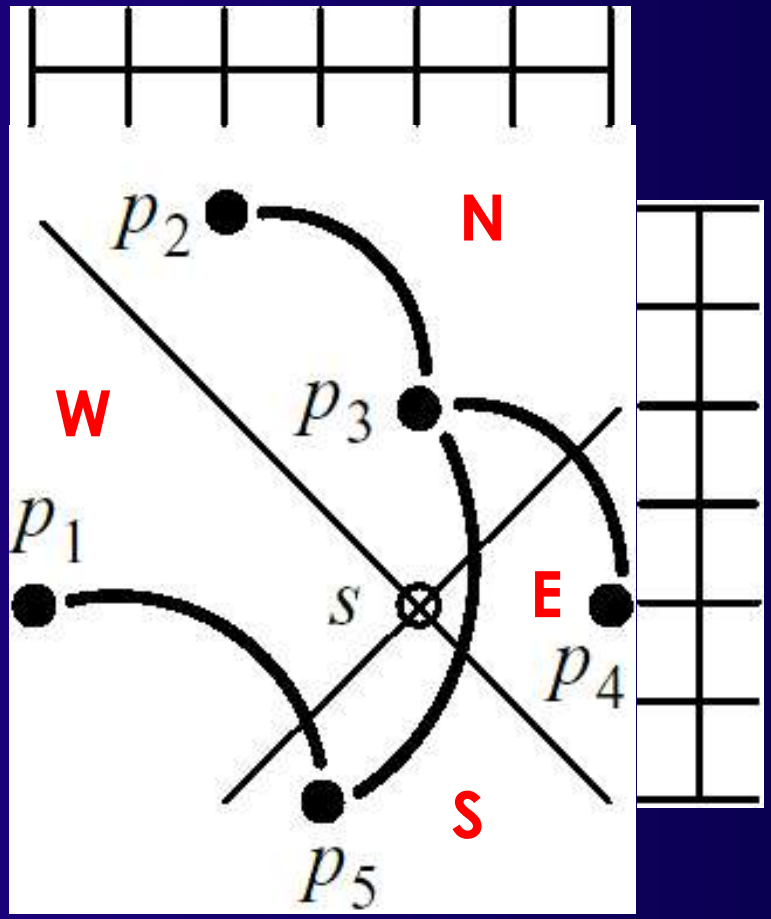
Eingabe: MRST, z.B. via Prims Algorithmus



Bögen geben nur Distanz an, noch keine genaue Führung

Beispiel Schritt 2

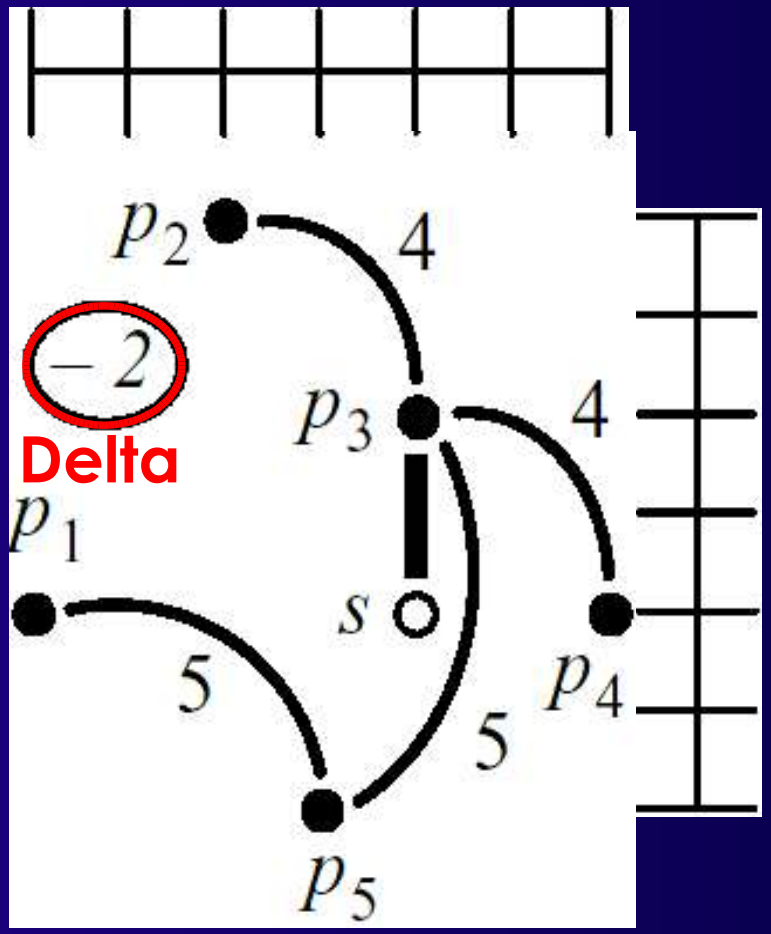
Hinzunahme eines ersten Hanan-Punktes *s*



***s* nahegelegenste Punkte aus *P*: p_3, p_4, p_5, p_1**

Beispiel Schritt 3

Anbinden an den ersten *s* benachbarten Punkt p_3 im N

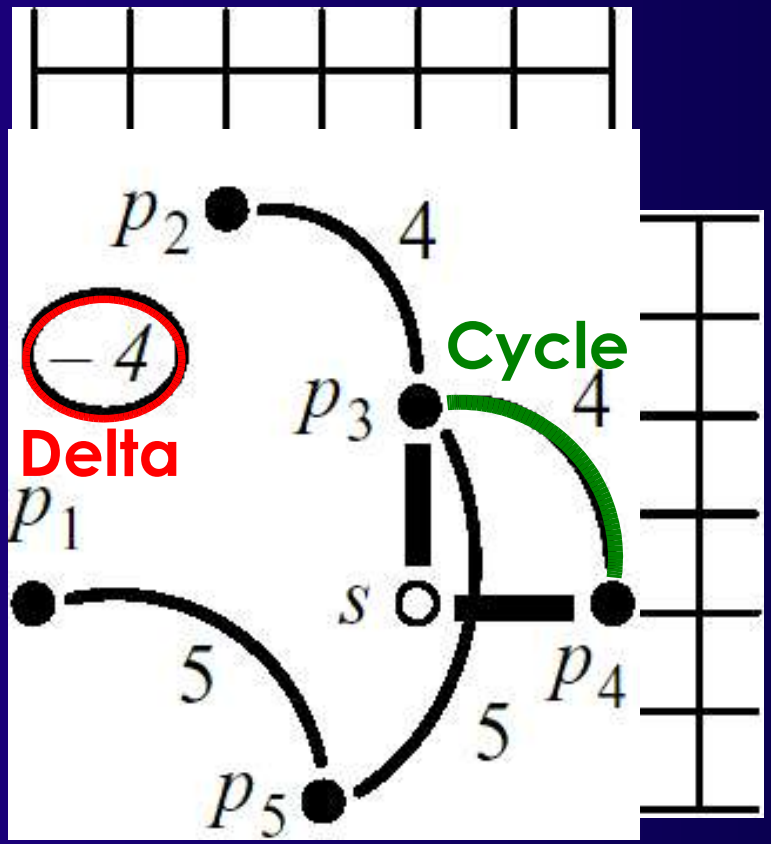


Nun festgelegte kürzeste Führung, Erhöhung der Länge

- **Feste Verbindung für Punkte auf selber Rasterlinie**

Beispiel Schritt 4

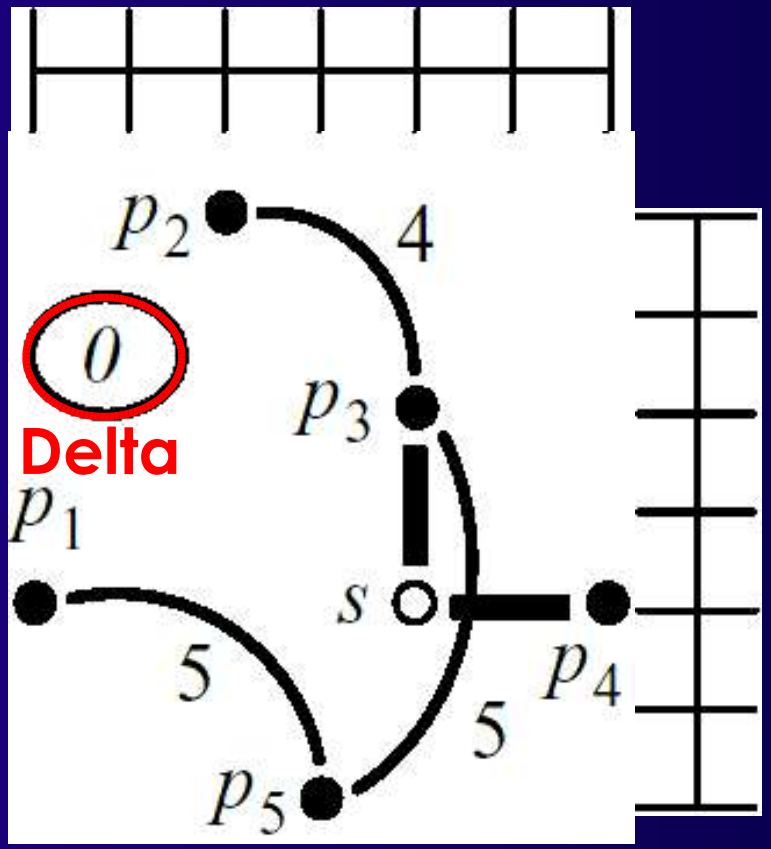
Anbinden an den zweiten *s* benachbarten Punkt p_4 im E



Auch festgelegte Führung und Erhöhung der Länge, Zyklus

Beispiel Schritt 5

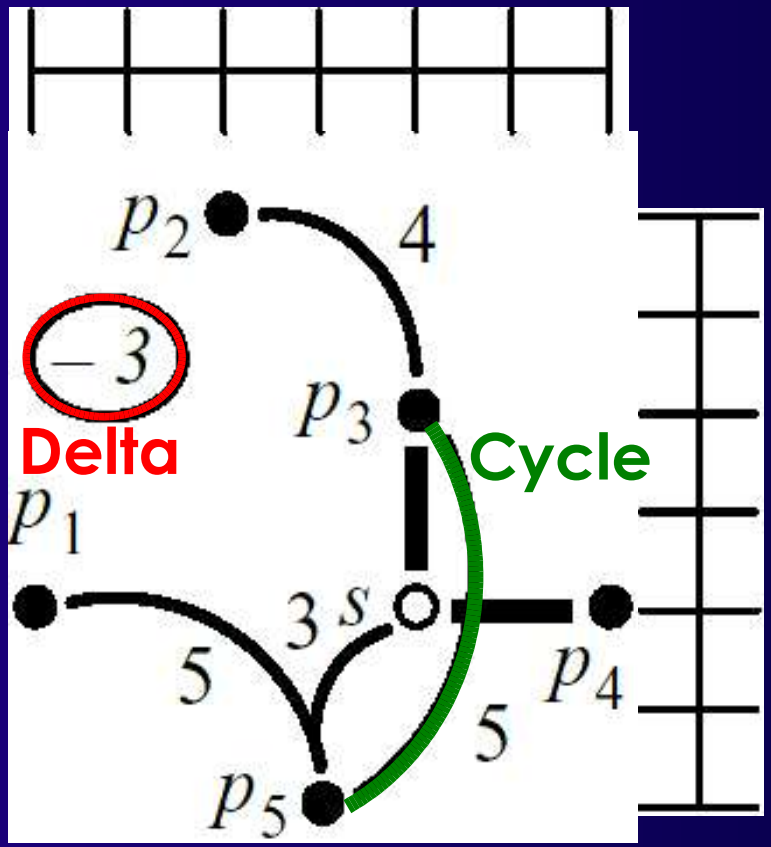
Entferne längste Kante d({p_4,p_3})=4 aus Zyklus



Gesamtlänge verkürzt sich nun um 4

Beispiel Schritt 6

Anbinden an den dritten *s* benachbarten Punkt p_5 im S

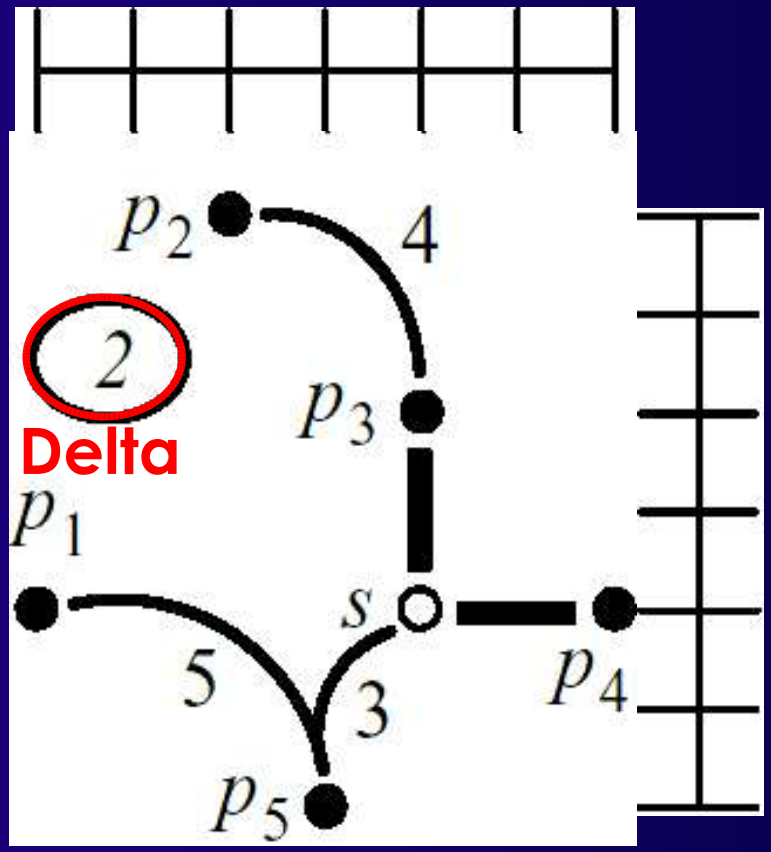


Noch keine feste Führung, Gesamtlänge erhöht sich, Zyklus

- ***s* und *p5* nicht auf selber Rasterlinie**

Beispiel Schritt 7

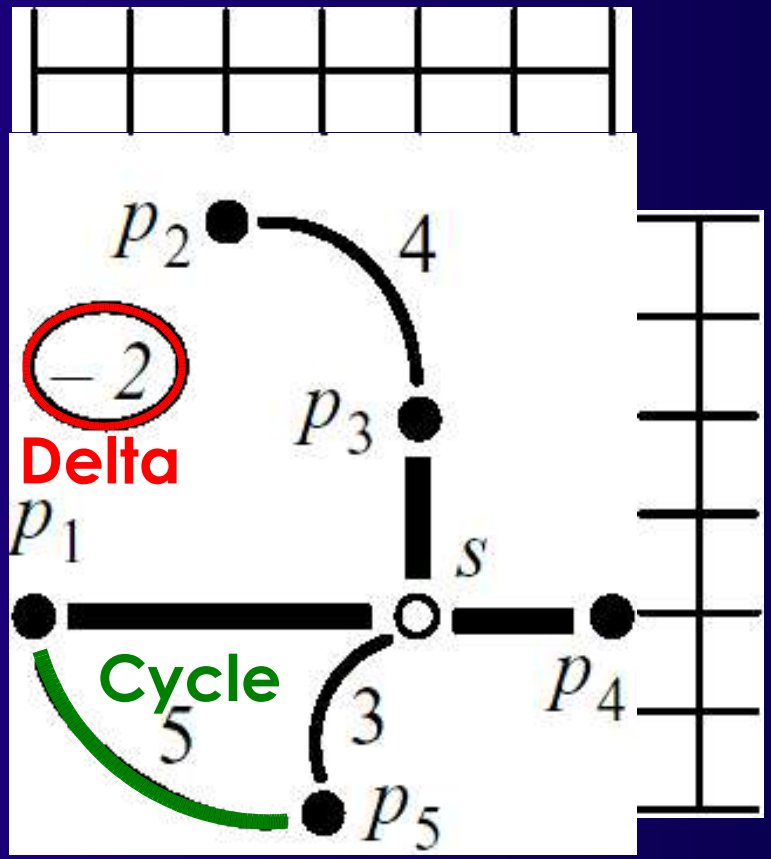
Entferne längste Kante $d(\{p_5,p_3\})=5$ aus Zyklus



Gesamtlänge verkürzt sich nun um 5

Beispiel Schritt 8

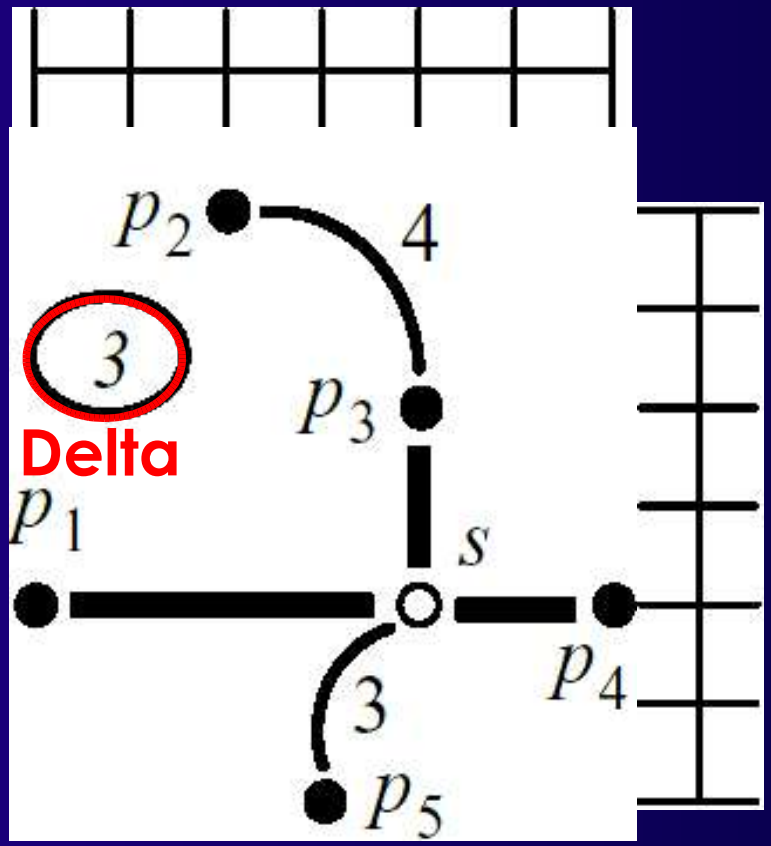
Anbinden an den vierten *s* benachbarten Punkt p_1 im W



Feste Führung, Gesamtlänge erhöht sich, Zyklus

Beispiel Schritt 9

Entferne längste Kante d({p_5,p_1})=5 aus Zyklus



Gesamtlänge verkürzt sich nun um 5, Gesamtgewinn ist 3

Komplexität

- spanningUpdate()
  - 4x closestPoint(): O(n)
  - hasCycle(): DFS mit History, O(n)
  - findLongestCycleSegment(): History, O(n)

  ⇨ Gesamt: O(n)
- Anzahl Hanan-Punkte: O(n^2)
- oneSteiner() Gesamt: O(n^3)
- steiner() Gesamt: O(n^5)

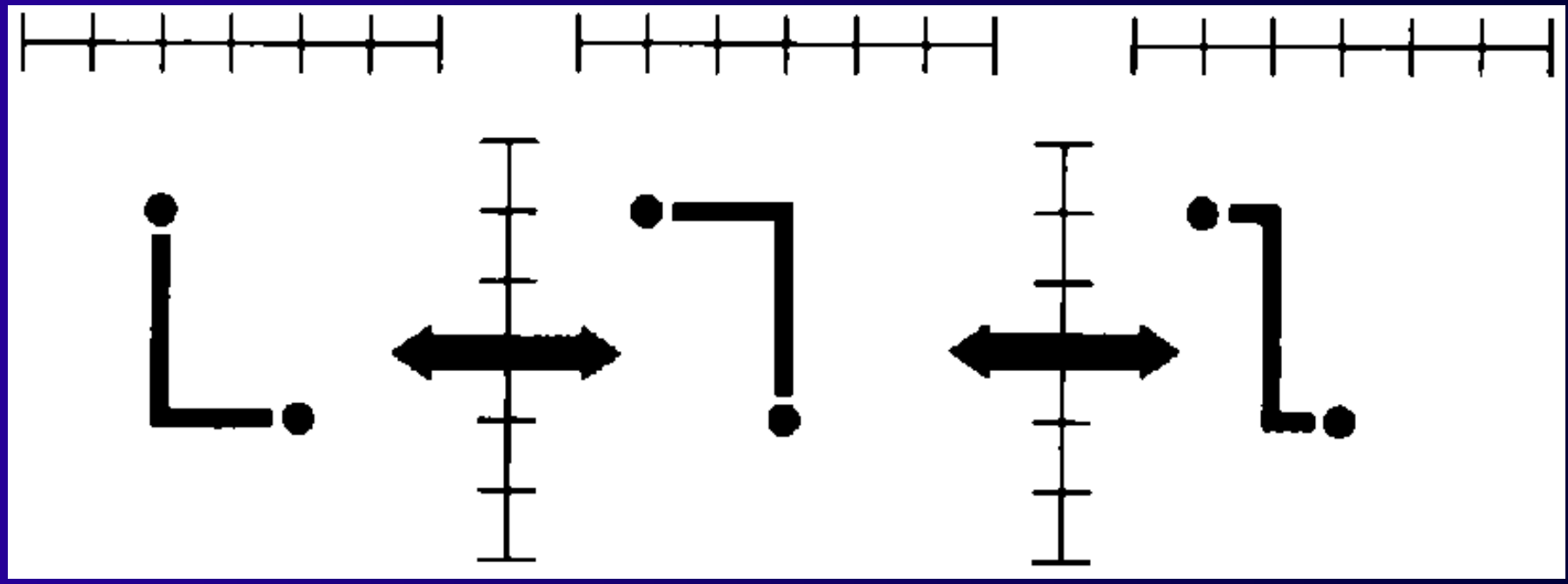
- Im Durchschnitt aber besser
  - z.B. oneSteiner() nur 2x aufgerufen bei n=40

  ⇨ O(n^3)

Beseitigen von Verstopfungen

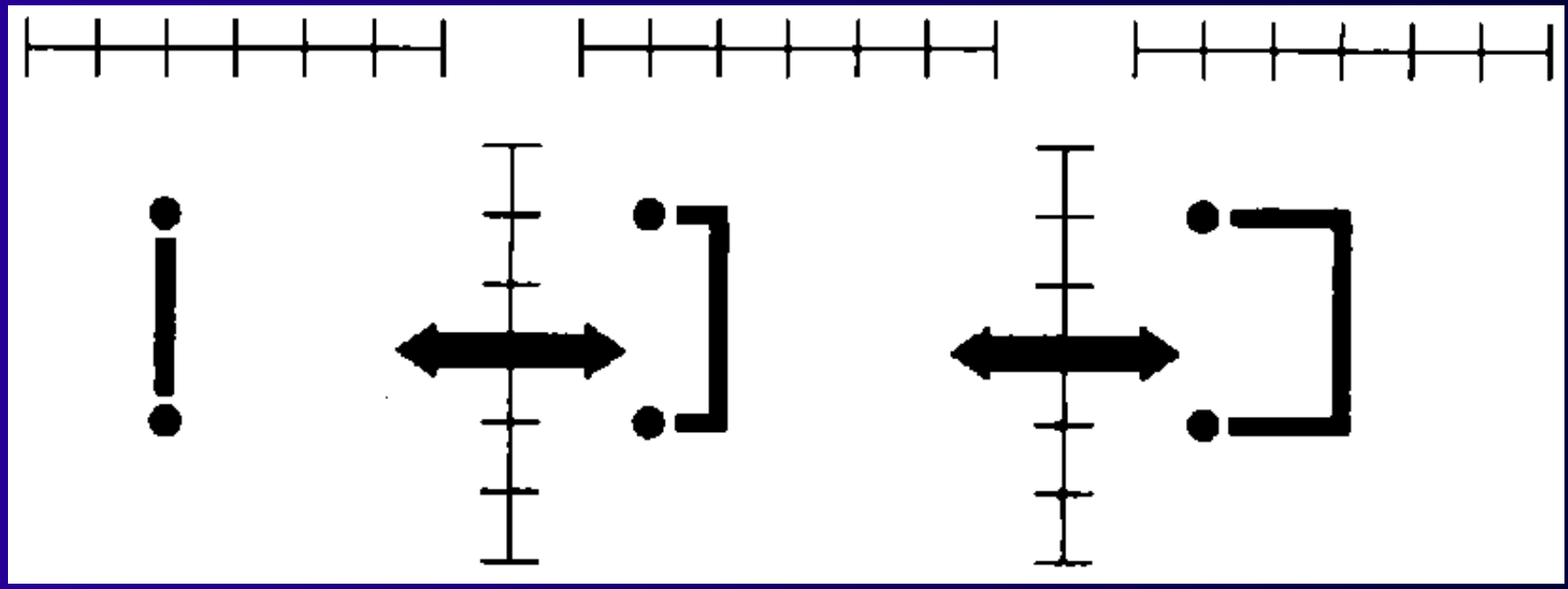
- **Bisher unabhängige RSMTs: Einer je Netz**
  - Ähnlich erster Durchgang bei PathFinder
- **Nachfrage nach V-Feedthroughs**
  - Bestimmen
  - Stark verstopfte Stellen entlasten
- **Lokale Transformation der einzelnen RSMTs**
  - Kontrolliert durch eigene Optimierung
    - Z.B. Simulated Annealing oder Nachbarsuche

Lokale Transformation 1



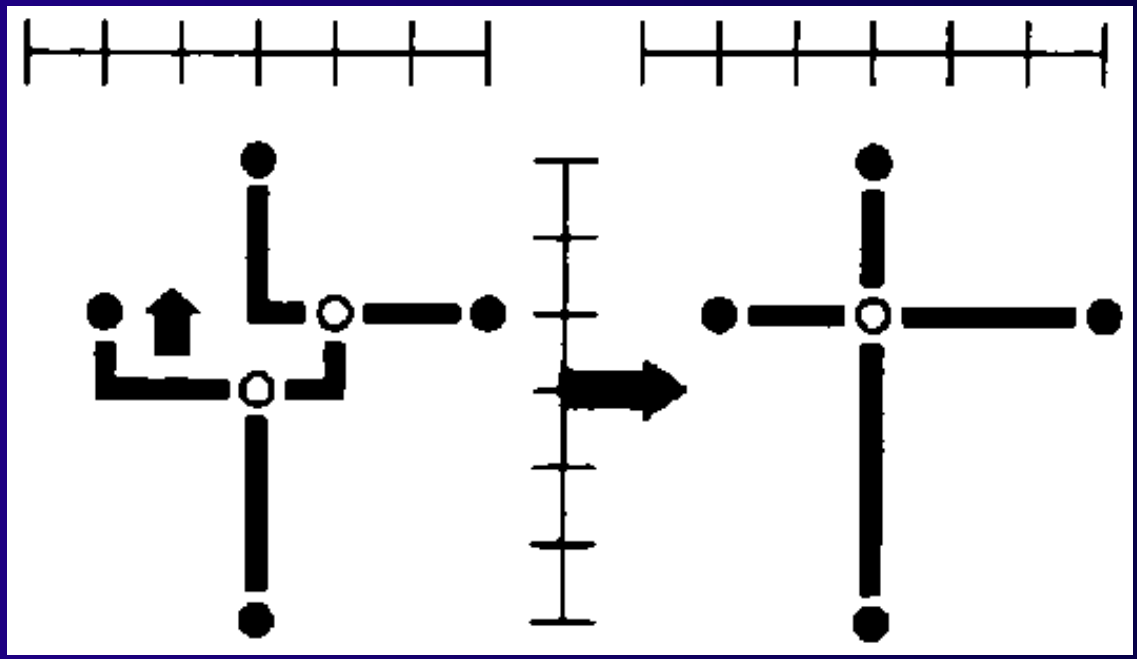
- Variiere konkrete Führung einer Kante
- Länge bleibt gleich

Lokale Transformation 2



- **Länge erhöht sich**
  - Kann aber Gesamtkosten senken

Lokale Transformation 3



- **Kompliziertere Verscheibung**
  - Vollständiges Entfernen von Steiner-Punkten

Ausblick

- **Nächste Vorlesung: Di, 3.1.2006**
  - Floorplanning, Kapitel 8

Zusammenfassung

- Abhängig von Zieltechnologien
- Steiner-Bäume
  - Optimierungsziele
- Routing in Slicing-Floorplans
  - CDP, COP
- Globale Verdrahtung für Standardzellen
  - Konstruktion von Steiner-Bäumen
  - Lokale Optimierung