Layout-Synthese: Platzierung

- Einführung; Kräftemodell, QAP, Partitionierung -

Peter Marwedel
TU Dortmund
Informatik 12

Layout-Synthese

Ziel: Erzeugung der geometrischen Strukturen mikroelektronischer Schaltungen, v.a. in Chips, aber auch auf Platinen

Literatur:

- Ohtsuki, 1986; Preas, 1988
- Th. Lengauer, 1990, Shahookar, 1991
- Marwedel, 1993; Brück, 1993
- Sherwani, 1999

Aus Komplexitätsgründen wird die Layouterzeugung in **Phasen** zerlegt:

- Platzierung
- Verdrahtung
  - Globale Verdrahtung
  - Lokale (detaillierte Verdrahtung)

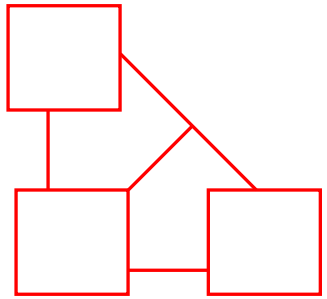
Platzierung: Einführung

Ziel: Anordnung von Bausteinen auf der verfügbaren Fläche (ICs, Platinen)

Vorgabe: **Netzliste** aus **Zellen, Anschlüssen, Netzen**

Beispielformat:

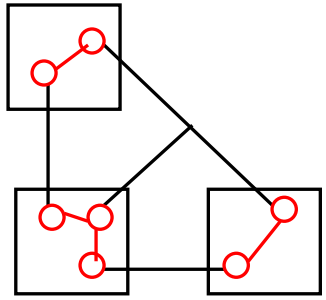
Netz, (*Zelle*, *Anschluss*) …(*Zelle*, *Anschluss*)
Netz, (*Zelle*, *Anschluss*) …(*Zelle*, *Anschluss*)
….



- **Zellen:** anderer Name für Bausteine.
- **Anschlüsse:** (engl. *pin*), bei VHDL: **Port**.
- **Netz:** leitende Verbindung zwischen Anschlüssen

Mathematisches Modell der Netzliste

- Graph?
  Nein, mehrere Kanten zwischen zwei Knoten möglich.
- Multigraph (mehrere Kanten zwischen 2 Knoten)?
  Nein, keine Unterscheidung zwischen Pins.
- Hypergraph (Knoten sind selbst wieder Graphen)?
  OK.



Kenntnis der Funktion der Zellen:

- Für Platzierung nicht erforderlich.
- Aus Namen direkt abgeleitet (z.B. `AND_gate`)
- Über Namen aus der Bibliothek abgeleitet.

Randbedingungen

Im Prinzip viele Randbedingungen

- kein Übersprechen zwischen Leitungen
- maximale Verlustleistung pro Fläche muss klein genug sein
- Verdrahtbarkeit zu gewährleisten.
- Taktrate muss erreicht werden
- Einhalten maximaler Stromdichte
- Unempfindlichkeit gegenüber Strahlung
- Symmetrieanforderungen (v.a. bei analogen Schaltungen),
- Widerstehen von *security*-Attacken
- …

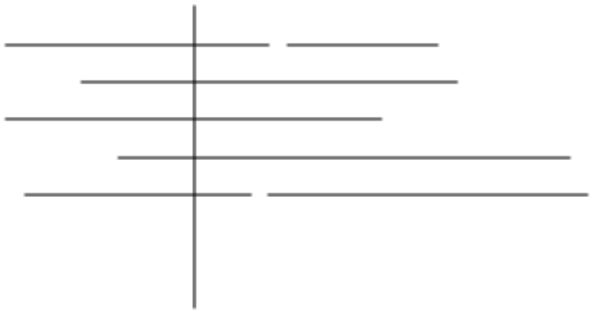
Zielfunktionen

Hoffnung, dass Zielfunktionen Randbedingungen berücksichtigen.

Praktisch anwendbar sind drei Zielfunktionen:

a) Gesamte Verdrahtungslänge $\rightarrow$ Min,

b) Maximum der von Schnittlinien durchschnittenen Netze $\rightarrow$ Min;



c) Maximale Dichte von Leitungen/Fläche $\rightarrow$ Min.

Überprüfung der Randbedigungen erforderlich.

Submicron-Technologie: zunehmend instabile Iterationen.

Platzierung nach dem Kräftemodell

Kräftemodell idealer Federn (Hookesches Gesetz):

Federn zwischen allen Zellen, die über Netze miteinander verbunden sind.

Kraft zwischen zwei Zellen i und j:

$$
\begin{aligned}
F_{i,j} &= -D * x_{i,j}, \text{ mit} \\
D &= \text{Anzahl der Netze zwischen } i \text{ und } j, \text{ sowie} \\
x_{i,j} &= \text{vorzeichenbehafteter Abstand zwischen } i \text{ und } j \\
&\quad \text{(im 1-dimens. Fall Differenz der Koordinaten)}
\end{aligned}
$$

Bewegung der Zelle so, dass

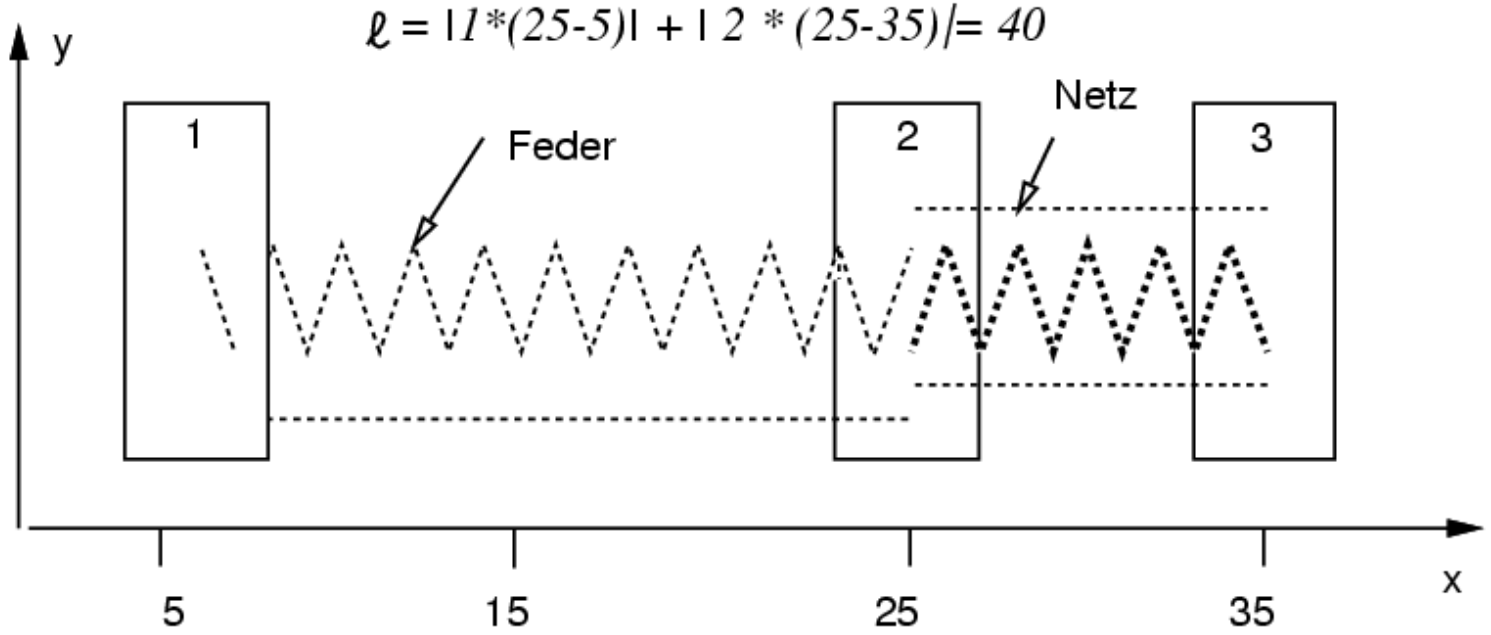
$$F_i = \sum_j F_{i,j} = 0$$

Beispiel

Zellen 1 und 3 fest:

$$F_2 = -1*(25-5) - 2 * (25-35) = 0$$

$$\ell = |1*(25-5)| + |2 * (25-35)| = 40$$

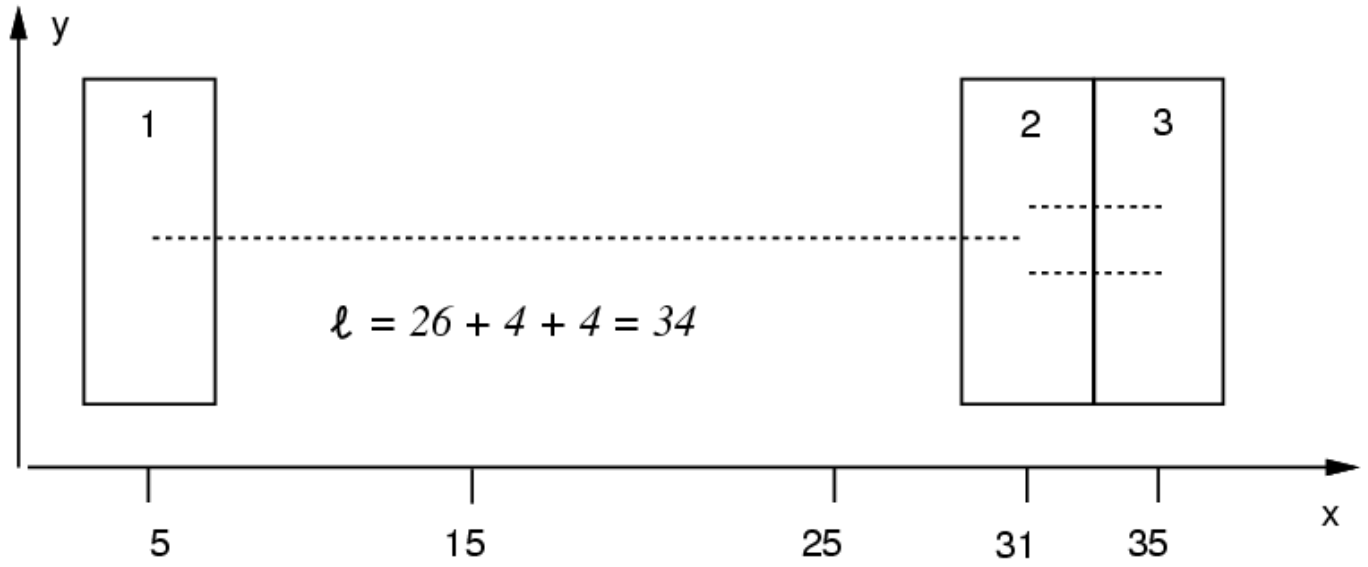


Platzierung der Zelle 2 lässt die resultierende Kraft auf diese Zelle zu Null werden.

Probleme des Modells

- Algorithmen nach diesem Modell tendieren dazu, alle Zellen demselben Platz zuzuorden.
- Platzierung minimaler resultierender Kräfte $\neq$ Platzierung mit minimaler Verdrahtungslänge ℓ.

  Beispiel: Platzierung minimaler Verdrahtungslänge:



Modellierung als Quadratisches Zuordnungsproblem

Explizite Minimierung der Verdrahtungslänge. Bei 2–Punkt–Netzen gilt für die Gesamtlänge T der Verdrahtung als Funktion der Platzierung p:

$$(1) \qquad T(p) = \sum_{i,j \in M; i<j} w_{i,j} * d_{p(i),p(j)}$$

$$(2) \qquad = \sum_{i,j,k,l; i<j} w_{i,j} * d_{k,l} * x_{i,k} * x_{j,l}$$

Darin bedeuten:

M : die Menge der Zellen;

$w_{i,j}$: Zahl der den Zellen i und j gemeinsamen Netze;

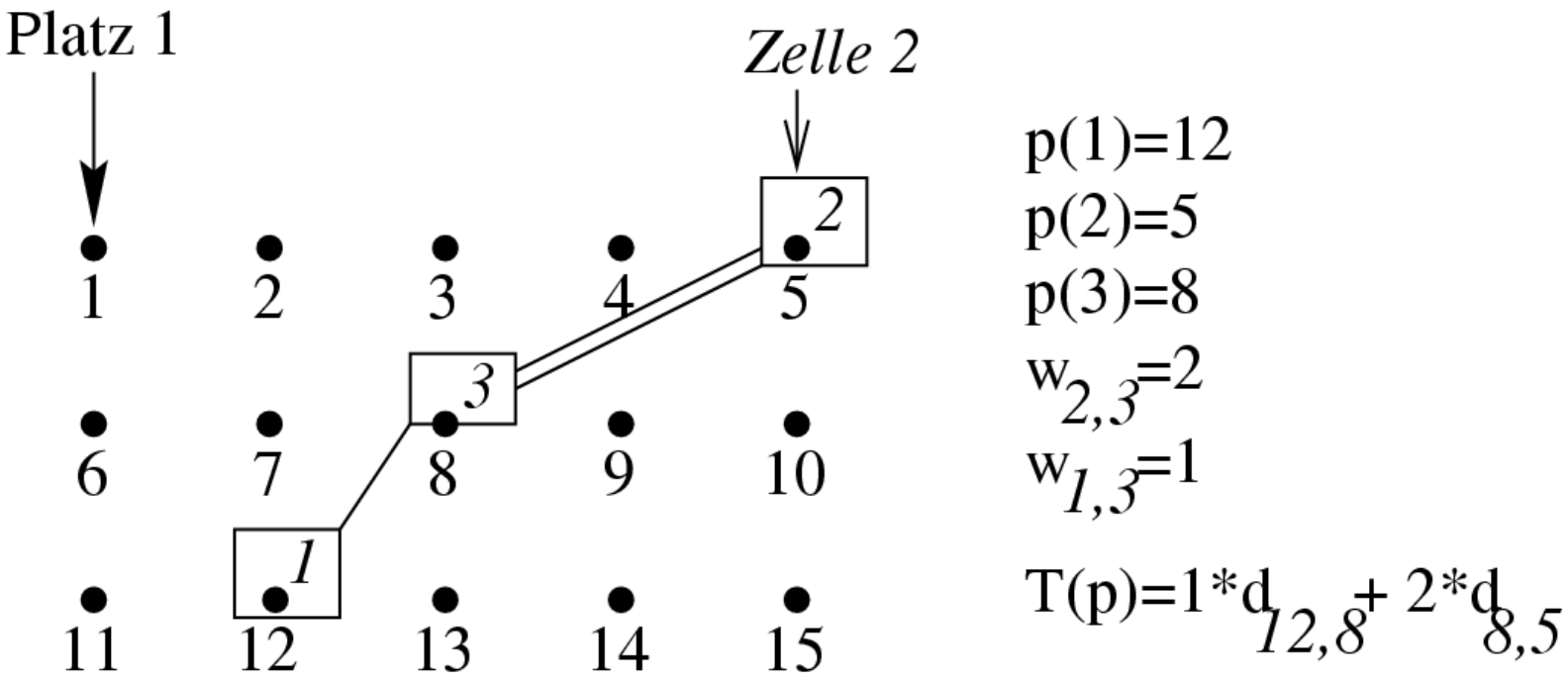
p : Funktion, die Zellen einen Platz zuordnet
mit: $\forall i, j \in M, i \neq j : p(i) \neq p(j)$;

$d_{k,l}$: Abstand der Plätze k und l voneinander;

$x_{i,k}$:= 1, falls $p(i) = k$ ist
0, sonst

Ziel: p so bestimmen, dass $T(p)$ minimal wird.

Beispiel


Platz 1
Zelle 2
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
1
2
3
p(1)=12
p(2)=5
p(3)=8
$w_{2,3}=2$
$w_{1,3}=1$
$T(p)=1*d_{12,8}+2*d_{8,5}$

Randbedingungen und Kontext

- Allen Zellen einen Platz zuordnen $\forall i : \sum_k x_{i,k} = 1$
- Allen Plätzen höchstens eine Zelle zuordnen. $\forall k : \sum_i x_{i,k} \leq 1$

Problem taucht auch in anderen Anwendungen auf:

Beispiel: Menge M von Maschinen mit einer Aufgabe, Fluß $w_{i,j}$ zwischen Maschinen, Aufstellung der Maschine i auf dem Platz $p(i)$.

$\longrightarrow$ Transportkosten $w_{i,j} * d_{p(i),p(j)}$.

Wahl von p? $\longrightarrow$ QAP.

QAP ist NP-hart.

Optimale Lösung in der Regel mit Branch–and–Bound–Verfahren.

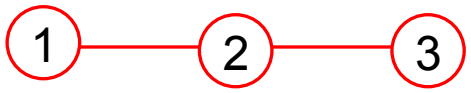
Für QAP sind im Ggs. zum allgemeinen Platzierungsproblem gute heuristische Verfahren bekannt [Preas].

n-Punkt-Netze

Bei n-Punkt-Netzen mit $n \geq 3$:

Nur noch Terme berücksichtigen, die **direkt** miteinander verbunden sind!

Beispiel:



$d_{p(1)p(2)} + d_{p(2)p(3)} + d_{p(1)p(3)}$

Es entfällt der Term für die Verbindung zwischen den Zellen 1 und 3.

Damit sind die $w_{i,j}$ nicht mehr allein aufgrund der Kenntnis der Netze zu bestimmen, sondern hängen von p ab.

Damit zumindest die Summe der $w_{i,j}$ für 2- und 3-Punkt-Netze den richtigen Wert ergibt, berücksichtigt man ein n–Punkt–Netz statt mit 1 mit $2/n$.

Platzierung mittels Partitionierung: Begriffe

Platzierung mittels fortgesetzter Aufteilung der Menge der Zellen in Teilmengen und der Zuordnung der Teilmengen zu Teilflächen der verfügbaren Fläche.

Gegeben:

- Graph $G = (V, E), E \subseteq V \times V$.
- Knotengewichte $k : V \rightarrow I\!R^+$
- Kantengewichte $w : E \rightarrow I\!R^+$

Def.: Die Zerlegung der Knotenmenge V in **Blöcke** P_i heißt Partitionierung des Graphen $G \Leftrightarrow$

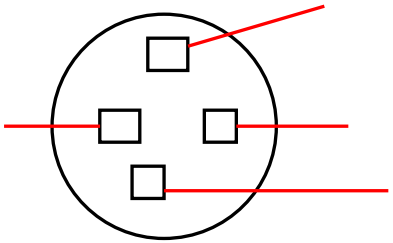
1. $\forall i : P_i \in \wp(V) \quad \wedge$
2. $\forall i, j \quad \text{mit} \quad i \neq j : P_i \cap P_j = \emptyset \quad \wedge$
3. $\cup_i P_i = V$.

Der Spezialfall der Zerlegung in zwei Blöcke $A = P_1, B = P_2$ heißt Bipartitionierung oder **Bisektion**.

Platzierung mittels Partitionierung: Begriffe (2)

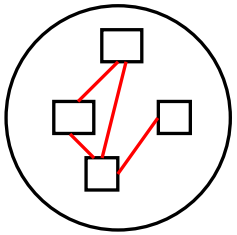
Externe Kosten E_i eines Knotens i in Block P_j sind:

$$E_i = \sum_{l \notin P_j} w_{i,l}$$



Interne Kosten I_i eines Knotens i in Block P_j sind

$$I_i = \sum_{l \in P_j} w_{i,l}$$



Ziel: Minimierung der gesamten externen Kosten:

$$\sum_j E_j \rightarrow \quad \text{Min.}$$

Beschränkungen einzuhalten, z.B.:

$$\forall i : E_i \leq E_{max};$$

$$\forall j : \sum_{i \in P_j} k_i \leq K_{max};$$

Platzierung mittels Partitionierung: Begriffe (3)

Meist die Einhaltung eines **Balancekriteriums** gefordert (sonst alle Knoten in einem Block).

Allgemeines Partitionierungsproblem ist NP–hart.

Heuristiken:

- **konstruktive Verfahren**
- **iterative Verfahren**.

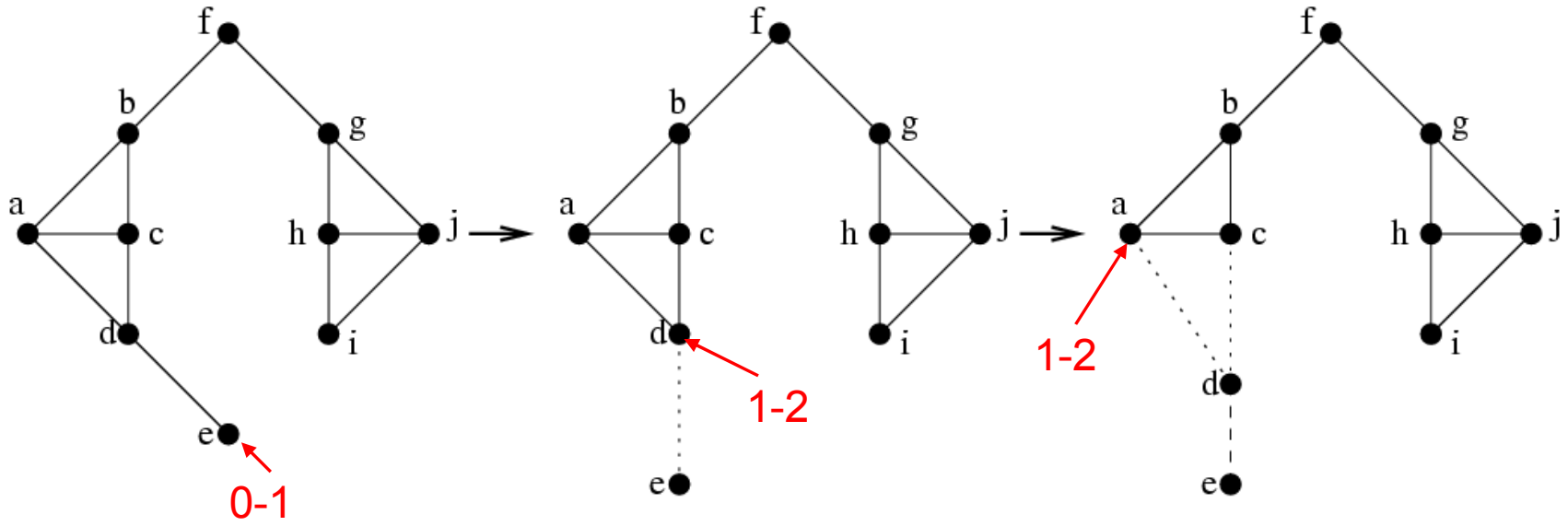
Konstruktive Partitionierung

Grundidee der meisten konstruktiven Verfahren:

$H := V; F := \emptyset;$
WHILE $H \neq \emptyset$ DO
  BEGIN
    Sei Q der Knoten aus H, für den
    (Zahl der Kanten zwischen Q und F)
    - (Zahl der Kanten zwischen Q und H)
    maximal ist.
    Ordne Q dem Block P_i zu, der die
    maximale Zahl von Verbindungen zu Q
    besitzt. Zusätzliche Beschränkungen
    sind hierbei zu beachten.
    $H := H - \{Q\}; F := F + \{Q\}$
  END

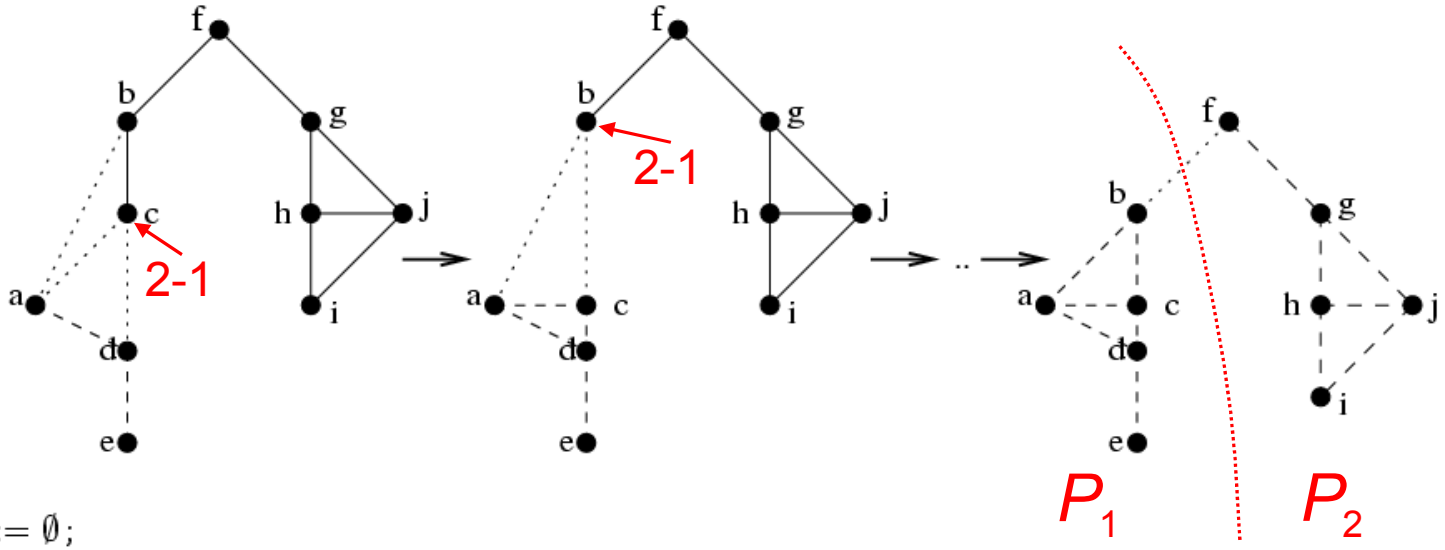
Beispiel

Gegeben Graph nach Abb. (links); Beschränkung: ca. 50% Zellen / Block.



$H := V; F := \emptyset;$
WHILE $H \neq \emptyset$ DO
  BEGIN
    Sei Q der Knoten aus H, für den (Zahl der Kanten zwischen Q und F)
    - (Zahl der Kanten zwischen Q und H) maximal ist. Ordne Q dem Block P_i zu, der die maximale Zahl von Verbindungen zu Q besitzt. Zusätzliche Beschränkungen sind hierbei zu beachten.
    $H := H - \{Q\}; F := F + \{Q\}$
  END

Weiterer Ablauf



$H := V; F := \emptyset;$
WHILE $H \neq \emptyset$ DO
  BEGIN
    Sei Q der Knoten aus H, für den (Zahl der Kanten zwischen Q und F)
    - (Zahl der Kanten zwischen Q und H) maximal ist. Ordne Q dem Block P_i zu, der die maximale Zahl von Verbindungen zu Q besitzt. Zusätzliche Beschränkungen sind hierbei zu beachten.
    $H := H - \{Q\}; F := F + \{Q\}$
  END

Einmal getroffene Zuordnungen bei den meisten Verfahren nicht wieder verworfen $\longrightarrow$ Häufig verbesserungsfähig.

Verfahren von Kernighan und Lin

Kernighan und Lin, 1970

Annahme: Startpartition $|A| = |B| = n$ gegeben.

Paarweise Vertauschung von Knoten aus A und B.

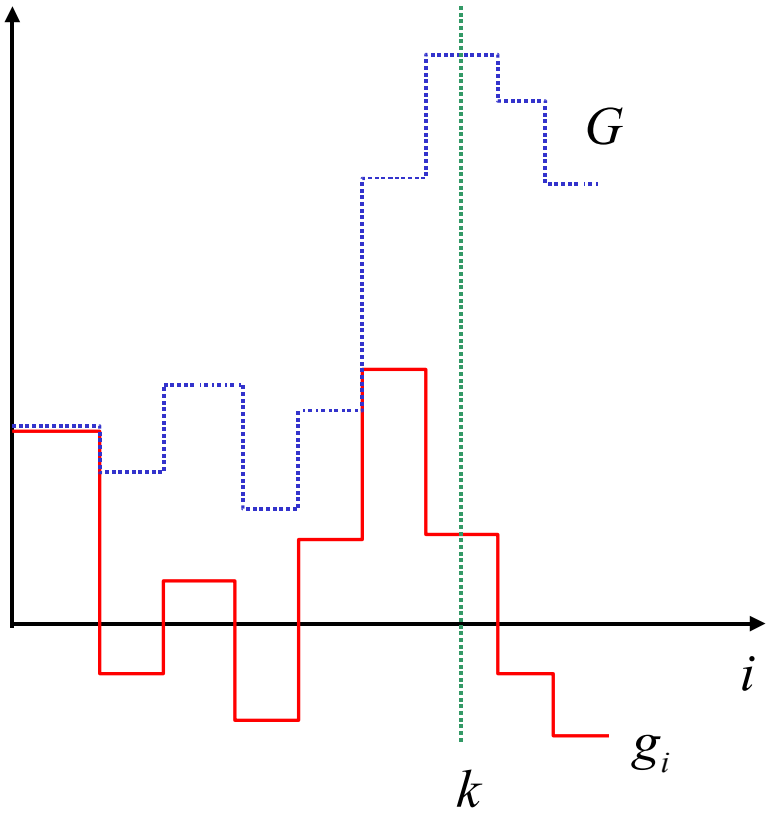
Fortführen des Vertauschens, auch wenn es momentan keinen Gewinn mehr erbringt und sich aber aber Ende ein positiver Gewinn einstellt.

Verfahren von Kernighan und Lin (2)

In äußerer Schleife Vertauschungen iteriert solange der Gesamtgewinn >0:

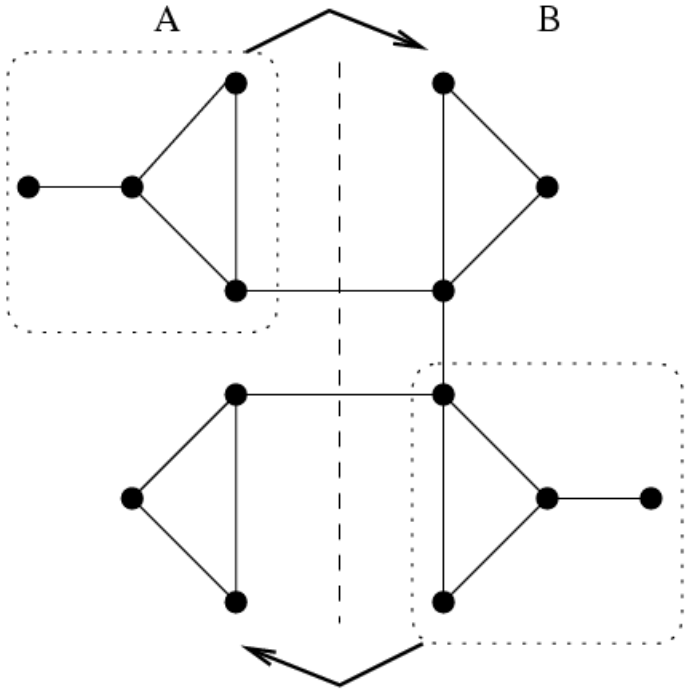
REPEAT $Max := 0; G := 0;$
  A';=A; B':=B;
  FOR i:=1 TO n DO
    BEGIN
      Wähle $a_i \in A', b_i \in B'$ so, dass das Vertauschen
      von a und b den Gewinn g maximal werden läßt.
      $A' := A' - \{a_i\}; B' := B' - \{b_i\};$
      g_i:=Gewinn durch das Vertauschen von a_i und b_i.
      $G := G + g_i;$
      IF $G > Max$ THEN BEGIN $k := i;$ $Max := G$ END;
    END;
  IF $Max > 0$ THEN
    BEGIN
      $A := A - \{a_1..a_k\} \cup \{b_1..b_k\};$
      $B := B - \{b_1..b_k\} \cup \{a_1..a_k\};$
    END;
UNTIL $Max \leq 0;$

Verhalten beim Überwinden vorübergehender Verschlechterungen



Beispiel

Partitionierter Graph, bei dem eine Reduktion der Anzahl der geschnittenen Kanten von 2 auf 1 nur über eine momentane Erhöhung der Anzahl der geschnittenen Kanten möglich ist:



Linearer Algorithmus von Fiduccia/Mattheyses

Komplexität des Kernighan–Lin–Verfahrens?

Für die Auswahl von a_i und b_i müßten ggf. alle Knotenpaare aus A und B betrachtet werden.

Schon ohne die Gewinnberechnung würde sich ein quadratischer Effekt ergeben.

Unerwünscht, da man Graphen mit einigen Hundert Knoten noch in vertretbarer Zeit partitionieren möchte.

Datenstrukturen nach Fiduccia und Mattheyses: lineares Laufzeitverhalten.

Nur die Bewegung eines Knotens in einen anderen Block.

Gewicht aller Kanten 1, Minimierung der **Anzahl** der geschnittenen Netze

Sorgfältige Buchführung über die durch Vertauschung möglichen Gewinne.

Typische Gesamtkomplexität von $O(P = \Sigma p_i)$
(mit $p_i =$ Zahl der Pins der Zelle i).

Ausgangssituation (1)

Im Folgenden:

1. Jede Zelle *i* hat p_i Pins.
2. Jede Zelle hat eine Größe w_i.

- Eine Zellen seien **fest** einem Block zugeordnet.
- Alle übrigen heißen **freie Zellen.**
- Statt der Gesamtgröße der Zellen wird nur noch deren Anzahl berücksichtigt (ok für geringe Größenvariation).
- Balancekriterium:
  $r(|A|+|B|) - s_{max} \leq |A| \leq r(|A|+|B|) + s_{max}$
  mit $s_{max} := \max_i (p_i)$
  s_{max} ist ausreichend groß, um stets eine Bewegung entweder einer Zelle aus A oder B zu garantieren.
  r ist das gewünschte Verhältnis ($|A|/(|A|+|B|)$)

|A|, |B| mit Anzahl der Pins gewichtet, sonst reicht s_{max}=1

Ausgangssituation (2)

Gegeben sei ein Netzwerk aus Zellen und Netzen.

Netze im Eingabefile wie folgt beschrieben:

$$n_1(c_{11},p_{11})(c_{12},p_{12})\ldots$$
$$n_2(c_{21},p_{21})(c_{22},p_{22})\ldots$$
$$\ldots.$$
$$n_N(c_{N1},p_{N1})(c_{N2},p_{N2})\ldots$$

Die Gesamtzahl der Netze sei N,

die Zahl der Zellen sei C.

Def.: Ein Netz heißt **geschnitten**, falls es Zellen in A und B enthält.

Def.: Die **Schnittmenge** (engl. *cutset*) ist die Menge aller geschnittenen Netze.

Gesamtalgorithmus

```
procedure Fiduccia;
  begin
    InitNetAndCell;
    CheckBalance; (*Stelle Balance her*)
    repeat
      InitBuckets;
      Pass(G);
    until G ≤ 0;
  end;
```

Einzelner Lauf der Vertauschungsroutine

```
procedure Pass(out G);
  begin
   Max:=-1; Blocked:={};
   repeat
    Betrachte die freien Zellen mit Gewinn MaxGain für A und B.
    Selektiere jene, deren Bewegung das Balancekriterium erhält.
    Falls derartige Zellen nicht existieren, dann exit;
    Wähle eine Zelle i mit dem höchsten Gewinn,
    bei Gewinngleichheit wähle aufgrund bester Balance,
    Entferne die Zelle i aus der Bucket-Liste;
    Korrigiere MaxGainA bzw. MaxGainB;
    AdjustGain(i); Blocked:=Blocked+{i};
    G:=Gewinn der gegenwärtigen Partition; (erzielte Reduktion)
    if G>Max then begin (A',B'):=gegenwärtige Partition; Max:=G end;
   until False;
   if Max>0 then begin A:=A'; B:=B'; G:=Max end else G:=0;
  end;
```

Pro Lauf maximal 4 Gewinnkorrekturen/Netz.
Komplexität also $O(P)$.

Typische Anwendungen des Algorithmus von Fiduccia/Mattheyses

Chip	Zellen	Netze	Pins	Läufe	CPU-Zeit [Sek] (VAX 780)
1	306	300	857	3	1,63
2	296	238	672	2	0,98
3	214	222	550	5	1,91
4	255	221	571	5	2,09

Die Komplexität von Pass ist $O(P)$. Laut Tabelle wird die äußere Schleife wird in der Regel 2- bis 5-mal durchlaufen. Damit ist die Gesamtkomplexität **auf der Basis typischer Anwendungen** wieder $O(P)$.

Kernighan-Lin/Fiduccia-Mattheyses ist Standard-Verfahren der Partitionierung auch außerhalb von EDA.☺

Datenstrukturen

Rasche Zuordnung zwischen Netzen und Zellen mittels Arrays:

```
type
 NetType  = set of  0..C;
 CellType = record
              s:set of 0..N; ...
            end;
var
 Net : array [1..N] of NetType; (* Abbildung Netz → Zellen *)
 Cell: array [1..C] of CellType; (* Abbildung Zelle → Netze *)
```

Berechnung der Arrays

```
procedure InitNetAndCell;
  begin
   for each net ∈ [1..N] do   (*Für alle Eingabezeilen    *)
     for each (c,p) do
       begin                          (*Für jedes Paar der Zeile*)
        Net [n] :=Net [n] +{c};
        Cell[c].s :=Cell[c].s +{n};
      end;
  end;
```

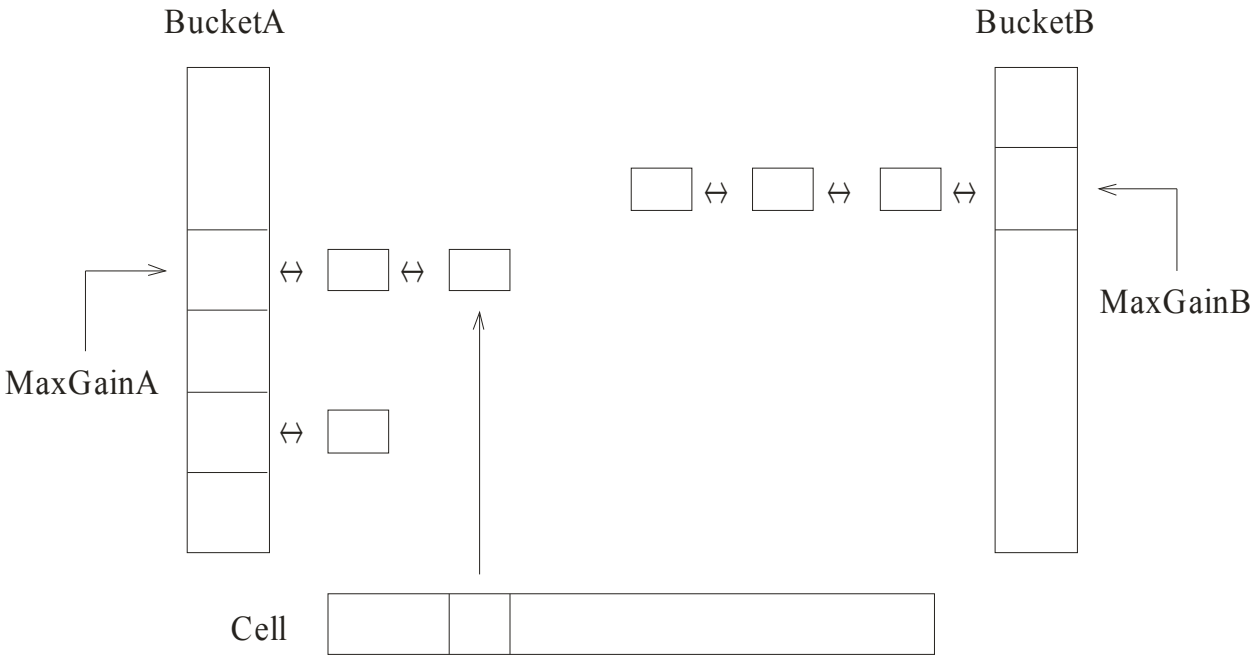
Zahl der Operationen ~ Gesamtzahl *P* der Pins:

$$P = \sum_{i=1}^{C} p_i$$

Selektion und Bewegung einer Zelle

Zelle, deren Bewegung von A nach B bzw. umgekehrt den maximalen Gewinn erbringt? Zugriff über Array in der Reihenfolge abnehmender Gewinne

Für A und B getrennte Arrays. Indices MaxGainA und MaxGainB für direkten Zugriff. Auf die Listenelemente kann vom Array Cell aus über die Zellnummer direkt zugegriffen werden.



Kritische Netze

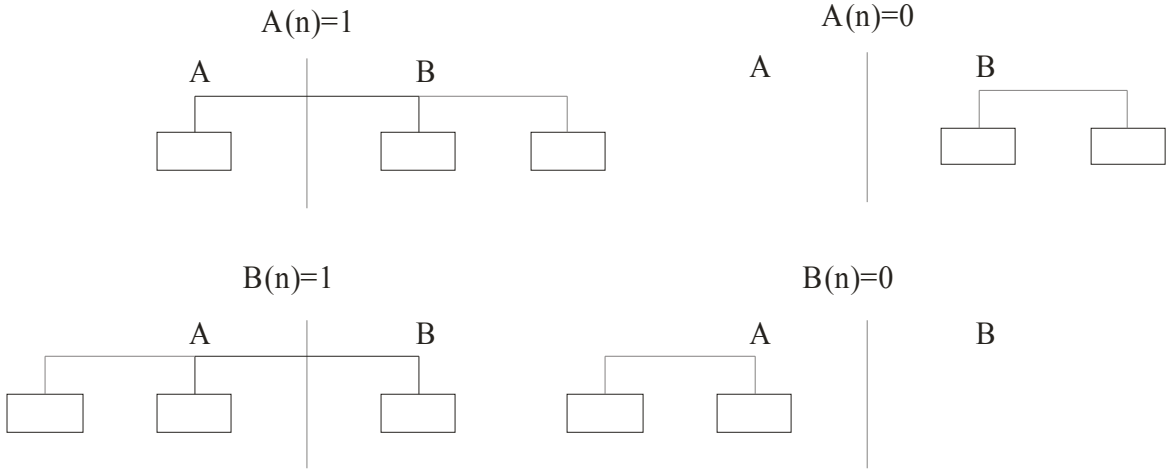
anz(A,*n*) = A(*n*) die Zahl der Zellen von *n*, die in A liegen.

anz(B,*n*) = B(*n*) die Zahl der Zellen von *n*, die in B liegen.

Def.: Ein Netz heißt **kritisch**, wenn es einen Anschluss an einer Zelle enthält, deren Bewegung den Schnittzustand des Netzes ändert, d.h. wenn das Netz vor der Bewegung geschnitten und hinterher ungeschnitten ist bzw. umgekehrt.

Lemma: *n* ist kritisch $\Leftrightarrow$ A(n)=0 $\vee$ A(n)=1 $\vee$ B(n)=0 $\vee$ B(n)=1

Vier mögliche Fälle kritischer Netze



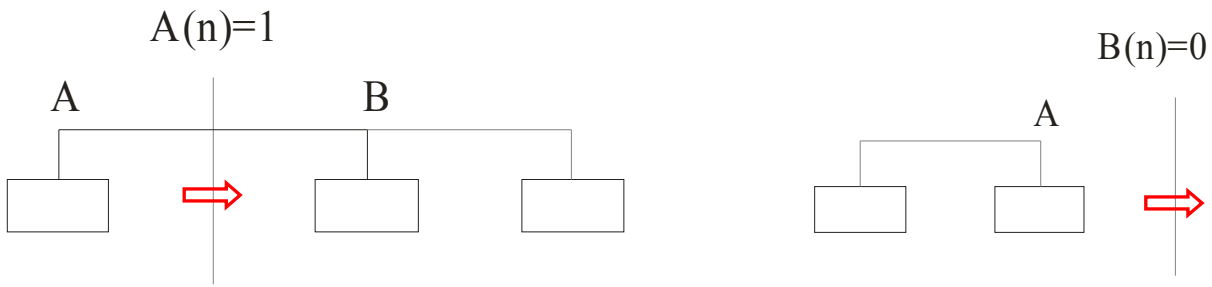
Initialisierung der Gewinne

Def.: Für jede Zelle *i* bezeichne *F*(*i*) den gegenwärtigen Block von *i* und *T*(*i*) dessen Komplement (*From-Block* bzw. *To-Block*)

Def.: Für jede Zelle *i* bezeichne *FS*(*i*) die Zahl der Netze, die *i* als einzige Zelle in *F*(*i*) enthalten.

Def.: Für jede Zelle *i* bezeichne *TE*(*i*) die Zahl der Netze, die *i* enthalten und die keine Zellen in *T*(*i*) enthalten.

Lemma: Die Bewegung der Zelle *i* von *F*(*i*) nach *T*(*i*) reduziert die Zahl der geschnittenen Netze um den Gewinn $g_i = FS(i)\text{-}TE(i)$



Aufbau der Arrays

procedure InitBuckets;
begin
Initialisiere BucketA und BucketB, d.h. berechne die Gewinne für die 1. Vertauschung. Hierzu ist das oben eingeführte $g_i = FS(i)\text{-}TE(i)$ durch Abzählen zu berechnen.
end;

Gewinnkorrekturen

Bei der Zellenbewegung Gewinne anpassen!

Lemma: Ein Netz n ist **vor** der Bewegung einer Zelle i kritisch $\Leftrightarrow$ anz($F(i)$,n)=1 $\vee$ anz($T(i)$,n)=0 $\vee$ anz($T(i)$,n)=1

Fall anz($F(i)$,n)=0 kann nicht eintreten.

Lemma: Ein Netz n ist **nach** der Bewegung einer Zelle i kritisch $\Leftrightarrow$ anz($F(i)$,n)=1 $\vee$ anz($F(i)$,n)=0 $\vee$ anz($T(i)$,n)=1

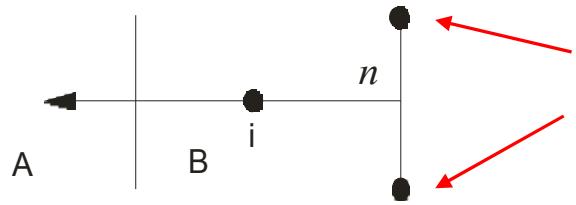
Der Fall anz($T(i)$,n)=0 kann nicht eintreten.

Lemma:

- anz($F(i)$,n)=1 vor der Bewegung $\Leftrightarrow$ anz($F(i)$,n)=0 nach der Bewegung.
- anz($T(i)$,n)=0 vor der Bewegung $\Leftrightarrow$ anz($T(i)$,n)=1 nach der Bewegung.

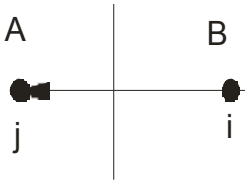
Gewinnkorrektur (1)

```
procedure AdjustGains(in i:1..C);
  begin
    for each net n of i do
     begin (* Operationen vor dem Wechsel von i:*)
      if anz(T(i),n)=0 then ++ für die Gewinne aller freien Zellen des
      Netzes n; Wechsel der übrigen Zellen des Netzes wäre vorteilhaft
```



Inkrementiere Gewinne dieser Zellen, da ihre Bewegung nach A die Schnittmenge reduzieren würde.

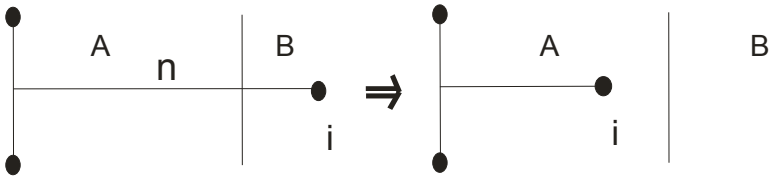
else if anz(T(i),n)=1 **then** -- für den Gewinn der einzigen Zelle j in T(i) (falls frei); Wechsel von j nach B bringt keinen Gewinn mehr.



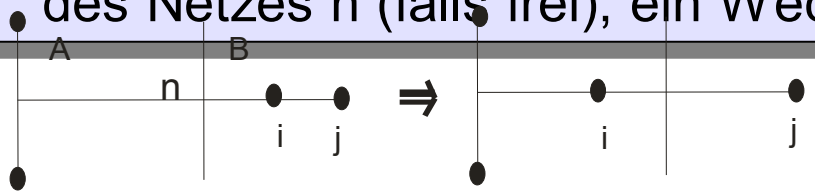
Dekrementiere den Gewinn der Zelle j, da deren Bewegung die Schnittmenge nicht mehr reduzieren würde.

Gewinnkorrektur (2)

anz(T(i),n):=anz(T(i),n)+1; anz(F(i),n):=anz(F(i),n)-1;
(* Wechsel; Operationen nach dem Wechsel:*)
if anz(F(i),n)=0 **then** -- für die Gewinne aller freien Zellen des Netzes n; kein Gewinn mehr bei Wechsel nach B.



else if anz(F(i),n)=1 **then** ++ für Gewinn der einzigen F-Zelle j
des Netzes n (falls frei); ein Wechsel würde Gewinn bringen.



end; Vertausche F(i) und T(i);
end;

Zahl der Operationen in der inneren Schleife ist $O(P)$

Zusammenfassung

Begriff der Netzliste
Platzierung

- Randbedingungen und Zielfunktionen
- ~ nach dem Kräftemodell
- Modell des Quadratischen Zuordnungsproblems
- Partititonierung
  - Kernighan/Lin
  - Fiduccia/Mattheyses