Algorithmen für Chip-Entwurfswerkzeuge

Partitionierung

TECHNISCHE UNIVERSITÄT DARMSTADT

Vorlesung
WS 2014/2015

Andreas Koch

Eingebette Systeme und Anwendungen
Technische Universität Darmstadt

Embedded Systems & Applications

Partitionierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Aufteilung einer Schaltung in Teilschaltungen (Partitionen, Blöcke)

Partitionierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Aufteilung einer Schaltung in Teilschaltungen (Partitionen, Blöcke)
  - Aufteilung auf mehrere Chips (auch System/Board/Package-Level)

Partitionierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Aufteilung einer Schaltung in Teilschaltungen (Partitionen, Blöcke)
  - Aufteilung auf mehrere Chips (auch System/Board/Package-Level)
  - Verkleinerung der Problemgröße
    (Vorbereitung auf anderen Algorithmus)

Partitionierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Aufteilung einer Schaltung in Teilschaltungen (Partitionen, Blöcke)
  - Aufteilung auf mehrere Chips (auch System/Board/Package-Level)
  - Verkleinerung der Problemgröße
    (Vorbereitung auf anderen Algorithmus)
- Optimierungsziele:

Partitionierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Aufteilung einer Schaltung in Teilschaltungen (Partitionen, Blöcke)
  - Aufteilung auf mehrere Chips (auch System/Board/Package-Level)
  - Verkleinerung der Problemgröße
    (Vorbereitung auf anderen Algorithmus)
- Optimierungsziele:
  - Minimierung der Verknüpfungen zwischen den Partitionen

Partitionierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Aufteilung einer Schaltung in Teilschaltungen (Partitionen, Blöcke)
  - Aufteilung auf mehrere Chips (auch System/Board/Package-Level)
  - Verkleinerung der Problemgröße
    (Vorbereitung auf anderen Algorithmus)
- Optimierungsziele:
  - Minimierung der Verknüpfungen zwischen den Partitionen
  - Flächenziele

Partitionierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Aufteilung einer Schaltung in Teilschaltungen (Partitionen, Blöcke)
  - Aufteilung auf mehrere Chips (auch System/Board/Package-Level)
  - Verkleinerung der Problemgröße
    (Vorbereitung auf anderen Algorithmus)
- Optimierungsziele:
  - Minimierung der Verknüpfungen zwischen den Partitionen
  - Flächenziele
    - Gleichmäßige Größe

Partitionierung

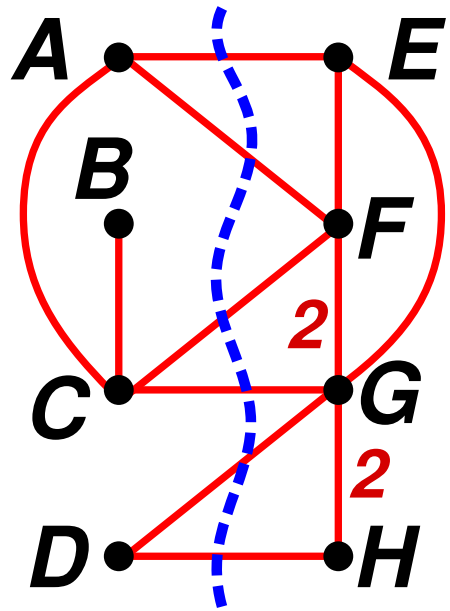
TECHNISCHE UNIVERSITÄT DARMSTADT

- Aufteilung einer Schaltung in Teilschaltungen (Partitionen, Blöcke)
  - Aufteilung auf mehrere Chips (auch System/Board/Package-Level)
  - Verkleinerung der Problemgröße
    (Vorbereitung auf anderen Algorithmus)
- Optimierungsziele:
  - Minimierung der Verknüpfungen zwischen den Partitionen
  - Flächenziele
    - Gleichmäßige Größe
    - Erreichen vorgegebener Größen

Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Problem:

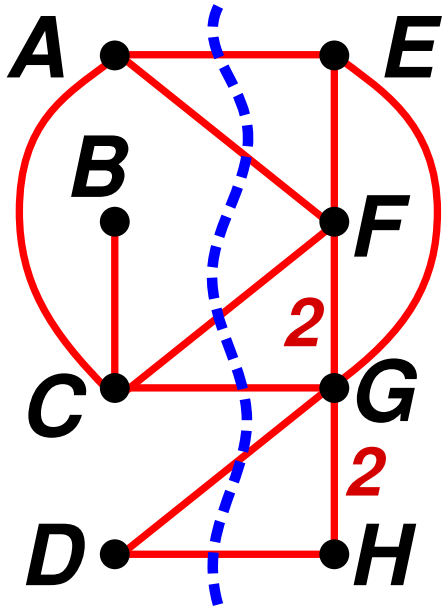


Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Problem:

  Gegeben: Graph mit $2n$ Knoten

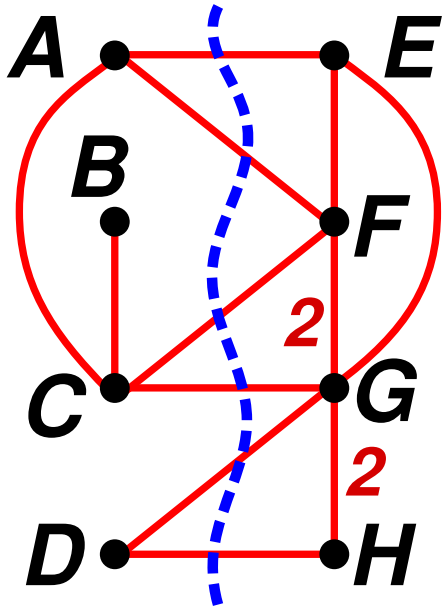


Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Problem:

  Gegeben: Graph mit $2n$ Knoten
  Gesucht: Partitionierung in 2 Knotenmengen A, B, so dass ...



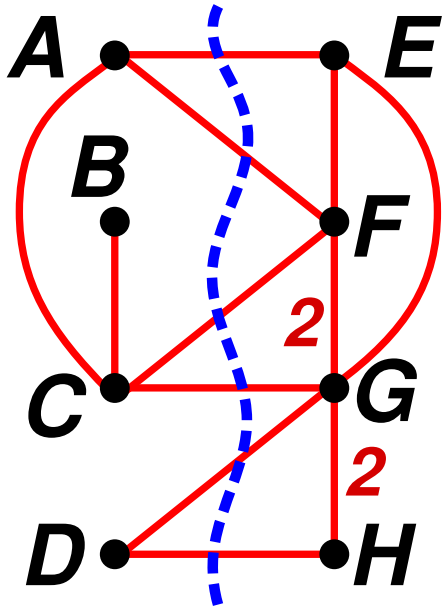
Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Problem:

  Gegeben: Graph mit $2n$ Knoten
  Gesucht: Partitionierung in 2 Knotenmengen A, B, so dass ...

  1. beide n Knoten enthalten und



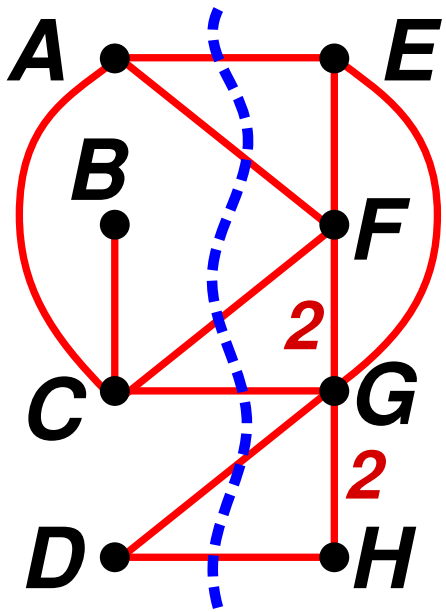
Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Problem:

  Gegeben: Graph mit $2n$ Knoten
  Gesucht: Partitionierung in 2 Knotenmengen A, B, so dass ...

  1. beide n Knoten enthalten und
  2. das Gewicht γ_{a_i,b_k} der geschnittenen Kanten $\sum_{e_i=(a_j,b_k),\ a_j \in A, b_k \in B} e_i$ (Schnittkosten) minimal ist (Min-Cut)



Problem

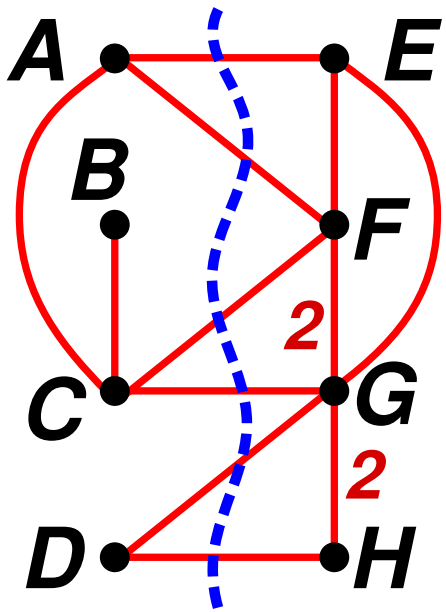
TECHNISCHE UNIVERSITÄT DARMSTADT

- Problem:

  Gegeben: Graph mit $2n$ Knoten
  Gesucht: Partitionierung in 2 Knotenmengen A, B, so dass ...

  1. beide n Knoten enthalten und
  2. das Gewicht γ_{a_i,b_k} der geschnittenen Kanten $\sum_{e_i=(a_j,b_k),\ a_j \in A, b_k \in B} e_i$ (Schnittkosten) minimal ist (Min-Cut)

- NP-vollständig



Problem

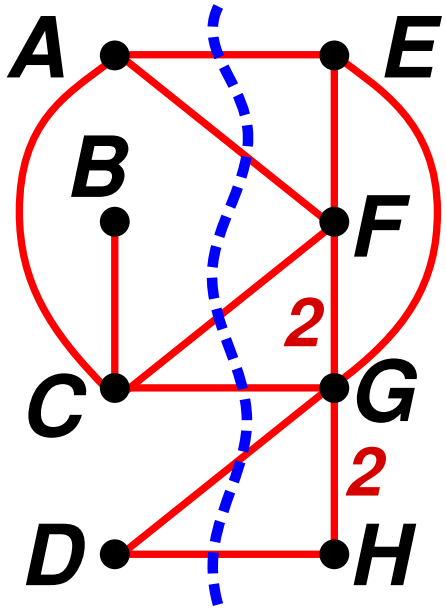
TECHNISCHE UNIVERSITÄT DARMSTADT

- Problem:

  Gegeben: Graph mit $2n$ Knoten
  Gesucht: Partitionierung in 2 Knotenmengen A, B, so dass ...

  1. beide n Knoten enthalten und
  2. das Gewicht γ_{a_i,b_k} der geschnittenen Kanten $\sum_{e_i=(a_j,b_k),\ a_j \in A, b_k \in B} e_i$ (Schnittkosten) minimal ist (Min-Cut)

- NP-vollständig
- Bewährte Heuristiken:



Problem

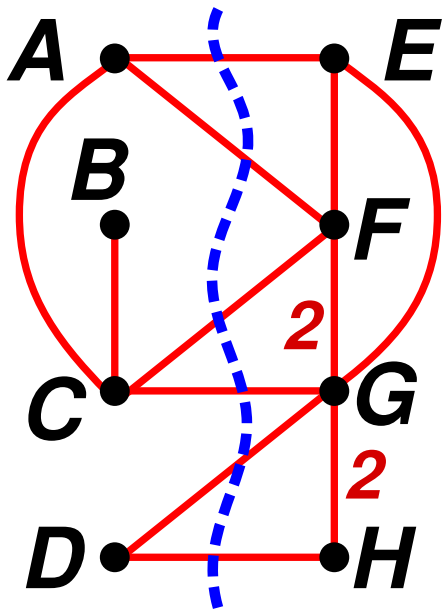
TECHNISCHE UNIVERSITÄT DARMSTADT

- Problem:

  Gegeben: Graph mit $2n$ Knoten
  Gesucht: Partitionierung in 2 Knotenmengen A, B, so dass ...

  1. beide n Knoten enthalten und
  2. das Gewicht γ_{a_i,b_k} der geschnittenen Kanten $\sum_{e_i=(a_j,b_k),\ a_j \in A, b_k \in B} e_i$ (Schnittkosten) minimal ist (Min-Cut)

- NP-vollständig
- Bewährte Heuristiken:
  - Kerninghan-Lin (KL)



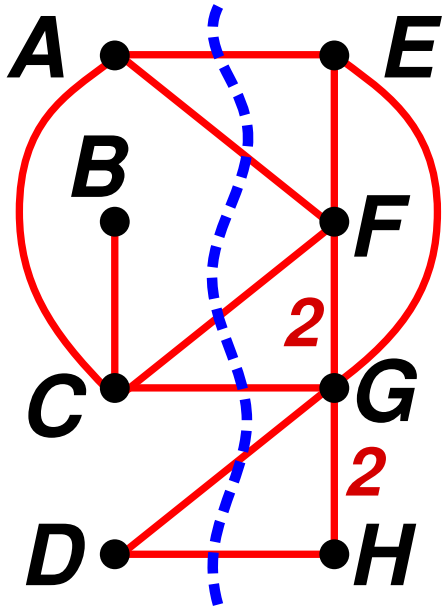
Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Problem:

  Gegeben: Graph mit $2n$ Knoten
  Gesucht: Partitionierung in 2 Knotenmengen A, B, so dass ...

  1. beide n Knoten enthalten und
  2. das Gewicht γ_{a_i,b_k} der geschnittenen Kanten $\sum_{e_i=(a_j,b_k),\ a_j \in A, b_k \in B} e_i$ (Schnittkosten) minimal ist (Min-Cut)



- NP-vollständig
- Bewährte Heuristiken:
  - Kerninghan-Lin (KL)
  - Fediuccia-Mattheyses (FM)

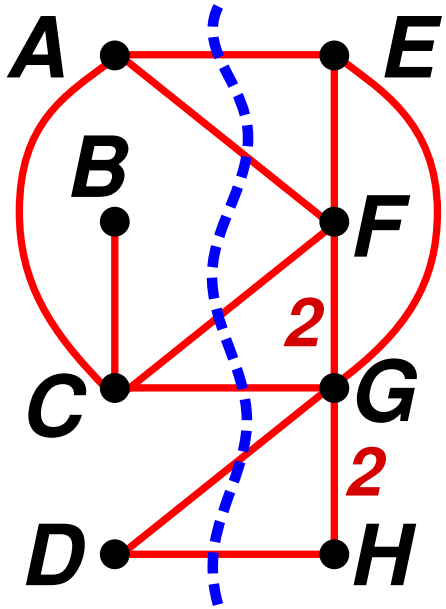
Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Problem:

  Gegeben: Graph mit $2n$ Knoten
  Gesucht: Partitionierung in 2 Knotenmengen A, B, so dass ...

  1. beide n Knoten enthalten und
  2. das Gewicht γ_{a_i,b_k} der geschnittenen Kanten $\sum_{e_i=(a_j,b_k),\ a_j \in A, b_k \in B} e_i$ (Schnittkosten) minimal ist (Min-Cut)

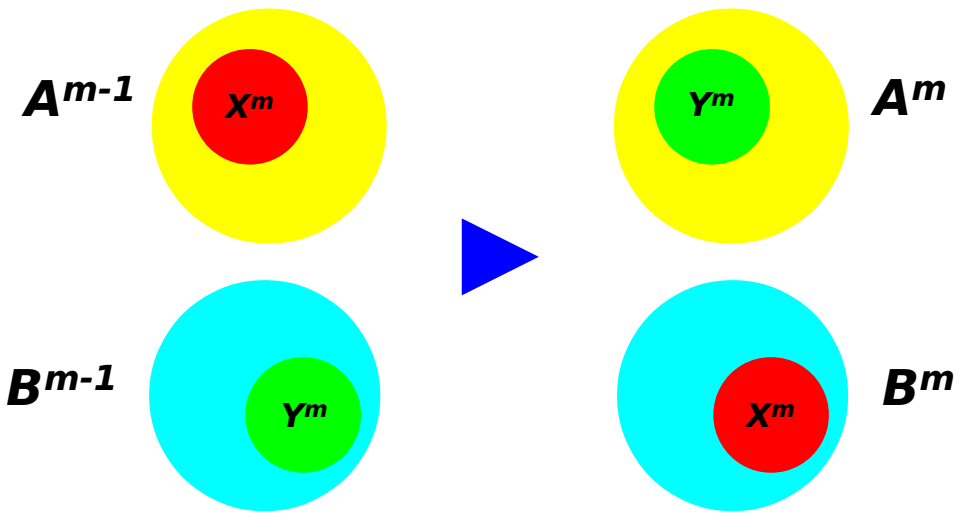


- NP-vollständig
- Bewährte Heuristiken:
  - Kerninghan-Lin (KL)
  - Fediuccia-Mattheyses (FM)
  - SA

Kerninghan-Lin-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

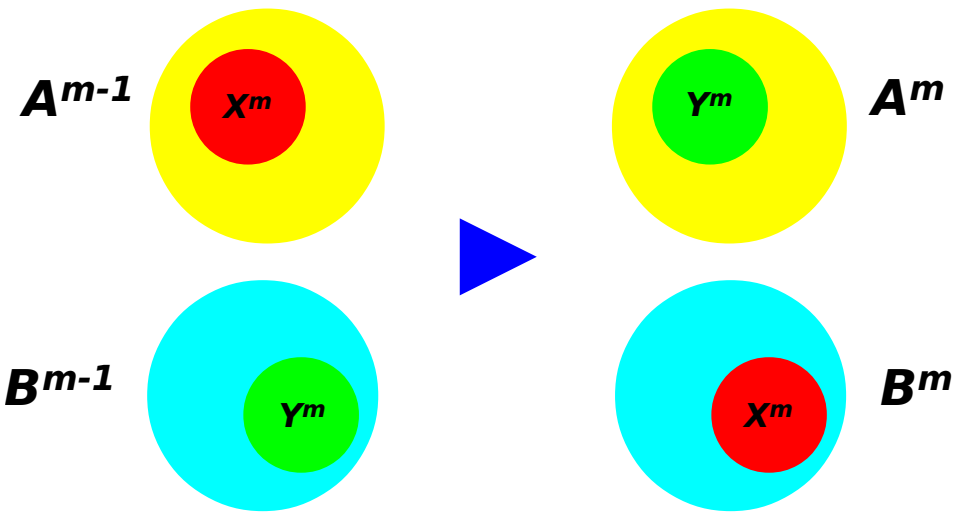
- 1970



Kerninghan-Lin-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

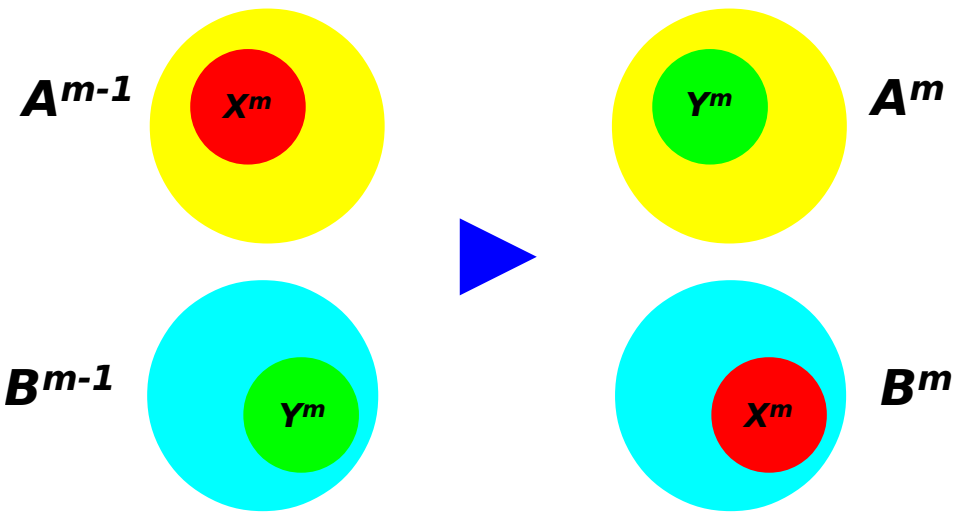
- 1970
- Idee:



Kerninghan-Lin-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

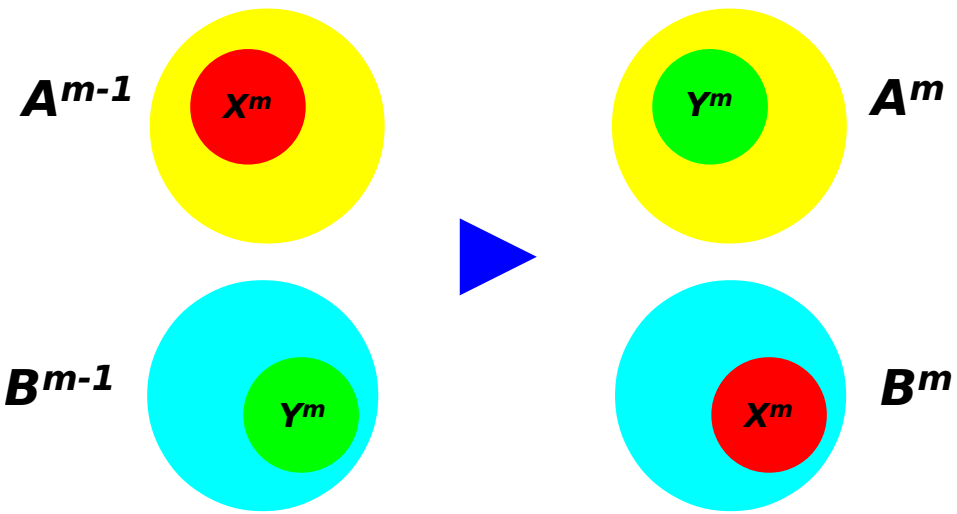
- 1970
- Idee:
  1. Anfangslösung mit beliebiger Startlösung A_0, B_o



Kerninghan-Lin-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

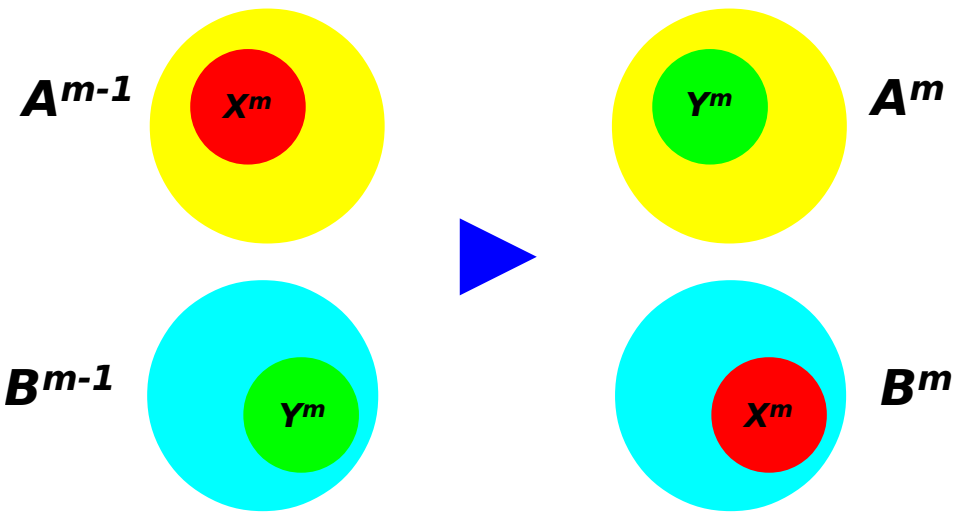
- 1970
- Idee:
  1. Anfangslösung mit beliebiger Startlösung A_0, B_o
  2. Isoliere Teilmenge $X_m \subset A_m$ und $Y_m \subset B_m$



Kerninghan-Lin-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

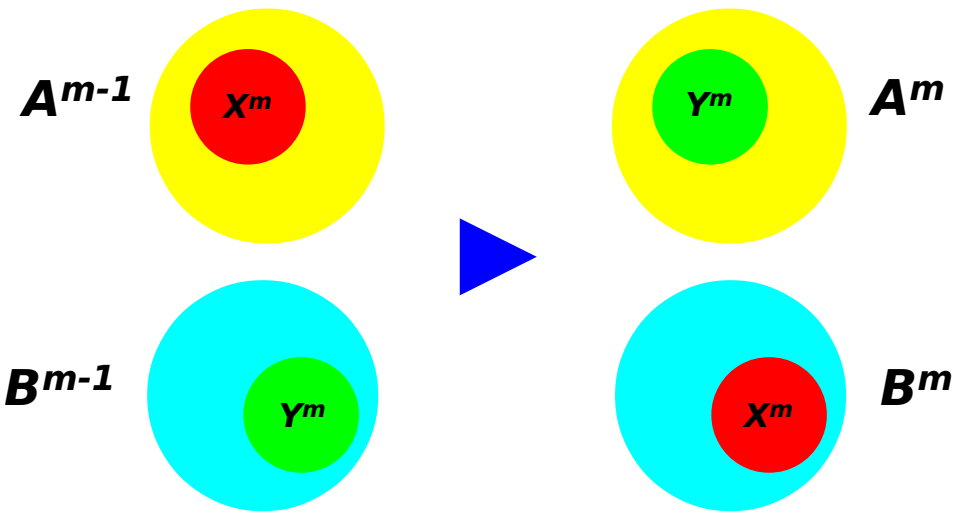
- 1970
- Idee:
  1. Anfangslösung mit beliebiger Startlösung A_0, B_o
  2. Isoliere Teilmenge $X_m \subset A_m$ und $Y_m \subset B_m$
  3. Tausche Teilmengen um A_{m+1} und B_{m+1} zu erzeugen



Kerninghan-Lin-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

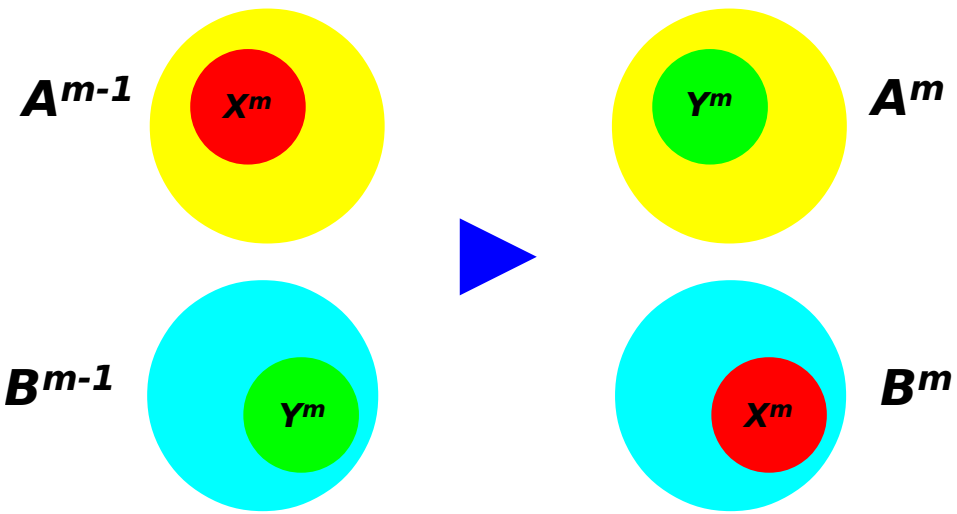
- 1970
- Idee:
  1. Anfangslösung mit beliebiger Startlösung A_0, B_o
  2. Isoliere Teilmenge $X_m \subset A_m$ und $Y_m \subset B_m$
  3. Tausche Teilmengen um A_{m+1} und B_{m+1} zu erzeugen
  4. Wiederhole bis keine Verbesserung mehr erreichbar



Kerninghan-Lin-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

- 1970
- Idee:
  1. Anfangslösung mit beliebiger Startlösung A_0, B_o
  2. Isoliere Teilmenge $X_m \subset A_m$ und $Y_m \subset B_m$
  3. Tausche Teilmengen um A_{m+1} und B_{m+1} zu erzeugen
  4. Wiederhole bis keine Verbesserung mehr erreichbar
- Arbeitet auf 2-Pin Netzen (Cliquen-Modell)



Kerninghan-Lin-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

- Optimum immer in einem Schritt erzielbar
  Bei geeignetem X_m und Y_m

Kerninghan-Lin-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

- Optimum immer in einem Schritt erzielbar
  Bei geeignetem X_m und Y_m
- Problem: Wie X_m und Y_m wählen?

Kerninghan-Lin-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

- Optimum immer in einem Schritt erzielbar
  Bei geeignetem X_m und Y_m
- Problem: Wie X_m und Y_m wählen?
  - Schwer zu finden

Kerninghan-Lin-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

- Optimum immer in einem Schritt erzielbar
  Bei geeignetem X_m und Y_m
- Problem: Wie X_m und Y_m wählen?
  - Schwer zu finden

$\Rightarrow$ Suche Lösung in mehreren Schritten
Solange bis keine Verbesserung mehr

Kerninghan-Lin-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

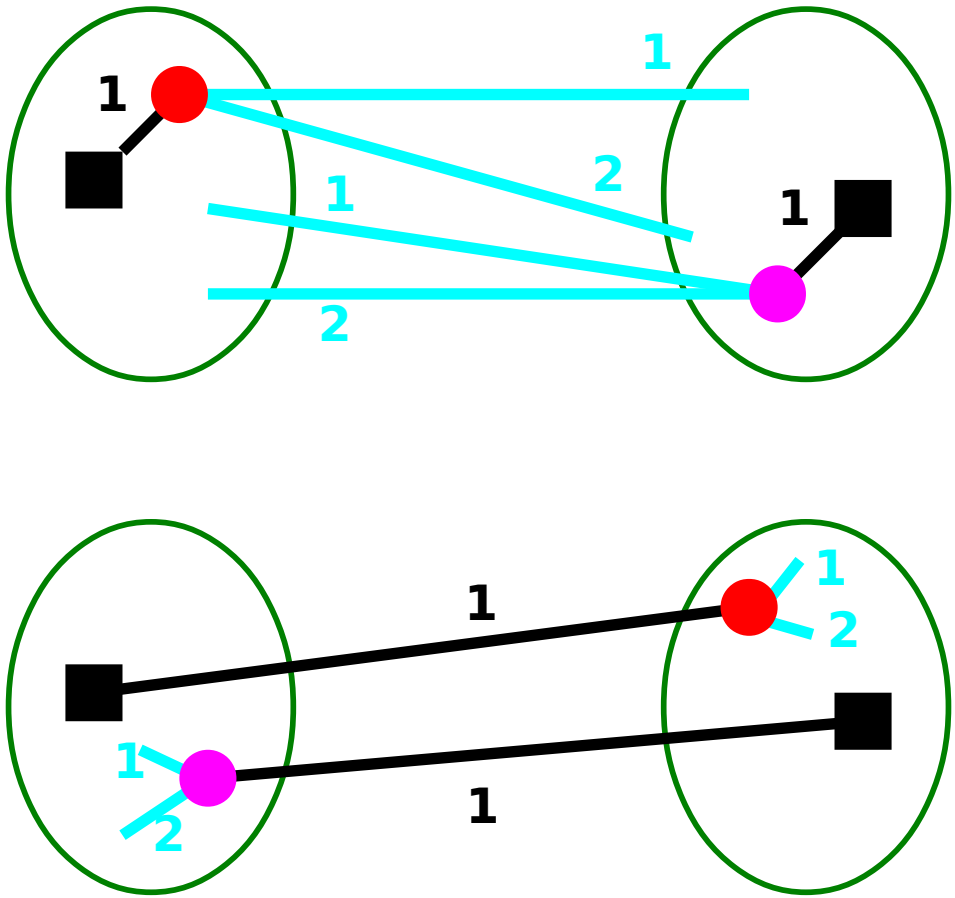
- Optimum immer in einem Schritt erzielbar
  Bei geeignetem X_m und Y_m
- Problem: Wie X_m und Y_m wählen?
  - Schwer zu finden

$\Rightarrow$ Suche Lösung in mehreren Schritten
Solange bis keine Verbesserung mehr

- Anzahl Schritte unabhängig von n
  In der Praxis ≤ 4

Kerninghan-Lin-Algorithmus
Kostenidee

TECHNISCHE UNIVERSITÄT DARMSTADT



Kerninghan-Lin-Algorithmus
Kosten

TECHNISCHE UNIVERSITÄT DARMSTADT

- Konstruktion von X_{m+1} und Y_{m+1}

Kerninghan-Lin-Algorithmus
Kosten

TECHNISCHE UNIVERSITÄT DARMSTADT

- Konstruktion von X_{m+1} und Y_{m+1}
- Externe Kosten

$$E(a) = \sum_{c \in B_m} \gamma_{a,c}, a \in A_m$$

Kerninghan-Lin-Algorithmus
Kosten

TECHNISCHE UNIVERSITÄT DARMSTADT

- Konstruktion von X_{m+1} und Y_{m+1}
- Externe Kosten

$$E(a) = \sum_{c \in B_m} \gamma_{a,c}, a \in A_m$$

- Interne Kosten

$$I(a) = \sum_{c \in A_m} \gamma_{a,c}, a \in A_m$$

Kerninghan-Lin-Algorithmus
Kosten

TECHNISCHE UNIVERSITÄT DARMSTADT

- Konstruktion von X_{m+1} und Y_{m+1}
- Externe Kosten

$$E(a) = \sum_{c \in B_m} \gamma_{a,c}, a \in A_m$$

- Interne Kosten

$$I(a) = \sum_{c \in A_m} \gamma_{a,c}, a \in A_m$$

- D-Kosten (Drang/Desirability) – Kosten falls Knoten wechselt

$$D(a) = E(a) - I(a)$$

Kerninghan-Lin-Algorithmus
Kosten

TECHNISCHE UNIVERSITÄT DARMSTADT

- Konstruktion von X_{m+1} und Y_{m+1}
- Externe Kosten

$$E(a) = \sum_{c \in B_m} \gamma_{a,c}, a \in A_m$$

- Interne Kosten

$$I(a) = \sum_{c \in A_m} \gamma_{a,c}, a \in A_m$$

- D-Kosten (Drang/Desirability) – Kosten falls Knoten wechselt

$$D(a) = E(a) - I(a)$$

  - Groß, $> 0 \Rightarrow$ Drang Partition zu wechseln

Kerninghan-Lin-Algorithmus
Kosten

TECHNISCHE UNIVERSITÄT DARMSTADT

- Konstruktion von X_{m+1} und Y_{m+1}
- Externe Kosten

$$E(a) = \sum_{c \in B_m} \gamma_{a,c}, a \in A_m$$

- Interne Kosten

$$I(a) = \sum_{c \in A_m} \gamma_{a,c}, a \in A_m$$

- D-Kosten (Drang/Desirability) – Kosten falls Knoten wechselt

$$D(a) = E(a) - I(a)$$

  - Groß, $> 0 \Rightarrow$ Drang Partition zu wechseln
  - Klein, $< 0 \Rightarrow$ Drang in Partition zu bleiben

Kerninghan-Lin-Algorithmus
Kosten

TECHNISCHE UNIVERSITÄT DARMSTADT

- Konstruktion von X_{m+1} und Y_{m+1}
- Externe Kosten

$$E(a) = \sum_{c \in B_m} \gamma_{a,c}, a \in A_m$$

- Interne Kosten

$$I(a) = \sum_{c \in A_m} \gamma_{a,c}, a \in A_m$$

- D-Kosten (Drang/Desirability) – Kosten falls Knoten wechselt

$$D(a) = E(a) - I(a)$$

  - Groß, $> 0 \Rightarrow$ Drang Partition zu wechseln
  - Klein, $< 0 \Rightarrow$ Drang in Partition zu bleiben
- Analog für B

Kerninghan-Lin
Beispiel Tauschkosten

TECHNISCHE UNIVERSITÄT DARMSTADT

- Verbesserung Δ der Schnittkosten, falls *a* mit *b* ausgetauscht wird

Kerninghan-Lin
Beispiel Tauschkosten

TECHNISCHE UNIVERSITÄT DARMSTADT

- Verbesserung Δ der Schnittkosten, falls a mit b ausgetauscht wird
- Bei Austausch von $a \in A_m$ und $b \in B_m$

$$\Delta = D(a) + D(b) - 2\gamma_{a,b}$$

Kerninghan-Lin
Beispiel Tauschkosten

TECHNISCHE UNIVERSITÄT DARMSTADT

- Verbesserung Δ der Schnittkosten, falls a mit b ausgetauscht wird
- Bei Austausch von $a \in A_m$ und $b \in B_m$

$$\Delta = D(a) + D(b) - 2\gamma_{a,b}$$

- Δ kann auch negativ sein!

Kerninghan-Lin Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

KernighanLin(V,E,γ): **begin**

- Init(A_0, B_0) ;
- m := 0 ;
- **repeat**
  - **foreach** $a \in A_m$ **do** *Berechne* $D(a)$;
  - **foreach** $b \in B_m$ **do** *Berechne* $D(b)$;
  - **for** *(i:=1; i* $\leq$ *n; i++)* **do**
    - *Finde freies* $a_i \in A_m, b_i \in B_m$ *mit* $\Delta_i := D(a_i) + D(b_i) - 2\gamma_{a_i,b_i}$ *maximal* ;
    - *Sperre* a_i *und* b_i ;
    - **foreach** *freie* $x \in A_m$ **do** $D(x) := D(x) + 2\gamma_{x,a_i} - 2\gamma_{x,b_i}$;
    - **foreach** *freie* $x \in B_m$ **do** $D(x) := D(x) - 2\gamma_{x,a_i} + 2\gamma_{x,b_i}$;
  - *Finde k mit* $G := \sum_{i=1}^{k} \Delta_i$ *das maximal ist* ;
  - **if** $G > 0$ **then**
    - $X_{m+1} := a_1, \ldots, a_k$;
    - $Y_{m+1} := b_1, \ldots, b_k$;
    - $A_{m+1} := (A_m \setminus X_{m+1}) \cup X_{m+1}$;
    - $B_{m+1} := (B_m \setminus Y_{m+1}) \cup Y_{m+1}$;
    - *Entsperre alle Knoten* ;
    - m := m+1 ;
- **until** $G \leq 0$;

Initialisierung

Startmengen A_0 und B_0 erzeugen

Kerninghan-Lin Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

KernighanLin(V,E,γ): **begin**
- Init(A_0, B_0) ;
- m := 0 ;
- **repeat**
  - **foreach** $a \in A_m$ **do** *Berechne D(a)* ;
  - **foreach** $b \in B_m$ **do** *Berechne D(b)* ;
  - **for** *(i:=1; i* $\leq$ *n; i++)* **do**
    - *Finde freies* $a_i \in A_m, b_i \in B_m$ *mit* $\Delta_i := D(a_i) + D(b_i) - 2\gamma_{a_i,b_i}$ *maximal* ;
    - *Sperre* a_i *und* b_i ;
    - **foreach** *freie* $x \in A_m$ **do** $D(x) := D(x) + 2\gamma_{x,a_i} - 2\gamma_{x,b_i}$;
    - **foreach** *freie* $x \in B_m$ **do** $D(x) := D(x) - 2\gamma_{x,a_i} + 2\gamma_{x,b_i}$;
  - *Finde k mit* $G := \sum_{i=1}^{k} \Delta_i$ *das maximal ist* ;
  - **if** $G > 0$ **then**
    - $X_{m+1} := a_1, \ldots, a_k$;
    - $Y_{m+1} := b_1, \ldots, b_k$;
    - $A_{m+1} := (A_m \setminus X_{m+1}) \cup X_{m+1}$;
    - $B_{m+1} := (B_m \setminus Y_{m+1}) \cup Y_{m+1}$;
    - *Entsperre alle Knoten* ;
    - m := m+1 ;
- **until** $G \leq 0$;

Berechnung der D-Kosten

- Für alle Knoten
- Komplexität: $\mathcal{O}(n^2)$

Kerninghan-Lin Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

```
KernighanLin(V,E,γ): begin
    Init(A_0, B_0) ;
    m := 0 ;
    repeat
        foreach a ∈ A_m do  Berechne D(a) ;
        foreach b ∈ B_m do  Berechne D(b) ;
        for (i:=1; i ≤ n; i++) do
            Finde freies a_i ∈ A_m, b_i ∈ B_m mit Δ_i := D(a_i) + D(b_i) − 2γ_{a_i,b_i} maximal ;
            Sperre a_i und b_i ;
            foreach freie x ∈ A_m do  D(x) := D(x) + 2γ_{x,a_i} − 2γ_{x,b_i} ;
            foreach freie x ∈ B_m do  D(x) := D(x) − 2γ_{x,a_i} + 2γ_{x,b_i} ;
        Finde k mit G := Σ_{i=1}^{k} Δ_i das maximal ist ;
        if G > 0 then
            X_{m+1} := a_1, ... , a_k ;
            Y_{m+1} := b_1, ... , b_k ;
            A_{m+1} := (A_m \ X_{m+1}) ∪ X_{m+1} ;
            B_{m+1} := (B_m \ Y_{m+1}) ∪ Y_{m+1} ;
            Entsperre alle Knoten ;
            m := m+1 ;
    until G ≤ 0;
```

Gewinn beim Tauschen

- Für alle Knotenpaare a_i und b_i
- Komplexität: $\mathcal{O}(n^2)$
- Paar mit größtmöglichem Gewinn wählen

Kerninghan-Lin Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

KernighanLin(V,E,γ): **begin**
- Init(A_0, B_0) ;
- m := 0 ;
- **repeat**
  - **foreach** $a \in A_m$ **do** *Berechne* $D(a)$;
  - **foreach** $b \in B_m$ **do** *Berechne* $D(b)$;
  - **for** *(i:=1; i* $\leq$ *n; i++)* **do**
    - *Finde freies* $a_i \in A_m, b_i \in B_m$ *mit* $\Delta_i := D(a_i) + D(b_i) - 2\gamma_{a_i,b_i}$ *maximal* ;
    - *Sperre* a_i *und* b_i ;
    - **foreach** *freie* $x \in A_m$ **do** $D(x) := D(x) + 2\gamma_{x,a_i} - 2\gamma_{x,b_i}$;
    - **foreach** *freie* $x \in B_m$ **do** $D(x) := D(x) - 2\gamma_{x,a_i} + 2\gamma_{x,b_i}$;
  - *Finde k mit* $G := \sum_{i=1}^{k} \Delta_i$ *das maximal ist* ;
  - **if** $G > 0$ **then**
    - $X_{m+1} := a_1, \ldots, a_k$;
    - $Y_{m+1} := b_1, \ldots, b_k$;
    - $A_{m+1} := (A_m \setminus X_{m+1}) \cup X_{m+1}$;
    - $B_{m+1} := (B_m \setminus Y_{m+1}) \cup Y_{m+1}$;
    - *Entsperre alle Knoten* ;
    - m := m+1 ;
- **until** $G \leq 0$;

Freie und gesperrte Knoten

- Einmal zum Tausch ausgewählte Elemente werden gesperrt
- Erst nächste Iteration wieder verfügbar

Kerninghan-Lin Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

KernighanLin(V,E,γ): **begin**
- Init(A_0, B_0) ;
- m := 0 ;
- **repeat**
  - **foreach** $a \in A_m$ **do** *Berechne* $D(a)$;
  - **foreach** $b \in B_m$ **do** *Berechne* $D(b)$;
  - **for** *(i:=1; i* $\leq$ *n; i++)* **do**
    - *Finde freies* $a_i \in A_m, b_i \in B_m$ *mit* $\Delta_i := D(a_i) + D(b_i) - 2\gamma_{a_i,b_i}$ *maximal* ;
    - *Sperre* a_i *und* b_i ;
    - **foreach** *freie* $x \in A_m$ **do** $D(x) := D(x) + 2\gamma_{x,a_i} - 2\gamma_{x,b_i}$;
    - **foreach** *freie* $x \in B_m$ **do** $D(x) := D(x) - 2\gamma_{x,a_i} + 2\gamma_{x,b_i}$;
  - *Finde k mit* $G := \sum_{i=1}^{k} \Delta_i$ *das maximal ist* ;
  - **if** $G > 0$ **then**
    - $X_{m+1} := a_1, \ldots, a_k$;
    - $Y_{m+1} := b_1, \ldots, b_k$;
    - $A_{m+1} := (A_m \setminus X_{m+1}) \cup X_{m+1}$;
    - $B_{m+1} := (B_m \setminus Y_{m+1}) \cup Y_{m+1}$;
    - *Entsperre alle Knoten* ;
    - m := m+1 ;
- **until** $G \leq 0$;

Update der D-Kosten

- $D(x)$ nicht mehr aktuell, falls x mit a_i oder b_i verbunden

$$
\begin{aligned}
D(x) &= E(x) - I(x) \\
&= E(x)^{old} + \gamma_{a,x} - \gamma_{b,x} - \\
&\quad (I(x)^{old} - \gamma_{a,x} + \gamma_{b,x}) \\
&= D(x)^{old} + 2\gamma_{a,x} - 2\gamma_{b,x}
\end{aligned}
$$

Kerninghan-Lin Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

KernighanLin(V,E,γ): **begin**
- Init(A_0, B_0) ;
- m := 0 ;
- **repeat**
  - **foreach** $a \in A_m$ **do** *Berechne* $D(a)$;
  - **foreach** $b \in B_m$ **do** *Berechne* $D(b)$;
  - **for** *(i:=1; i* $\leq$ *n; i++)* **do**
    - *Finde freies* $a_i \in A_m, b_i \in B_m$ *mit* $\Delta_i := D(a_i) + D(b_i) - 2\gamma_{a_i,b_i}$ *maximal* ;
    - *Sperre* a_i *und* b_i ;
    - **foreach** *freie* $x \in A_m$ **do** $D(x) := D(x) + 2\gamma_{x,a_i} - 2\gamma_{x,b_i}$;
    - **foreach** *freie* $x \in B_m$ **do** $D(x) := D(x) - 2\gamma_{x,a_i} + 2\gamma_{x,b_i}$;
  - *Finde k mit* $G := \sum_{i=1}^{k} \Delta_i$ *das maximal ist* ;
  - **if** $G > 0$ **then**
    - $X_{m+1} := a_1, \ldots, a_k$;
    - $Y_{m+1} := b_1, \ldots, b_k$;
    - $A_{m+1} := (A_m \setminus X_{m+1}) \cup X_{m+1}$;
    - $B_{m+1} := (B_m \setminus Y_{m+1}) \cup Y_{m+1}$;
    - *Entsperre alle Knoten* ;
    - m := m+1 ;
- **until** $G \leq 0$;

Auswahl der zu tauschenden Knoten

- n vorläufige Tauschpartner
- Greedy Verfahren um tatsächliche Austauschungen zu wählen
- Δ_i kann negativ werden
- $\sum \Delta_i$ kann zeitweise auch negativ sein
  - Bei dicht verbundenen Teilmengen
  - Keine Verbesserung bei Austausch von Einzelknoten
  - Erst bei Tausch der gesamten Teilmenge

Kerninghan-Lin Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

KernighanLin(V,E,γ): **begin**
- Init(A_0, B_0) ;
- m := 0 ;
- **repeat**
  - **foreach** $a \in A_m$ **do** *Berechne* $D(a)$;
  - **foreach** $b \in B_m$ **do** *Berechne* $D(b)$;
  - **for** *(i:=1; i* $\leq$ *n; i++)* **do**
    - *Finde freies* $a_i \in A_m, b_i \in B_m$ *mit* $\Delta_i := D(a_i) + D(b_i) - 2\gamma_{a_i,b_i}$ *maximal* ;
    - *Sperre* a_i *und* b_i ;
    - **foreach** *freie* $x \in A_m$ **do** $D(x) := D(x) + 2\gamma_{x,a_i} - 2\gamma_{x,b_i}$;
    - **foreach** *freie* $x \in B_m$ **do** $D(x) := D(x) - 2\gamma_{x,a_i} + 2\gamma_{x,b_i}$;
  - *Finde k mit* $G := \sum_{i=1}^{k} \Delta_i$ *das maximal ist* ;
  - **if** $G > 0$ **then**
    - $X_{m+1} := a_1, \ldots, a_k$;
    - $Y_{m+1} := b_1, \ldots, b_k$;
    - $A_{m+1} := (A_m \setminus X_{m+1}) \cup X_{m+1}$;
    - $B_{m+1} := (B_m \setminus Y_{m+1}) \cup Y_{m+1}$;
    - *Entsperre alle Knoten* ;
    - m := m+1 ;
- **until** $G \leq 0$;

Austausch

- Wenn Gewinn gewählten Austausch durchführen
- Bis kein Gewinn mehr möglich

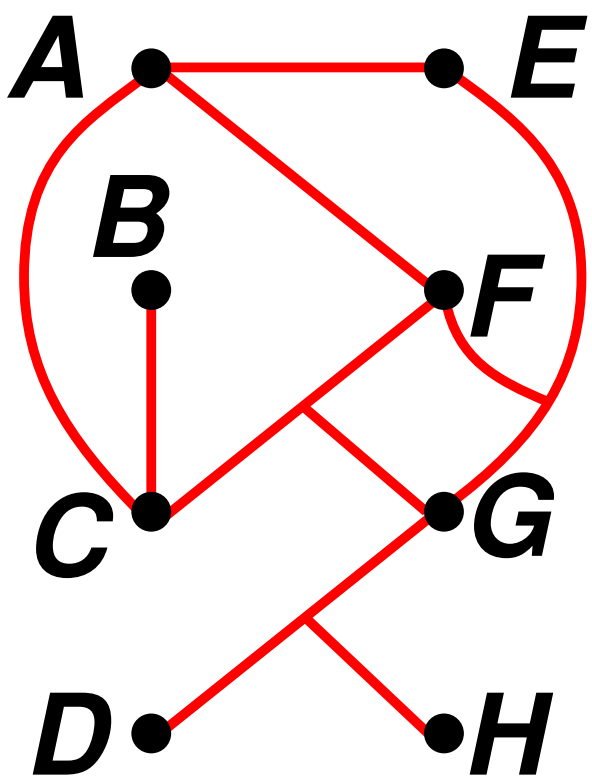
Kerninghan-Lin Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

KernighanLin(V,E,γ): **begin**
- Init(A_0, B_0) ;
- m := 0 ;
- **repeat**
  - **foreach** $a \in A_m$ **do** *Berechne* $D(a)$;
  - **foreach** $b \in B_m$ **do** *Berechne* $D(b)$;
  - **for** *(i:=1; i* $\leq$ *n; i++)* **do**
    - *Finde freies* $a_i \in A_m, b_i \in B_m$ *mit* $\Delta_i := D(a_i) + D(b_i) - 2\gamma_{a_i,b_i}$ *maximal* ;
    - *Sperre* a_i *und* b_i ;
    - **foreach** *freie* $x \in A_m$ **do** $D(x) := D(x) + 2\gamma_{x,a_i} - 2\gamma_{x,b_i}$;
    - **foreach** *freie* $x \in B_m$ **do** $D(x) := D(x) - 2\gamma_{x,a_i} + 2\gamma_{x,b_i}$;
  - *Finde k mit* $G := \sum_{i=1}^{k} \Delta_i$ *das maximal ist* ;
  - **if** $G > 0$ **then**
    - $X_{m+1} := a_1, \ldots, a_k$;
    - $Y_{m+1} := b_1, \ldots, b_k$;
    - $A_{m+1} := (A_m \setminus X_{m+1}) \cup X_{m+1}$;
    - $B_{m+1} := (B_m \setminus Y_{m+1}) \cup Y_{m+1}$;
    - *Entsperre alle Knoten* ;
    - m := m+1 ;
- **until** $G \leq 0$;

Kerninghan-Lin
Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

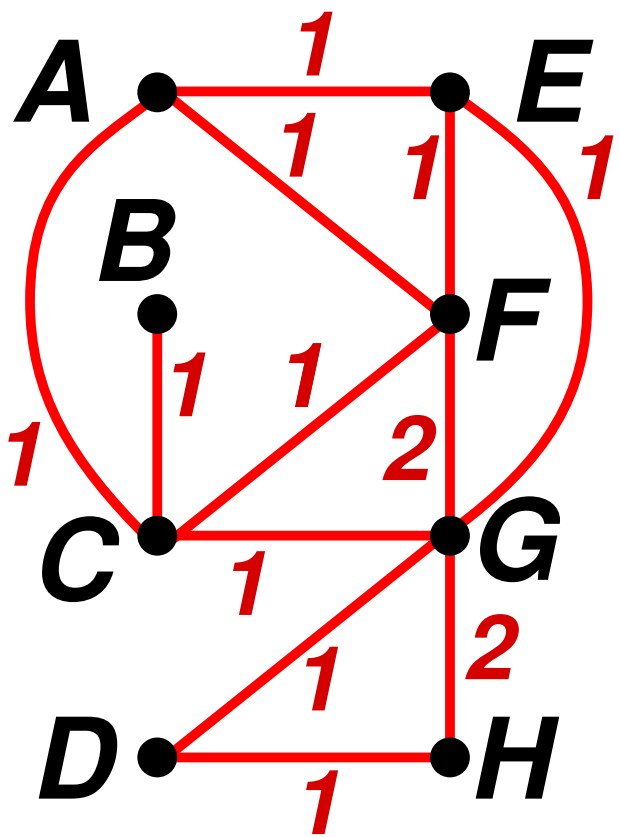


Beispielhypergraph

- Einfachheit halber: Kantengewichte alle 1

Kerninghan-Lin
Beispiel

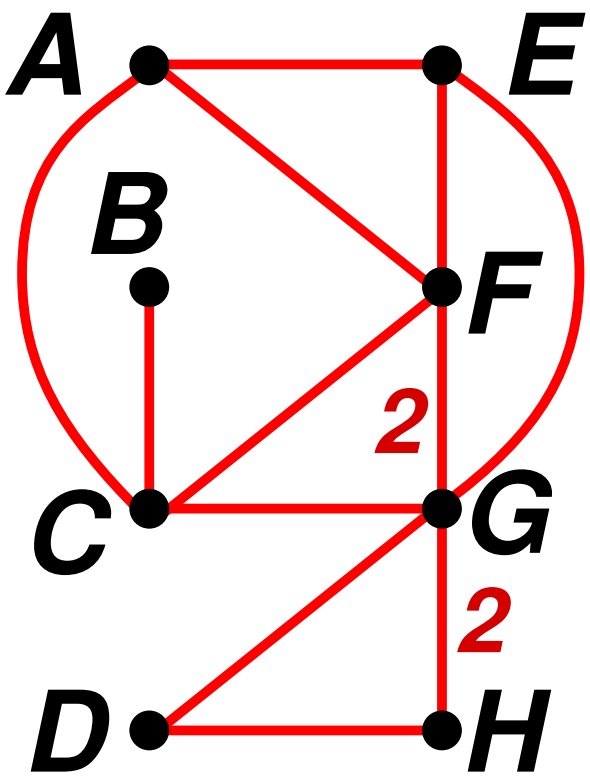
TECHNISCHE UNIVERSITÄT DARMSTADT



Beispielgraph

- Hyperkanten durch Cliquen ersetzt
- Falls bereits Kante vorhanden, werden Gewichte addiert

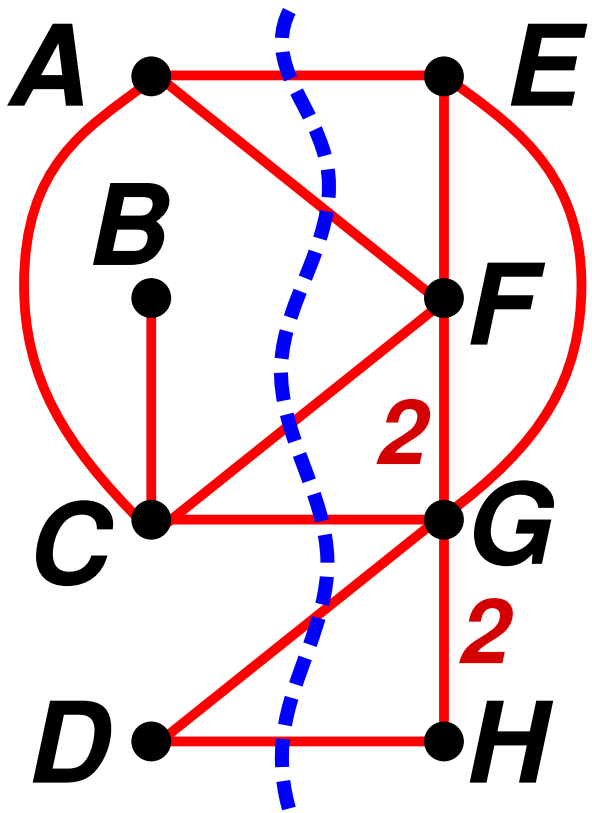
Kerninghan-Lin
Beispiel



Beispielgraph

- Nur Kantengewichte $\neq 1$ annotiert

Kerninghan-Lin
Beispiel



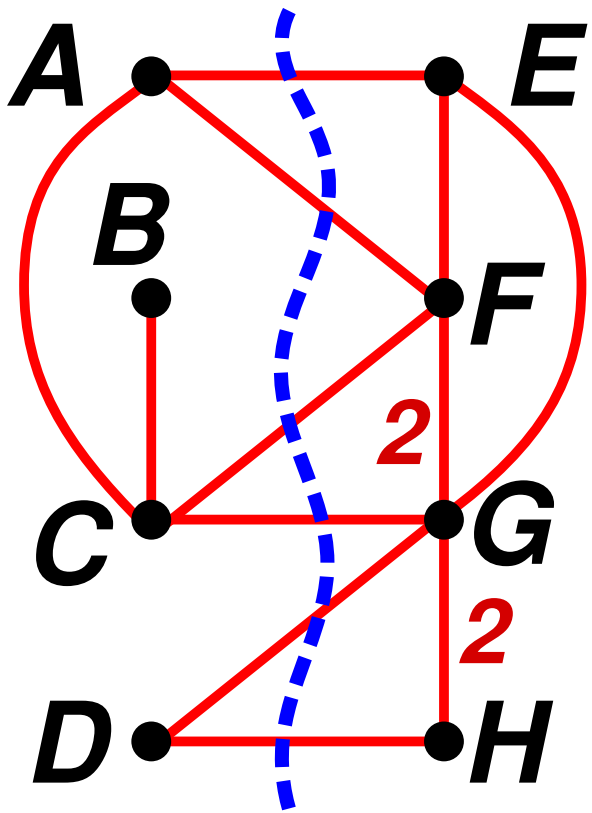
Beispielgraph

- Nur Kantengewichte $\neq 1$ annotiert
- Startpartitionen
  $A_0 = \{A, B, C, D\}, B_0 = \{E, F, G, H\}$
- Kosten: 6

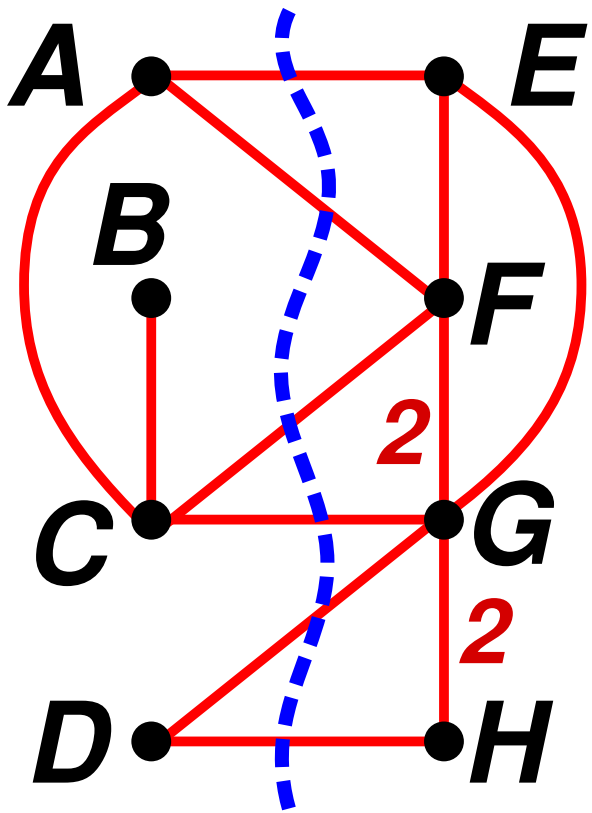
Kerninghan-Lin
Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

- 1. Durchlauf



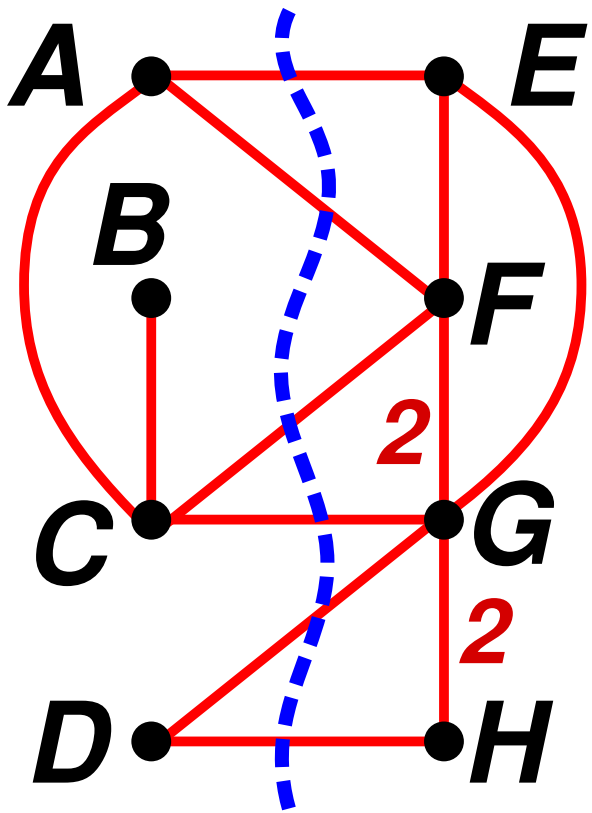
Kerninghan-Lin
Beispiel



- 1. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$E(x)$	2	0	2	2	1	2	2	1
$I(x)$	1	1	2	0	2	3	5	2
$D(x)$	1	-1	0	2	-1	-1	-3	-1

Kerninghan-Lin
Beispiel

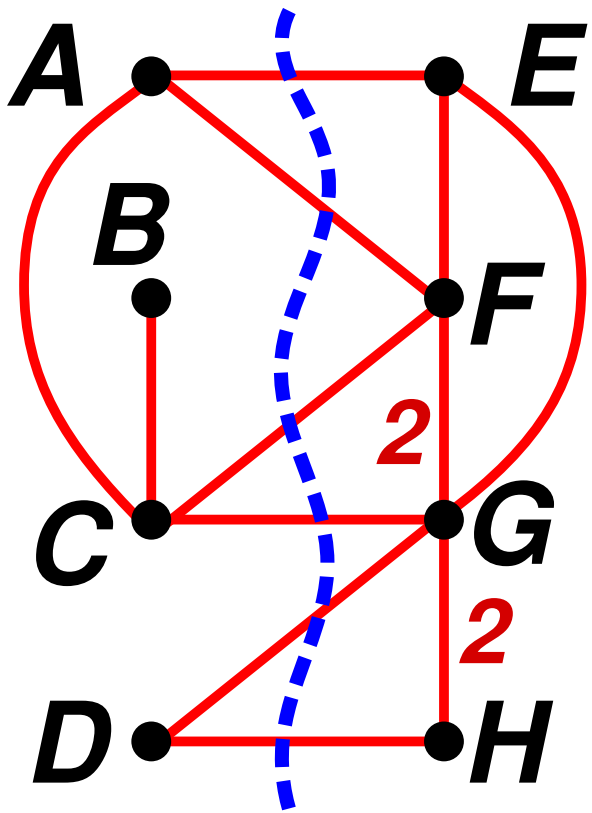


- 1. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	1	-1	0	2	-1	-1	-3	-1

- Bestimmung von Δ_1

Kerninghan-Lin
Beispiel



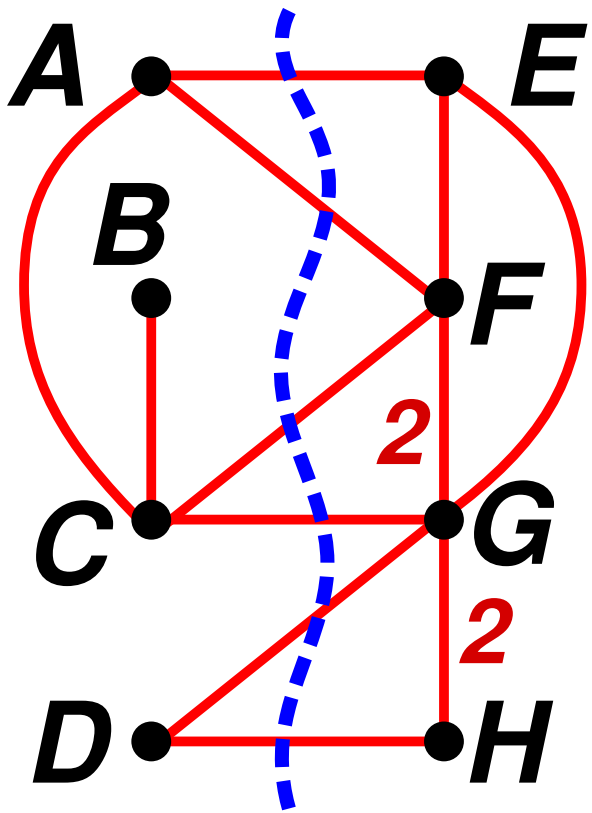
- 1. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	1	-1	0	2	-1	-1	-3	-1

- Bestimmung von Δ_1

	E	F	G	H
A	-2	-2	-2	0
B	-2	-2	-4	-2
C	-1	-3	-5	0
D	1	1	1	-1

Kerninghan-Lin
Beispiel



- 1. Durchlauf
- Bestimmung der $D(x)$

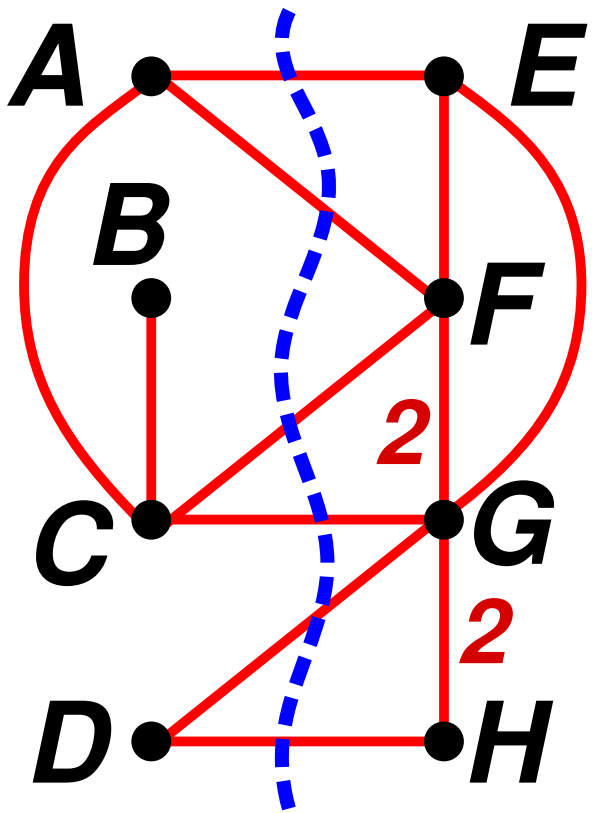
	A	B	C	D	E	F	G	H
$D(x)$	1	-1	0	2	-1	-1	-3	-1

- Bestimmung von Δ_1

	E	F	G	H
A	-2	-2	-2	0
B	-2	-2	-4	-2
C	-1	-3	-5	0
D	1	1	1	-1

$\Delta_1 = 1$ $\quad D \leftrightarrow E$

Kerninghan-Lin
Beispiel



- 1. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	2	-1	1	-3	-3

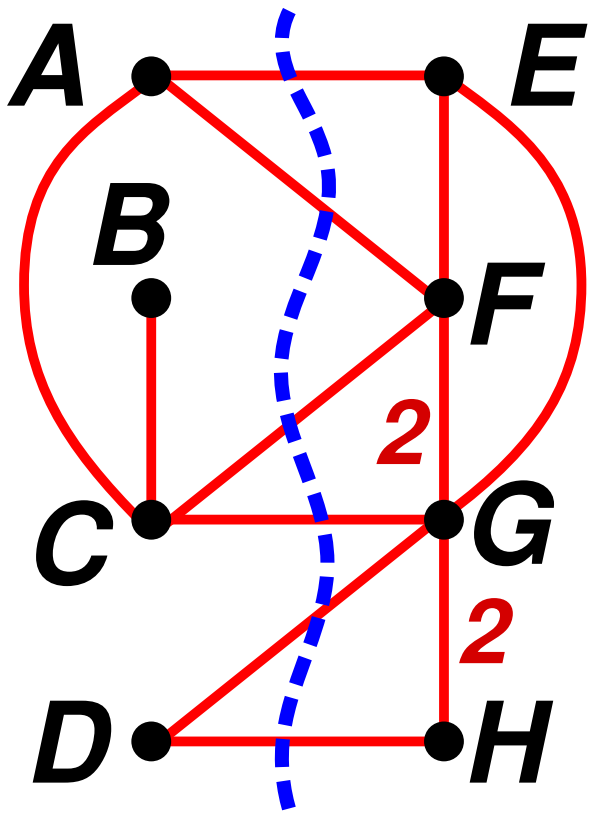
- Bestimmung von Δ_1

	E	F	G	H
A	-2	-2	-2	0
B	-2	-2	-4	-2
C	-1	-3	-5	0
D	1	1	1	-1

$\Delta_1 = 1 \quad D \leftrightarrow E$

- Aktualisierung $D(x)$

Kerninghan-Lin
Beispiel

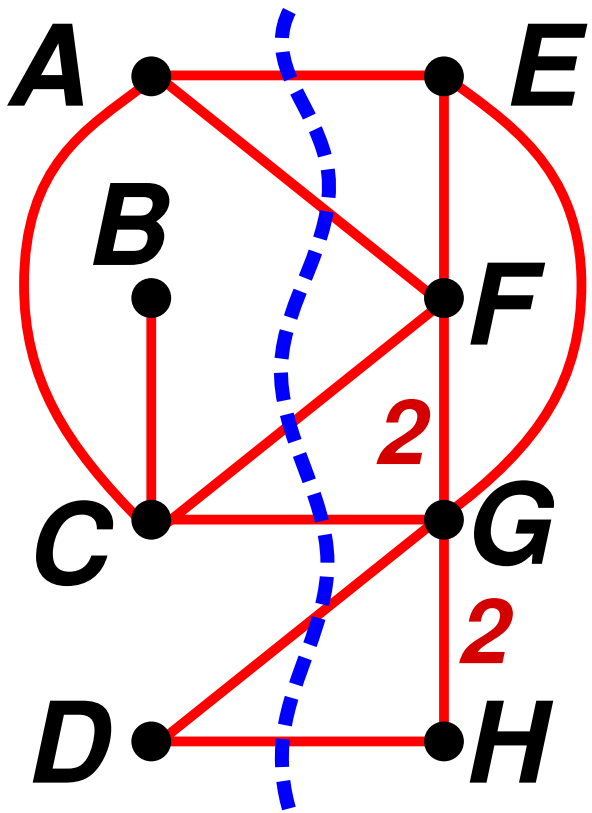


- 1. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	2	-1	1	-3	-3

- Bestimmung von Δ_2

Kerninghan-Lin
Beispiel



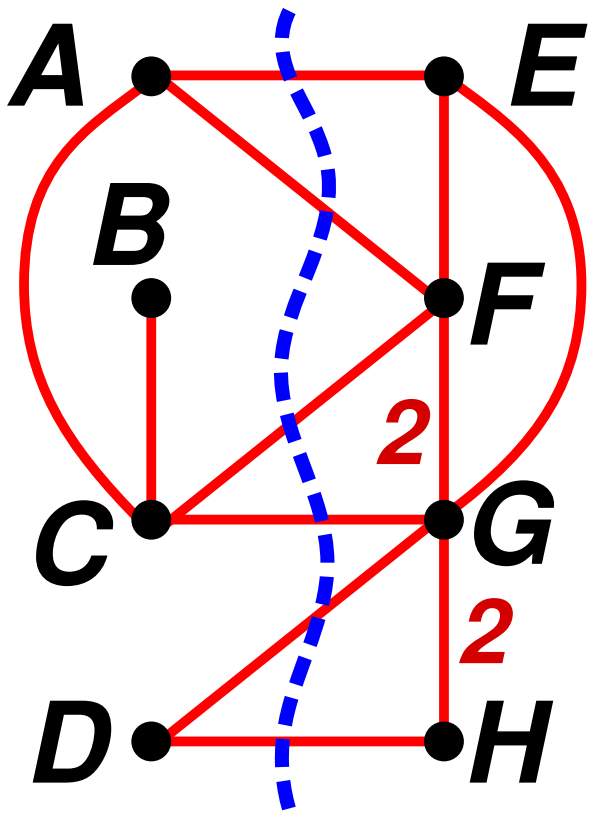
- 1. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	2	-1	1	-3	-3

- Bestimmung von Δ_2

	E	F	G	H
A		-2	-4	-4
B		0	-4	-4
C		-1	-5	-3
D				

Kerninghan-Lin
Beispiel



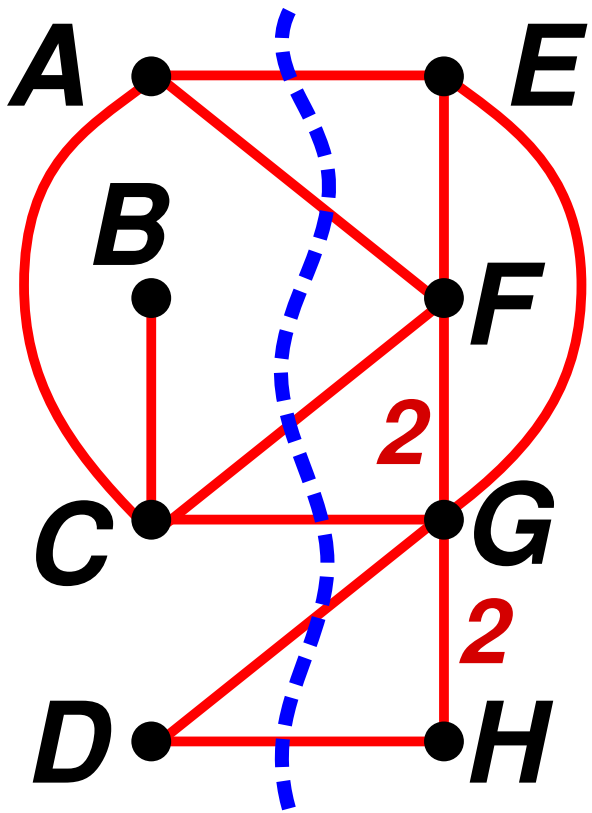
- 1. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	2	-1	1	-3	-3

- Bestimmung von Δ_2

	E	F	G	H
A		-2	-4	-4
B		0	-4	-4
C		-1	-5	-3
D				

Kerninghan-Lin
Beispiel



- 1. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	2	-1	1	-3	-3

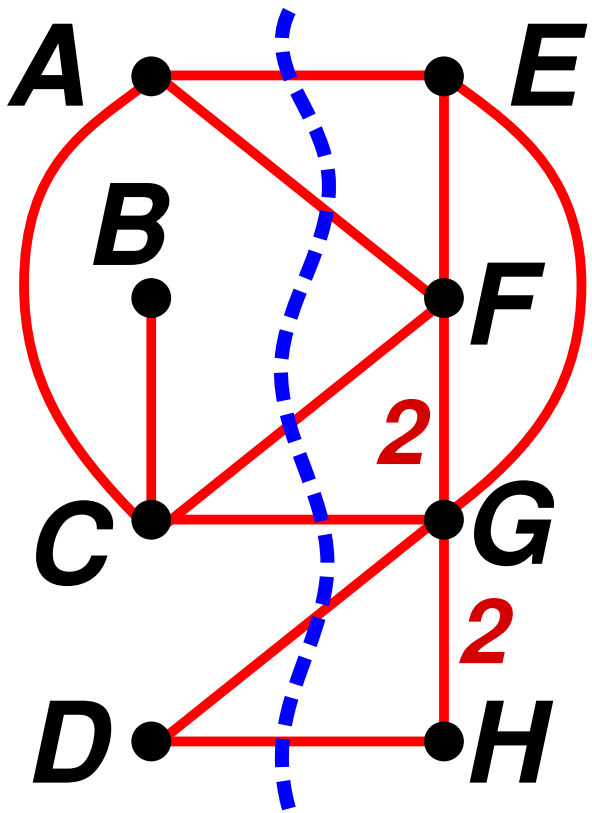
- Bestimmung von Δ_2

	E	F	G	H
A		-2	-4	-4
B		0	-4	-4
C		-1	-5	-3
D				

$\Delta_1 = 1$ D↔E

$\Delta_2 = 0$ B↔F

Kerninghan-Lin
Beispiel



- 1. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-3	-1	0	2	-1	1	-7	-3

- Bestimmung von Δ_2

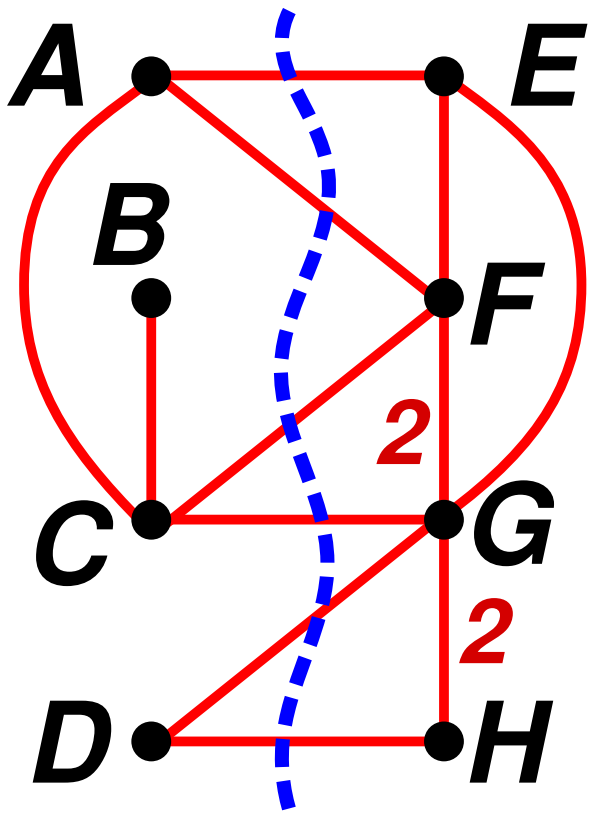
	E	F	G	H
A		-2	-4	-4
B		0	-4	-4
C		-1	-5	-3
D				

$\Delta_1 = 1$ $\quad D \leftrightarrow E$

$\Delta_2 = 0$ $\quad B \leftrightarrow F$

- Aktualisierung $D(x)$

Kerninghan-Lin
Beispiel

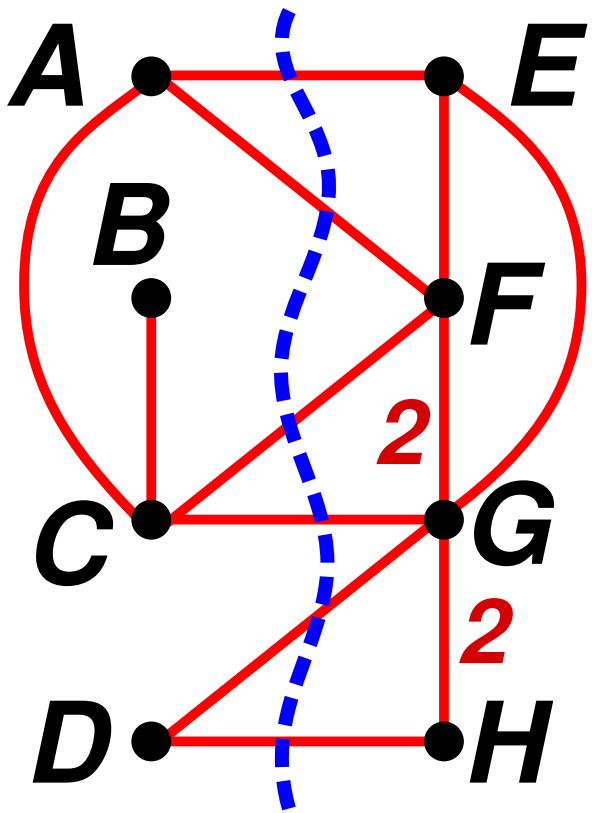


- 1. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-3	-1	0	2	-1	1	-7	-3

- Bestimmung von Δ_3

Kerninghan-Lin
Beispiel



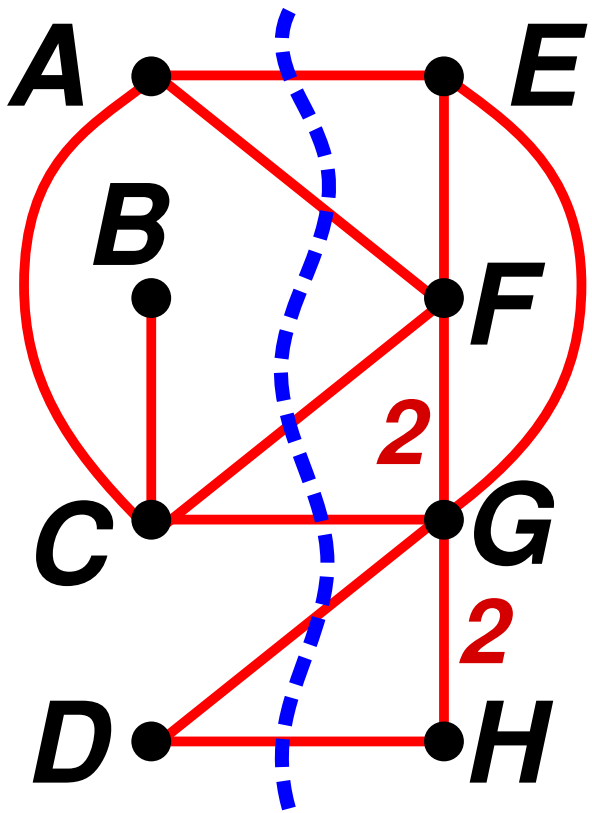
- 1. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-3	-1	0	2	-1	1	-7	-3

- Bestimmung von Δ_3

	E	F	G	H
A			-10	-6
B				
C			-9	-3
D				

Kerninghan-Lin
Beispiel



- 1. Durchlauf
- Bestimmung der $D(x)$

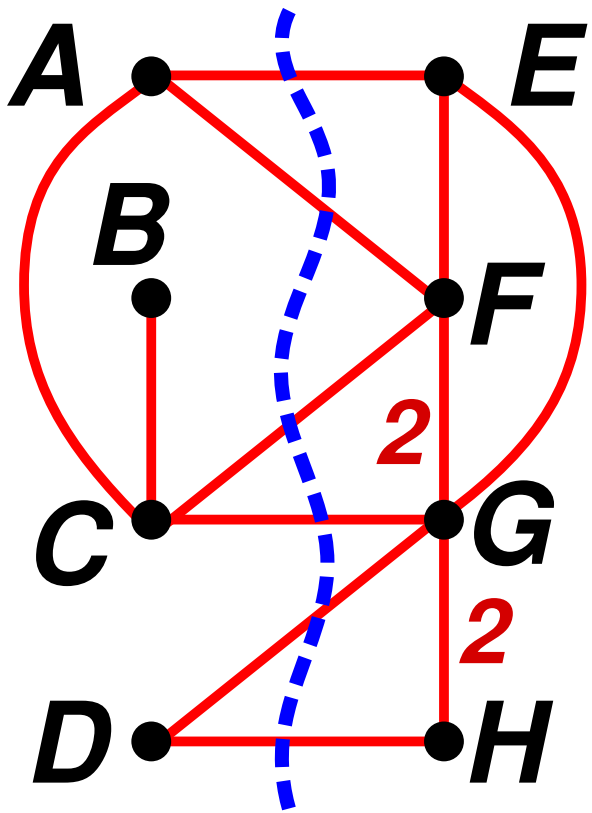
	A	B	C	D	E	F	G	H
$D(x)$	-3	-1	0	2	-1	1	-7	-3

- Bestimmung von Δ_3

	E	F	G	H
A			-10	-6
B				
C			-9	-3
D				

$\Delta_1 = 1$ D$\leftrightarrow$E
$\Delta_2 = 0$ B$\leftrightarrow$F
$\Delta_3 = -3$ C$\leftrightarrow$H

Kerninghan-Lin
Beispiel



- 1. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	2	-1	1	-5	-3

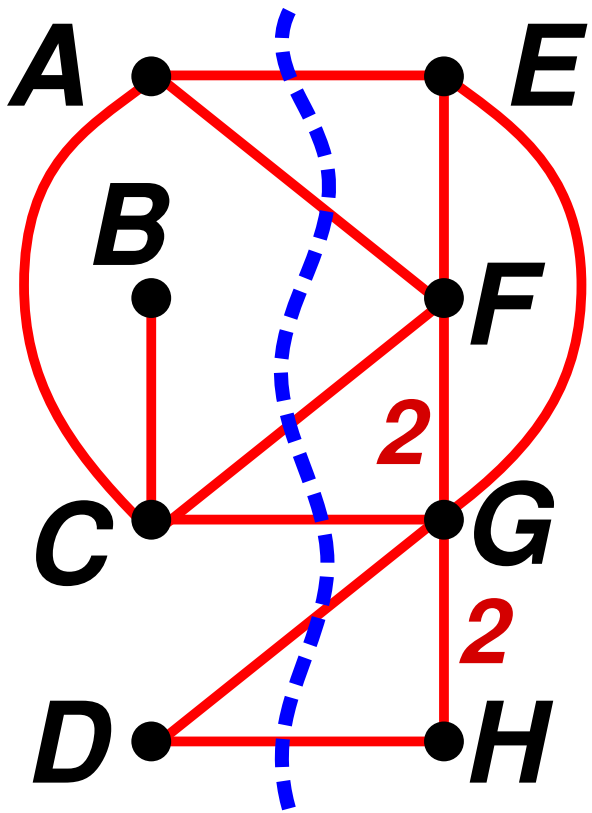
- Bestimmung von Δ_3

	E	F	G	H
A			-10	-6
B				
C			-9	-3
D				

$\Delta_1 = 1$ D↔E
$\Delta_2 = 0$ B↔F
$\Delta_3 = -3$ C↔H

- Aktualisierung $D(x)$

Kerninghan-Lin
Beispiel

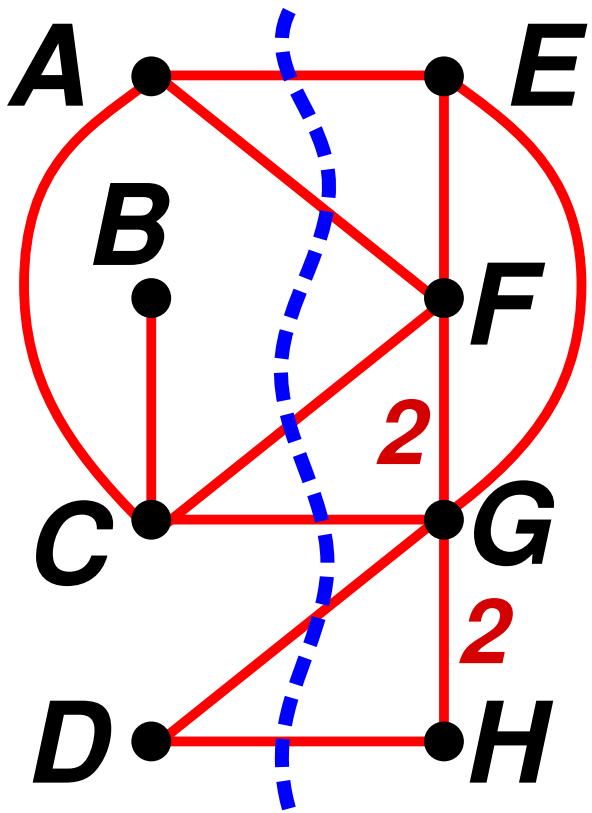


- 1. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	2	-1	1	-5	-3

- Bestimmung von Δ_4

Kerninghan-Lin
Beispiel



- 1. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	2	-1	1	-5	-3

- Bestimmung von Δ_4

	E	F	G	H
A			-6	
B				
C				
D				

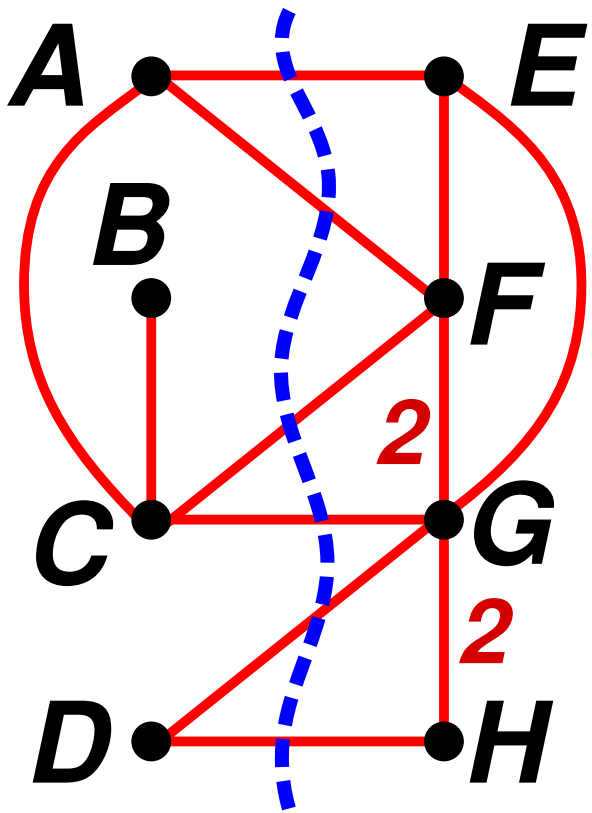
$\Delta_1 = 1$ $\quad D \leftrightarrow E$

$\Delta_2 = 0$ $\quad B \leftrightarrow F$

$\Delta_3 = -3$ $\quad C \leftrightarrow H$

$\Delta_4 = -6$ $\quad A \leftrightarrow G$

Kerninghan-Lin
Beispiel



- 1. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	2	-1	1	-5	-3

- Bestimmung von Δ_4

	E	F	G	H
A			-6	
B				
C				
D				

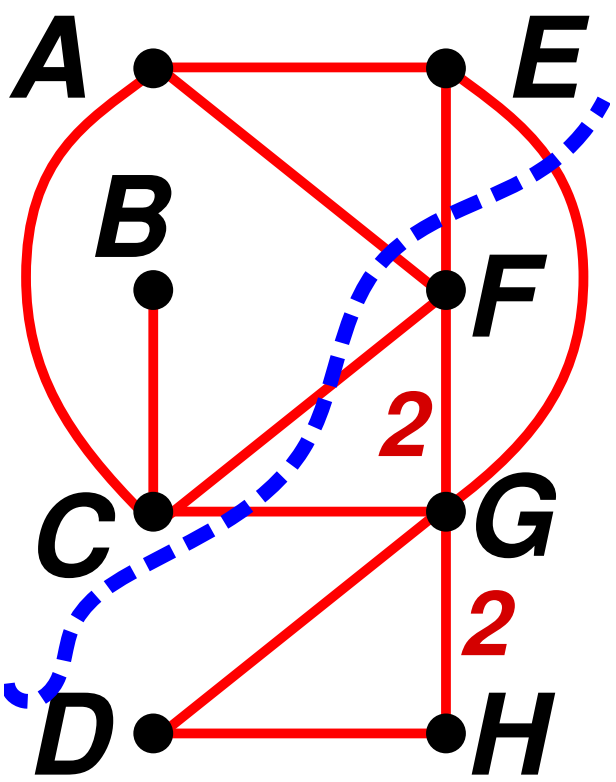
$\Delta_1 = 1$ $\quad D \leftrightarrow E$

$\Delta_2 = 0$ $\quad B \leftrightarrow F$

$\Delta_3 = -3$ $\quad C \leftrightarrow H$

$\Delta_4 = -6$ $\quad A \leftrightarrow G$

Kerninghan-Lin
Beispiel



- 1. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	2	-1	1	-5	-3

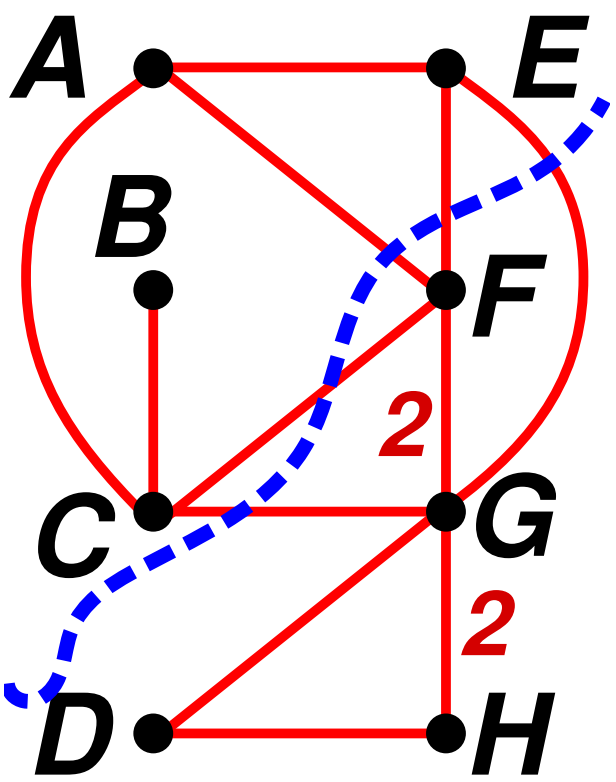
- Bestimmung von Δ_4

	E	F	G	H
A			-6	
B				
C				
D				

$\Delta_1 = 1$ D↔E
$\Delta_2 = 0$ B↔F
$\Delta_3 = -3$ C↔H
$\Delta_4 = -6$ A↔G

- $\sum_i^k \Delta_i$ wird maximal bei $k = 1$
  $\Rightarrow$ Nur D und E austauschen

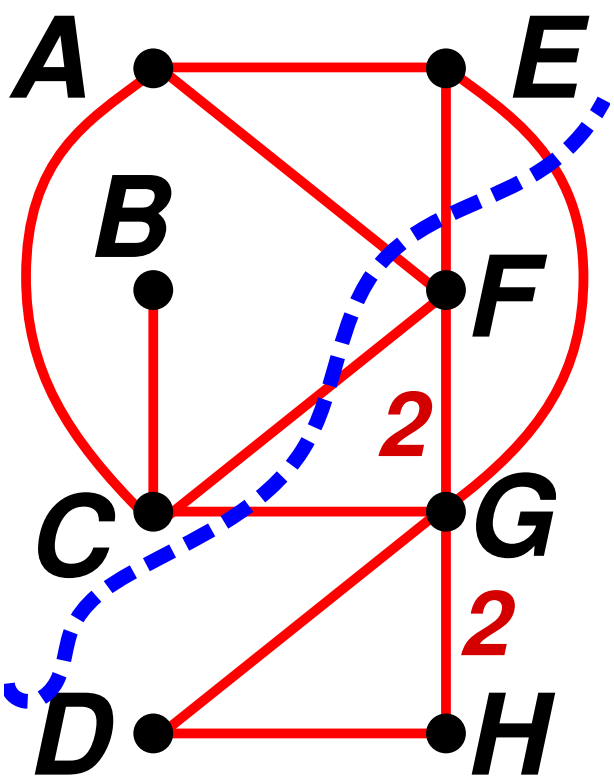
Kerninghan-Lin
Beispiel



- 2. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	-2	1	1	-5	-3

Kerninghan-Lin
Beispiel

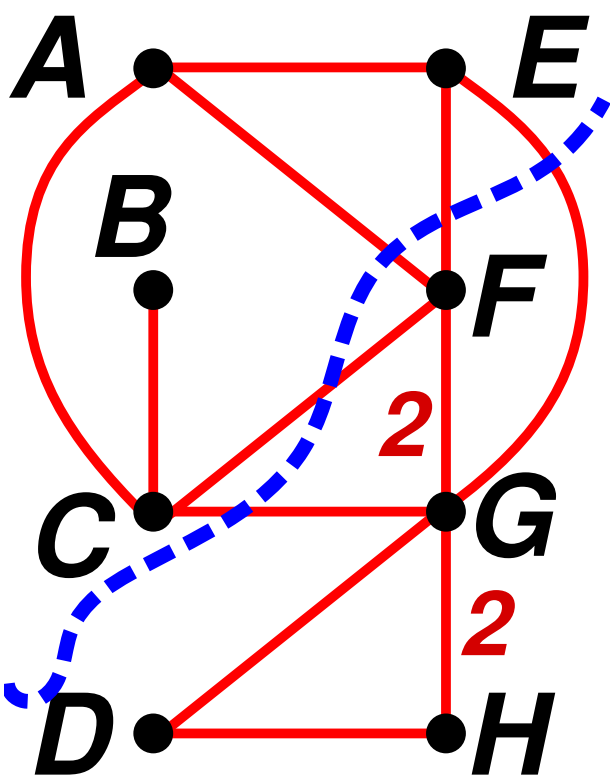


- 2. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	-2	1	1	-5	-3

- Bestimmung von Δ_1

Kerninghan-Lin
Beispiel



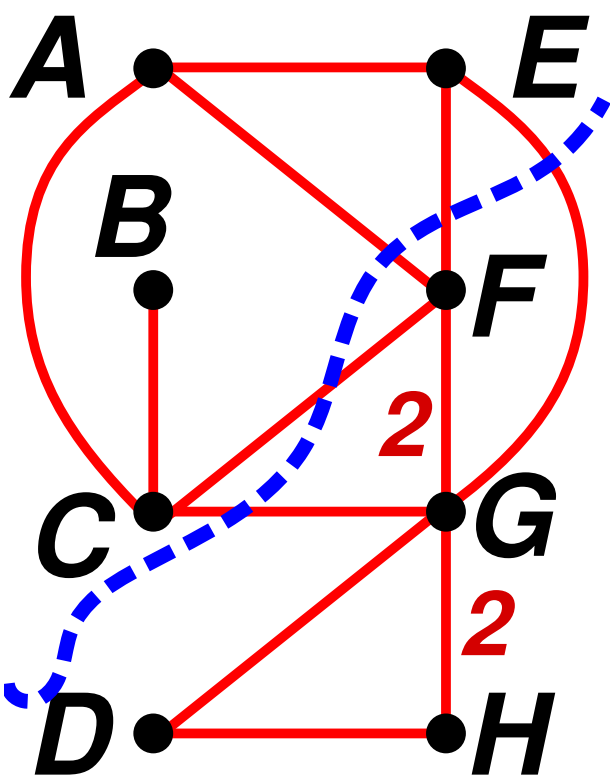
- 2. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	-2	1	1	-5	-3

- Bestimmung von Δ_1

	D	F	G	H
A	-3	-2	-5	-4
B	-3	0	-5	-4
C	-2	-1	-6	-3
E	-1	0	-5	-2

Kerninghan-Lin
Beispiel



- 2. Durchlauf
- Bestimmung der $D(x)$

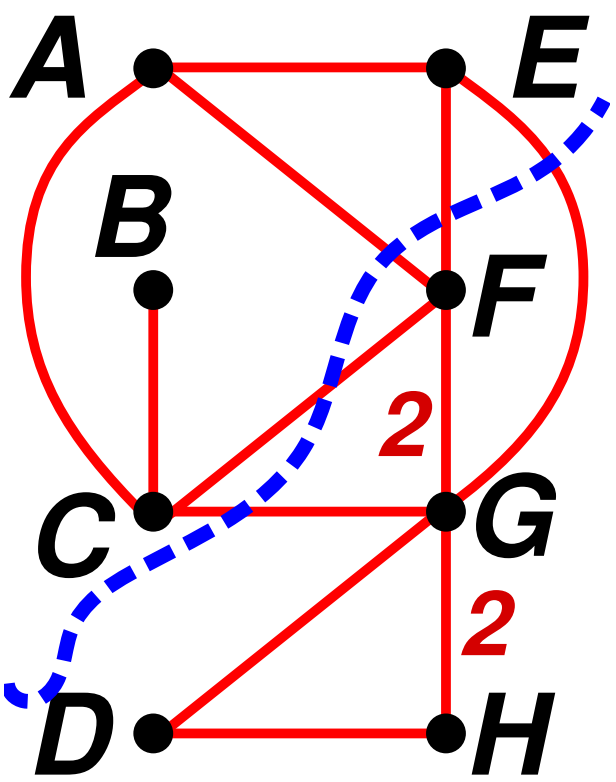
	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	-2	1	1	-5	-3

- Bestimmung von Δ_1

	D	F	G	H
A	-3	-2	-5	-4
B	-3	0	-5	-4
C	-2	-1	-6	-3
E	-1	0	-5	-2

$\Delta_1 = 0 \quad B \leftrightarrow F$

Kerninghan-Lin
Beispiel



- 2. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-3	-1	0	-2	-1	1	1	-3

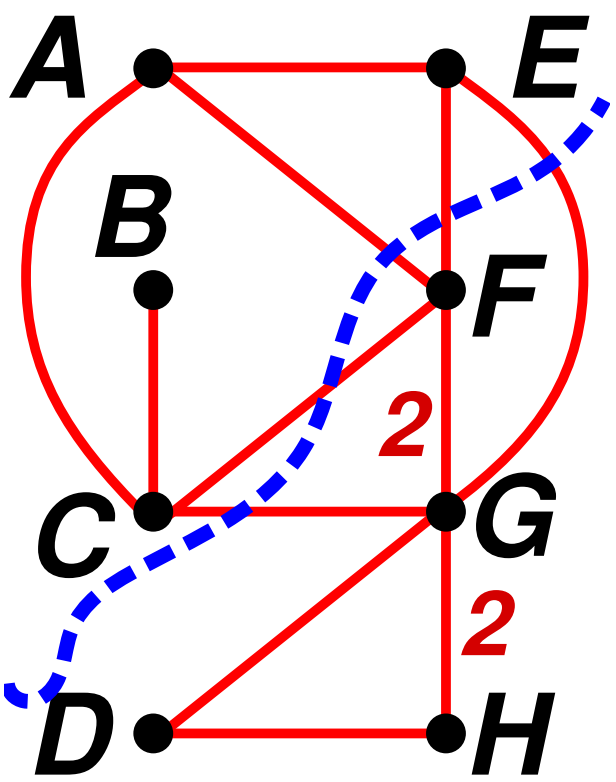
- Bestimmung von Δ_1

	D	F	G	H
A	-3	-2	-5	-4
B	-3	0	-5	-4
C	-2	-1	-6	-3
E	-1	0	-5	-2

$\Delta_1 = 0 \quad B \leftrightarrow F$

- Aktualisierung $D(x)$

Kerninghan-Lin
Beispiel

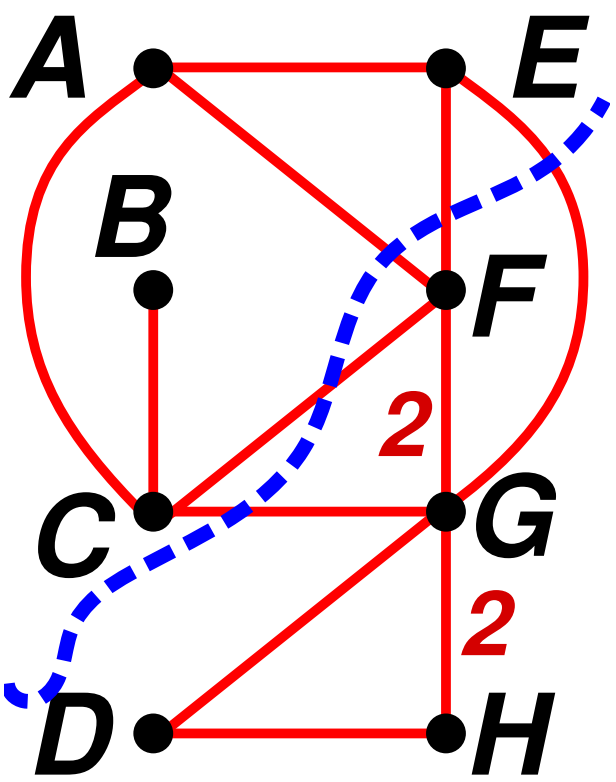


- 2. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-3	-1	0	-2	-1	1	1	-3

- Bestimmung von Δ_2

Kerninghan-Lin
Beispiel



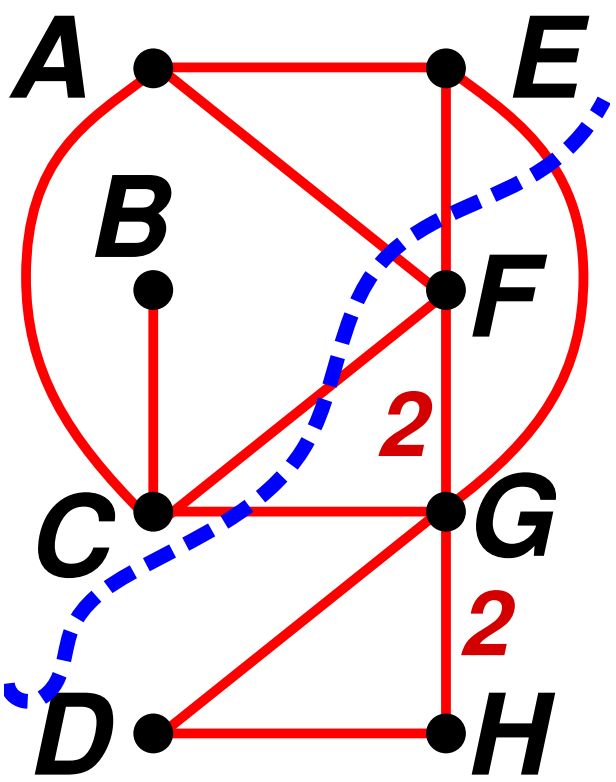
- 2. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-3	-1	0	-2	-1	1	1	-3

- Bestimmung von Δ_2

	D	F	G	H
A	-5		-3	-6
B				
C	-2		-1	-3
E	-3		-2	-4

Kerninghan-Lin
Beispiel



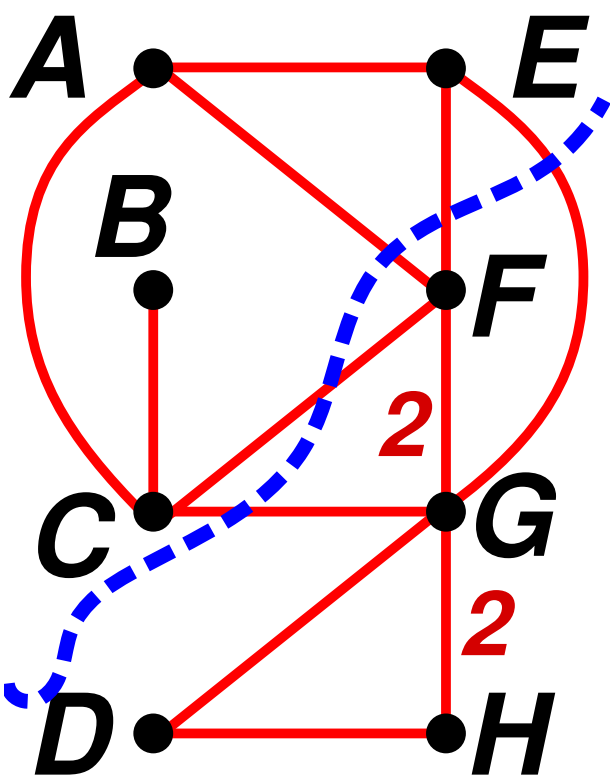
- 2. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-3	-1	0	-2	-1	1	1	-3

- Bestimmung von Δ_2

	D	F	G	H
A	-5		-3	-6
B				
C	-2		-1	-3
E	-3		-2	-4

Kerninghan-Lin
Beispiel



- 2. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-3	-1	0	-2	-1	1	1	-3

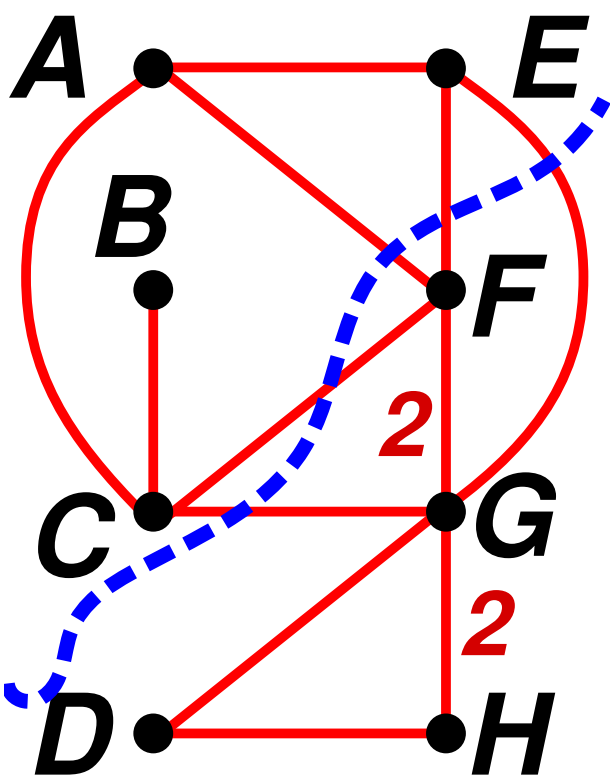
- Bestimmung von Δ_2

	D	F	G	H
A	-5		-3	-6
B				
C	-2		-1	-3
E	-3		-2	-4

$\Delta_1 = 0$ B↔F

$\Delta_2 = -1$ C↔G

Kerninghan-Lin
Beispiel



- 2. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	0	-3	1	1	1

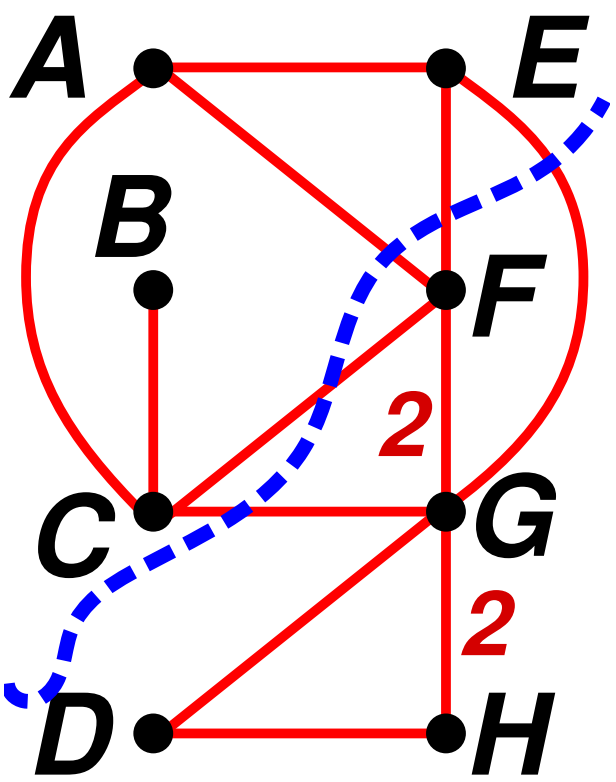
- Bestimmung von Δ_2

	D	F	G	H
A	-5		-3	-6
B				
C	-2		-1	-3
E	-3		-2	-4

$\Delta_1 = 0$ B$\leftrightarrow$F
$\Delta_2 = -1$ C$\leftrightarrow$G

- Aktualisierung $D(x)$

Kerninghan-Lin
Beispiel

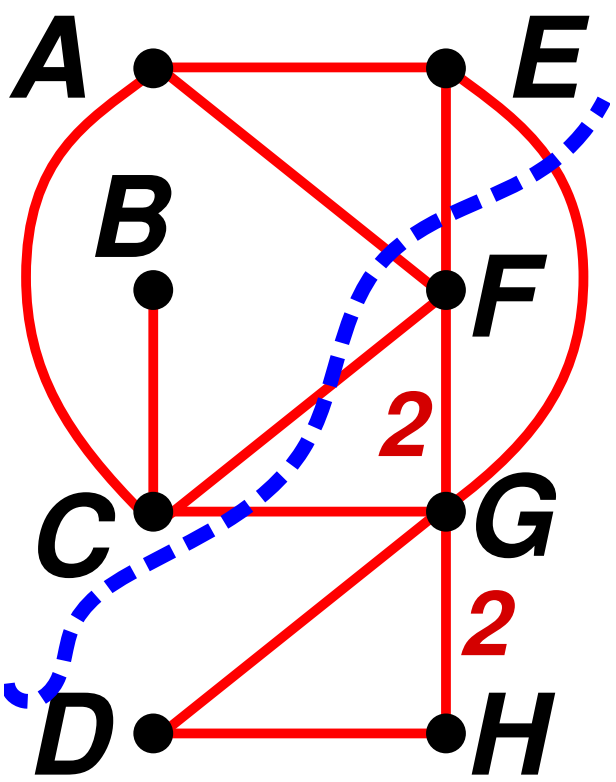


- 2. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	0	-3	1	1	1

- Bestimmung von Δ_3

Kerninghan-Lin
Beispiel



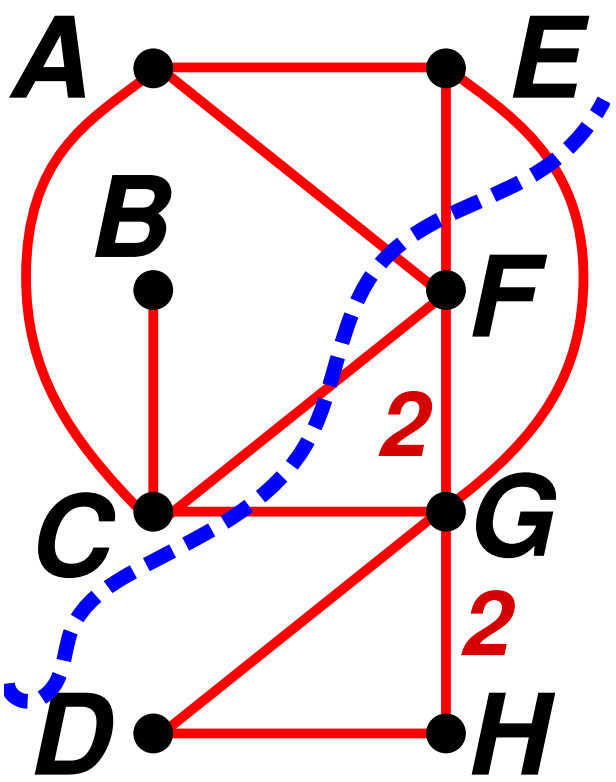
- 2. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	0	-3	1	1	1

- Bestimmung von Δ_3

	D	F	G	H
A	-1			0
B				
C				
E	-3			-2

Kerninghan-Lin
Beispiel



- 2. Durchlauf
- Bestimmung der $D(x)$

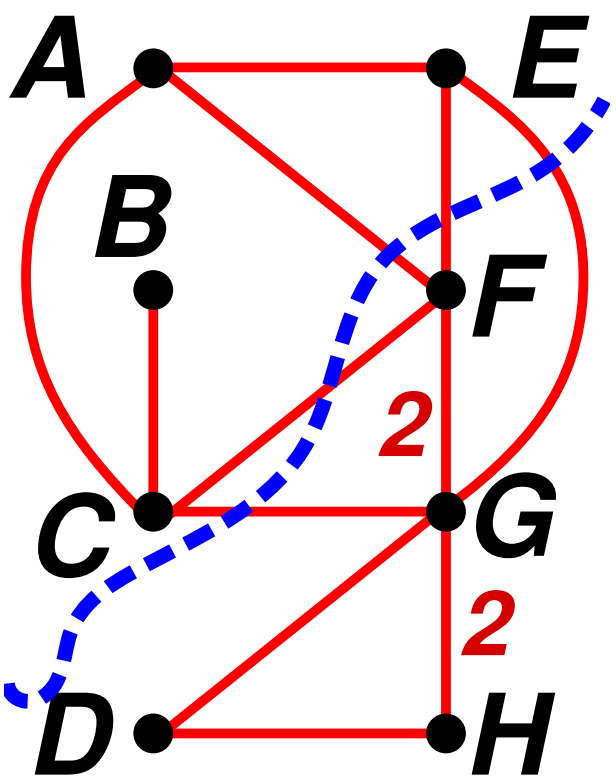
	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	0	-3	1	1	1

- Bestimmung von Δ_3

	D	F	G	H
A	-1			0
B				
C				
E	-3			-2

$\Delta_1 = 0$ B$\leftrightarrow$F
$\Delta_2 = -1$ C$\leftrightarrow$G
$\Delta_3 = 0$ A$\leftrightarrow$H

Kerninghan-Lin
Beispiel



- 2. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	2	-1	1	1	1

- Bestimmung von Δ_3

	D	F	G	H
A	-1			0
B				
C				
E	-3			-2

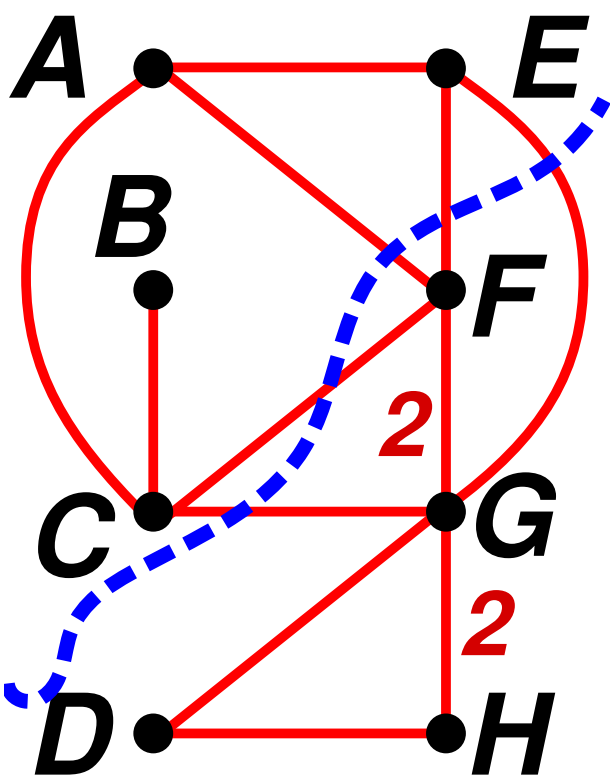
$\Delta_1 = 0$ $\quad B \leftrightarrow F$

$\Delta_2 = -1$ $\quad C \leftrightarrow G$

$\Delta_3 = 0$ $\quad A \leftrightarrow H$

- Aktualisierung $D(x)$

Kerninghan-Lin
Beispiel



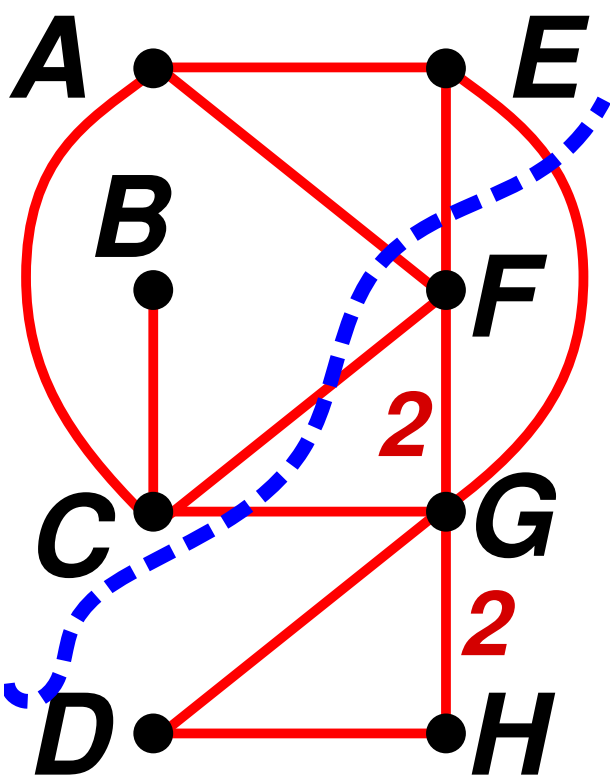
- 2. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	2	-1	1	1	1

- Bestimmung von Δ_4

	E	F	G	H
A				
B				
C				
D	1			

Kerninghan-Lin
Beispiel



- 2. Durchlauf
- Bestimmung der $D(x)$

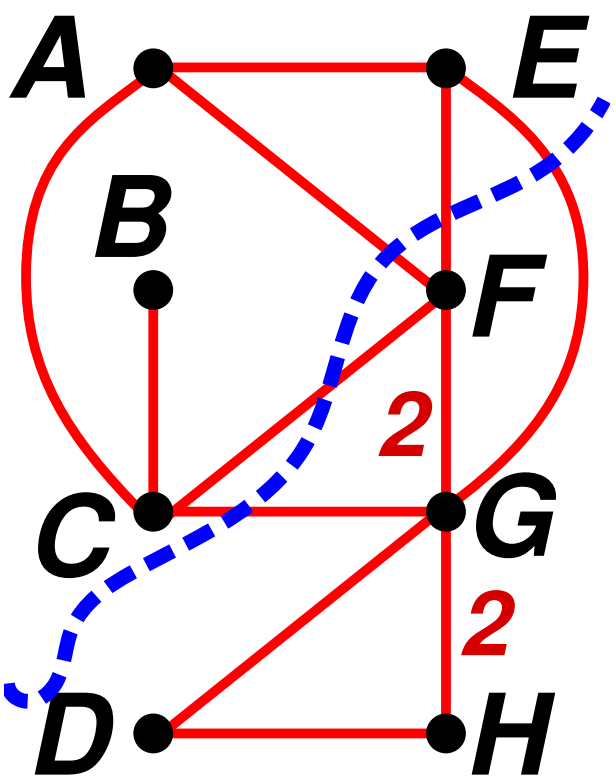
	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	2	-1	1	1	1

- Bestimmung von Δ_4

	E	F	G	H
A				
B				
C				
D	1			

$\Delta_1 = 0$ B$\leftrightarrow$F
$\Delta_2 = -1$ C$\leftrightarrow$G
$\Delta_3 = 0$ A$\leftrightarrow$H
$\Delta_4 = -1$ E$\leftrightarrow$D

Kerninghan-Lin
Beispiel



- 2. Durchlauf
- Bestimmung der $D(x)$

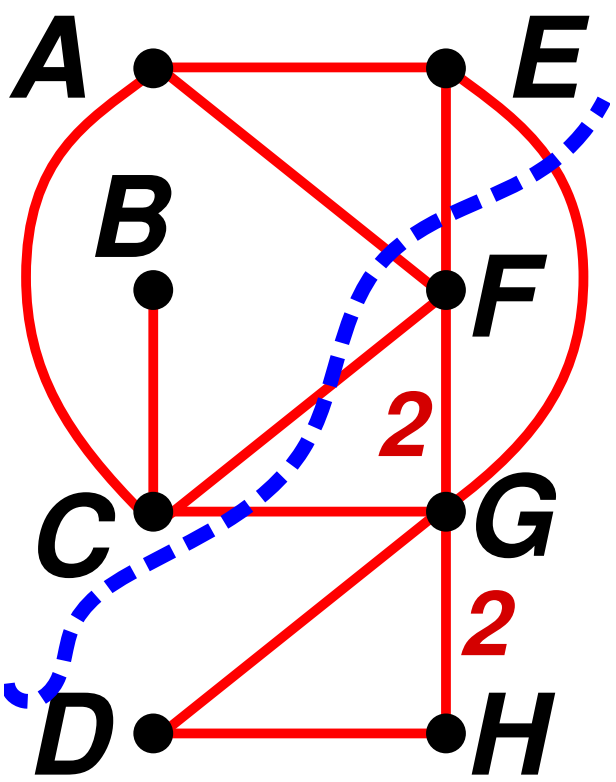
	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	2	-1	1	1	1

- Bestimmung von Δ_4

	E	F	G	H
A				
B				
C				
D	1			

$\Delta_1 = 0$ B$\leftrightarrow$F
$\Delta_2 = -1$ C$\leftrightarrow$G
$\Delta_3 = 0$ A$\leftrightarrow$H
$\Delta_4 = -1$ E$\leftrightarrow$D

Kerninghan-Lin
Beispiel



- 2. Durchlauf
- Bestimmung der $D(x)$

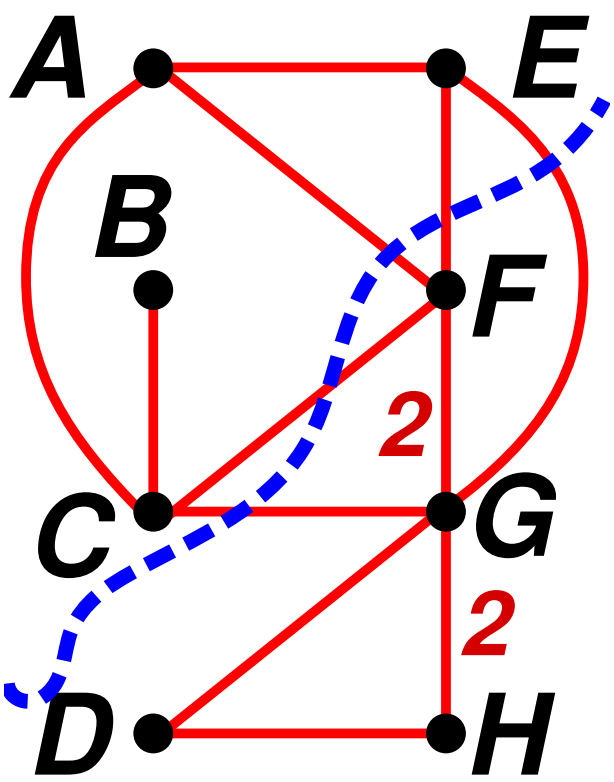
	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	2	-1	1	1	1

- Bestimmung von Δ_4

	E	F	G	H
A				
B				
C				
D	1			

$\Delta_1 = 0$ B$\leftrightarrow$F
$\Delta_2 = -1$ C$\leftrightarrow$G
$\Delta_3 = 0$ A$\leftrightarrow$H
$\Delta_4 = -1$ E$\leftrightarrow$D

Kerninghan-Lin
Beispiel



- 2. Durchlauf
- Bestimmung der $D(x)$

	A	B	C	D	E	F	G	H
$D(x)$	-1	-1	0	2	-1	1	1	1

- Bestimmung von

	E	F	G	H
A				
B				
C				
D	1			

$\Delta_1 = 0$ B↔F
$\Delta_2 = -1$ C↔G
$\Delta_3 = 0$ A↔H
$\Delta_4 = -1$ E↔D

- Maximale $\sum_i^k \Delta_i \leq 0$
  $\Rightarrow$ Keine weitere Verbesserung möglich

Lösung: Partitionierung zu Beginn des Durchlaufes

Kerninghan-Lin
Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Lokale Suche mit variabler Nachbarschaft

Kerninghan-Lin
Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Lokale Suche mit variabler Nachbarschaft
- Komplexität: $\mathcal{O}(n^3)$ pro Durchlauf

Kerninghan-Lin
Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Lokale Suche mit variabler Nachbarschaft
- Komplexität: $\mathcal{O}(n^3)$ pro Durchlauf
- Nachteile:

Kerninghan-Lin
Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Lokale Suche mit variabler Nachbarschaft
- Komplexität: $\mathcal{O}(n^3)$ pro Durchlauf
- Nachteile:
  - Kein Hypergraph

Kerninghan-Lin
Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Lokale Suche mit variabler Nachbarschaft
- Komplexität: $\mathcal{O}(n^3)$ pro Durchlauf
- Nachteile:
  - Kein Hypergraph
  - Exakte Bisektion
    $\Rightarrow$ Partitionen müssen gleich groß sein
    ggf. Dummy-Knoten einfügen

Kerninghan-Lin
Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Lokale Suche mit variabler Nachbarschaft
- Komplexität: $\mathcal{O}(n^3)$ pro Durchlauf
- Nachteile:
  - Kein Hypergraph
  - Exakte Bisektion
    $\Rightarrow$ Partitionen müssen gleich groß sein
    ggf. Dummy-Knoten einfügen
  - Unterstützt keine Knotengewichte

Kerninghan-Lin
Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Lokale Suche mit variabler Nachbarschaft
- Komplexität: $\mathcal{O}(n^3)$ pro Durchlauf
- Nachteile:
  - Kein Hypergraph
  - Exakte Bisektion
    $\Rightarrow$ Partitionen müssen gleich groß sein
    ggf. Dummy-Knoten einfügen
  - Unterstützt keine Knotengewichte
- Allgemeiner: k-Wege Partitionierung (bisher $k = 2$)

Kerninghan-Lin
Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Lokale Suche mit variabler Nachbarschaft
- Komplexität: $\mathcal{O}(n^3)$ pro Durchlauf
- Nachteile:
  - Kein Hypergraph
  - Exakte Bisektion
    $\Rightarrow$ Partitionen müssen gleich groß sein
    ggf. Dummy-Knoten einfügen
  - Unterstützt keine Knotengewichte
- Allgemeiner: k-Wege Partitionierung (bisher $k = 2$)
  1. Partitionieren des Graph in k gleich große Mengen

Kerninghan-Lin
Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Lokale Suche mit variabler Nachbarschaft
- Komplexität: $\mathcal{O}(n^3)$ pro Durchlauf
- Nachteile:
  - Kein Hypergraph
  - Exakte Bisektion
    $\Rightarrow$ Partitionen müssen gleich groß sein
    ggf. Dummy-Knoten einfügen
  - Unterstützt keine Knotengewichte
- Allgemeiner: k-Wege Partitionierung (bisher $k = 2$)
  1. Partitionieren des Graph in k gleich große Mengen
  2. KL auf jedes Teilmengenpaar anwenden

Fiduccia-Mattheyses-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

- 1982
- Arbeitet mit Hypergraphen (Multi-Pin-Netze)
  $\Rightarrow$ Reduziert Netzkosten, nicht Kantenkosten
- Partitionen können verschieden Größe haben
  Nur eine ca.-Verhältnisvorgabe für die beiden Partitionen
- Immer nur eine Zelle auf einmal bewegen (d.h. kein Austausch)
- Sehr schnell

Fiduccia-Mattheyses-Algorithmus
Notation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gegeben: Netze 1, ... , N, Zellen 1, ... , C

Fiduccia-Mattheyses-Algorithmus
Notation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gegeben: Netze $1, \ldots, N$, Zellen $1, \ldots, C$

  $n(i)$: Anzahl Zellen von Netz i

Fiduccia-Mattheyses-Algorithmus
Notation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gegeben: Netze $1, \ldots, N$, Zellen $1, \ldots, C$

  $n(i)$: Anzahl Zellen von Netz i

  $s(i)$: Größe von Zelle i

  $|x_1, x_x, \ldots| := \sum_i x_i$

  $S_{max} := max_i s(i)$

Fiduccia-Mattheyses-Algorithmus
Notation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gegeben: Netze $1, \ldots, N$, Zellen $1, \ldots, C$

  $n(i)$: Anzahl Zellen von Netz i

  $s(i)$: Größe von Zelle i

  $|x_1, x_x, \ldots| := \sum_i x_i$

  $S_{max} := max_i s(i)$

  $p(i)$: Anzahl Pins von Zelle i

  $P := \sum_i p(i)$ Anzahl aller Pins

Fiduccia-Mattheyses-Algorithmus
Notation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gegeben: Netze $1, \ldots, N$, Zellen $1, \ldots, C$

  $n(i)$: Anzahl Zellen von Netz i

  $s(i)$: Größe von Zelle i

  $|x_1, x_x, \ldots| := \sum_i x_i$

  $S_{max} := max_i s(i)$

  $p(i)$: Anzahl Pins von Zelle i

  $P := \sum_i p(i)$ Anzahl aller Pins

- Zu gegebenen Verhältnis $0 < r < 1$, partitioniert FM den Graph $G = (V, E)$ in Knotensets A und B, so dass ...

Fiduccia-Mattheyses-Algorithmus
Notation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gegeben: Netze $1, \ldots, N$, Zellen $1, \ldots, C$

  $n(i)$: Anzahl Zellen von Netz i

  $s(i)$: Größe von Zelle i

  $|x_1, x_x, \ldots| := \sum_i x_i$

  $S_{max} := max_i s(i)$

  $p(i)$: Anzahl Pins von Zelle i

  $P := \sum_i p(i)$ Anzahl aller Pins

- Zu gegebenen Verhältnis $0 < r < 1$, partitioniert FM den Graph $G = (V, E)$ in Knotensets A und B, so dass ...
  - die Schnittkosten minimiert werden

Fiduccia-Mattheyses-Algorithmus
Notation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gegeben: Netze $1, \ldots, N$, Zellen $1, \ldots, C$

  $n(i)$: Anzahl Zellen von Netz i

  $s(i)$: Größe von Zelle i

  $|x_1, x_x, \ldots| := \sum_i x_i$

  $S_{max} := max_i s(i)$

  $p(i)$: Anzahl Pins von Zelle i

  $P := \sum_i p(i)$ Anzahl aller Pins

- Zu gegebenen Verhältnis $0 < r < 1$, partitioniert FM den Graph $G = (V, E)$ in Knotensets A und B, so dass ...
  - die Schnittkosten minimiert werden
  - $\frac{|A|}{|A|+|B|} \approx r$

Fiduccia-Mattheyses-Algorithmus
Ideen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Schnittkosten: Kosten der Netze in der Schnittmenge (Menge der Netze die beide Partitionen verbinden)

Fiduccia-Mattheyses-Algorithmus
Ideen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Schnittkosten: Kosten der Netze in der Schnittmenge (Menge der Netze die beide Partitionen verbinden)
- Zellgewinnwert (gain) $g(v)$ einer Zelle v: Veränderung der Schnittkosten beim Wechsel der Zelle

Fiduccia-Mattheyses-Algorithmus
Ideen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Schnittkosten: Kosten der Netze in der Schnittmenge (Menge der Netze die beide Partitionen verbinden)
- Zellgewinnwert (gain) $g(v)$ einer Zelle v: Veränderung der Schnittkosten beim Wechsel der Zelle
- $-p(v) \leq g(v) \leq p(v)$

Fiduccia-Mattheyses-Algorithmus
Ideen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Schnittkosten: Kosten der Netze in der Schnittmenge (Menge der Netze die beide Partitionen verbinden)
- Zellgewinnwert (gain) $g(v)$ einer Zelle v: Veränderung der Schnittkosten beim Wechsel der Zelle
- $-p(v) \leq g(v) \leq p(v)$
- Nur eine Zelle bewegen

Fiduccia-Mattheyses-Algorithmus
Ideen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Schnittkosten: Kosten der Netze in der Schnittmenge (Menge der Netze die beide Partitionen verbinden)
- Zellgewinnwert (gain) $g(v)$ einer Zelle v: Veränderung der Schnittkosten beim Wechsel der Zelle
- $-p(v) \leq g(v) \leq p(v)$
- Nur eine Zelle bewegen
- Versuchen balanciert zu bleiben
  (inkl. etwas *Spiel*) $\Rightarrow$ max. Partitionsgröße

$$r|V| - S_{max} \leq |A| \leq r|V| + S_{max}$$

Fiduccia-Mattheyses-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

```
FiducciaMattheyses() (V,E,r,n,s,): begin
    Init(A_0, B_0) ;
    m := 0 ;
    repeat
        Entsperre alle Knoten ;
        for (i:=1; i ≤ |V| ; i++) do
            Finde freies v_i ∈ A_m ∪ B_m mit
              maximalem Gain, dass die Größen-
              bedingungen nicht verletzt ;
            g_i := gain(v_i) ;
            WechselVorläufig(v_i) ;
            Sperre v_i ;
        Finde k mit G := Σ_{i=1}^{k} g_i das maximal ist ;
        if G > 0 then
            MacheWechselPermanent(v_1, ... , v_k) ;
            VerwerfeWechsel(v_{k+1}, ... , v_{|V|}) ;
            m := m+1 ;
    until G ≤ 0;
```

Initialisierung

Startmengen A_0 und B_0 erzeugen (zufällig)

Fiduccia-Mattheyses-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

```
FiducciaMattheyses() (V,E,r,n,s,): begin
    Init(A_0, B_0) ;
    m := 0 ;
    repeat
        Entsperre alle Knoten ;
        for (i:=1; i ≤ |V| ; i++) do
            Finde freies v_i ∈ A_m ∪ B_m mit
              maximalem Gain, dass die Größen-
              bedingungen nicht verletzt ;
            g_i := gain(v_i) ;
            WechselVorläufig(v_i) ;
            Sperre v_i ;
        Finde k mit G := Σ_{i=1}^{k} g_i das maximal ist ;
        if G > 0 then
            MacheWechselPermanent(v_1, ... , v_k) ;
            VerwerfeWechsel(v_{k+1}, ... , v_{|V|}) ;
            m := m+1 ;
    until G ≤ 0;
```

Freie und gesperrte Knoten

- Einmal zum Wechsel ausgewählte Zellen werden gesperrt
- Erst nächste Iteration (*Pass*) wieder verfügbar

Fiduccia-Mattheyses-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

```
FiducciaMattheyses() (V,E,r,n,s,): begin
    Init(A_0, B_0) ;
    m := 0 ;
    repeat
        Entsperre alle Knoten ;
        for (i:=1; i ≤ |V| ; i++) do
            Finde freies v_i ∈ A_m ∪ B_m mit
              maximalem Gain, dass die Größen-
              bedingungen nicht verletzt ;
            g_i := gain(v_i) ;
            WechselVorläufig(v_i) ;
            Sperre v_i ;
        Finde k mit G := Σ_{i=1}^{k} g_i das maximal ist ;
        if G > 0 then
            MacheWechselPermanent(v_1, ... , v_k) ;
            VerwerfeWechsel(v_{k+1}, ... , v_{|V|}) ;
            m := m+1 ;
    until G ≤ 0;
```

Wechselkandiaten auswählen

- Zelle v so wählen, dass
  - v nicht gesperrt ist
  - Ein Wechsel von v die Größenbedingung nicht verletzt
  - Zellgewinnwert von v maximal ist
- Wechsel nur vorläufig

Fiduccia-Mattheyses-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

```
FiducciaMattheyses() (V,E,r,n,s,): begin
    Init(A_0, B_0) ;
    m := 0 ;
    repeat
        Entsperre alle Knoten ;
        for (i:=1; i ≤ |V| ; i++) do
            Finde freies v_i ∈ A_m ∪ B_m mit
              maximalem Gain, dass die Größen-
              bedingungen nicht verletzt ;
            g_i := gain(v_i) ;
            WechselVorläufig(v_i) ;
            Sperre v_i ;
        Finde k mit G := Σ_{i=1}^{k} g_i das maximal ist ;
        if G > 0 then
            MacheWechselPermanent(v_1, ... , v_k) ;
            VerwerfeWechsel(v_{k+1}, ... , v_{|V|}) ;
            m := m+1 ;
    until G ≤ 0;
```

Auswahl der zu wechselnden Knoten

- Greedy Verfahren um tatsächliche Wechsel zu wählen
- g_i kann negativ werden
- $\sum g_i$ kann zeitweise auch negativ sein
  - Bei dicht verbundenen Teilmengen
  - Keine Verbesserung beim Wechsel von Einzelknoten
  - Erst bei Wechsel der gesamten Teilmenge

Fiduccia-Mattheyses-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

```
FiducciaMattheyses() (V,E,r,n,s,): begin
    Init(A_0, B_0) ;
    m := 0 ;
    repeat
        Entsperre alle Knoten ;
        for (i:=1; i ≤ |V| ; i++) do
            Finde freies v_i ∈ A_m ∪ B_m mit
              maximalem Gain, dass die Größen-
              bedingungen nicht verletzt ;
            g_i := gain(v_i) ;
            WechselVorläufig(v_i) ;
            Sperre v_i ;
        Finde k mit G := Σ_{i=1}^{k} g_i das maximal ist ;
        if G > 0 then
            MacheWechselPermanent(v_1, ... , v_k) ;
            VerwerfeWechsel(v_{k+1}, ... , v_{|V|}) ;
            m := m+1 ;
    until G ≤ 0;
```

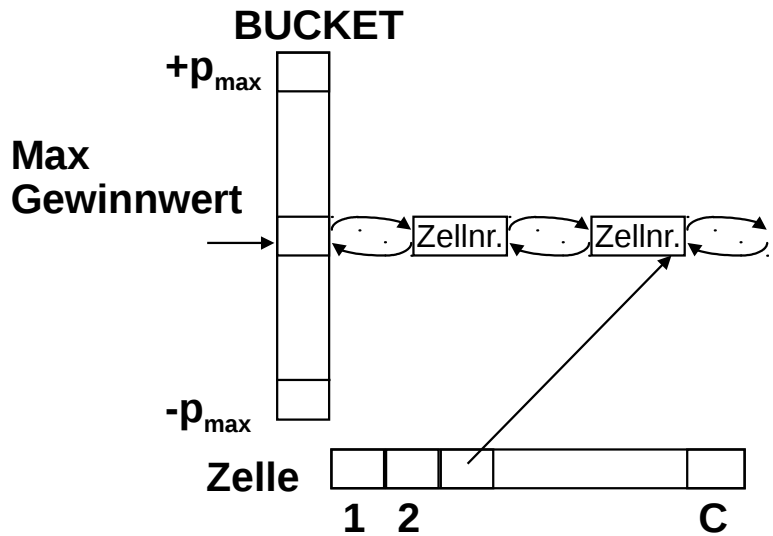
Wechsel

- Wenn Gewinn gewählten Wechsel durchführen
- Bis kein Gewinn mehr möglich

FM-Implementierungsdetails
Datenstrukturen

TECHNISCHE UNIVERSITÄT DARMSTADT

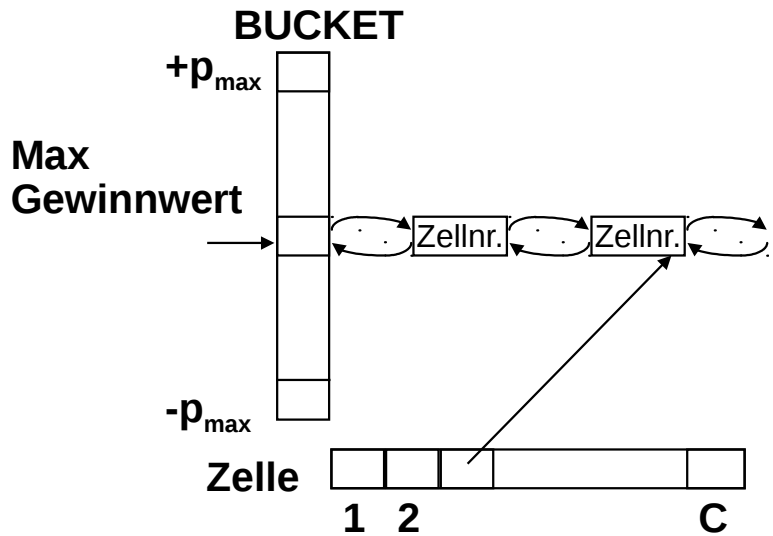
- Zwei Bucket-List Strukturen, für jede Partition eine



FM-Implementierungsdetails
Datenstrukturen

TECHNISCHE UNIVERSITÄT DARMSTADT

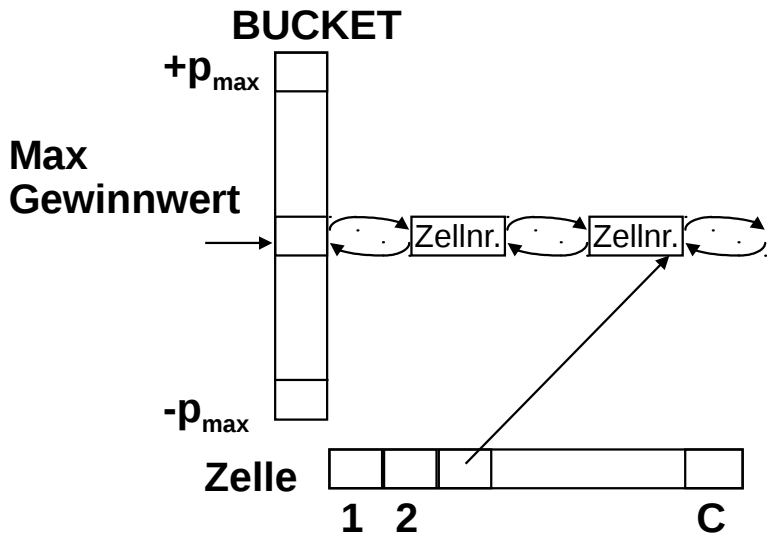
- Zwei Bucket-List Strukturen, für jede Partition eine
- Erlaubt in $\mathcal{O}(1)$:



FM-Implementierungsdetails
Datenstrukturen

TECHNISCHE UNIVERSITÄT DARMSTADT

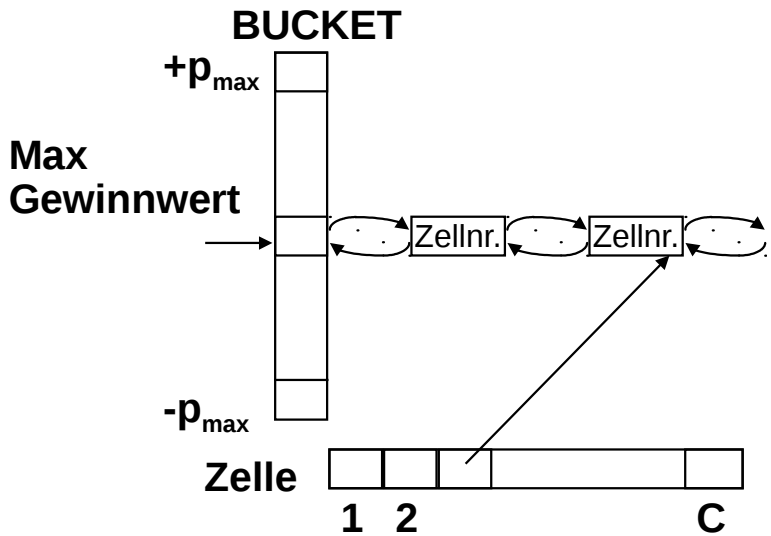
- Zwei Bucket-List Strukturen, für jede Partition eine
- Erlaubt in $\mathcal{O}(1)$:
  - Zelle mit maximalem Gewinnwert finden



FM-Implementierungsdetails
Datenstrukturen

TECHNISCHE UNIVERSITÄT DARMSTADT

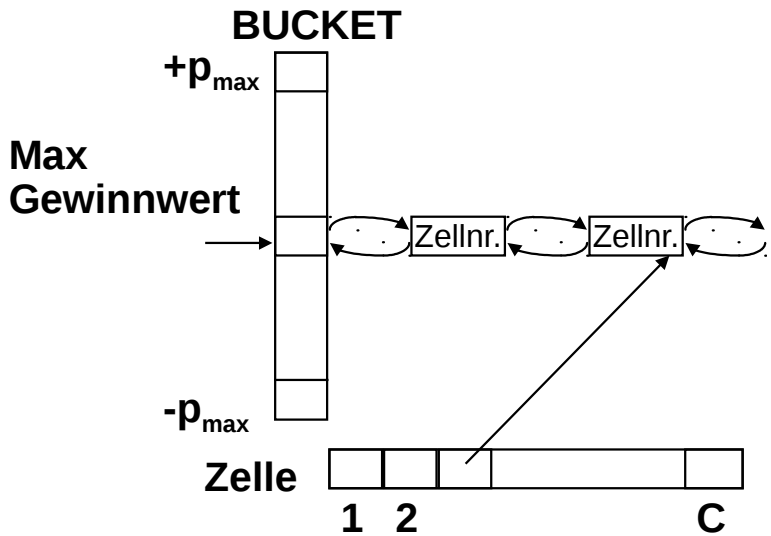
- Zwei Bucket-List Strukturen, für jede Partition eine
- Erlaubt in $\mathcal{O}(1)$:
  - Zelle mit maximalem Gewinnwert finden
  - Aktualisieren des max. Gewinnwert-Zeigers



FM-Implementierungsdetails
Datenstrukturen

TECHNISCHE UNIVERSITÄT DARMSTADT

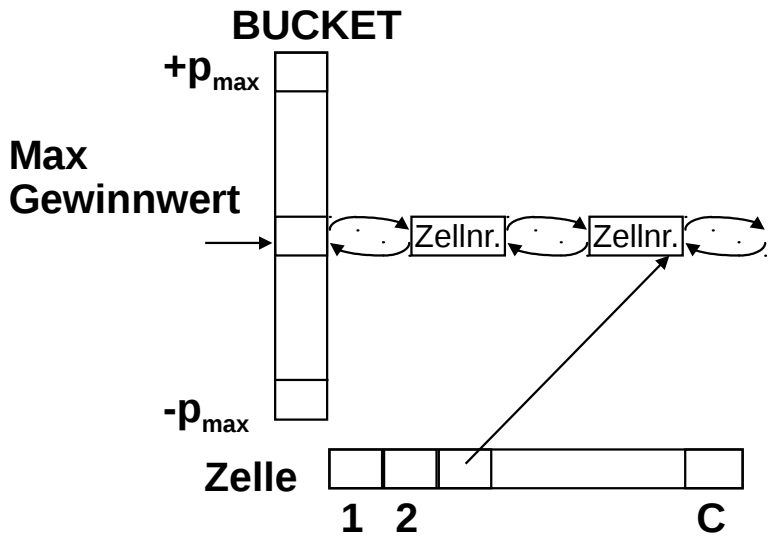
- Zwei Bucket-List Strukturen, für jede Partition eine
- Erlaubt in $\mathcal{O}(1)$:
  - Zelle mit maximalem Gewinnwert finden
  - Aktualisieren des max. Gewinnwert-Zeigers
  - Zelle *v* löschen



FM-Implementierungsdetails
Datenstrukturen

TECHNISCHE UNIVERSITÄT DARMSTADT

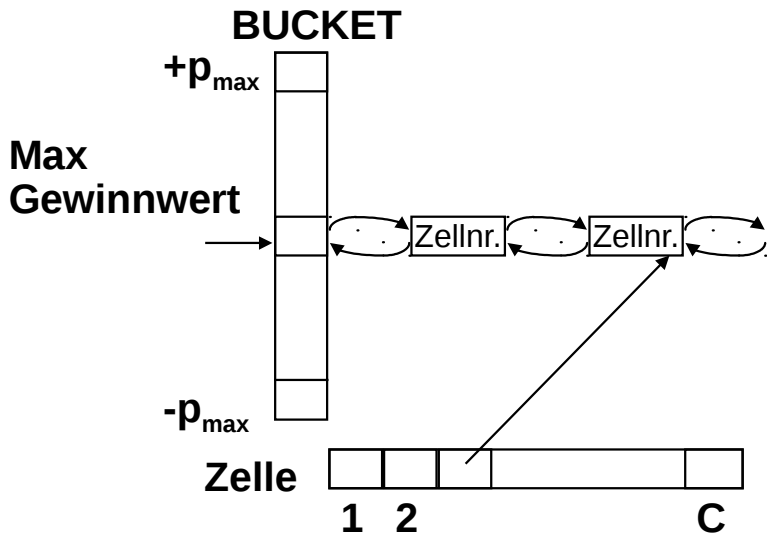
- Zwei Bucket-List Strukturen, für jede Partition eine
- Erlaubt in $\mathcal{O}(1)$:
  - Zelle mit maximalem Gewinnwert finden
  - Aktualisieren des max. Gewinnwert-Zeigers
  - Zelle *v* löschen
  - Zelle *v* einfügen



FM-Implementierungsdetails
Datenstrukturen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Zwei Bucket-List Strukturen, für jede Partition eine
- Erlaubt in $\mathcal{O}(1)$:
  - Zelle mit maximalem Gewinnwert finden
  - Aktualisieren des max. Gewinnwert-Zeigers
  - Zelle v löschen
  - Zelle v einfügen
  - Ändern von $g(v)$



FM-Implementierungsdetails
Kritische Netze

TECHNISCHE UNIVERSITÄT DARMSTADT

- Neuberechnung aller Zellgewinnwerte $\mathcal{O}(C^2)$

FM-Implementierungsdetails
Kritische Netze

TECHNISCHE UNIVERSITÄT DARMSTADT

- Neuberechnung aller Zellgewinnwerte $\mathcal{O}(C^2)$
- Besser:

FM-Implementierungsdetails
Kritische Netze

TECHNISCHE UNIVERSITÄT DARMSTADT

- Neuberechnung aller Zellgewinnwerte $\mathcal{O}(C^2)$
- Besser:
  - Nur bewegte Zellen und mit ihr verbundene neuberechnen!

FM-Implementierungsdetails
Kritische Netze

TECHNISCHE UNIVERSITÄT DARMSTADT

- Neuberechnung aller Zellgewinnwerte $\mathcal{O}(C^2)$
- Besser:
  - Nur bewegte Zellen und mit ihr verbundene neuberechnen!
  - Noch Besser: Dies nur für kritische Netze tun!

FM-Implementierungsdetails
Kritische Netze

TECHNISCHE UNIVERSITÄT DARMSTADT

- Neuberechnung aller Zellgewinnwerte $\mathcal{O}(C^2)$
- Besser:
  - Nur bewegte Zellen und mit ihr verbundene neuberechnen!
  - Noch Besser: Dies nur für kritische Netze tun!
- Kritische Netze: Netze deren Schnittzustand (d.h. geschnitten oder nicht) sich beim Zellbewegen ändert

FM-Implementierungsdetails
Kritische Netze

TECHNISCHE UNIVERSITÄT DARMSTADT

- Neuberechnung aller Zellgewinnwerte $\mathcal{O}(C^2)$
- Besser:
  - Nur bewegte Zellen und mit ihr verbundene neuberechnen!
  - Noch Besser: Dies nur für kritische Netze tun!
- Kritische Netze: Netze deren Schnittzustand (d.h. geschnitten oder nicht) sich beim Zellbewegen ändert
- Zellgewinnwert hängt nur von kritischen Netzen ab.

FM-Implementierungsdetails
Kritische Netze

TECHNISCHE UNIVERSITÄT DARMSTADT

- Neuberechnung aller Zellgewinnwerte $\mathcal{O}(C^2)$
- Besser:
  - Nur bewegte Zellen und mit ihr verbundene neuberechnen!
  - Noch Besser: Dies nur für kritische Netze tun!
- Kritische Netze: Netze deren Schnittzustand (d.h. geschnitten oder nicht) sich beim Zellbewegen ändert
- Zellgewinnwert hängt nur von kritischen Netzen ab.
- Bestimmung ob ein Netz kritisch ist:

FM-Implementierungsdetails
Kritische Netze

TECHNISCHE UNIVERSITÄT DARMSTADT

- Neuberechnung aller Zellgewinnwerte $\mathcal{O}(C^2)$
- Besser:
  - Nur bewegte Zellen und mit ihr verbundene neuberechnen!
  - Noch Besser: Dies nur für kritische Netze tun!
- Kritische Netze: Netze deren Schnittzustand (d.h. geschnitten oder nicht) sich beim Zellbewegen ändert
- Zellgewinnwert hängt nur von kritischen Netzen ab.
- Bestimmung ob ein Netz kritisch ist:
  - Netz *i* ist kritisch, gdw. Anzahl Zellen am Netz in einer Partition 0 oder 1 ist.

FM-Implementierungsdetails
Kritische Netze

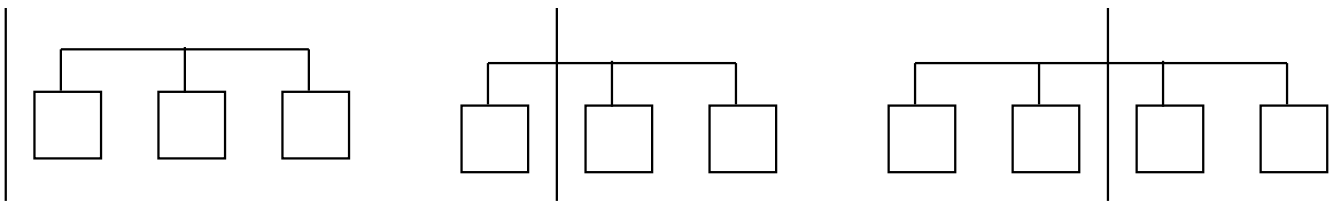
TECHNISCHE UNIVERSITÄT DARMSTADT

- Neuberechnung aller Zellgewinnwerte $\mathcal{O}(C^2)$
- Besser:
  - Nur bewegte Zellen und mit ihr verbundene neuberechnen!
  - Noch Besser: Dies nur für kritische Netze tun!
- Kritische Netze: Netze deren Schnittzustand (d.h. geschnitten oder nicht) sich beim Zellbewegen ändert
- Zellgewinnwert hängt nur von kritischen Netzen ab.
- Bestimmung ob ein Netz kritisch ist:
  - Netz *i* ist kritisch, gdw. Anzahl Zellen am Netz in einer Partition 0 oder 1 ist.

FM-Implementierungsdetails
Kritische Netze

TECHNISCHE UNIVERSITÄT DARMSTADT

- Neuberechnung aller Zellgewinnwerte $\mathcal{O}(C^2)$
- Besser:
  - Nur bewegte Zellen und mit ihr verbundene neuberechnen!
  - Noch Besser: Dies nur für kritische Netze tun!
- Kritische Netze: Netze deren Schnittzustand (d.h. geschnitten oder nicht) sich beim Zellbewegen ändert
- Zellgewinnwert hängt nur von kritischen Netzen ab.
- Bestimmung ob ein Netz kritisch ist:
  - Netz *i* ist kritisch, gdw. Anzahl Zellen am Netz in einer Partition 0 oder 1 ist.



A=0, B=3
kritisch

A=1, B=2
kritisch

A=2, B=2
nicht kritisch

FM-Implementierungsdetails
Berechnung Zellgewinnwerte

TECHNISCHE UNIVERSITÄT DARMSTADT

```
InitGain(Zelle z) begin
    z.gain := 0 ;
    F := FromPartition(z) ;
    T := ToPartition(z) ;
    foreach Netz n verbunden mit z do
        if #Pins von n in F = 1 then
            z.gain++ ;
        if #Pins von n in T = 0 then
            z.gain−− ;
```

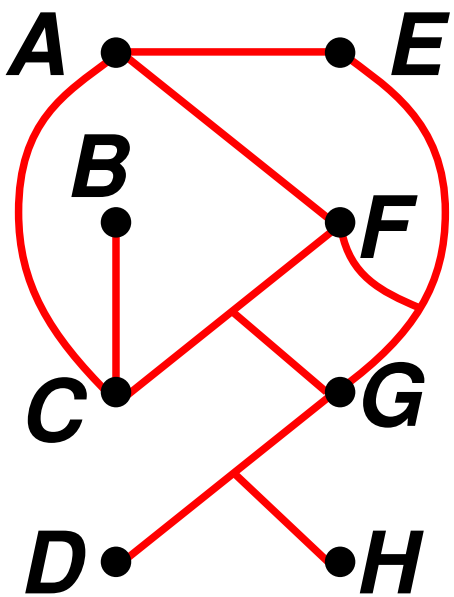
Komplexität: $\mathcal{O}(P)$

```
UpdateGain(Zelle z) begin
    F := FromPartition(z) ;
    T := ToPartition(z) ;
    foreach Netz n verbunden mit z do
        if #Pins von n in T = 0 then
            foreach Freie Zelle a verbunden mit z do
                a.gain++ ;
        if #Pins von n in T = 1 && die Zelle a ist frei then
            a.gain−− ;
        #Pins von n in T++ ; #Pins von n in F– ;
        if #Pins von n in F = 0 then
            foreach Freie Zelle a verbunden mit z do
                a.gain−− ;
        if #Pins von n in F = 1 && die Zelle a ist frei then
            a.gain++ ;
```

Komplexität: $\mathcal{O}(P)$ für einen FM-Durchlauf

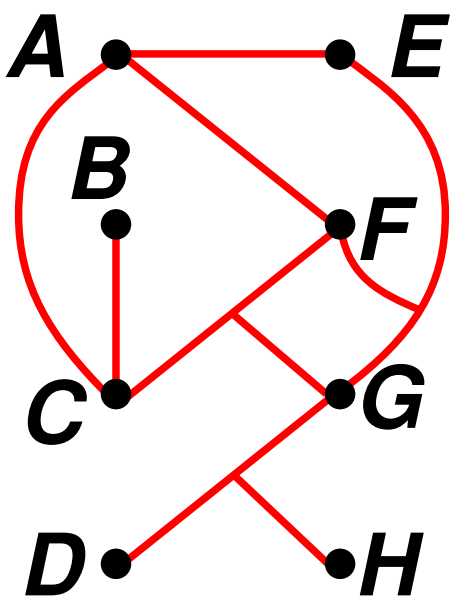
Fiduccia-Mattheyses-Algorithmus
Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT



- $r = \frac{4}{8} = 0.5$
- Zur Vereinfachung im Beispiel:
  - Alle Zellen Einheitsgröße $s = 1$
  - Keine besonderen Datenstrukturen
- Zufällige Startpartionen
  $(\{A, B, C, D\}, \{E, F, G, H\})$
  $\Rightarrow$ Schnittkosten = 4

Fiduccia-Mattheyses-Algorithmus
Beispiel

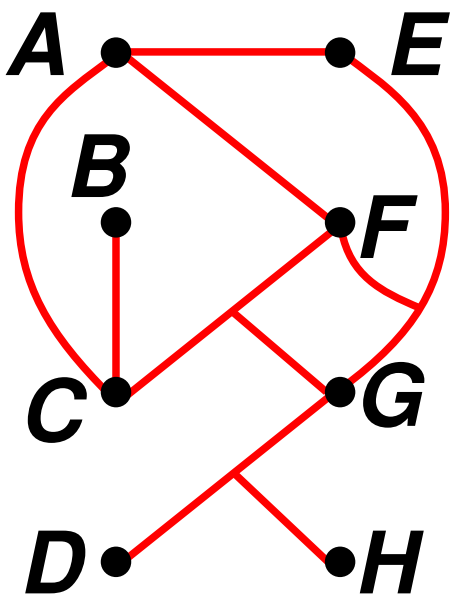


Durchlauf 1

Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$

Fiduccia-Mattheyses-Algorithmus
Beispiel

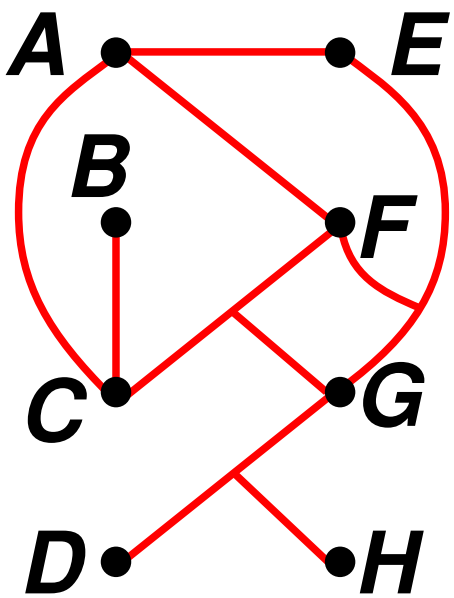


Durchlauf 1

Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$
1	1	-1	-1	-1	0	0	-1	0	1	1

Fiduccia-Mattheyses-Algorithmus
Beispiel

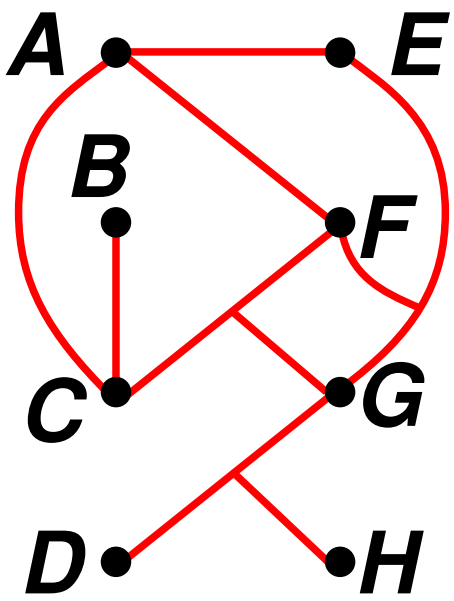


Durchlauf 1

Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$
1	1	-1	-1	-1	0	0	-1	0	1	1
2		-1	1	1	-2	-1	-1	0	0	1

Fiduccia-Mattheyses-Algorithmus
Beispiel

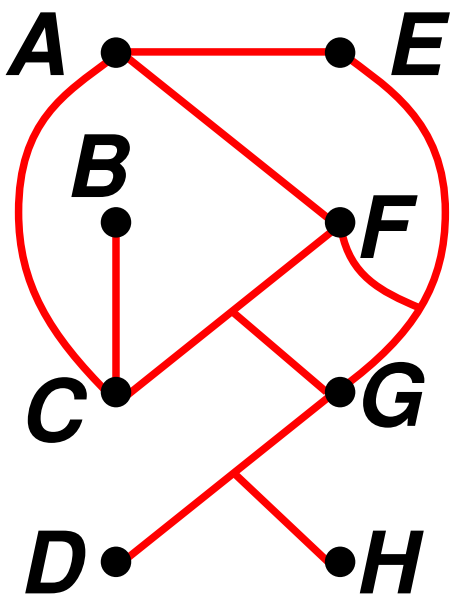


Durchlauf 1

Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$
1	1	-1	-1	-1	0	0	-1	0	1	1
2		-1	1	1	-2	-1	-1	0	0	1
3		0	1	0	-2	-1	-1		1	2

Fiduccia-Mattheyses-Algorithmus
Beispiel

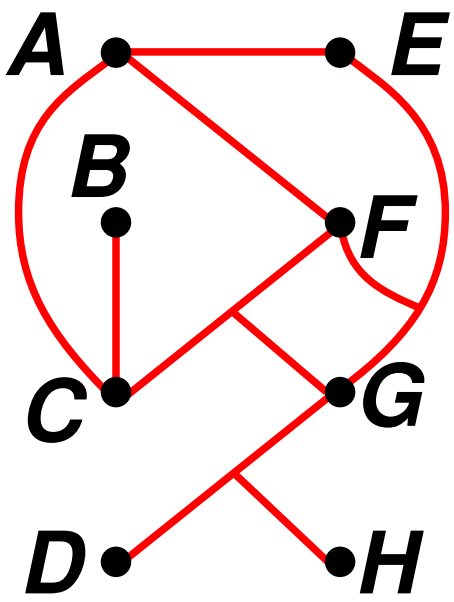


Durchlauf 1

Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$
1	1	-1	-1	-1	0	0	-1	0	1	1
2		-1	1	1	-2	-1	-1	0	0	1
3		0	1	0	-2	-1	-1		1	2
4		1		0	-2	-3	-1		-1	1

Fiduccia-Mattheyses-Algorithmus
Beispiel

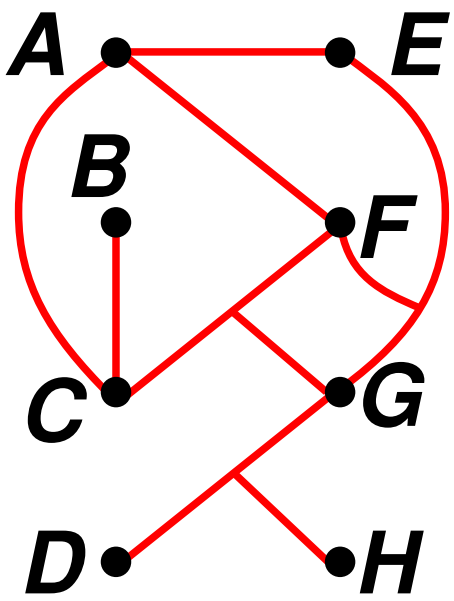


Durchlauf 1

Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$
1	1	-1	-1	-1	0	0	-1	0	1	1
2		-1	1	1	-2	-1	-1	0	0	1
3		0	1	0	-2	-1	-1		1	2
4		1		0	-2	-3	-1		-1	1
5		1		-1	-1	-1			1	2

Fiduccia-Mattheyses-Algorithmus
Beispiel

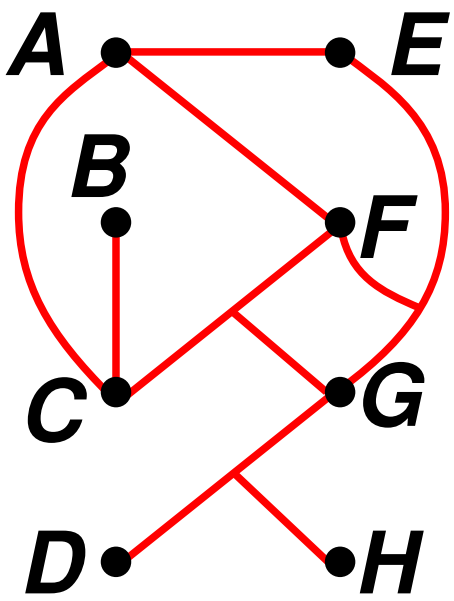


Durchlauf 1

Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$
1	1	-1	-1	-1	0	0	-1	0	1	1
2		-1	1	1	-2	-1	-1	0	0	1
3		0	1	0	-2	-1	-1		1	2
4		1		0	-2	-3	-1		-1	1
5		1		-1	-1	-1			1	2
6				-1	-1	-1			-1	1

Fiduccia-Mattheyses-Algorithmus
Beispiel



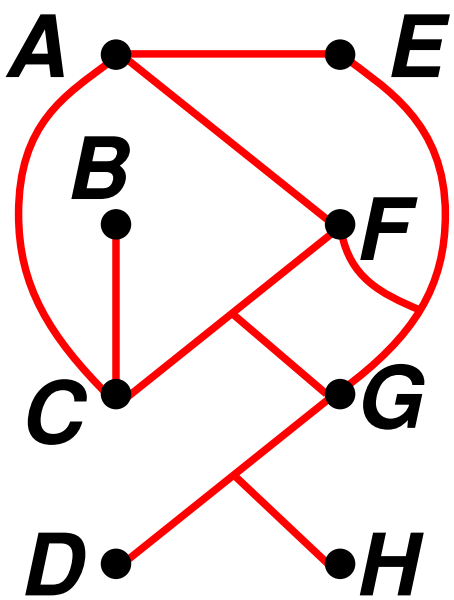
Durchlauf 1

Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$
1	1	-1	-1	-1	0	0	-1	0	1	1
2		-1	1	1	-2	-1	-1	0	0	1
3		0	1	0	-2	-1	-1		1	2
4		1		0	-2	-3	-1		-1	1
5		1		-1	-1	-1			1	2
6				-1	-1	-1			-1	1
7				-1		0			0	1

Fiduccia-Mattheyses-Algorithmus
Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

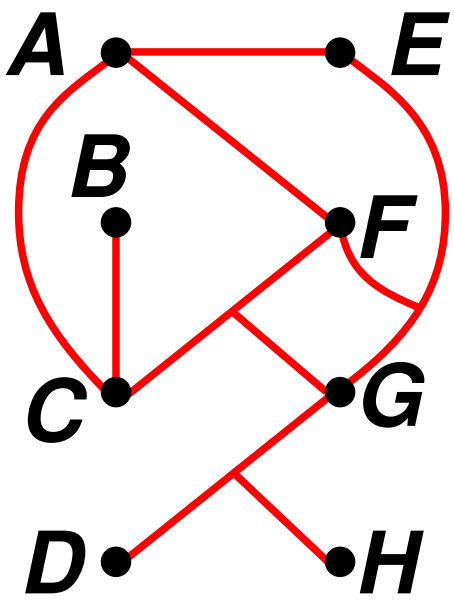


Durchlauf 1

Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$
1	1	-1	-1	-1	0	0	-1	0	1	1
2		-1	1	1	-2	-1	-1	0	0	1
3		0	1	0	-2	-1	-1		1	2
4		1		0	-2	-3	-1		-1	1
5		1		-1	-1	-1			1	2
6				-1	-1	-1			-1	1
7				-1		0			0	1
8				-1					-1	0

Fiduccia-Mattheyses-Algorithmus
Beispiel



Durchlauf 1

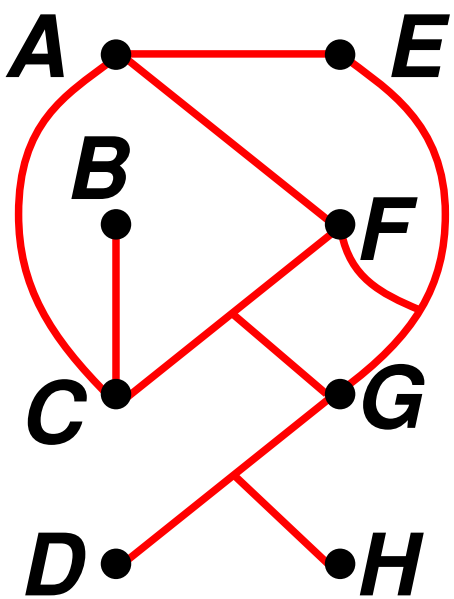
Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$
1	1	-1	-1	-1	0	0	-1	0	1	1
2		-1	1	1	-2	-1	-1	0	0	1
3		0	1	0	-2	-1	-1		1	2
4		1		0	-2	-3	-1		-1	1
5		1		-1	-1	-1			1	2
6				-1	-1	-1			-1	1
7				-1		0			0	1
8				-1					-1	0

$\sum$ maximal bei $i = 3 \Rightarrow$ Zellen A, H, C Wechseln

Fiduccia-Mattheyses-Algorithmus
Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

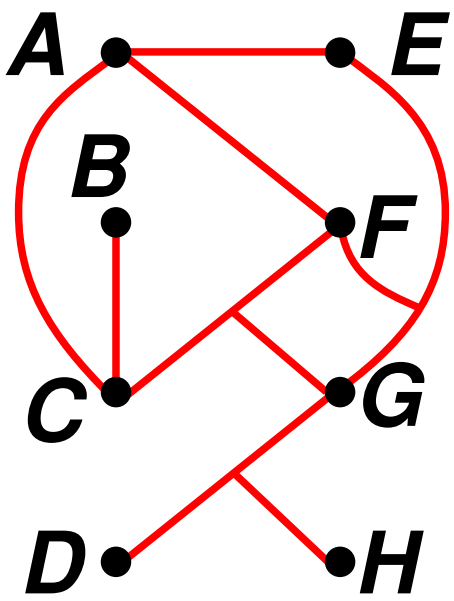


Durchlauf 2

Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$

Fiduccia-Mattheyses-Algorithmus
Beispiel

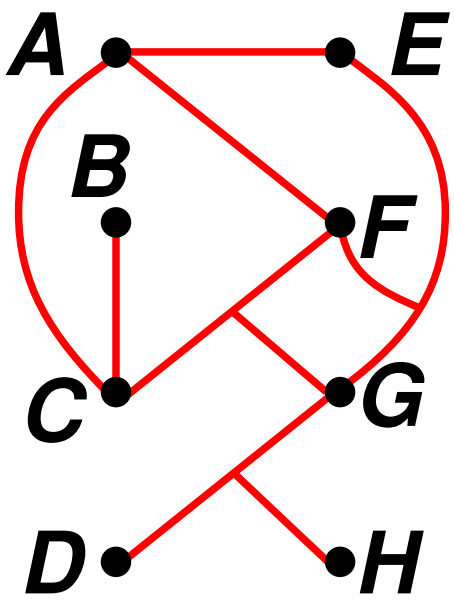


Durchlauf 2

Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$
1	-3	1	-1	0	-2	-3	-1	0	-1	-1

Fiduccia-Mattheyses-Algorithmus
Beispiel

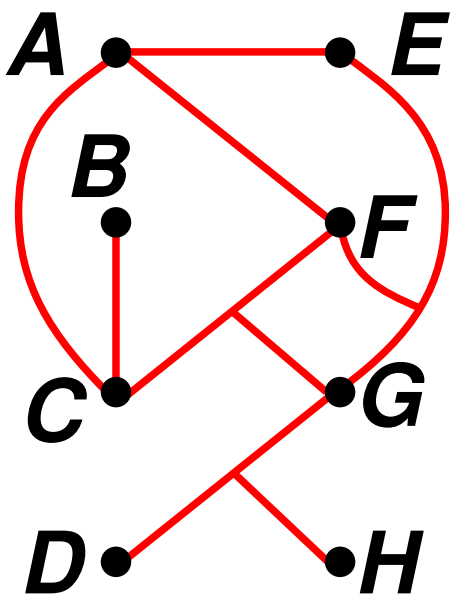


Durchlauf 2

Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$
1	-3	1	-1	0	-2	-3	-1	0	-1	-1
2	-1	-1		0	-2	-3	-1	0	0	-1

Fiduccia-Mattheyses-Algorithmus
Beispiel

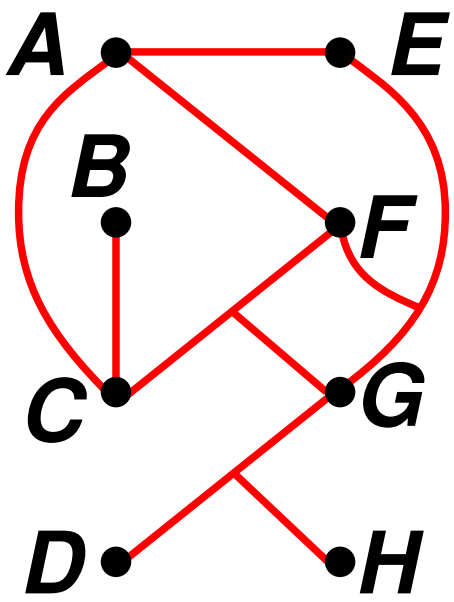


Durchlauf 2

Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$
1	-3	1	-1	0	-2	-3	-1	0	-1	-1
2	-1	-1		0	-2	-3	-1	0	0	-1
3	-1	-1			-2	-3	-1	1	-1	-2

Fiduccia-Mattheyses-Algorithmus
Beispiel

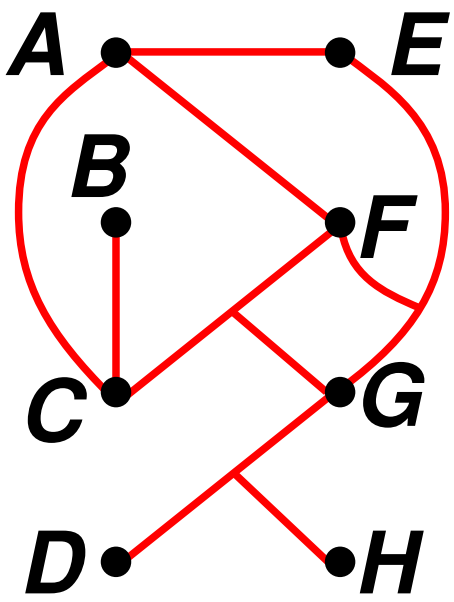


Durchlauf 2

Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$
1	-3	1	-1	0	-2	-3	-1	0	-1	-1
2	-1	-1		0	-2	-3	-1	0	0	-1
3	-1	-1			-2	-3	-1	1	-1	-2
4		-1			0	0	-1	1	1	-1

Fiduccia-Mattheyses-Algorithmus
Beispiel



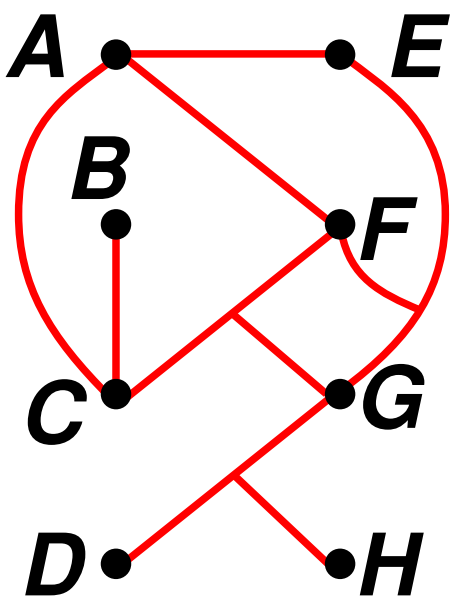
Durchlauf 2

Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$
1	-3	1	-1	0	-2	-3	-1	0	-1	-1
2	-1	-1		0	-2	-3	-1	0	0	-1
3	-1	-1			-2	-3	-1	1	-1	-2
4		-1			0	0	-1	1	1	-1
5		-1			0	0	-2		0	-1

Fiduccia-Mattheyses-Algorithmus
Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

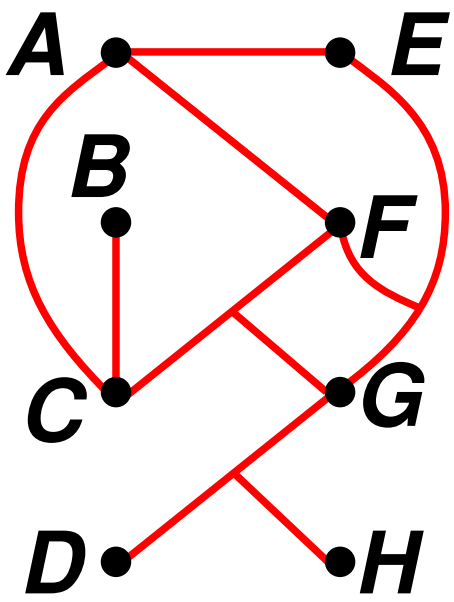


Durchlauf 2

Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$
1	-3	1	-1	0	-2	-3	-1	0	-1	-1
2	-1	-1		0	-2	-3	-1	0	0	-1
3	-1	-1			-2	-3	-1	1	-1	-2
4		-1			0	0	-1	1	1	-1
5		-1			0	0	-2		0	-1
6		-1				1	-2		1	0

Fiduccia-Mattheyses-Algorithmus
Beispiel

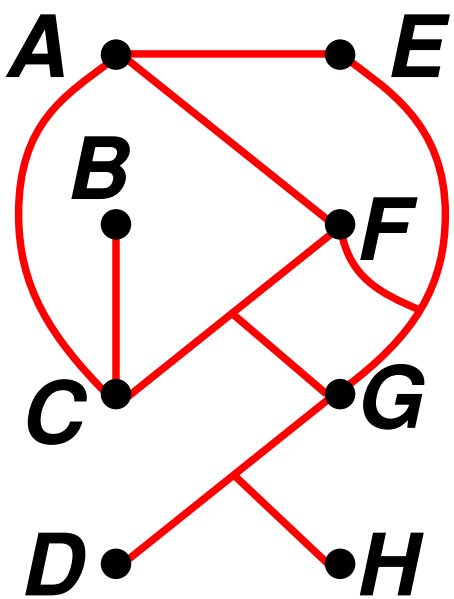


Durchlauf 2

Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$
1	-3	1	-1	0	-2	-3	-1	0	-1	-1
2	-1	-1		0	-2	-3	-1	0	0	-1
3	-1	-1			-2	-3	-1	1	-1	-2
4		-1			0	0	-1	1	1	-1
5		-1			0	0	-2		0	-1
6		-1				1	-2		1	0
7		-1					1		-1	-1

Fiduccia-Mattheyses-Algorithmus
Beispiel



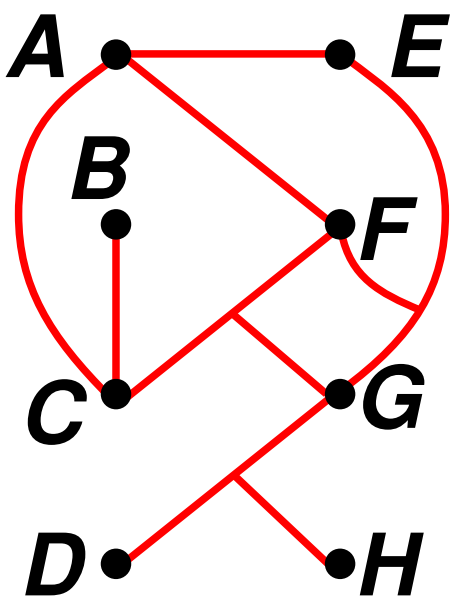
Durchlauf 2

Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$
1	-3	1	-1	0	-2	-3	-1	0	-1	-1
2	-1	-1		0	-2	-3	-1	0	0	-1
3	-1	-1			-2	-3	-1	1	-1	-2
4		-1			0	0	-1	1	1	-1
5		-1			0	0	-2		0	-1
6		-1				1	-2		1	0
7		-1					1		-1	-1
8							1		1	0

Fiduccia-Mattheyses-Algorithmus
Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT



Durchlauf 2

Gewinnwerte:

i	A	B	C	D	E	F	G	H	g_i	$\sum g_i$
1	-3	1	-1	0	-2	-3	-1	0	-1	-1
2	-1	-1		0	-2	-3	-1	0	0	-1
3	-1	-1			-2	-3	-1	1	-1	-2
4		-1			0	0	-1	1	1	-1
5		-1			0	0	-2		0	-1
6		-1				1	-2		1	0
7		-1					1		-1	-1
8							1		1	0

$\sum$ maximal = 0 $\Rightarrow$ kein Wechsel, FM fertig

Finale Schnittkosten = 2

FM-Algorithmus Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Komplexität: $\mathcal{O}(P)$ pro Durchlauf $\Rightarrow \mathcal{O}(PlogP)$

FM-Algorithmus
Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Komplexität: $\mathcal{O}(P)$ pro Durchlauf $\Rightarrow \mathcal{O}(PlogP)$
- Sehr schnell durch geschickte Wahl der Datenstrukturen

FM-Algorithmus
Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Komplexität: $\mathcal{O}(P)$ pro Durchlauf $\Rightarrow \mathcal{O}(PlogP)$
- Sehr schnell durch geschickte Wahl der Datenstrukturen
- Nicht optimal, aber relativ gute Qualität

Multilevel Partitionierer

TECHNISCHE UNIVERSITÄT DARMSTADT

- Multilevel Partitionierer

Multilevel Partitionierer

TECHNISCHE UNIVERSITÄT DARMSTADT

- Multilevel Partitionierer
- Ablauf:

Multilevel Partitionierer

TECHNISCHE UNIVERSITÄT DARMSTADT

- Multilevel Partitionierer
- Ablauf:

  Vergröbern: Rekursiv Knoten zusammengruppieren

  (Basierend auf einer Konnektivitätsmetrik) bis Anzahl Knoten einer Gruppe kleiner als kleiner Wert c (Größenordnung $c = 100$)

Multilevel Partitionierer

TECHNISCHE UNIVERSITÄT DARMSTADT

- Multilevel Partitionierer
- Ablauf:

  Vergröbern: Rekursiv Knoten zusammengruppieren

  (Basierend auf einer Konnektivitätsmetrik) bis Anzahl Knoten einer Gruppe kleiner als kleiner Wert c (Größenordnung $c = 100$)

Multilevel Partitionierer

TECHNISCHE UNIVERSITÄT DARMSTADT

- Multilevel Partitionierer
- Ablauf:

  Vergröbern: Rekursiv Knoten zusammengruppieren
  (Basierend auf einer Konnektivitätsmetrik) bis Anzahl Knoten einer Gruppe kleiner als kleiner Wert c (Größenordnung $c = 100$)

  Verfeinern:
  - Ergebnis einer gröberen Stufe nehmen und auf feinere projezieren

Multilevel Partitionierer

TECHNISCHE UNIVERSITÄT DARMSTADT

- Multilevel Partitionierer
- Ablauf:

  Vergröbern: Rekursiv Knoten zusammengruppieren
  (Basierend auf einer Konnektivitätsmetrik) bis Anzahl Knoten einer Gruppe kleiner als kleiner Wert c (Größenordnung $c = 100$)

  Verfeinern:
  - Ergebnis einer gröberen Stufe nehmen und auf feinere projezieren
  - Jedesmal Partionierungsalgorithmus benutzen, um Qualität schrittweise zu verbessern

Multilevel Partitionierer

TECHNISCHE UNIVERSITÄT DARMSTADT

- Multilevel Partitionierer
- Ablauf:

  Vergröbern: Rekursiv Knoten zusammengruppieren
  (Basierend auf einer Konnektivitätsmetrik) bis Anzahl Knoten einer Gruppe kleiner als kleiner Wert c (Größenordnung $c = 100$)

  Verfeinern:
  - Ergebnis einer gröberen Stufe nehmen und auf feinere projezieren
  - Jedesmal Partionierungsalgorithmus benutzen, um Qualität schrittweise zu verbessern
- Beispiel:

Multilevel Partitionierer

TECHNISCHE UNIVERSITÄT DARMSTADT

- Multilevel Partitionierer
- Ablauf:

  Vergröbern: Rekursiv Knoten zusammengruppieren
  (Basierend auf einer Konnektivitätsmetrik) bis Anzahl Knoten einer Gruppe kleiner als kleiner Wert c (Größenordnung $c = 100$)

  Verfeinern:
  - Ergebnis einer gröberen Stufe nehmen und auf feinere projezieren
  - Jedesmal Partionierungsalgorithmus benutzen, um Qualität schrittweise zu verbessern
- Beispiel:
  - hMetis (Karypis, 1997)

Multilevel Partitionierer

TECHNISCHE UNIVERSITÄT DARMSTADT

- Multilevel Partitionierer
- Ablauf:

  Vergröbern: Rekursiv Knoten zusammengruppieren
  (Basierend auf einer Konnektivitätsmetrik) bis Anzahl Knoten einer Gruppe kleiner als kleiner Wert c (Größenordnung $c = 100$)

  Verfeinern:
  - Ergebnis einer gröberen Stufe nehmen und auf feinere projezieren
  - Jedesmal Partionierungsalgorithmus benutzen, um Qualität schrittweise zu verbessern
- Beispiel:
  - hMetis (Karypis, 1997)
  - Benutzt zufällige Initiale Partitionierung

Multilevel Partitionierer

TECHNISCHE UNIVERSITÄT DARMSTADT

- Multilevel Partitionierer
- Ablauf:

  Vergröbern: Rekursiv Knoten zusammengruppieren
  (Basierend auf einer Konnektivitätsmetrik) bis Anzahl Knoten einer Gruppe kleiner als kleiner Wert c (Größenordnung $c = 100$)

  Verfeinern:
  - Ergebnis einer gröberen Stufe nehmen und auf feinere projezieren
  - Jedesmal Partionierungsalgorithmus benutzen, um Qualität schrittweise zu verbessern
- Beispiel:
  - hMetis (Karypis, 1997)
  - Benutzt zufällige Initiale Partitionierung
  - Für die Verfeinerung FM

Multilevel Partitionierer

TECHNISCHE UNIVERSITÄT DARMSTADT

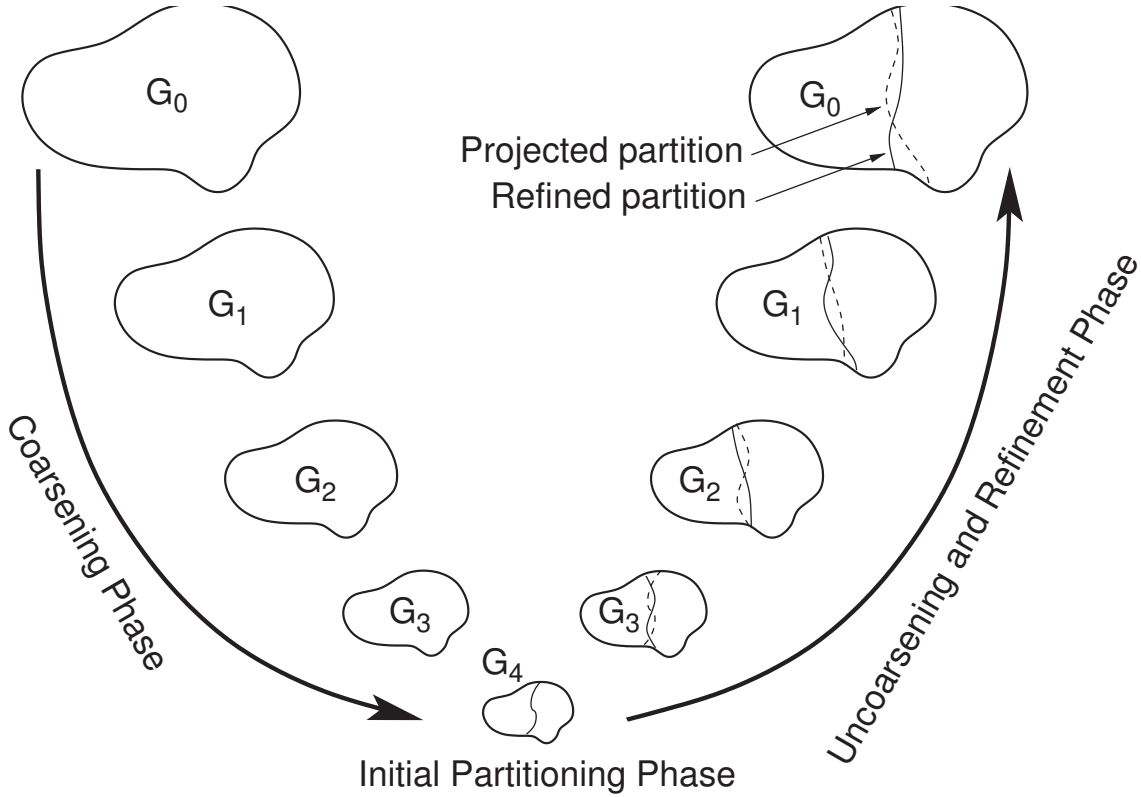
- Multilevel Partitionierer
- Ablauf:

  Vergröbern: Rekursiv Knoten zusammengruppieren
  (Basierend auf einer Konnektivitätsmetrik) bis Anzahl Knoten einer Gruppe kleiner als kleiner Wert c (Größenordnung $c = 100$)

  Verfeinern:
  - Ergebnis einer gröberen Stufe nehmen und auf feinere projezieren
  - Jedesmal Partionierungsalgorithmus benutzen, um Qualität schrittweise zu verbessern
- Beispiel:
  - hMetis (Karypis, 1997)
  - Benutzt zufällige Initiale Partitionierung
  - Für die Verfeinerung FM
  - Insgesamt sehr gute Lösungsqualität

Multilevel Partitionierer

TECHNISCHE UNIVERSITÄT DARMSTADT



Quelle: Y.-W. Chang

Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Partitionierung
- KL
- FM
  - Schneller als KL, aber schlechtere Lösungsqualität
- Multilevel Partitionierung
- Andere akademische Ansätze: Spektral Methode, Netzwerkfluß, ILP, Hybrid-Varianten, ...