Platzierung

- Algorithmen von Breuer & Lauther; SA & GA-Verfahren -

Peter Marwedel
Universität Dortmund, Informatik 12

Grundidee von Mincut-Algorithmen

Platzierung durch mehrfache Zerlegung der Netzliste.

Verfahren heissen **Min–Cut–Algorithmen**.

Sei C eine Menge von Schnittlinien durch die Platzierungsfläche.

Sei $v(c)$ für $c \in C$ eine Funktion, die einer Schnittlinie die Zahl der geschnittenen Netze zuordnet. Ideales Min–Cut–Verfahren würde das Maximum der Zahl der geschnittenen Netze minimieren:

$$\max_{c \in C}(v(c)) \rightarrow \text{min}$$

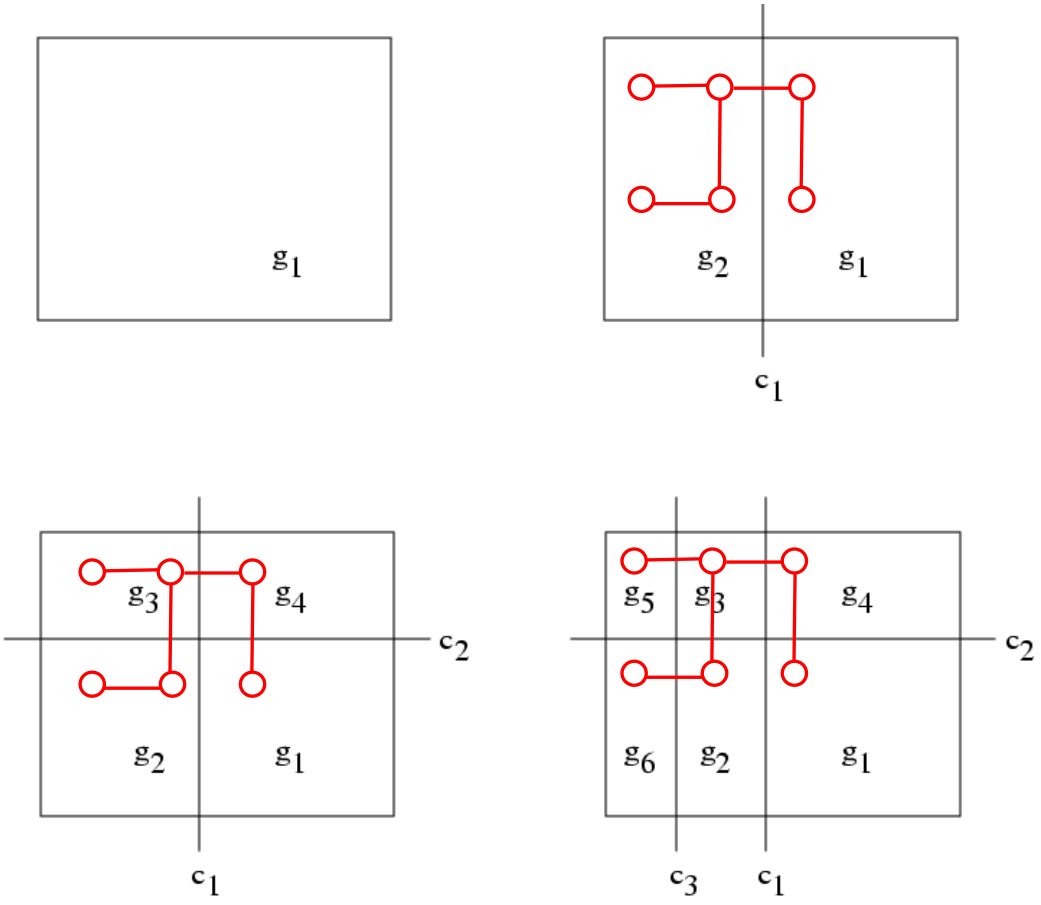
Zu komplex. Statt dessen:

1. Minimierung im 1. Schnitt geschnittener Netze.
2. Minimierung im 2. Schnitt geschnittener Netze.

usw.

Bis alle Zellen genau platziert sind oder bis ein anderes Abbruchkriterium erfüllt ist.

Beispiel für das Verfahren nach Breuer



Algorithmus von Breuer

Def.: Ein **Block** sei eine Menge von Zellen.

1. Der Block $g_j = g_1$ enthalte zunächst alle Zellen.
2. Wähle Reihenfolge der Schnittlinien-Bearbeitung.
3. Wähle die nächste Schnittlinie $c_i \in C$.
4. Platziere die Zellen aus g_j auf beiden Seiten der Schnittlinie c_i so, dass $v(c_i)$ unter Einhaltung eines Balancekriteriums minimal wird. Bilde aus den Zellen jeder Seite je einen neuen Block.
5. Für jeden neuen Block g_j wiederhole das Verfahren ab 3. solange das gewählte Abbruchkriterium nicht erfüllt ist.

Rekursiv immer genauere Platzierung.

Teilprobleme

a) Partitionierung eines Graphen in zwei Teilgraphen mit minimalem Schnitt (s.o.).

b) Selektion der Schnittlinien. Nach Breuer:

  b1) abwechselnd horizontale/vertikale Schnitte;

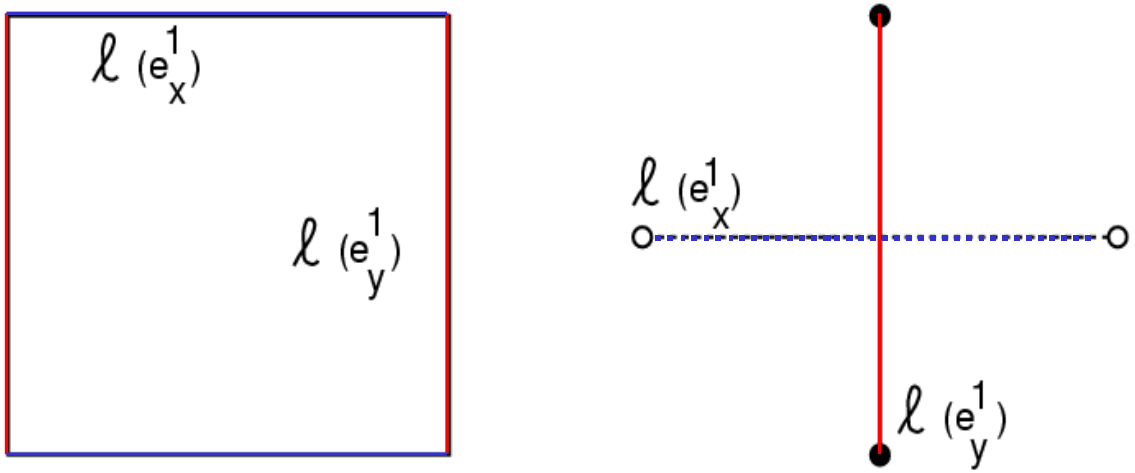
  b2) zunächst "dünne" Scheiben in einer Richtung, dann nach Art der binären Suche in der dazu senkrechten Richtung.

Immer Schnitt durch gesamte Länge.

$\longrightarrow$ ungeeignet, Zellen unterschiedlicher Größe zu platzieren.

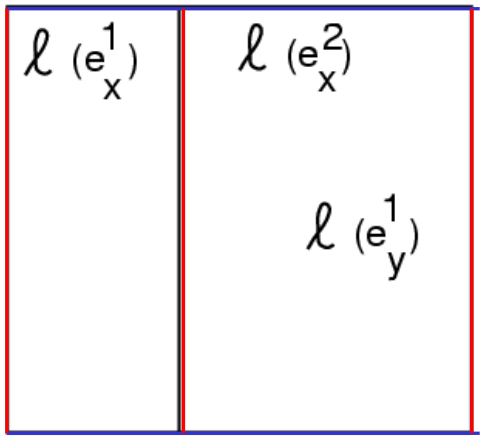
Min-Cut-Verfahren für beliebige Zellen

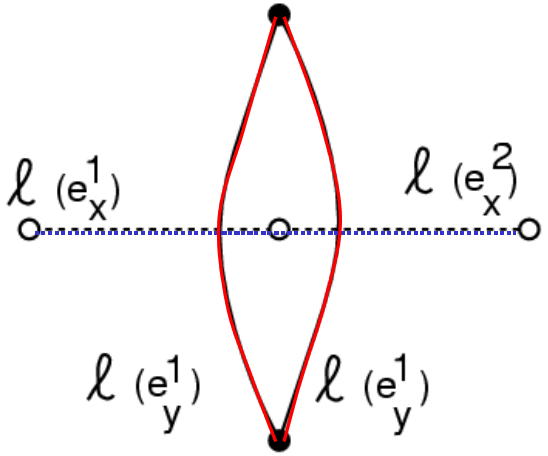
Grundlegendes Verfahren zur Min–Cut–Platzierung beliebiger Zellen stammt von Lauther: eine Fläche wird durch je eine Kante zweier einander zugeordneter Graphen dargestellt. Die Kanten e_x^i eines der Graphen enthalten die Abmessungen in x–Richtung, die Kanten e_y^i des anderen die der y–Richtung. Die Zuordnung der Kanten beider Graphen zueinander dadurch ausgedrückt, dass sich die beiden Kanten schneiden. Einer der Graphen mit gestrichelten, der andere mit einer durchgezogenen Kante dargestellt:



Darstellung von *n* Flächen

Zur Darstellung von n Flächen werden $2 * n$ Kanten benötigt. Die relative Lage der Flächen zueinander wird durch eine partielle Ordnung der Kanten ausgedrückt. Die Flächen 1 und 2 liegen neben**einander**, folglich sind auch die Kanten e_y^1 und e_y^2 entsprechend geordnet:



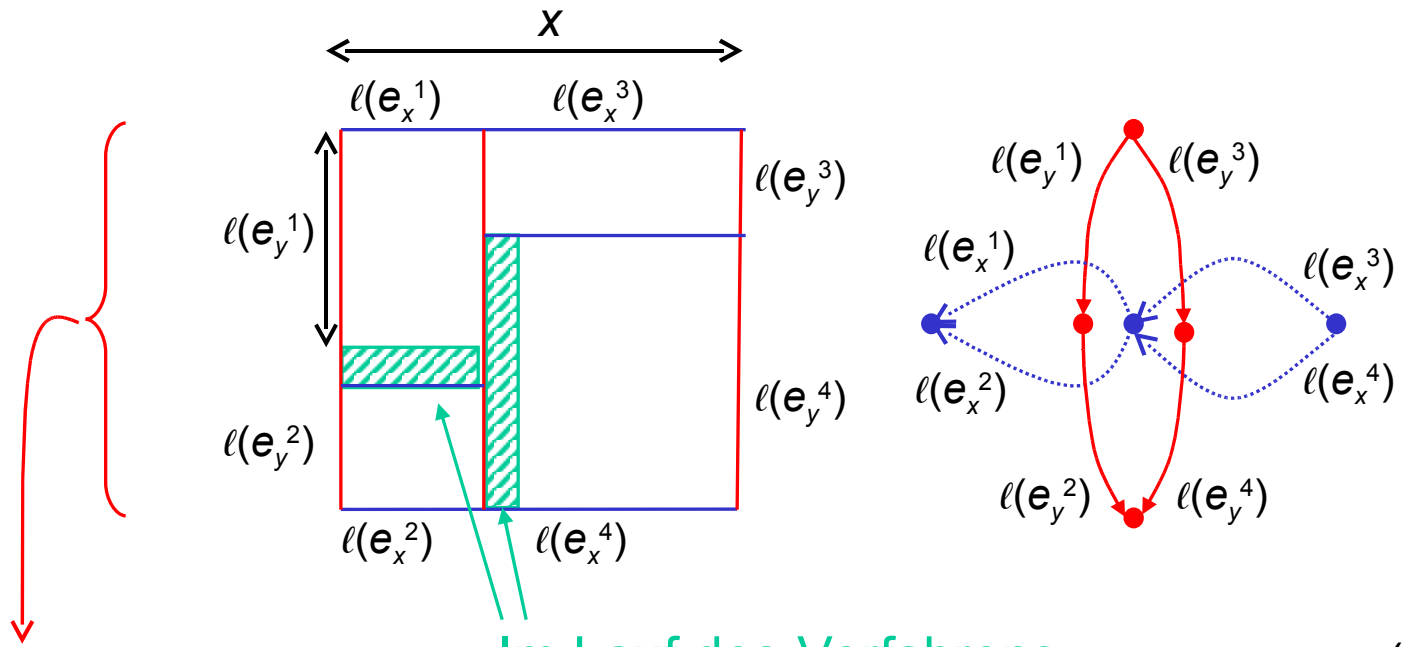


Entstehende Graphen enthalten stets je zwei ausgezeichnete Knoten, welche die vier Begrenzungslinien darstellen. Diese Graphen heißen **Polargraphen**.

Beziehung $\ell(e_y^1) = \ell(e_y^2)$ hier per Konstruktion; kann später entfallen.

Flächenberechung mittels Polargraphen

Ziel der Verwendung von Polargraphen ist die rasche Berechung der Abmessungen des umgebenden Rechtecks:



y=max((ℓ(e_y^1)+ ℓ(e_y^2)), (ℓ(e_y^3)+ ℓ(e_y^4)))

Im Lauf des Verfahrens möglicherweise entstehende Leerflächen

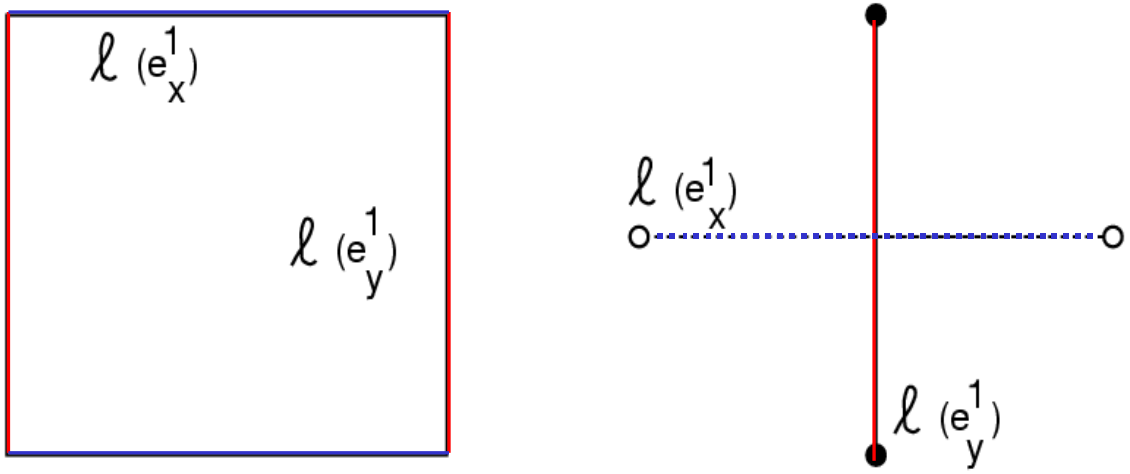
x= max(ℓ(e_x^1),ℓ(e_x^2))+ max(ℓ(e_x^3),ℓ(e_x^4))

Algorithmus von Lauther (1)

1. Die insgesamt benötigte Fläche wird zunächst durch die Summe der Flächen der Zellen angenähert. Es wird eine quadratische Fläche angenommen:

$$\ell(e_x^1) = \ell(e_y^1) = \sqrt{(\text{Summe der Zellflächen})}$$

Initialer Graph:



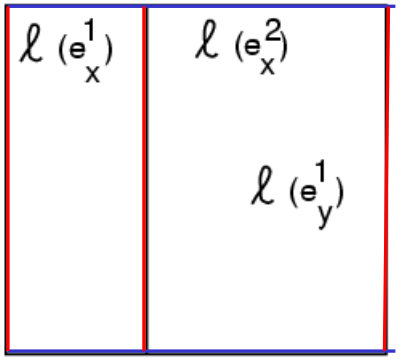
Algorithmus von Lauther (2)

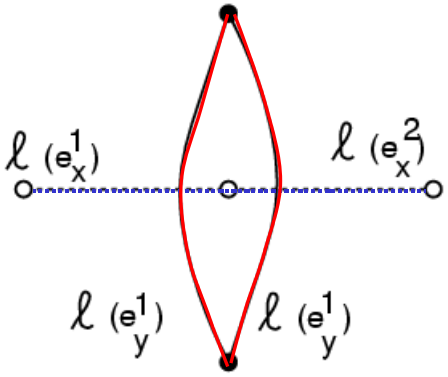
2. Netzgraph wird mit Bisektion geschnitten. Balancekriterium beachten! Platzierungsfläche so geschnitten, dass die Teilflächen so groß sind wie die Summe der Flächen der darin enthaltenen Zellen. Bei vertikalem Schnitt:

$$\ell(e_x^1) = (\text{Summe der Zellflächen aus Teilgraph 1})/\ell(e_y^1)$$

$$\ell(e_x^2) = (\text{Summe der Zellflächen aus Teilgraph 2})/\ell(e_y^2)$$

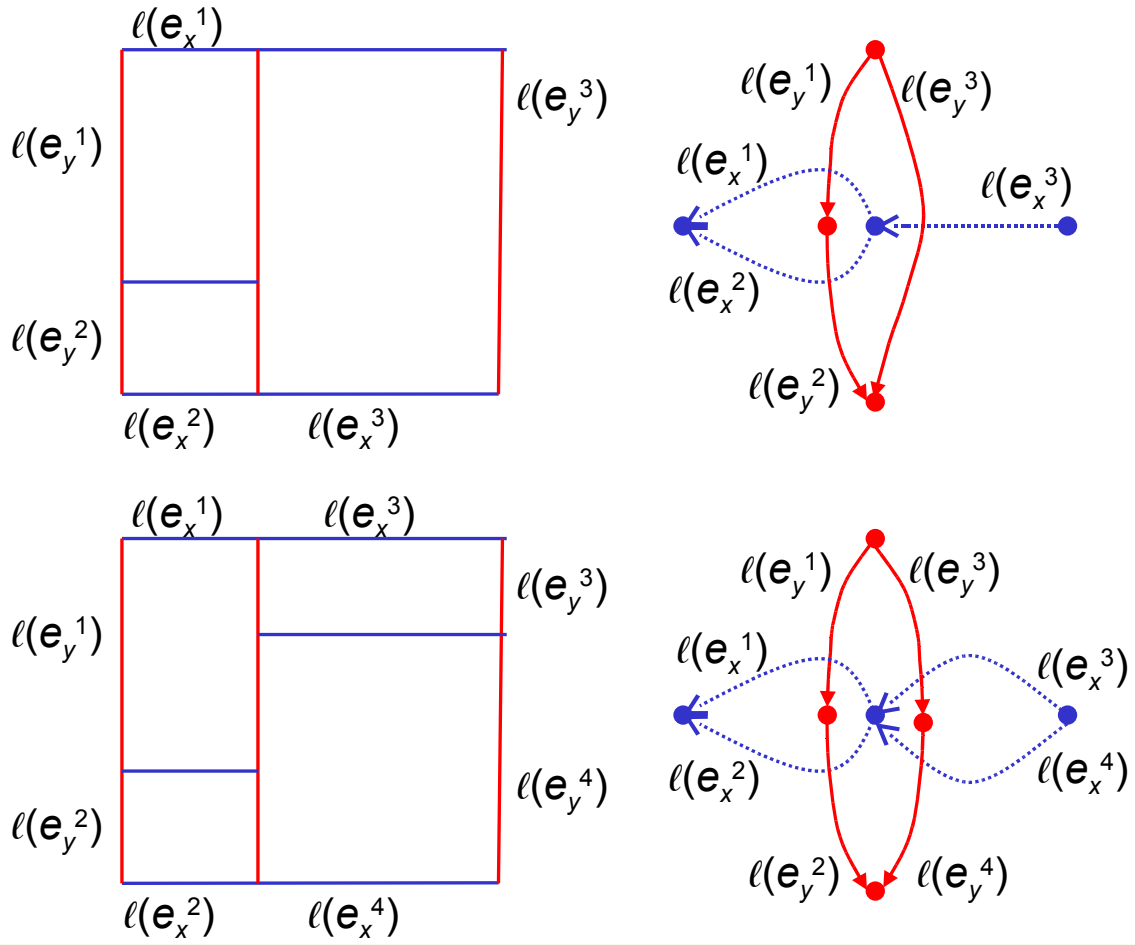
Daraus ergibt sich die Aufteilung:



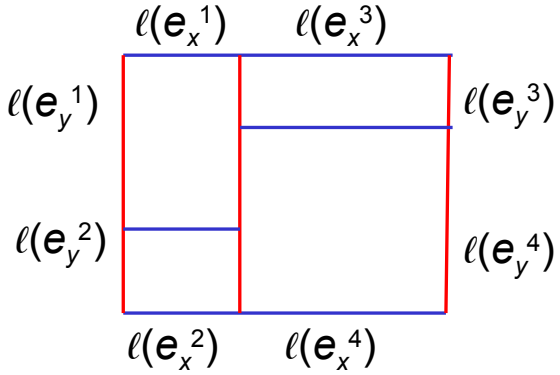


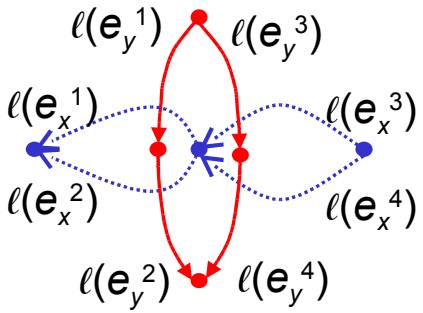
Algorithmus von Lauther (3)

3. Für jeden der resultierenden Teilgraphen wird der Schritt 2 wiederholt, und zwar abwechselnd mit horizontalem und vertikalem Schnitt.

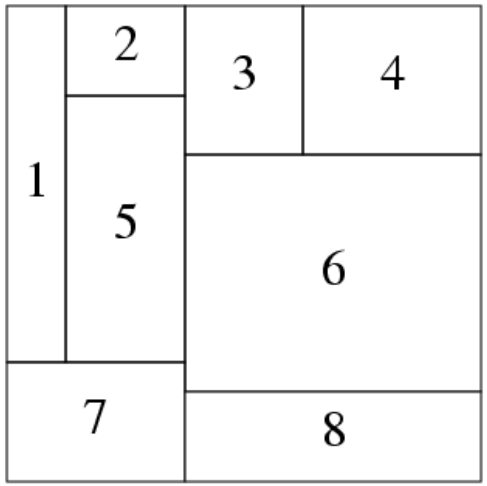


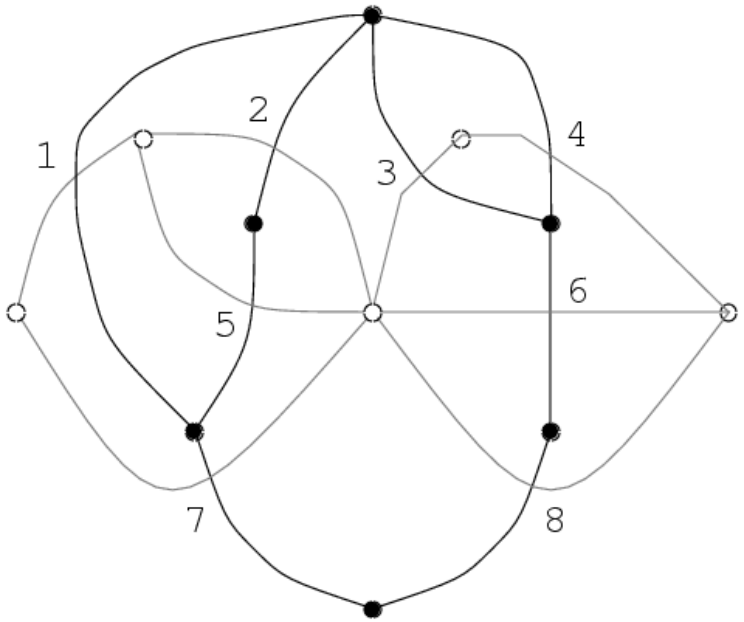
Algorithmus von Lauther (4)





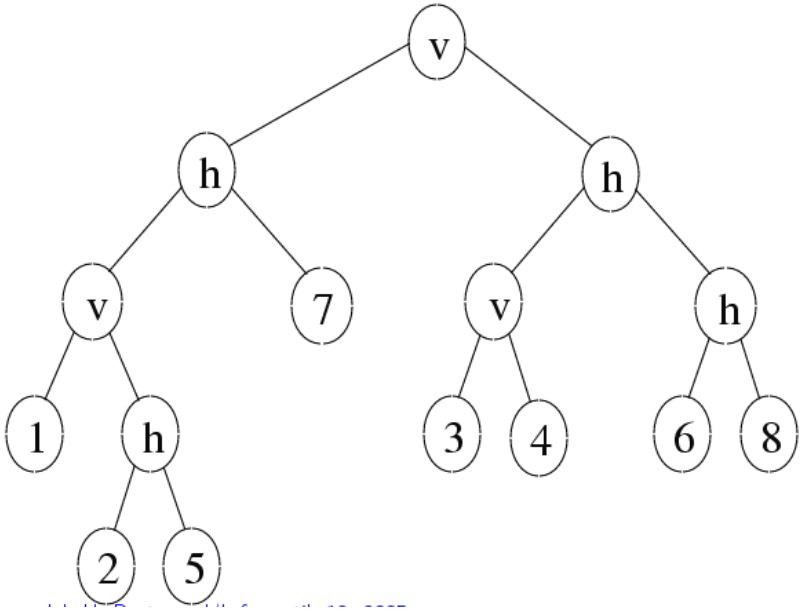
Nach mehrfachem Schneiden:

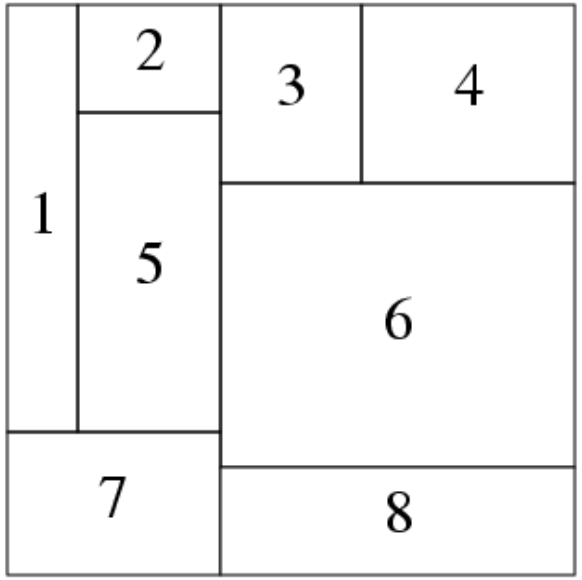




Einschränkung auf *Slicing trees*

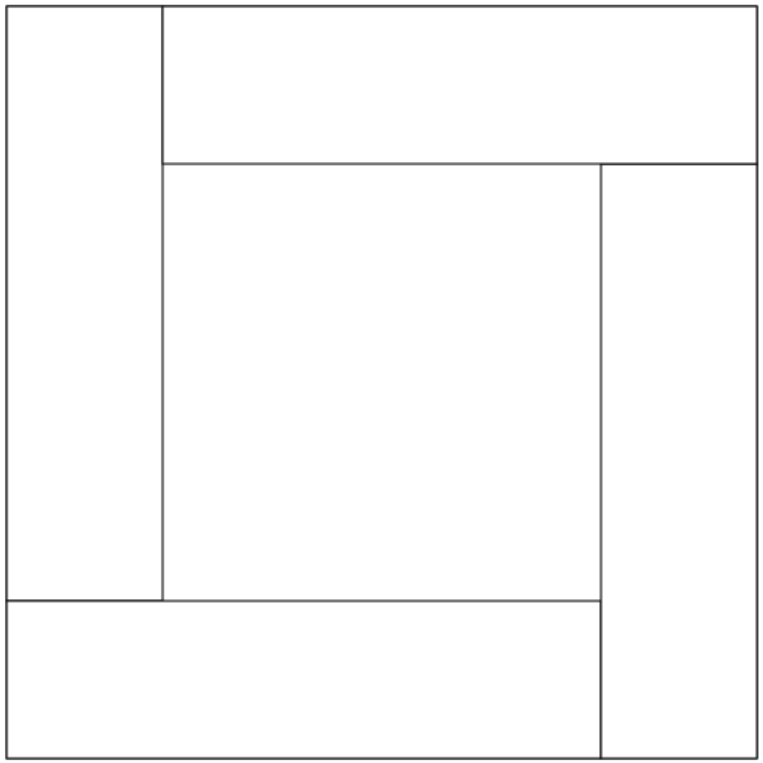
Das Min–Cut–Verfahren liefert nur Lösungen, die durch fortgesetzte Flächenaufteilungen entstehen. Solche Flächenaufteilungen kann man auch durch sog. *slicing–trees* darstellen. Die Knoten enthalten dabei jeweils die Information, ob vertikal oder horizontal zu schneiden ist. Ferner kann man verabreden, dass der linke Teilbaum jeweils die linke bzw. die obere Fläche beschreibt. Slicing–tree des vorherigen Beispiels:





Pin Wheels

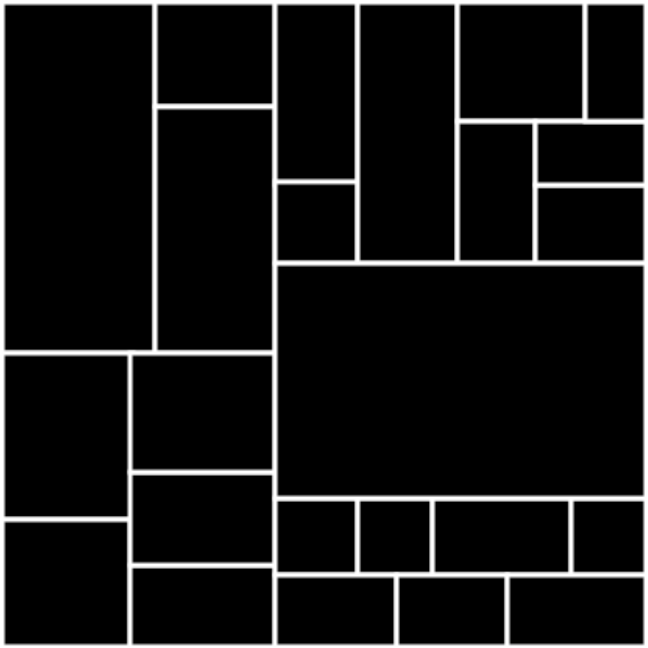
Slicing–trees sind keine allgemeine Darstellung von Flächenaufteilungen, da sie sog. *pin–wheels* nicht beschreiben können.



Orientierung (1)

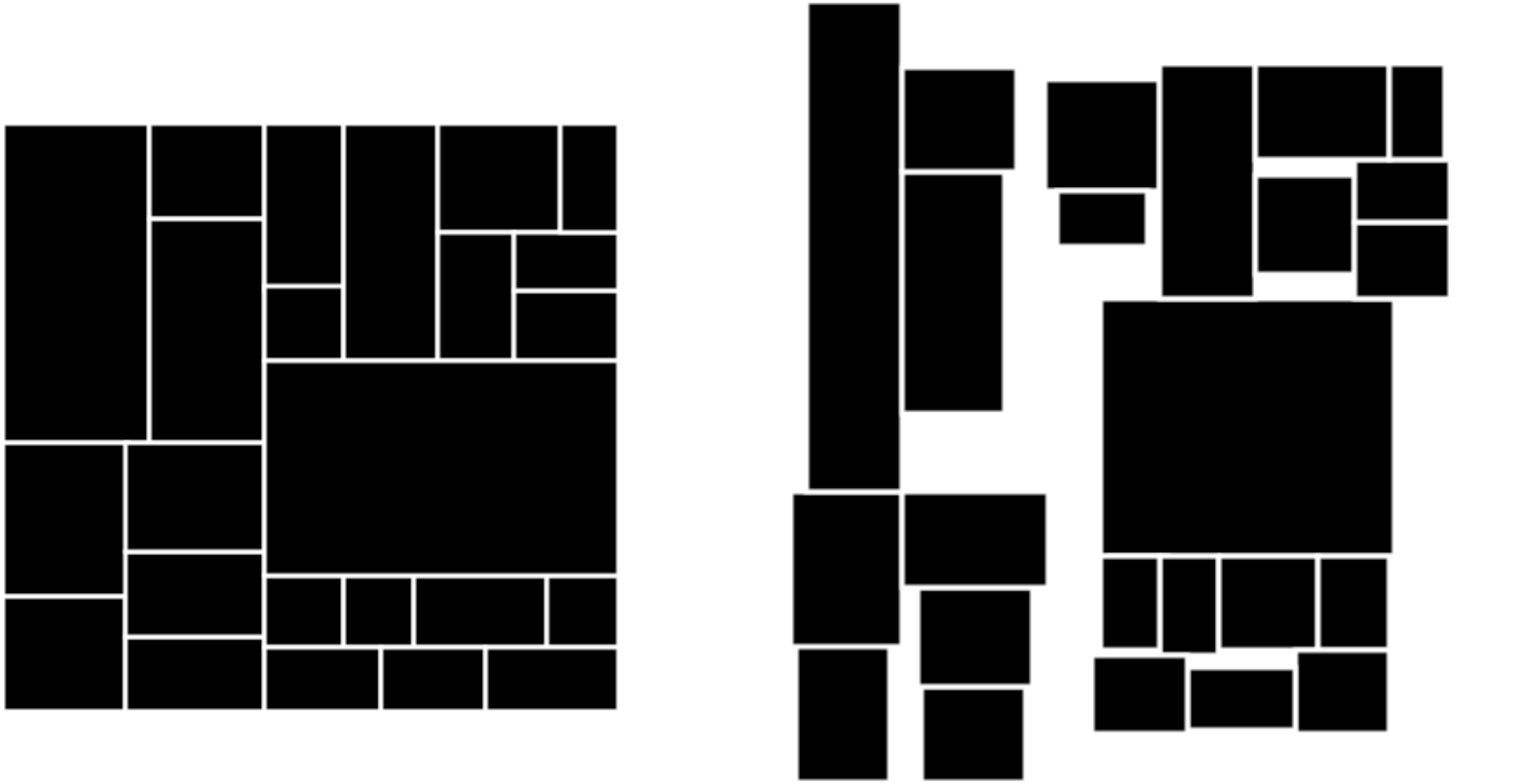
Die erzeugten Teilflächen besitzen aufgrund der Konstruktion genau die Fläche der zu platzierenden Zellen. Das Breiten/Längenverhältnis (engl. *aspect ratio*) wird jedoch noch nicht richtig wiedergegeben.

Die Zellen müssen jetzt die richtigen Längen-/Breiten-verhältnisse annehmen; anschließend Orientierung ….



Orientierung (2)

… die längste Seite der Zellen wird parallel zur längsten Seite der ihr zugeteilten Teilfläche gelegt. ☞ Leerflächen:

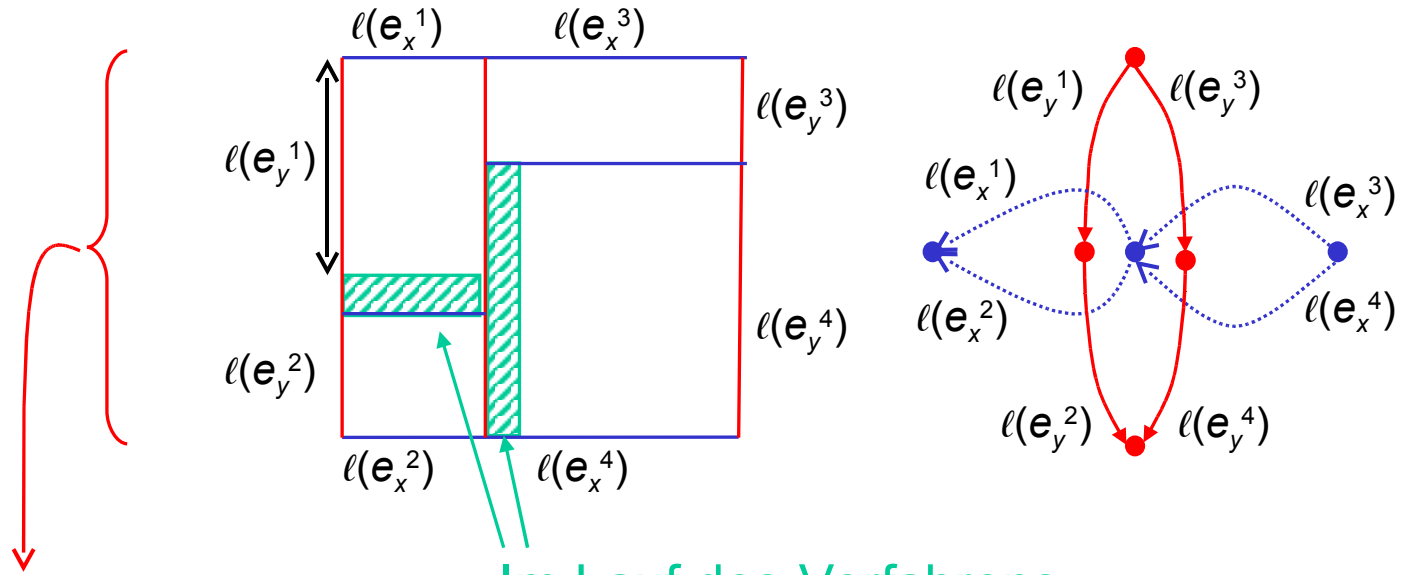


a) Initiales Placement

b) Berücksichtigung der Zellabmessungen

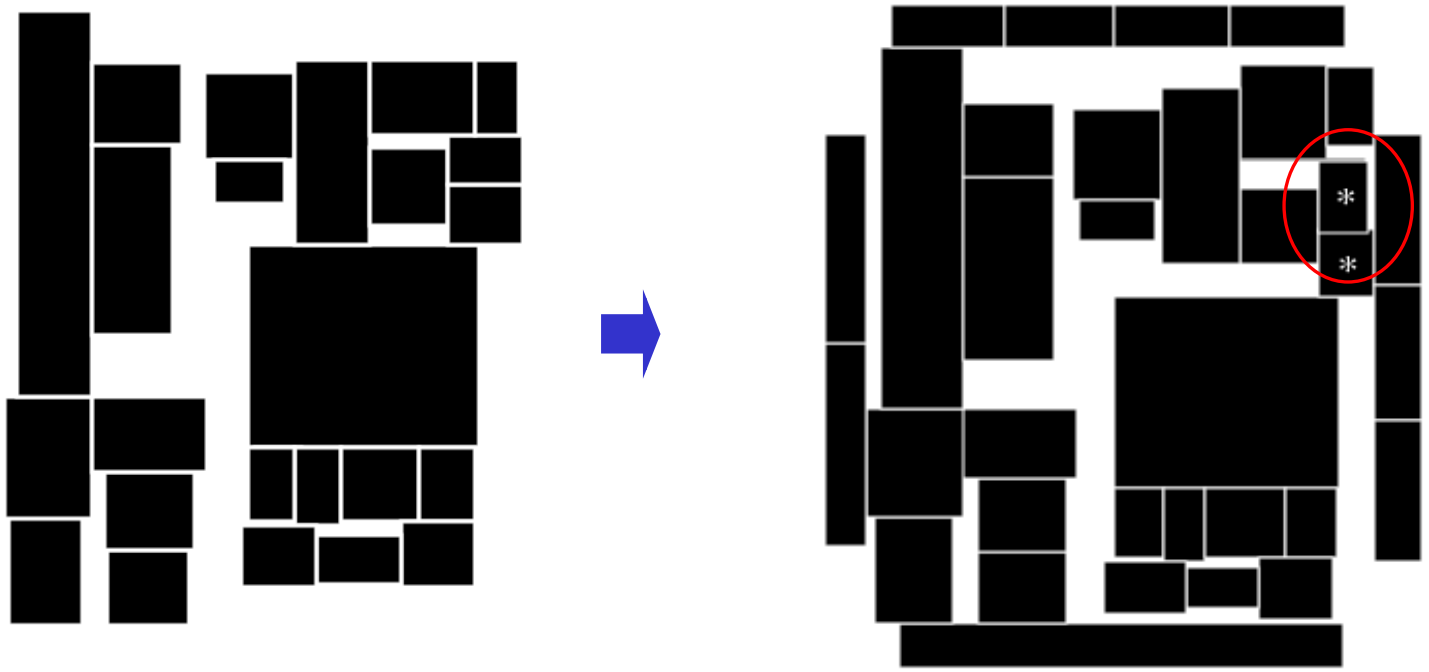
Nutzung der Polargraphen

Die absolute Lage kann jetzt einfach mittels der Polargraphen berechnet werden: die Koordinaten der linken unteren Ecke einer Zelle i ergeben sich als Länge des längsten Weges vom linken bzw. unteren Polknoten bis zur Kante e_x^i bzw. e_y^i.



Verbesserungen: Rotation

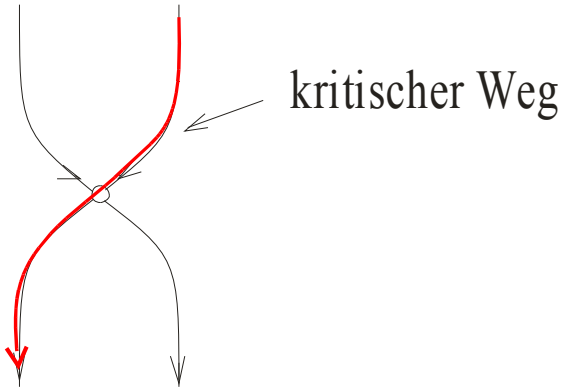
Vertauschung von $\ell(e_x^i)$ und $\ell(e_y^i)$ (**Rotation**) kann zu einer Flächenreduktion führen. Beispiel:



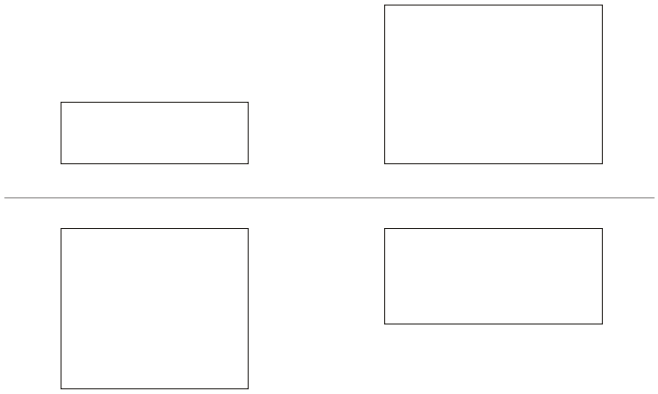
*: Gedrehte Zellen; dadurch horizontale Verkürzung.
Zusätzlich E/A-Zellen eingezeichnet.
Flächeneinsparung wegen vorheriger Orientierung meist gering.

Verbesserungen: *Squeezing* (1)

Ausgangssituation



a) Polargraph

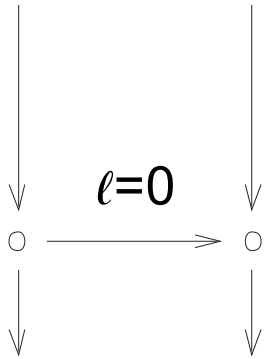


b) Flächenaufteilung

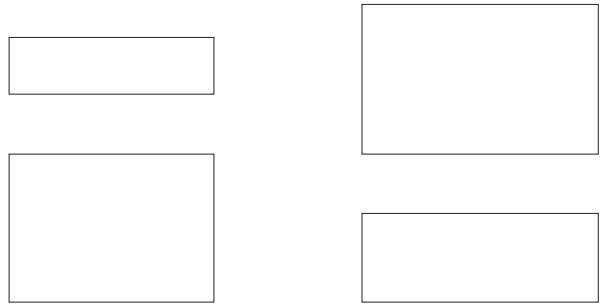
a) zeigt einen kritischen Pfad durch einen Polargraphen. Dieser entsteht durch eine Schnittlinie in der Flächenaufteilung (siehe b)).

Verbesserungen: *Squeezing* (2)

Die Schnittlinie ist für das weitere Verfahren unerheblich, die Zellen können verschoben werden.
Im Polargraphen kann die resultierende neue Länge einfach durch Einführung einer Kante der Länge 0 bestimmt werden:



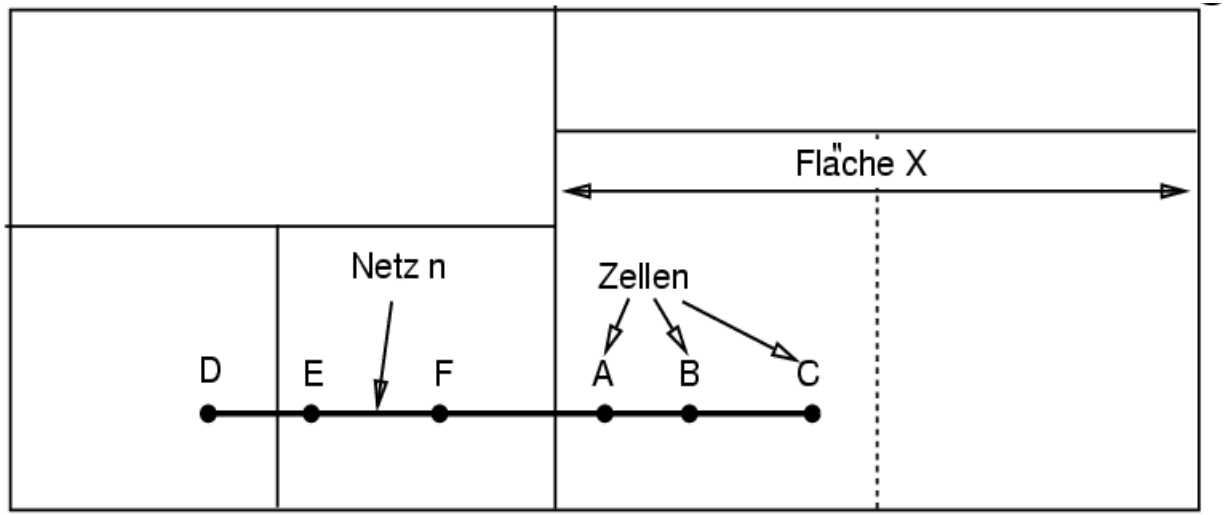
a) Polargraph



b) Flächenaufteilung

Berücksichtigung von Netzanschlüssen außerhalb der gegenwärtigen Fläche (1)

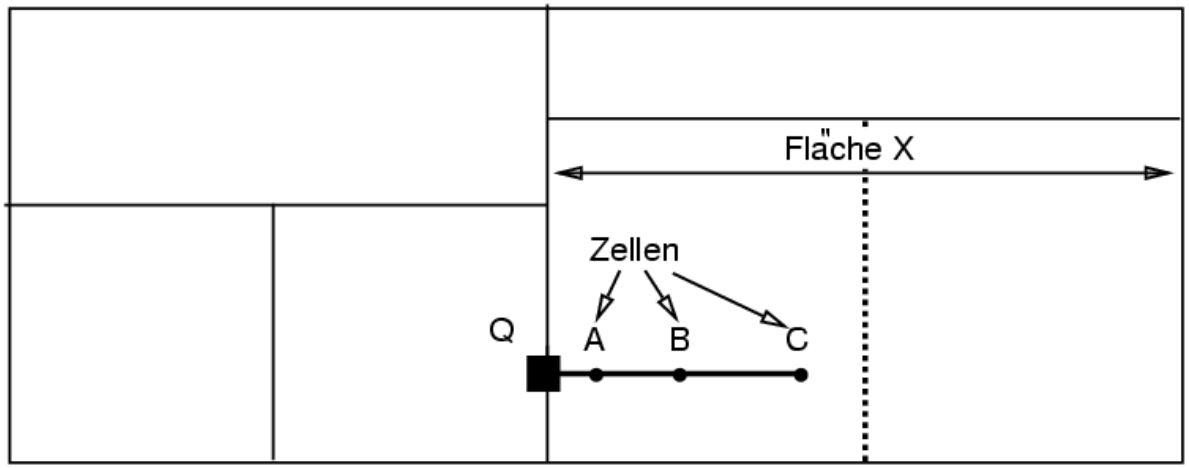
Grundproblem der bislang vorgestellten fortgesetzten Bipartitionierung: Netzanschlüsse außerhalb der zu teilenden Fläche nicht berücksichtigt:



Zu partitionieren: Fläche X. Netz n hat, Anschlüsse an Zellen A, B, C innerhalb und an Zellen D, E, F außerhalb von X. Beachtet man lediglich die in X enthaltenen Zellen und die Schnitte von Netzen an der gestrichelten Linie, bietet es keinen Kostenvorteil, die Zellen A, B und C in der linken Teilfläche zu platzieren.

Berücksichtigung von Netzanschlüssen außerhalb der gegenwärtigen Fläche (2)

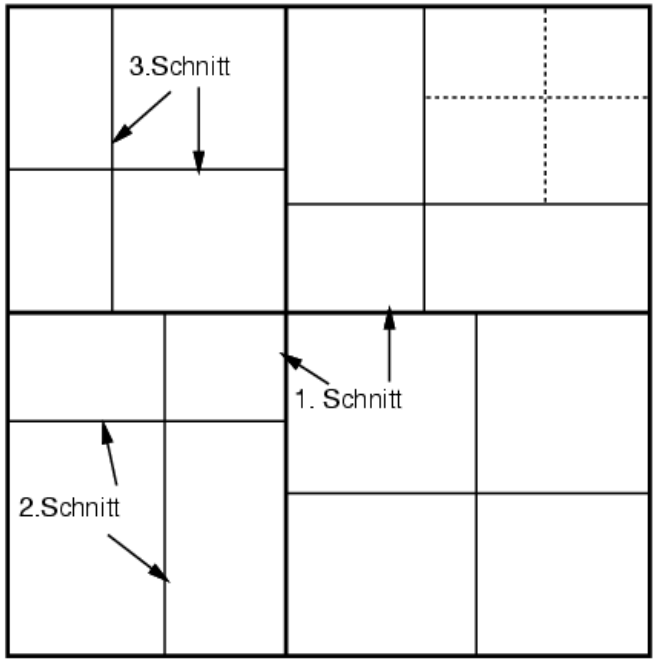
Um den Effekt der äußeren Anschlüsse zu berücksichtigen, ersetzen Dunlop und Kernighan die Anschlüsse des Netzes `n` außerhalb von `X` durch eine "Dummy-Zelle" `Q` am linken Rand von `X`



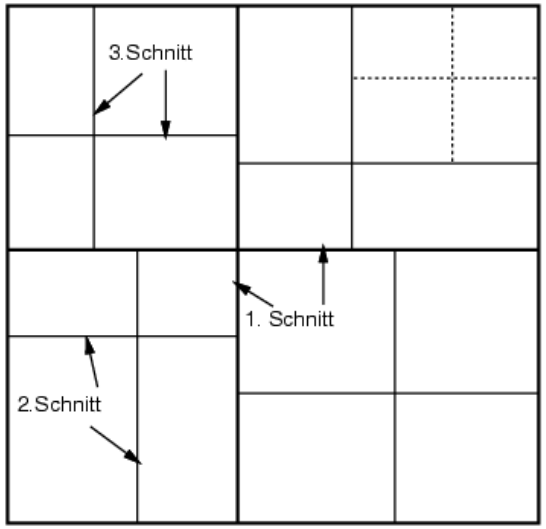
Technik kann bei jeder Partitionierung benutzt werden, insbesondere auch für die externen Anschlüsse eines Chips.

Quadripartitionierung (1)

Bei der Bipartitionierung bleiben die Entscheidungen relativ lokal. Die echte spätere Entfernung der Zellen wird nicht berücksichtigt. Als Verbesserung haben Suaris und Kedem die Aufteilung der Zellen in vier Teilmengen und ihre Zuordnung zu vier Quadranten vorgeschlagen:

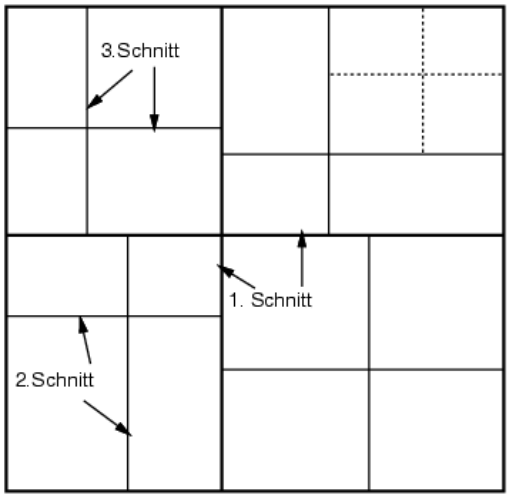


Quadripartitionierung (2)



Zelle A kann in einem bestimmten Quadranten mit einer Zelle B aus einem anderen Quadranten vertauscht werden. Gewählt wird das Paar (A, B), dessen Vertauschung den größten Gewinn erbringt. Für das Vertauschen von Zellen zwischen allen vier Quadranten müssen separate Gewinn–Arrays geführt werden. Zur Auswahl von 2 aus 4 Quadranten gibt es 6 Möglichkeiten. Statt zweier Arrays bei der Bipartitionierung benötigt man daher $2 * 6 = 12$ Arrays.

Quadripartitionierung (3)



Höhere Kosten für diagonal gegenüberliegende Quadranten möglich.

Quadripartitionierung erlaubt Unterscheidung zwischen diagonal und nebeneinander liegenden Zellen, kann aber den endgültigen Abstand zwischen Zellen nach fortgesetztem Partitionieren nicht berücksichtigen.

→ andere Optimierungsmethoden.

Simulated Annealing

- General method for solving combinatorial optimization problems.
- Based the model of slowly cooling crystal liquids.
- Some configuration is subject to changes.
- Special property of Simulated annealing: Changes leading to a poorer configuration (with respect to some cost function) are accepted with a certain probability.
- This probability is controlled by a temperature parameter: the probability is smaller for smaller temperatures.

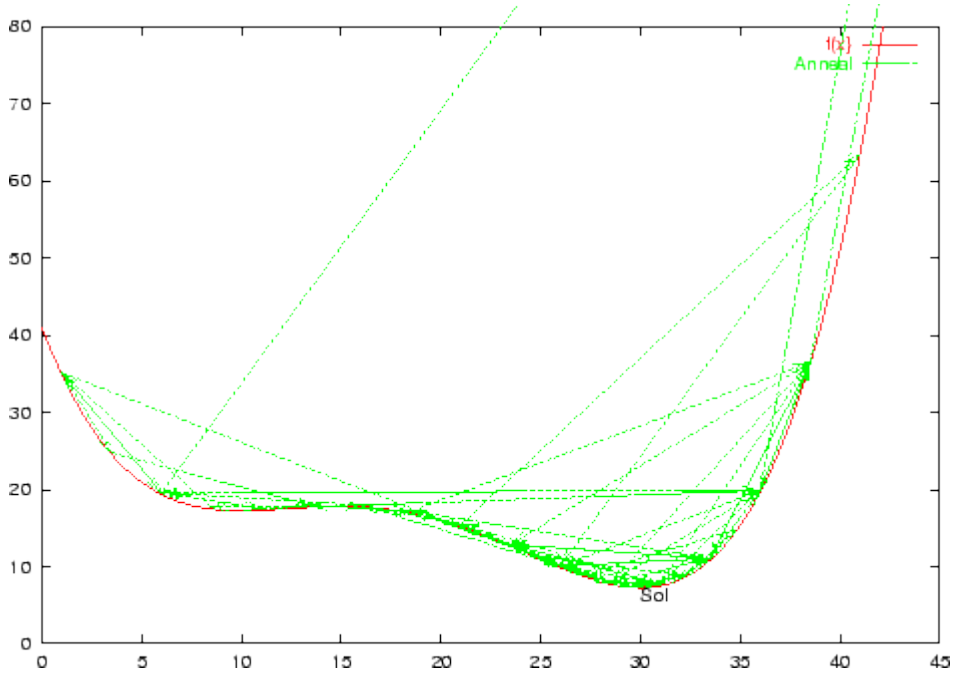
Simulated Annealing Algorithm

```
procedure SimulatedAnnealing;
var i, T: integer;
 begin
   i := 0; T := MaxT;
   configuration:= <some initial configuration>;
   while not terminate(i, T) do
    begin
     while InnerLoop do
       begin NewConfig := variation(configuration);
         delta := evaluation(NewConfig,configuration);
         if delta < 0
         then configuration := NewConfig;
         else if SmallEnough(delta, T, random(0,1))
           then configuration := Newconfiguration;
       end;
   T:= NewT(i,T); i:=i+1;
 end; end;
```

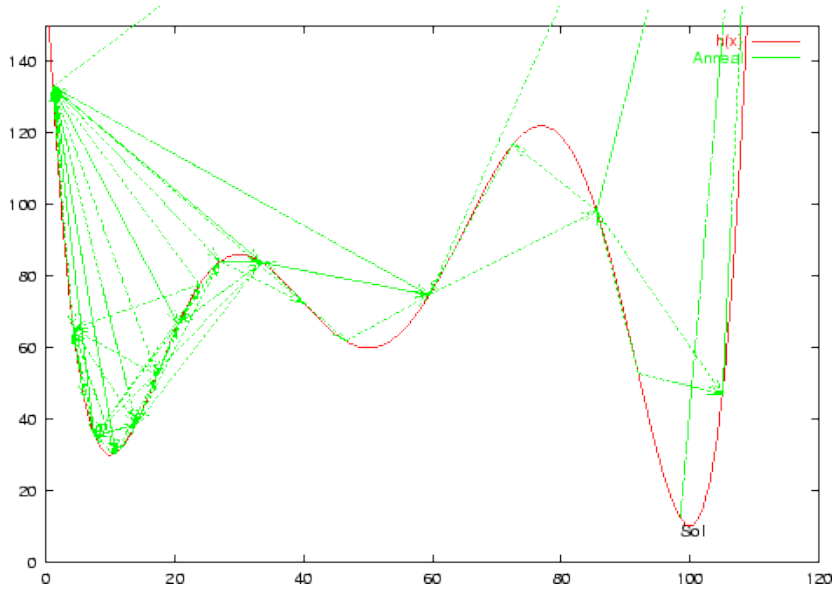
Explanation

- Initially, some random initial configuration is created.
- Current temperature is set to a large value.
- Outer loop:
  - Temperature is reduced for each iteration
  - Terminated if (temperature ≤ lower limit) or (number of iterations ≥ upper limit).
- Inner loop: For each iteration:
  - New configuration generated from current configuration
  - Accepted if (new cost ≤ cost of current configuration)
  - Accepted with temperature-dependent probability if (cost of new config. > cost of current configuration).

Behavior for actual functions



130 steps



200 steps

[people.equars.com/~marco/poli/phd/node57.html]

http://foghorn.cadlab.lafayette.edu/cadapplets/fp/fpIntro.html

TimberWolf

Anwendung von *simulated annealing* zur Platzierung und Verdrahtung.

- TimberWolf 3.2: 2 Methoden zur Erzeugung neuen Konfigurationen
  1. Zufällige Auswahl/zufällige Platzierung einer Zelle
  2. Zufällige Vertauschung zweier Zellen

  Entfernung größer bei hohen Temperaturen;
  Feste Zahl von Iterationen der inneren Schleife.
- Heute (2008) verfügbar:
  - TimberWolf 6.2 (public domain: opencircuitdesign.com/magic/download.html)
  - Kommerziell von itools (www.internetcad.com)

Genetische Algorithmen

- Genetische Algorithmen sind eine allgemeine Technik zur Lösung von kombinatorischen Optimierungsproblemen.
- Beginn mit einer Menge zulässiger Lösungen. Jede Lösung nennt man in diesem Zusammenhang ein **Individuum**.
- Die Menge der aktuellen Lösungen heißt **Population** .
- Jedes Individuum wird als String von **Symbolen** dargestellt. Die Symbole heißen **Gene** und der String heißt **Chromosom**.
- Alle Individuen einer Population werden anhand einer **Fitness-Funktion** beurteilt.
- Paare von leistungsfähigen Individuen werden als Eltern der nächsten Generation von Individuen ausgewählt.

Genetische Operatoren

Chromosomen der Kinder werden mittels dreier genetischer Operatoren aus den Chromosomen der Eltern abgeleitet. Diese sind:

- Die **Kreuzung** (engl. *cross-over)*: Aus den Chromosomen der Eltern wird jeweils ein Teil übernommen.
- Die **Mutation**: Im Chromosom des Kindes wird zufallsgesteuert ein Gen geändert.
- Die **Selektion**: Mittels einer Funktion wird aus den Kindern und der Elterngeneration die neue Generation ausgewählt.

Anwendung auf Platzierungsprobleme, Eigenschaften

Jedes einzelne Symbol repräsentiert i.d.R. die Platzierung einer Zelle (Zellname und Koordinaten).

Vorteile:

- in einer Population stets eine Menge guter Lösungen der Optimierungsaufgabe verfügbar ist. Es muss mit einer gewissen Wahrscheinlichkeit stets auch eine weniger gute Lösung akzeptiert werden. Beim SA-Verfahren ist dies dann die einzige gespeicherte Lösung. Bei genetischen Algorithmen bleiben weitere Lösungen erhalten.
- aus guten Lösungen können mittels komplexer Operatoren noch bessere Lösungen generiert werden. Genetische Algorithmen bilden daher eine viel versprechende Optimierungstechnik.

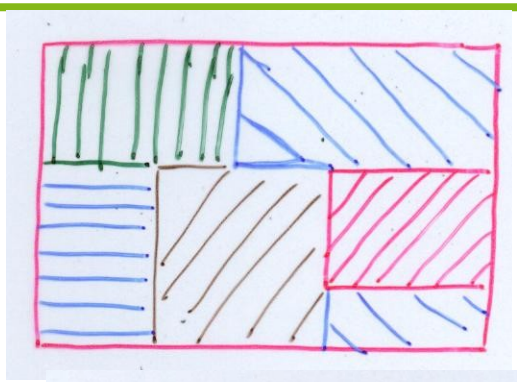
Benötigen mehr Speicherplatz als SA-Verfahren.

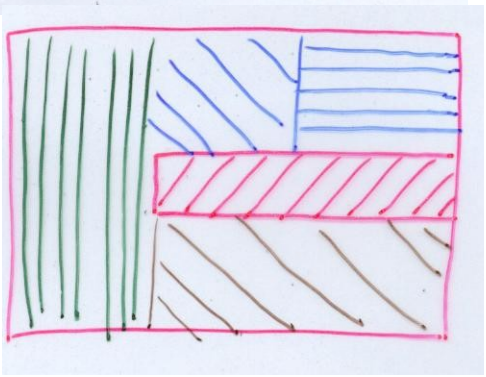
Chip-Planning

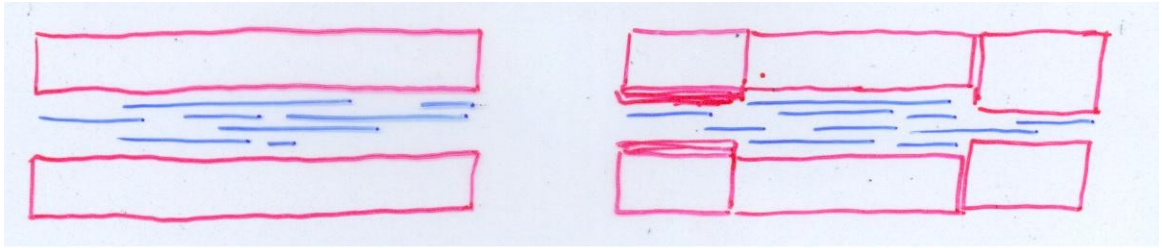
- Bisherige Verfahren: Längen-/Breitenverhältnisse der Zellen fest.
- Bei komplexen Zellen: i.d.R. eine Vielzahl von Alternativen. Günstig: Wahl einer Alternative anhand der Umgebung im Layout.

☞ ***Chip-Planning*- oder** *floor planning* **Algorithmen**

☞ Beispielsweise Abkehr von konstant breiten Kanälen







☞ Erheblich kompaktere Layouts (bis 50% Einsparung).

Zusammenfassung

- *Min-cut* Algorithmus von Breuer
- *Min-cut* Algorithmus von Lauther
- Erweiterungen
  - Netzanschlüsse außerhalb des Schnittbereichs
  - Quadripartitionierung
- *Simulated Annealing*
- Genetische Algorithmen (kurz)
- *Floor planning* (kurz)