technische universität
dortmund

fakultät für informatik
informatik 12

Layout-Synthese

- Detaillierte Verdrahtung: Kanalverdrahtung -

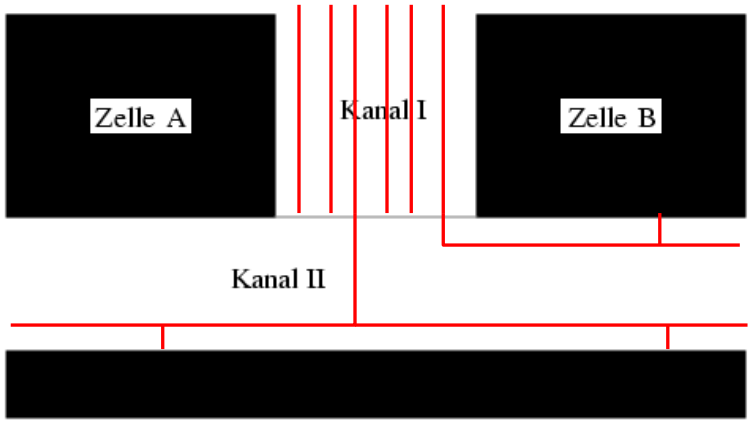
Peter Marwedel
Universität Dortmund

2008/07/09

Sortierung der Verdrahtungsregionen

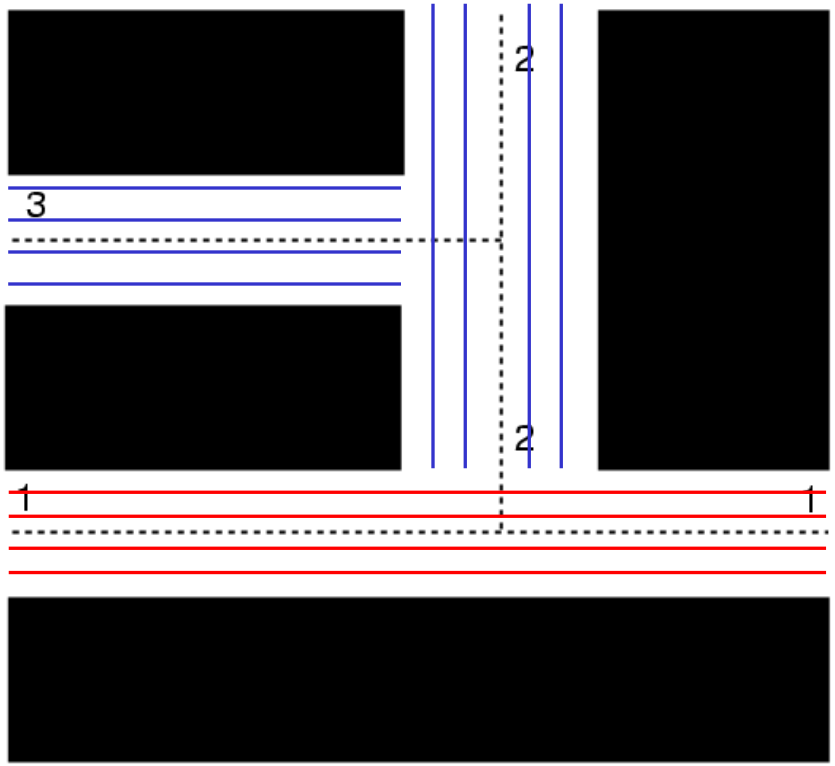
Während der detaillierten Verdrahtung werden wir v.a. eine Verdrahtung innerhalb der Kanäle betrachten,

Um die Kanalverdrahtungs-Algorithmen aus dem nächsten Abschnitt anwenden zu können, muß die Verdrahtung der Kanäle in einer bestimmten Reihenfolge erfolgen: Beispiel: Kanal I muß vor Kanal II verdrahtet werden, damit während der Kanalverdrahtung des Kanals II bereits die Lage der Netze am Rand des Kanals bekannt ist und damit der Abstand zwischen den Zellen A und B während der Verdrahtung des Kanals I noch verändert werden kann:



Konsequenz

Als Konsequenz muß die Verdrahtung einer Slicing–Struktur in einer gegenüber der Schnittabfolge umgekehrten Reihenfolge durchgeführt werden.



Verdrahtung bei einer Verdrahtungsebene

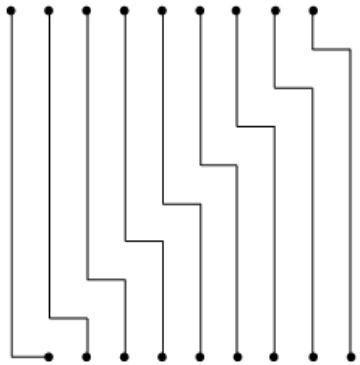
Selbst dann wichtig, wenn mehrere Verdrahtungsebenen existieren:

Netze für Betriebsspannung und Taktsignal.

Ein Netzgraph kann in einer Ebene realisiert werden, wenn der Netzgraph kreuzungsfrei gezeichnet werden kann (wenn er **planarisierbar** ist).

Wenn alle Netze Zweipunkt-Netze sind, dann bezeichnet man die Verdrahtung in einer Ebene als *river routing*.

Beispiel:



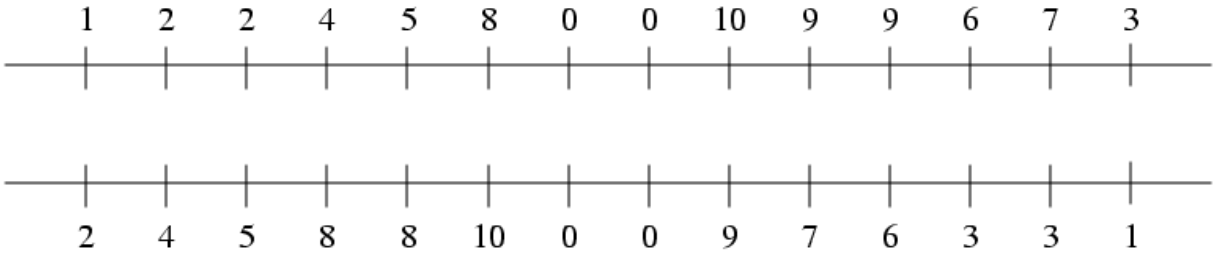
Kanalverdrahtung in zwei Ebenen

Problemstellung: Modell

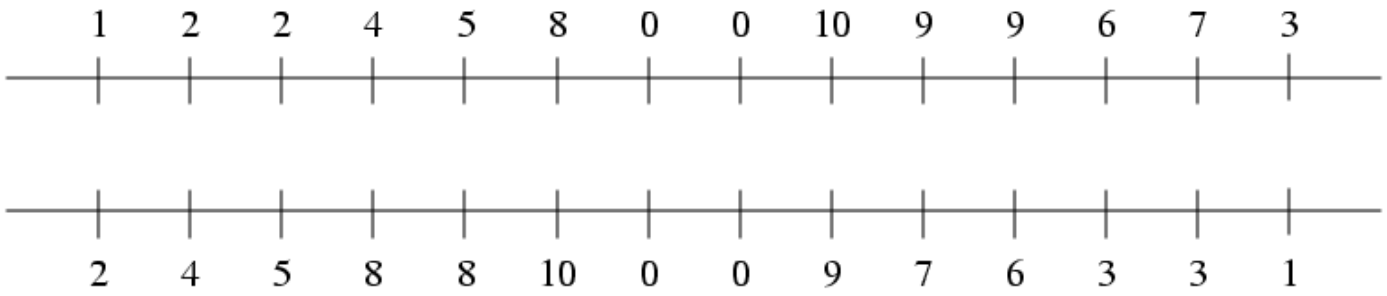
- Ein Kanal ist ein Rechteck, das frei von a priori blockierten Flächen ist.
- Die Anschlüsse der Netze an zwei gegenüberliegenden Seiten des Rechtecks.

  Anschlüsse mit Nummer des Netzes beschriftet. 0= nicht beschaltet.
- Die Länge der beiden anderen Seiten ist (in der Regel) variabel und sollte so klein wie möglich sein.

Beispiel:



Kodierung des Kanalverdrahtungsproblems



Kodierung in zwei Arrays `top` und `bottom`:

Top [1..14] = 1,2,2,4,5,8,0,0,10,9,9,6,7,3

Bottom[1..14] = 2,4,5,8,8,10,0,0,9,7,6,3,3,1

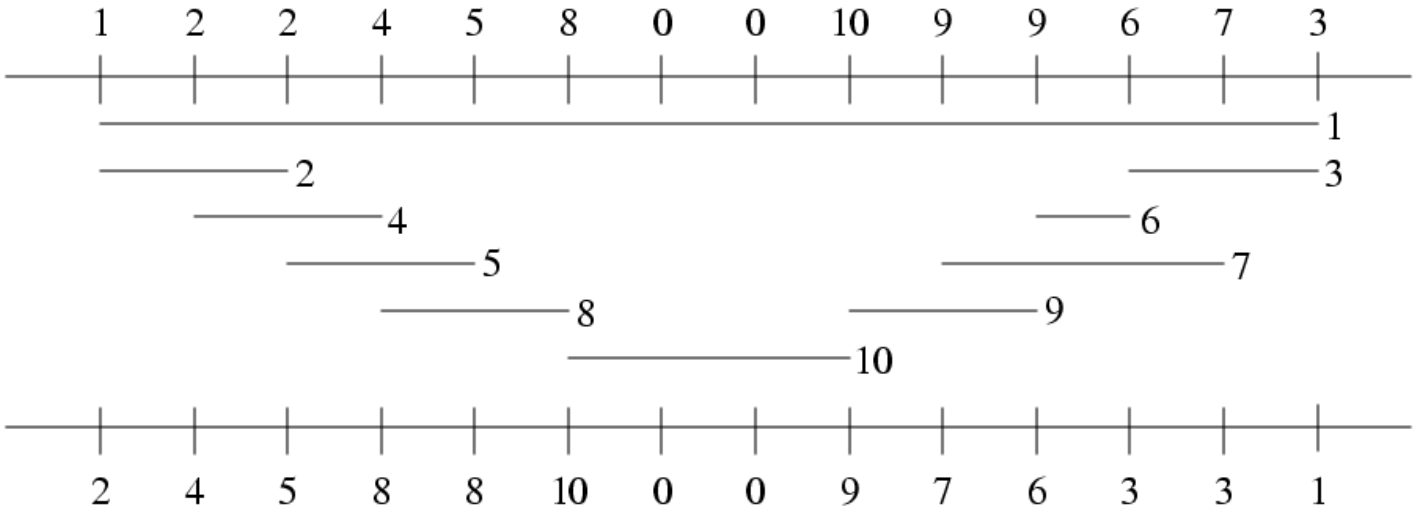
Kanalverdrahtung

Im Folgenden 2–Lagen–Verdrahtung:

eine Lage vertikal, eine Lage horizontal benutzt.

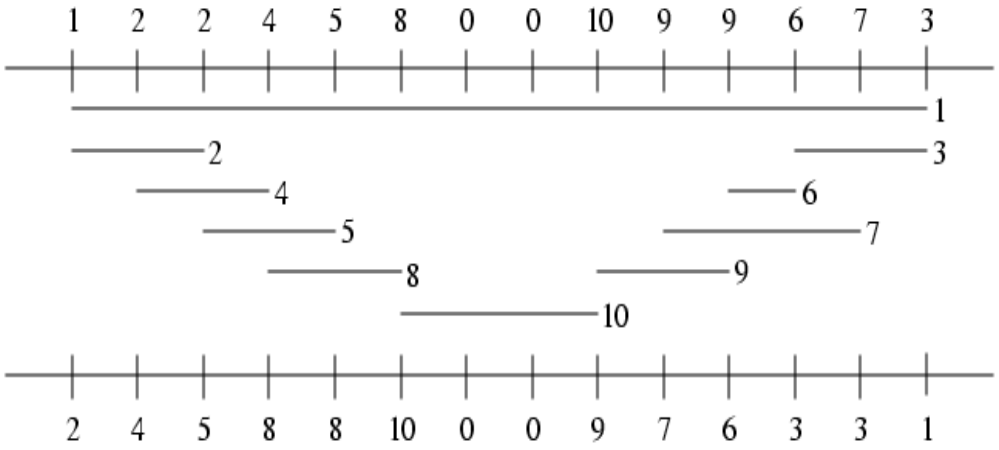
Def.: Eine Spalte heißt **von einem Netz belegt**, wenn die Spalte im Intervall zwischen dem äußersten linken und dem äußersten rechten Anschluß des Netzes enthalten ist.

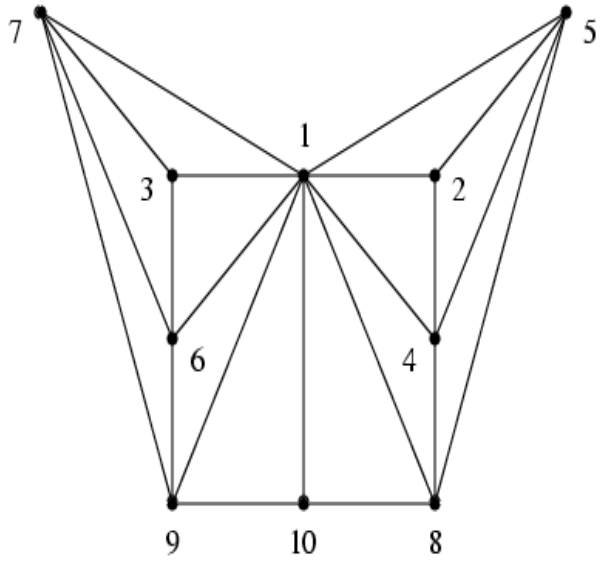
Beispiel: Intervalle der Netze für obiges Beispiel



Horizontal constraints graph

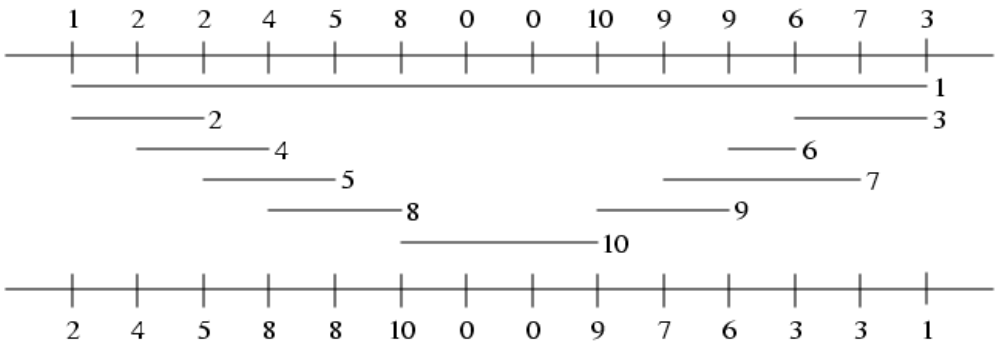
Def.: Der *horizontal constraints graph* (HCG) ist ein ungerichteter Graph, der für jedes Netz genau einen Knoten enthält. Eine Kante zwischen zwei Knoten i und j existiert genau dann, wenn eine Spalte existiert, die von beiden Netzen belegt wird.

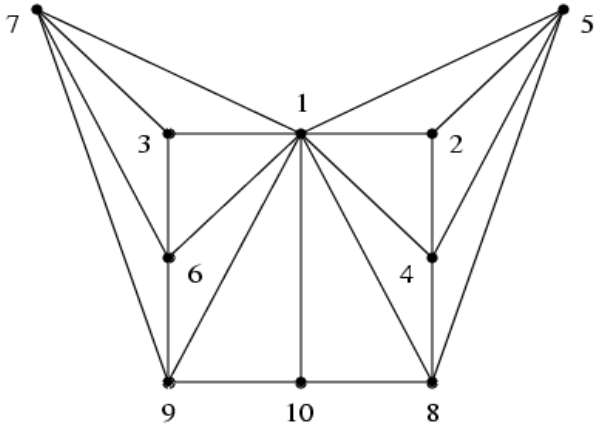




Intervallgraphen

Def.: Graph $G = (V, E)$ heißt Intervallgraph $\Leftrightarrow$ Es existiert eine eineindeutige Abbildung der Knoten $v \in V$ auf Intervalle einer linear geordneten Menge derart, dass Knoten genau dann mit einer Kante $e \in E$ verbunden sind, wenn sich die Intervalle schneiden.





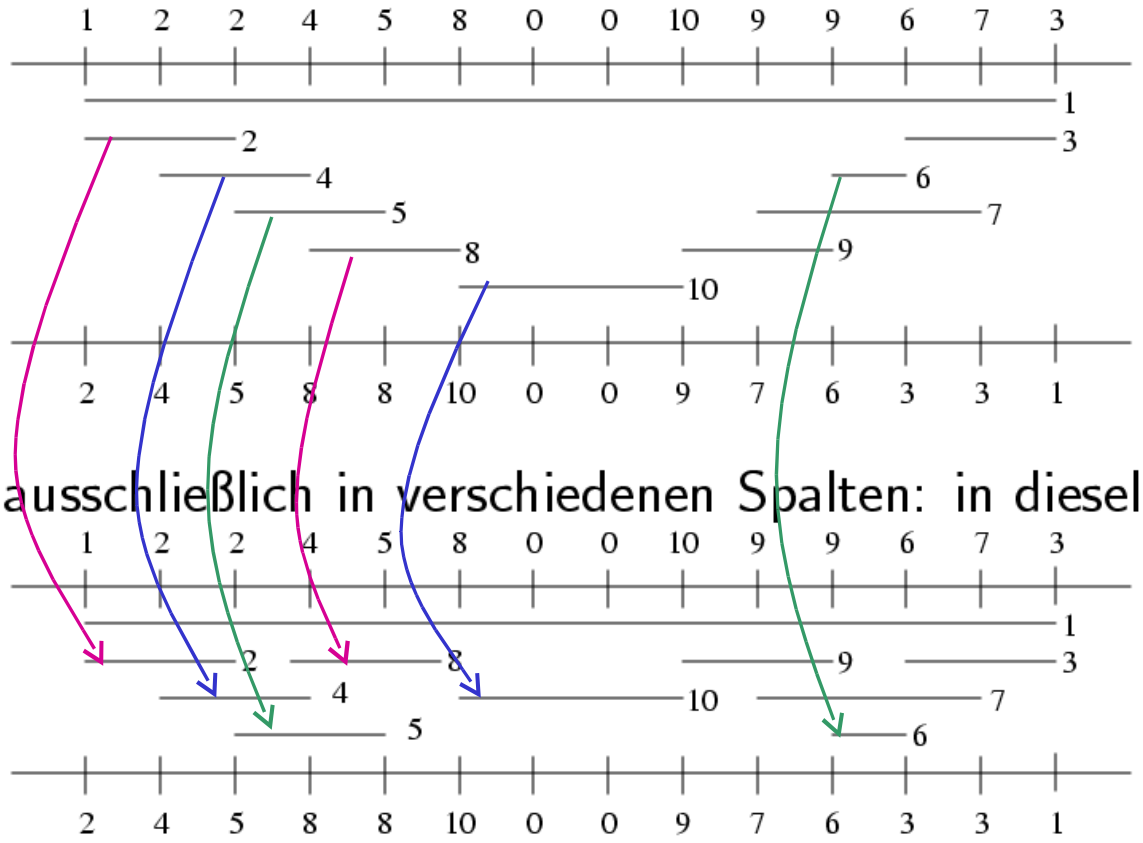
Def.: Der *horizontal constraints graph* (HCG) ist ein ungerichteter Graph, der für jedes Netz genau einen Knoten enthält. Eine Kante zwischen zwei Knoten i und j existiert genau dann, wenn eine Spalte existiert, die von beiden Netzen belegt wird.

☞ Der HCG ist per Konstruktion ein Intervallgraph

Restrictive routing

Maximal ein horizontales Segment pro Netz.

Horizontale Segmente der Netze $\rightarrow$ **Spuren** (*tracks*)



≥ 2 Segmente ausschließlich in verschiedenen Spalten: in dieselbe Spur!

Zusammenhang mit dem Färbungsproblem

Def.: Gegeben sei ein Graph $G = (V, E)$. Das Problem, eine Abbildung der Knoten V auf eine minimale Anzahl von Farben zu finden derart, dass je zwei über eine Kante verbundene Knoten eine verschiedene Farbe haben, heißt **Färbungsproblem**.

Minimierung der Spuranzahl = Färbungsproblem.

Satz: Das allgemeine Färbeproblem ist NP-hart.

Satz: Intervallgraphen lassen sich in linearer Zeit optimal färben.

Der *left-edge* Algorithmus

`first[i]`	Die Menge Netze mit linkem Rand bei Spalte `i`
`last[i]`	Die Menge Netze mit rechtem Rand bei Spalte `i`
`imax`	Die Anzahl der Spalten
`kmax`	Eine obere Schranke für die Anzahl der Spuren
`assign[j]`	Dem Netz `j` zugeordnete Spur.

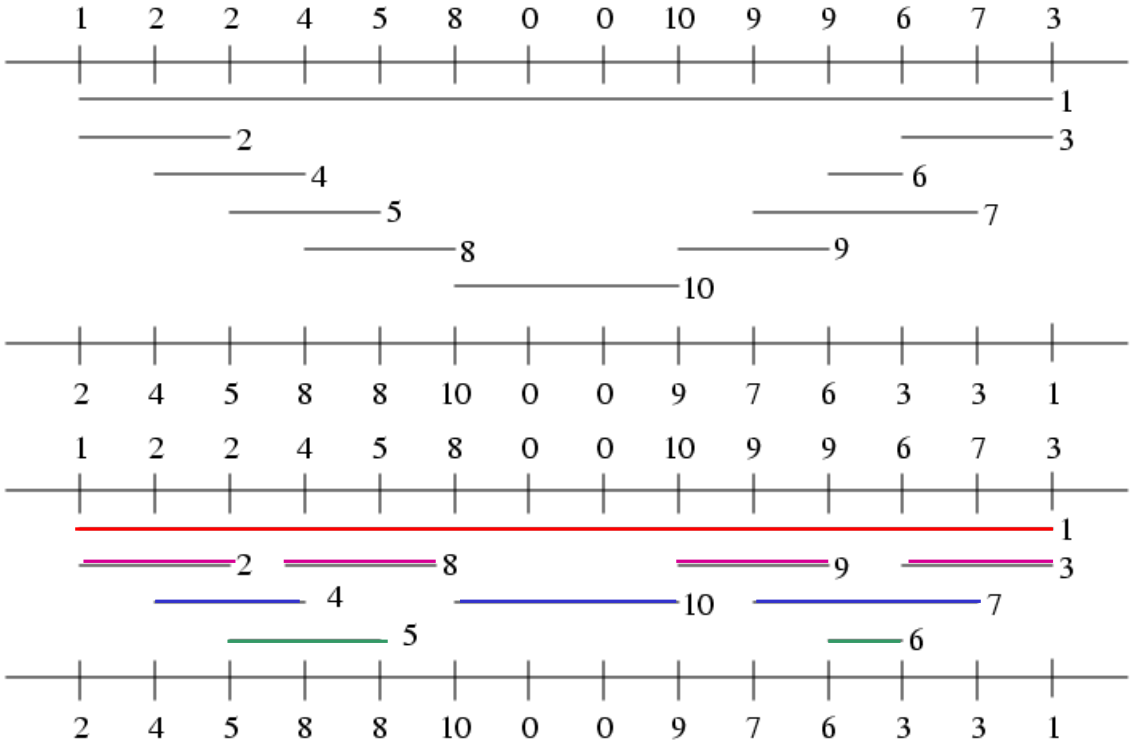
Beginnt Netz in Spalte: ordne freie Spur zu; endet ein Netz: gib Spur frei.

Der *left-edge* Algorithmus (2)

```
Free := {1..kmax}                    % alle Spuren sind frei
FOR i:= 1 to imax DO                 % Schleife über Spalten
 BEGIN
  FOR EACH j IN first[i] DO
   BEGIN
    k := das kleinste Element aus Free;
    assign[j] := k;                  % Zuordnung Netz -> Spur
    Free := Free - {k};              % Spur ist belegt
   END;
  FOR EACH j IN last[i] DO
   Free := Free + assign[j]; % gebe Spur frei
 END;
```

Lösung mit minimaler Spurenzahl, 'linear'.

Beispiel

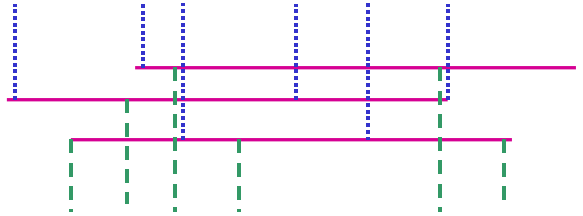


Zahl der benötigten Spuren = *channel density*;

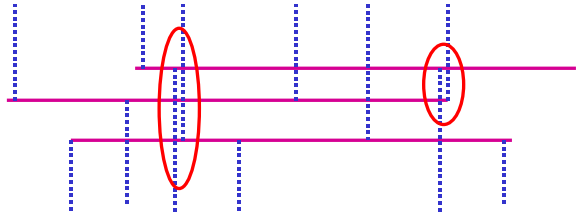
Untere Schranke für die Zahl von Spuren, die für das allgemeine Problem (d.h. unter Berücksichtigung der vertikalen Segmente) notwendig sind.

Berücksichtigung der vertikalen Segmente

- drei Verdrahtungsebenen: Unmittelbare Realisierung: Je eine Verdrahtungsebene für
  - die horizontalen Segmente
  - die vertikalen Segmente nach oben
  - die vertikalen Segmente nach unten
- zwei Verdrahtungsebenen
  Keine Überlappung der vertikalen Segmente;
  Modellierung durch *vertical constraints graph*

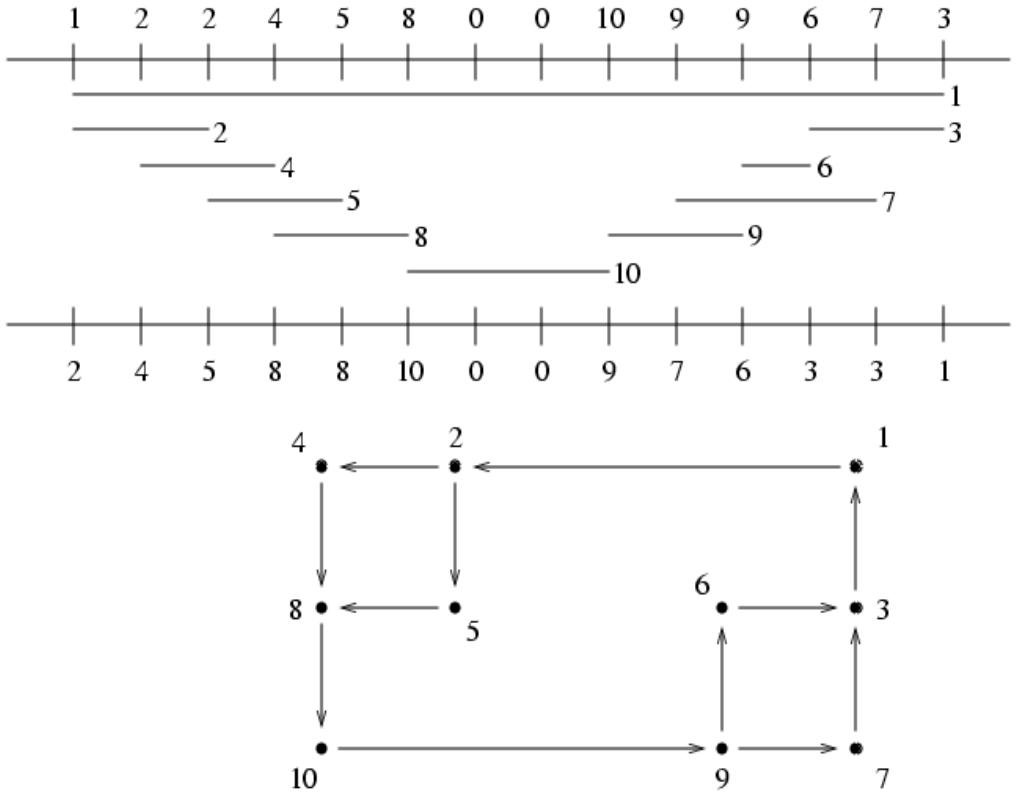


(je Ebene eine Farbe)

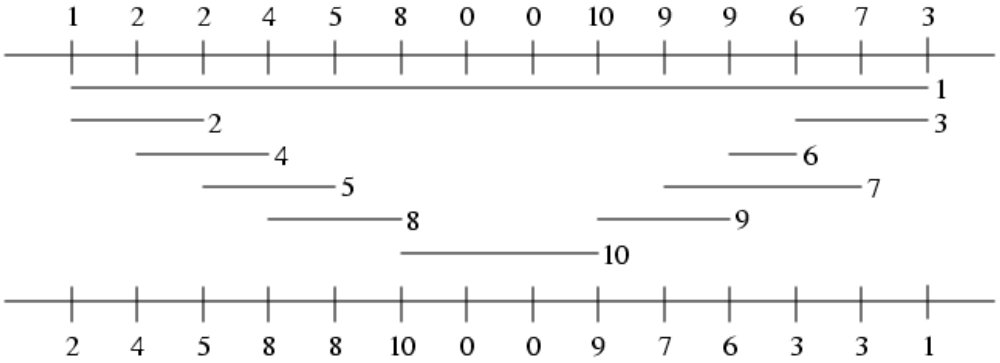


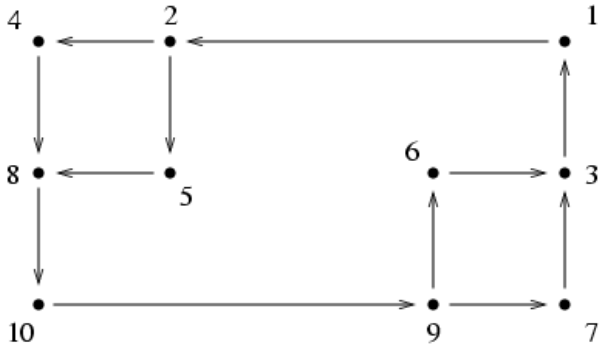
Vertical constraints graph

Def.: Der *vertical constraints graph* (VCG) ist ein gerichteter Graph, der für jedes Netz genau einen Knoten enthält. Eine Kante von Knoten i zum Knoten j existiert genau dann, wenn eine Spalte k existiert mit $\text{Top}[k] = i$ und $\text{Bottom}[k] = j$.



VCG





VCG modelliert die Relation "horizontales Segment des Netzes i muß oberhalb des horizontalen Segments des Netzes j liegen".

Falls der VCG azyklisch ist, beschreibt er eine partielle Ordnung. Die Segmente können dann vertikal in einer totalen Ordnung angeordnet werden, die mit der partiellen Ordnung des VCG (sowie dem HCG) verträglich ist.

Den Vorgang der Bestimmung einer solchen Ordnung nennt man **topologisches Sortieren**.

Im azyklischen Fall ist die Länge des längsten Weges gleichzeitig eine untere Schranke für die *channel density*.

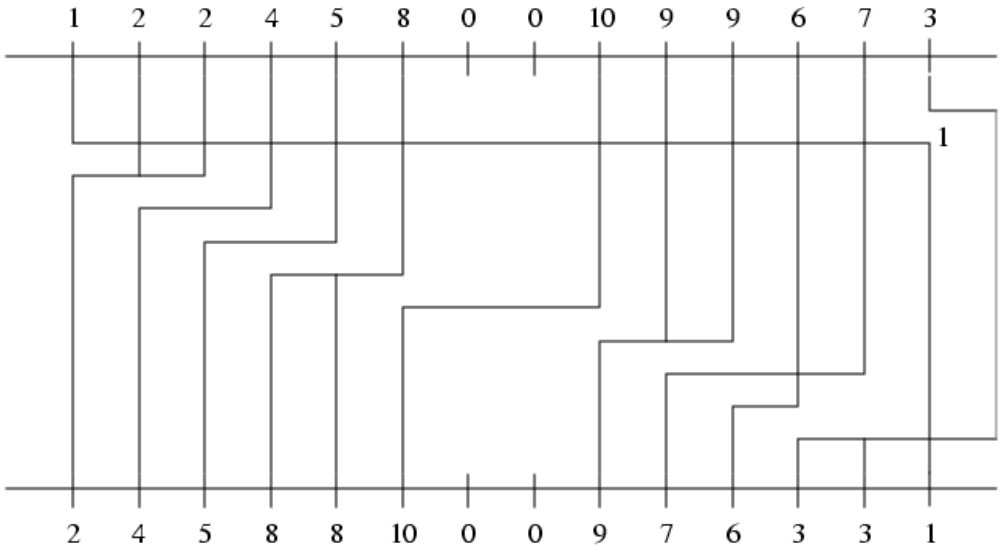
Left edge Algorithmus mit Berücksichtigung des VCG

Optimales *restrictive routing* mit vertikalen Segmenten ist NP–hart.

Heuristische Lösung: erweiterter *left edge*–Algorithmus:

Beim Durchlauf von links nach rechts werden dann nur solche Spuren zugeordnet, die mit dem *vertical constraints*–Graphen verträglich sind.

Beispiel:



Jeweils Start einer neuen Spur, wenn dies wegen des VCGs erforderlich ist.

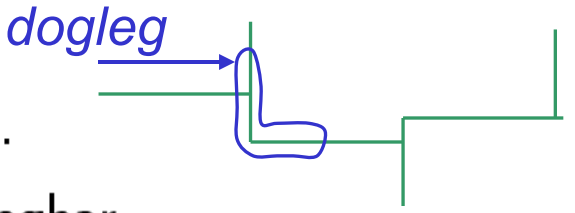
Ohne Netz 3 ok; Mit Netz 3 Verdrahtung mittels *left edge*-Algorithmus nicht möglich. 2. horizont. Segment erforderlich.

http://foghorn.cadlab.lafayette.edu/cadapplets/ChannelRouter.html

Dogleg-Kanalverdrahtung nach Deutsch

Def.: Vertikale Segmente zur Überbrückung zweier horizontaler Segmente heißen *doglegs*. Das entsprechende Verdrahtungsproblem heißt *dogleg channel routing* und ist ebenfalls NP–hart.

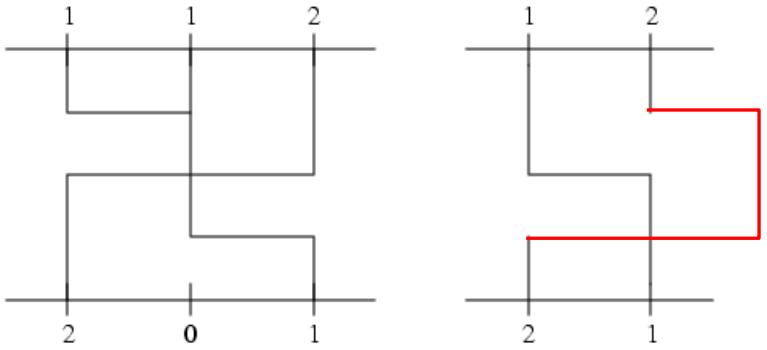
Dogleg–Router von Deutsch:



- Doglegs nur in Spalten mit Netzanschlüssen .
  n-Punkt-Netze in $n-1$ 2–Punkt–Netze zerlegbar.
- Der Abstand aufeinanderfolgender Doglegs ist stets größer als eine Systemkonstante.
- Pro Spalte und Netz existiert höchstens ein horizontales Segment.

Aufbrechen aller Zyklen?

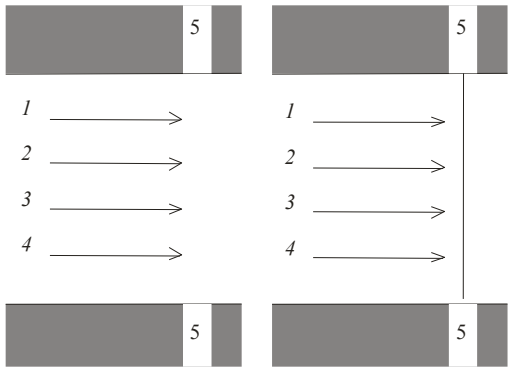
Einige Zyklen können aufgebrochen werden, nicht alle (wegen 3. Einschränkung):



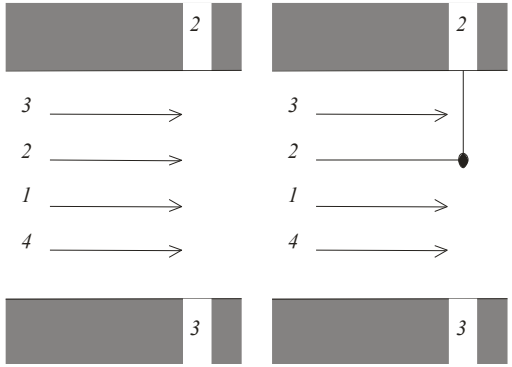
Allgemeines Aufbrechen von Zyklen: Aufheben der dritten Einschränkung.

„*Greedy channel router*“ von Rivest und Fiduccia

Beliebige Zahl horizontaler Segmente pro Spalte und Netz.
Durchlauf durch Kanal von links nach rechts

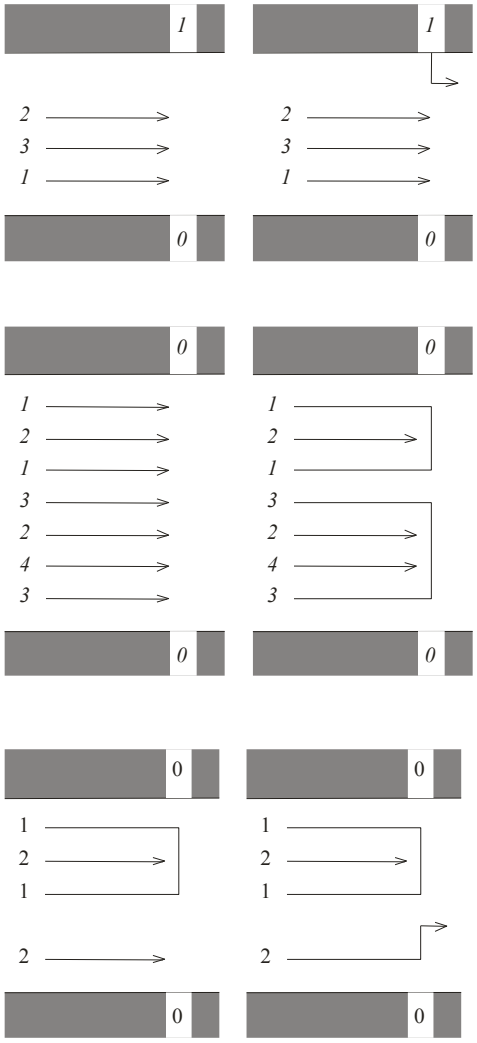


i:=1 (*Spaltennummer*)
repeat (*von links nach rechts*)
Erzeuge mögliche Top- und Bottom-Verbindungen:
1: Falls (Top[i]=Bottom[i]≠0)
dann verbinde Top und Bottom



2: Falls (Top[i] ≠ Bottom[i]) und (Bottom[i] ≠ 0) und
es existiert ein Konflikt,
dann erzeuge kürzeste Verbindung

„*Greedy channel router*“ von Rivest und Fiduccia (2)



3: Falls (Top[i] ≠ 0) oder (Bottom[i] ≠0)
dann bringe das Netz an diejenige nächste Spur, die entweder frei oder bereits vom Netz belegt ist. Hier wird nicht mit der belegten Spur verbunden!

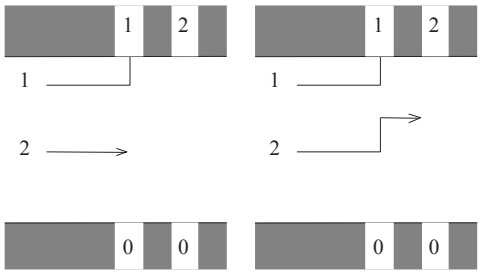
Vereinigung von geteilten Netzen:

Vollständige Suche nach *Doglegs*, die den größten Gewinn erbringen.
Gewinn: pro vereinigtem Netz eine Spur sowie für jedes Netz, das beendet wird, eine weitere Spur.

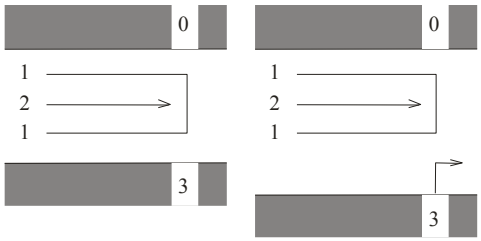
Annäherung von geteilten Netzen:

Äußere Spuren von geteilten Netzen rücken so weit, wie möglich, in die Mitte (Vorbereitung der Vereinigung).

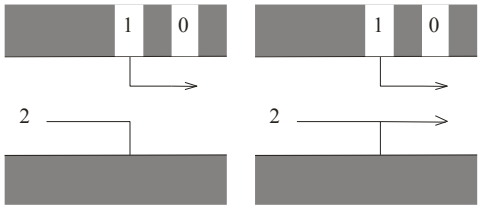
„*Greedy channel router*“ von Rivest und Fiduccia (3)



Annäherung an den nächsten Anschluss:
Dogleg nach oben, wenn der nächste Anschluss des Netzes sich oben befindet und in den nächsten k Spalten (k: Systemkonstante) kein Anschluss unten existiert.
Entsprechend für unten.

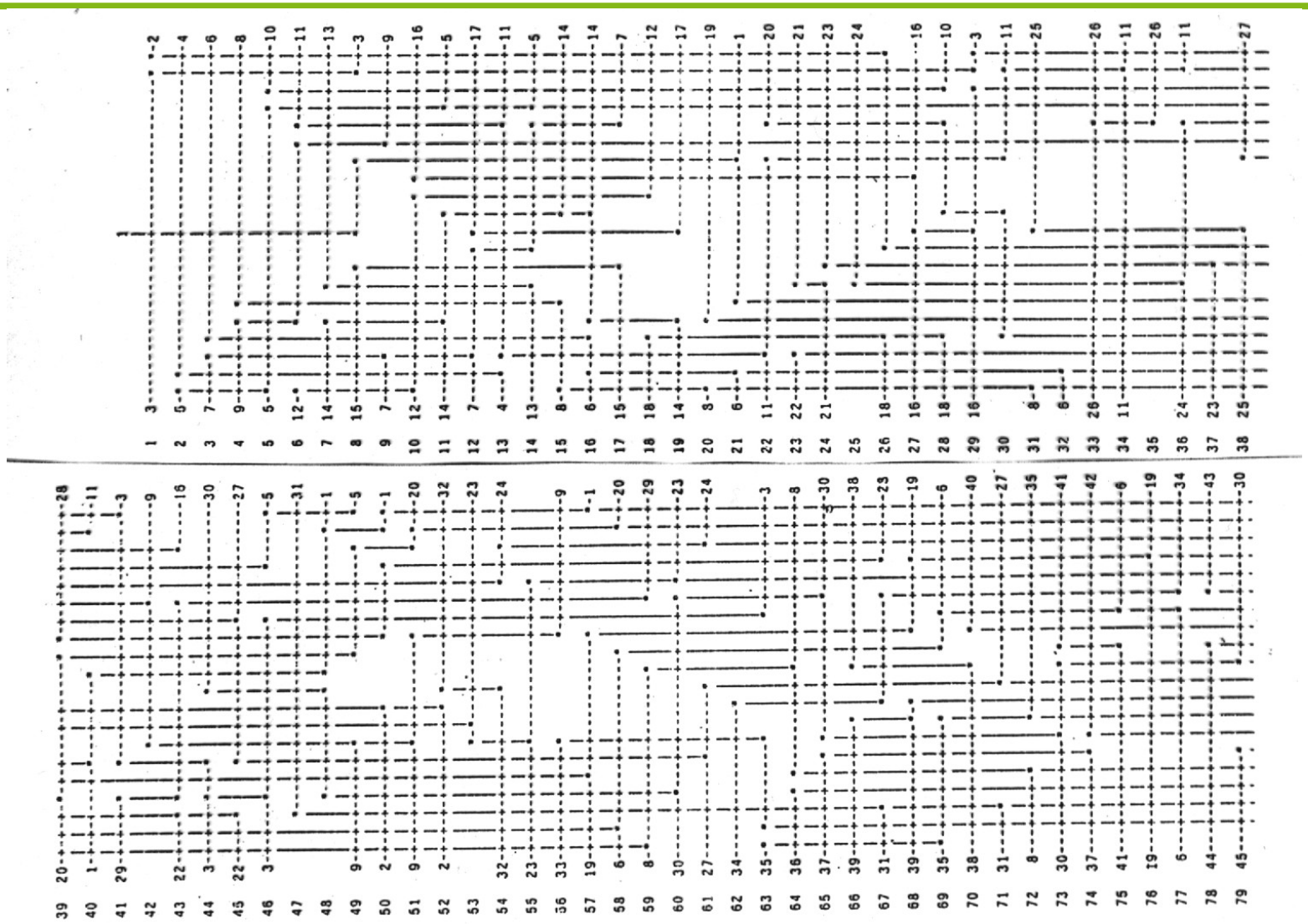


Kanalverbreiterung:
Verbreitere den Kanal, falls Top oder Bottom-Anschluss nicht möglich war.

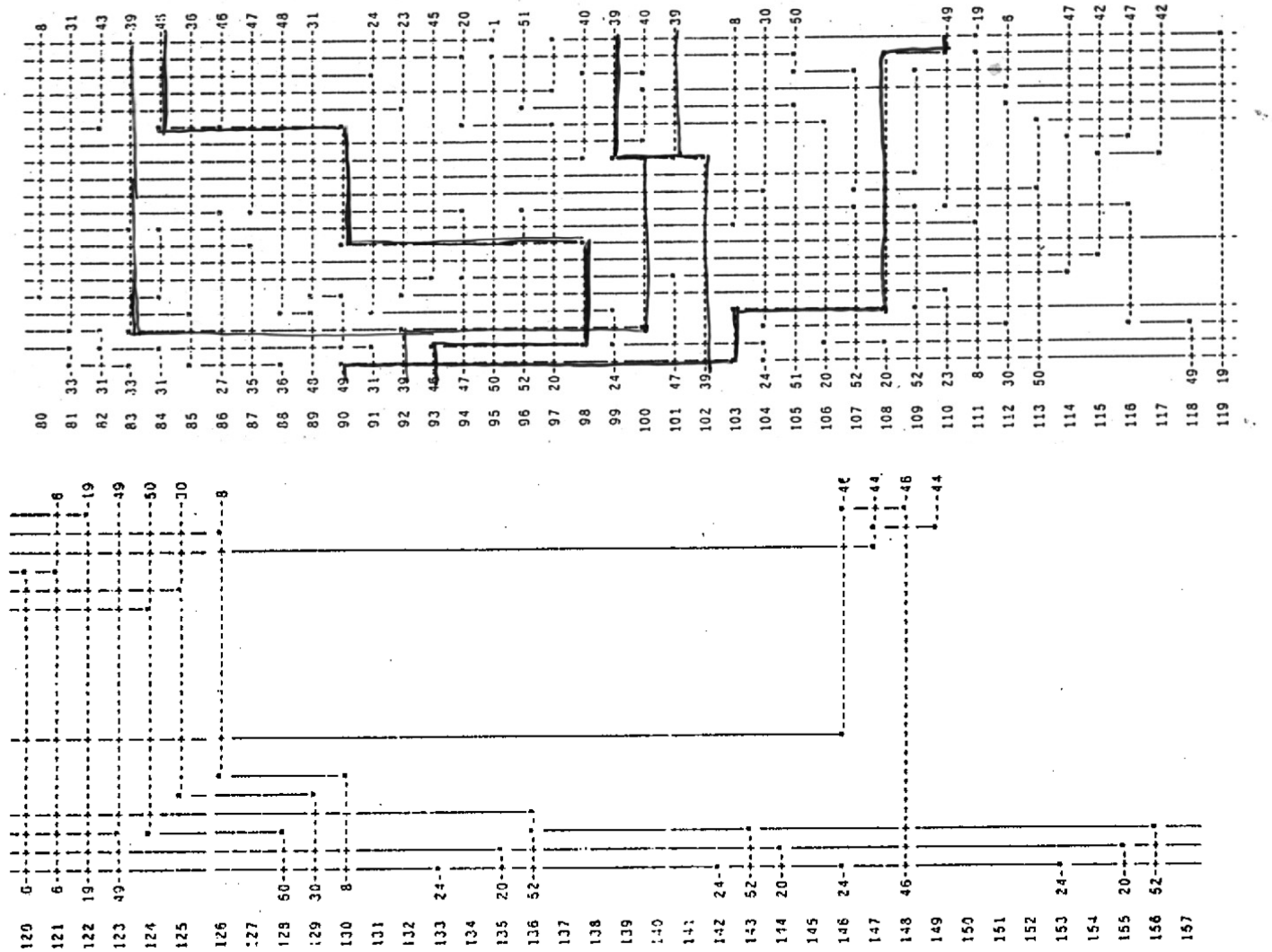


Erzeuge horizontale Segmente für den Übergang zur nächsten Spalte;
i:=i+1;
until (i>rechter Kanalrand) und
es existieren keine geteilten Netze

Anwendungs-Beispiel: das schwierige Beispiel von Deutsch

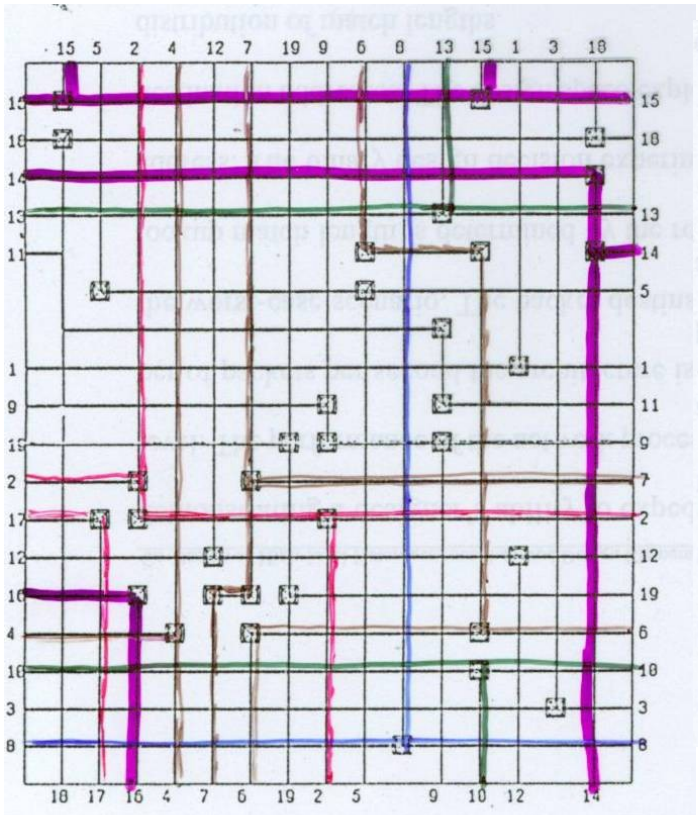


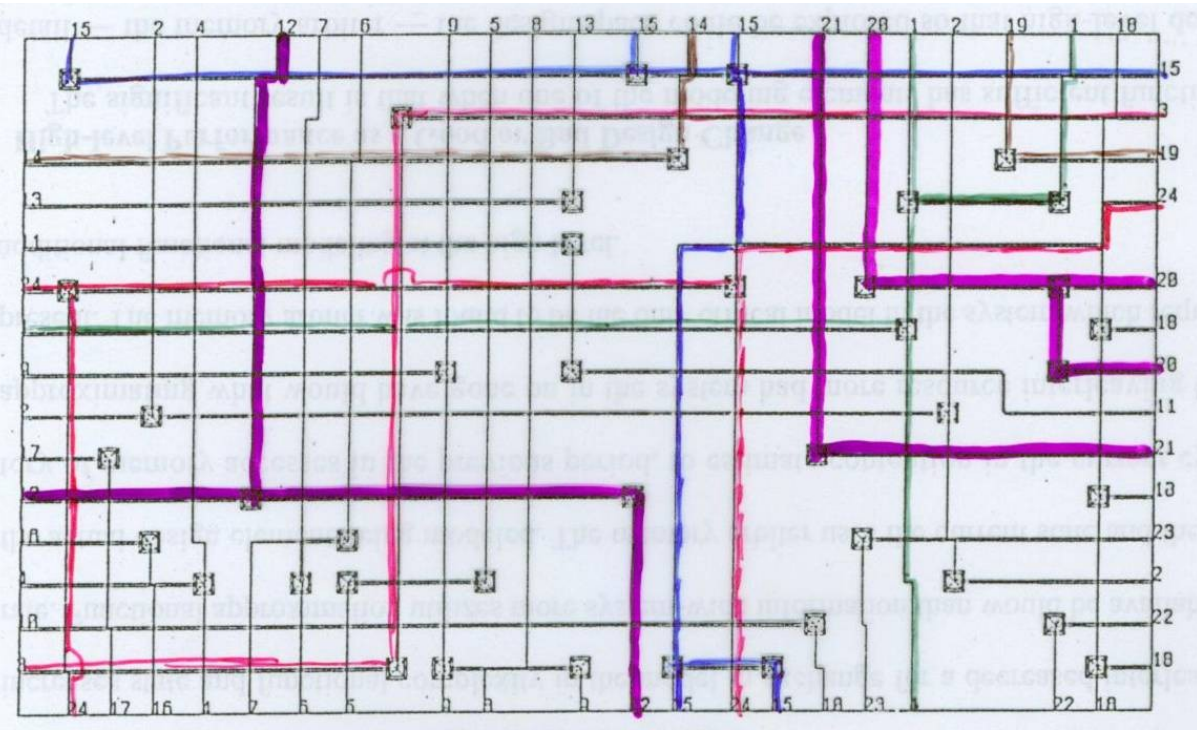
Anwendungs-Beispiel: das schwierige Beispiel von Deutsch (2)



Switchbox-Routing

An allen 4 Seiten Anschlüsse. Eine Kanalverbreiterung ist dann nicht mehr möglich.
Beispiele:



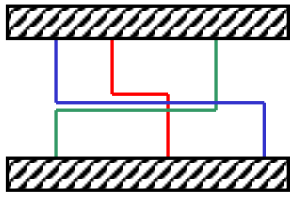


3-Lagen-Verdrahtung

1. Nicht reservierte Ebenen

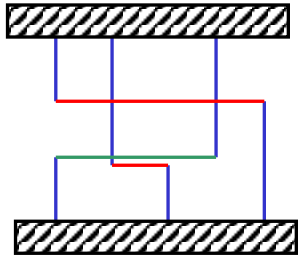
Beispiel

1 Spur



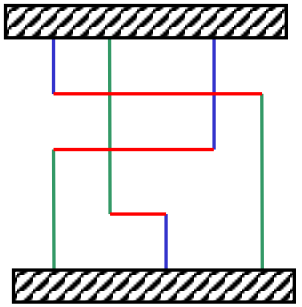
1. reservierte Ebenen
   a) HVH: geringe Kanalbreite

2 Spuren



a) VHV: kein Problem mit vertikalen Segmenten

3 Spuren



Neuere Router

- YACR2:yet another channel router, Berkeley
- Hierarchical channel router

Zusammenfassung

- Sortierung von Verdrahtungsregionen
- *River routing*
- 2-Lagen Kanalverdrahtung
  - HCG
  - Intervallgraphen, left edge-Algorithmus
  - VCG
  - *Dogleg channel routing*: Deutsch; Rivest&Fiduccia
- Switchbox-Routing
- 3-Lagen-Verdrahtung