Karlsruher Institut für Technologie

Institut für Technik der Informationsverarbeitung

Institutsleitung
Prof. Dr.-Ing. Dr. h. c. J. Becker
Prof. Dr.-Ing. E. Sax
Prof. Dr. rer. nat. W. Stork

Übung 7

Dipl.-Inform. Leonard Masing
Dr.-Ing. Oliver Sander

Institut für Technik der Informationsverarbeitung (ITIV)

Hardware/Software Co-Design

Agenda

- Wiederholung ausgewählter Themen
  - Profiling
  - Partitionierungsalgorithmen
    - Hierarchical clustering
    - Tabu-search
- Gruppenarbeit
- Vorstellung der Lösung

3.6.2 Software-Performanz: Profiling & Tracing (II)

- Das Profiling und Tracing erfolgt i.a. durch Einfügen von zusätzlichen Monitor- Code an bestimmten Stellen, der während der Programmausführung dann das Programmausführungsverhalten protokolliert.

- Das QPT- Tool verwendet dazu zwei Methoden:
  - **Profiling:** Es werden die Verzweigungs- bzw. Ausführungshäufigkeiten jedes sequentiellen Programm-Basisblocks gemessen.
  - **Tracing:** Die Ausführungsreihenfolge der Programm-Basisblöcke wird protokolliert.

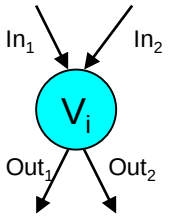
- Diese daraus gewonnenen Daten erlauben die Berechnung der Ausführungskosten der Prozeduren in einem Programm.

3.6.2 Software-Performanz: Profiling & Tracing mit WARTS (I)

- **Definition: Basisblock**
  - Ein Basisblock ist eine Sequenz von aufeinanderfolgenden Befehlen, in welche der Kontrollfluß eintritt und diese wieder verläßt, ohne Halt oder Möglichkeit daraus zu verzweigen, ausgenommen an deren Ende.

- **Definition: Kontrollflußgraph (CFG)**
  - Ein Kontrollflußgraph (CFG) ist ein gerichteter Graph (mit Wurzel) G=(V, E) mit einem speziellen EXIT- Knoten (≠ Wurzel), so daß dieser Graph mit einem Programm auf folgende Weise korrespondiert:
    Jeder Knoten in V repräsentiert einen Basisblock und jede Kante in E repräsentiert eine Kontrollverzweigung von einem Basisblock zu einem anderen.

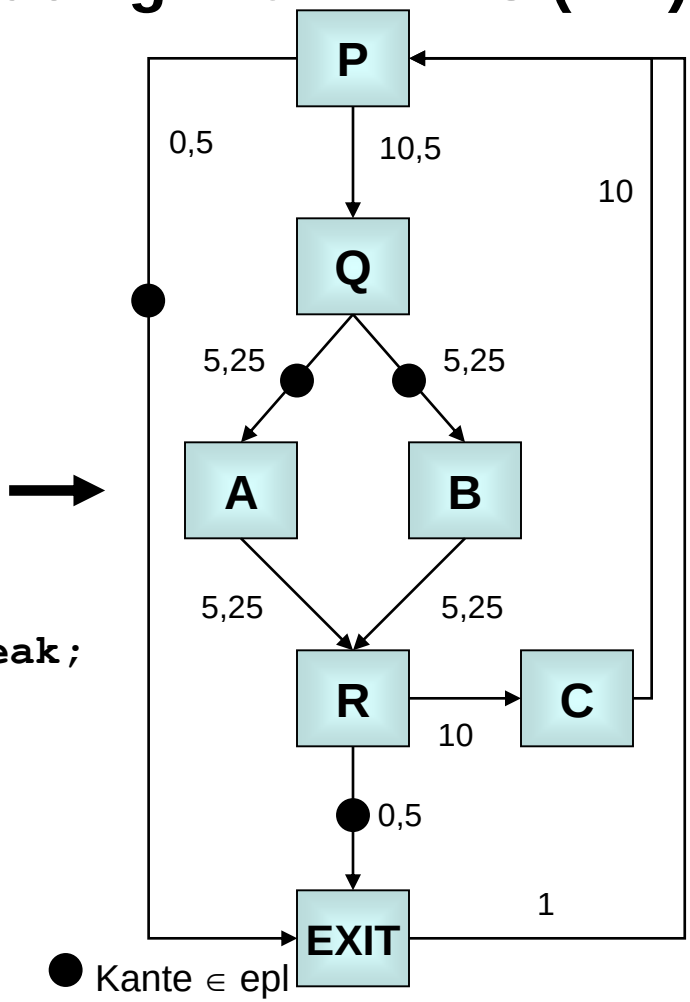
- **Definition: Gewichte W eines CFG**

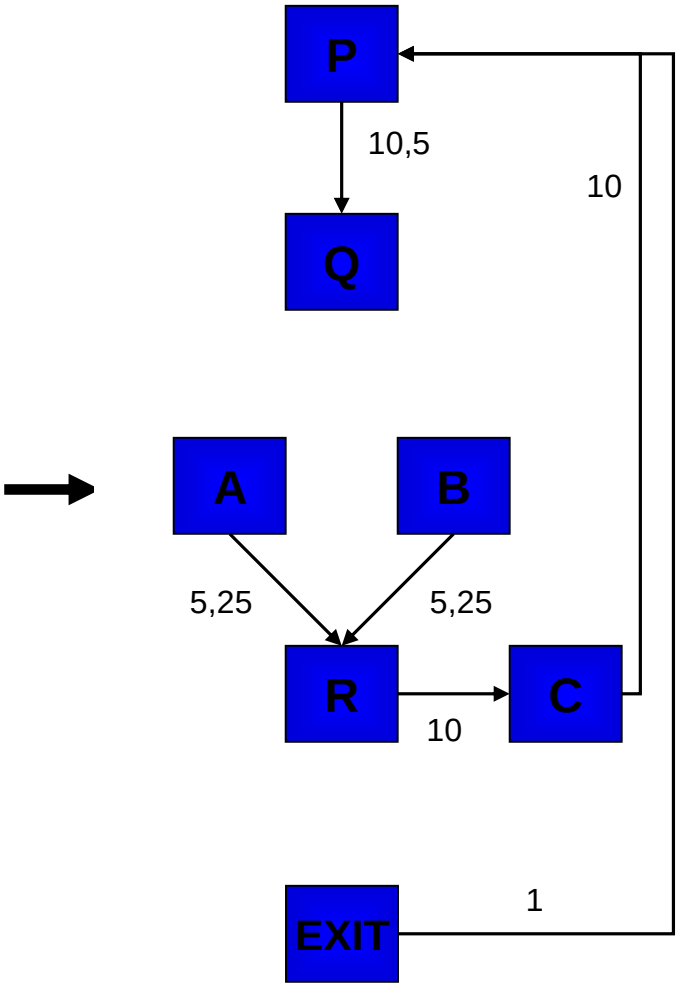


  - Als Gewichte W des CFG werden allen Kanten aus E ein nicht-negativer Wert zugeordnet, wobei die Summe der Gewichte der eingehenden bzw. ausgehenden Kanten eines Knotens gleich sind (Bilanz: $\Sigma\ in_i = \Sigma\ out_i$).
  - Das Gewicht eines Knotens aus V ist einfach die Summe der eingehenden bzw. nur ausgehenden Kanten (=Ausführungshäufigkeit B_i).

3.6.2 Software-Performanz: Profiling & Tracing mit WARTS (VIII)

```
program
  while P do
  begin
    if Q then
      A;
    else
      B;
    if R then break;
    C;
  end;
end
```





Programmbeispiel

CFG mit Gewichten W

Maximal-aufspannender Baum (zusammenhängend + zyklenfrei !)

- epl liefert eine minimale Lösung bzgl. EF(G, epl) ⇔ (E- epl) bildet einen maximal aufspannenden Baum G´ von G

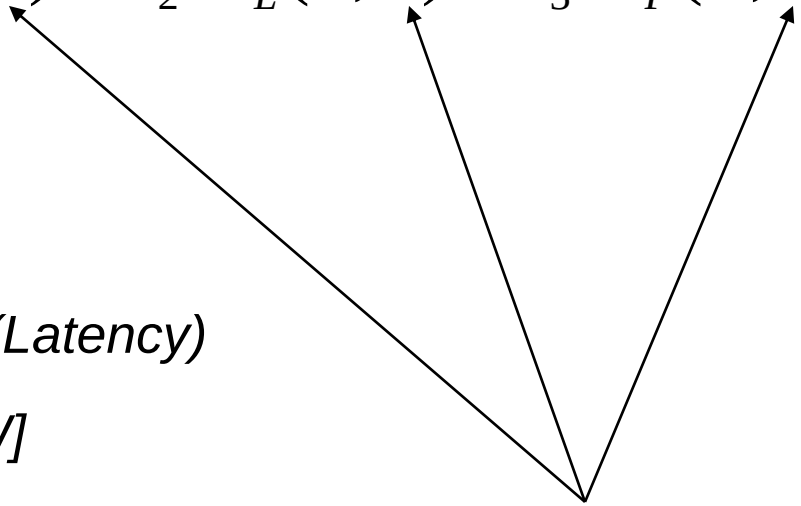
Partitionierung - Überblick

- **4.1 Einführung, Partitionierungsansätze, Komplexität**
- 4.2 Klassifikation von Partitionierungsalgorithmen
- 4.3 Konstruktive Algorithmen
  - 4.3.1 Hierarchisches Clustering
- 4.4 Iterative Algorithmen
  - 4.4.1 Tabu-Search
  - 4.4.2 Kernighan Lin
  - 4.4.3 Fiduccia Mattheyses
  - 4.4.4 Integer Linear Programming - ILP
  - 4.4.5 Simulated Annealing
  - 4.4.6 Genetische Algorithmen
- 4.5 Hardware/Software Partitionierungsverfahren und Co-Design-Systeme

4.1 Kostenfunktionen zur Hardware/Software-Partitionierung

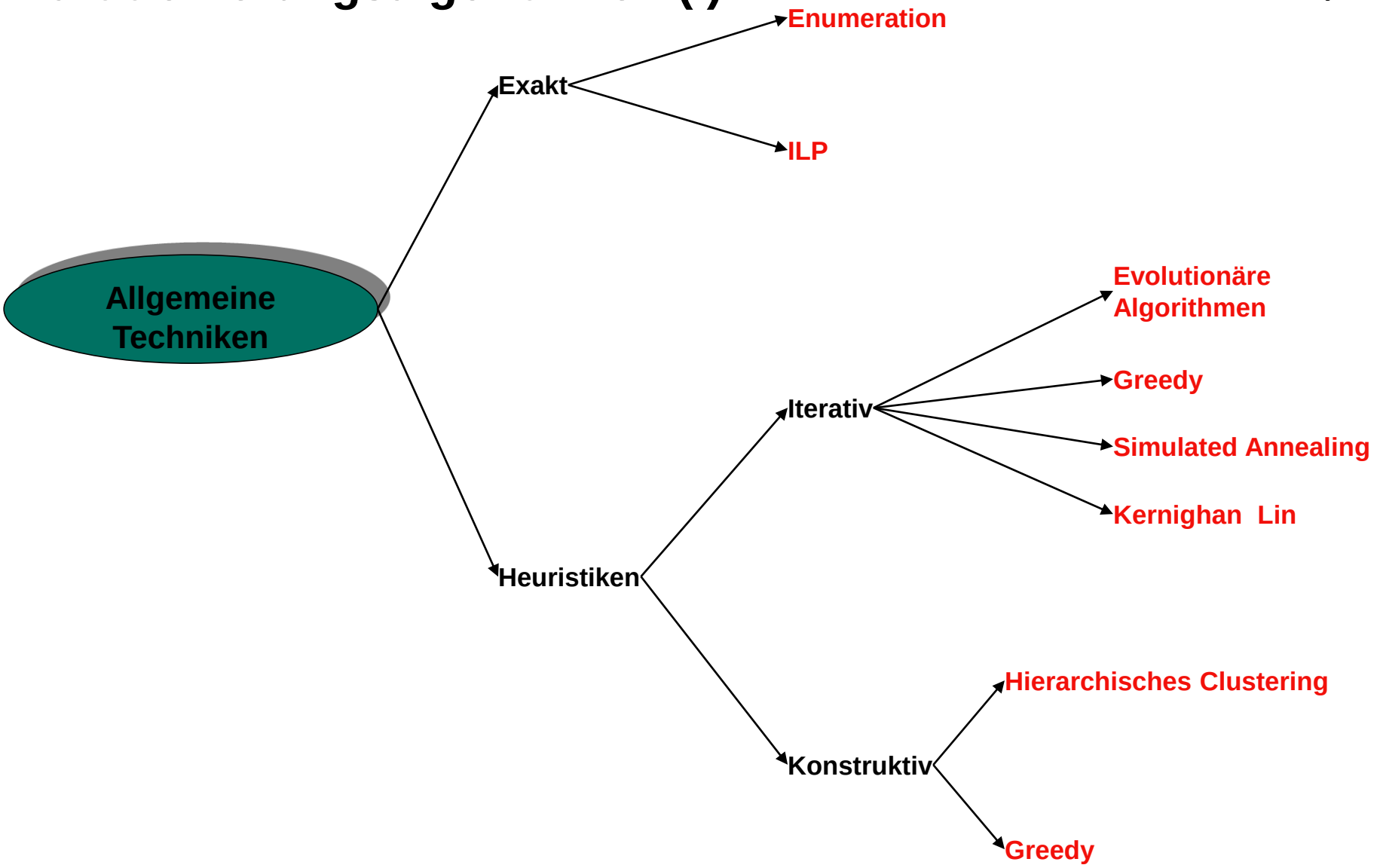
- **Beispiel für eine Kostenfunktion:**

$$f(C,L,P) = k_1 \cdot h_c(C,\overline{C}) + k_2 \cdot h_L(L,\overline{L}) + k_3 \cdot h_P(P,\overline{P})$$

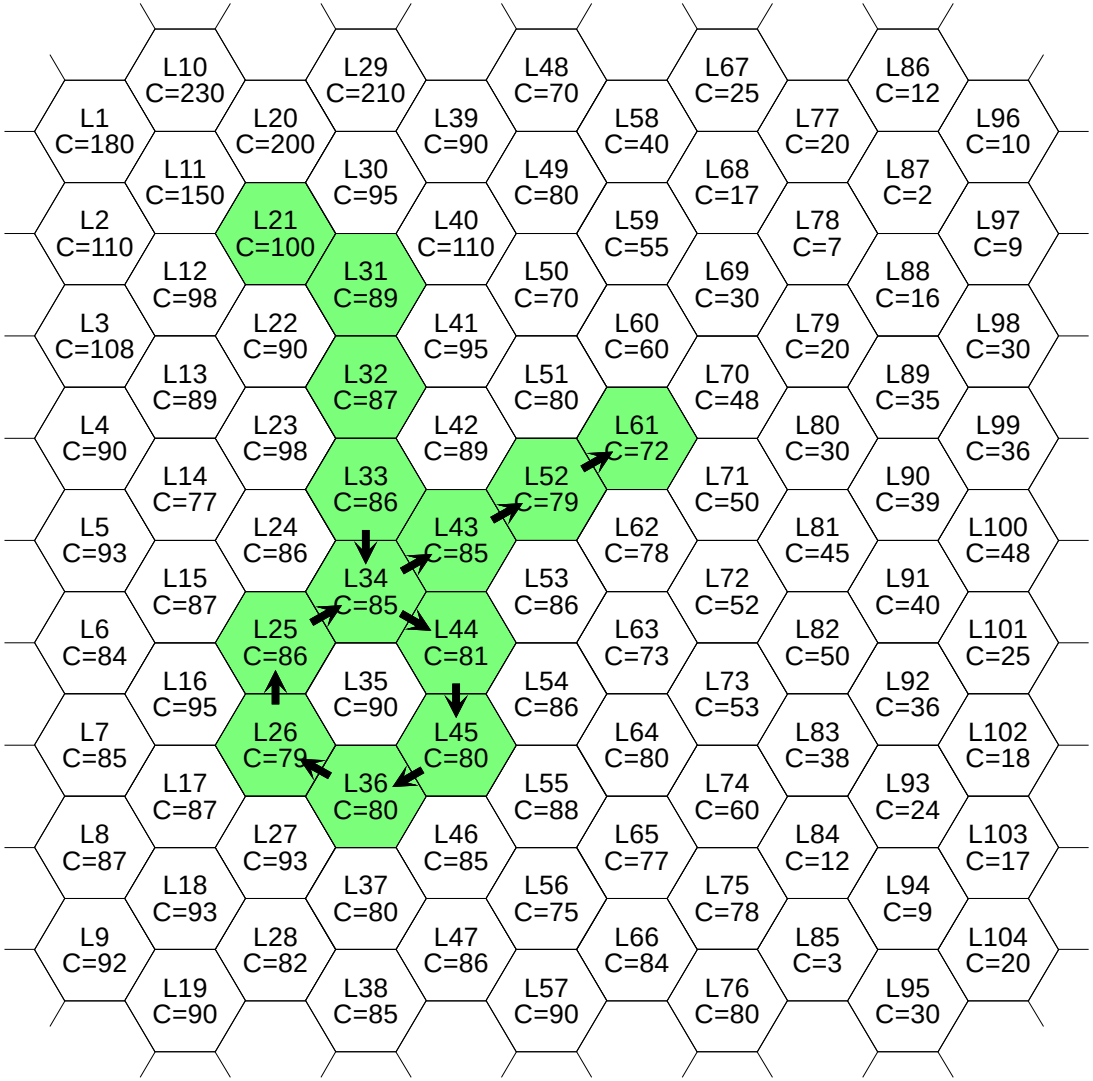


- *C ≙ Systemkosten in [Euro]*
- *L ≙ Ausführungszeit in [sec] (Latency)*
- *P ≙ Leistungsaufnahme in [W]*
- *h_C, h_L, h_P, geben an, wie stark C, L, P die Entwurfsbedingungen (Constraints C, L, P überstrichen) verletzen.*
- *k_1, k_2, k_3 ≙ Gewichtung und Normierung*

4.2 Klassifikation von Partitionierungsalgorithmen (I)



4.4.1 Tabu-Search Algorithmus: Beispiel (I)



n_{Tabu_Liste}=10 k=6

Stop_Cnd: BS <5

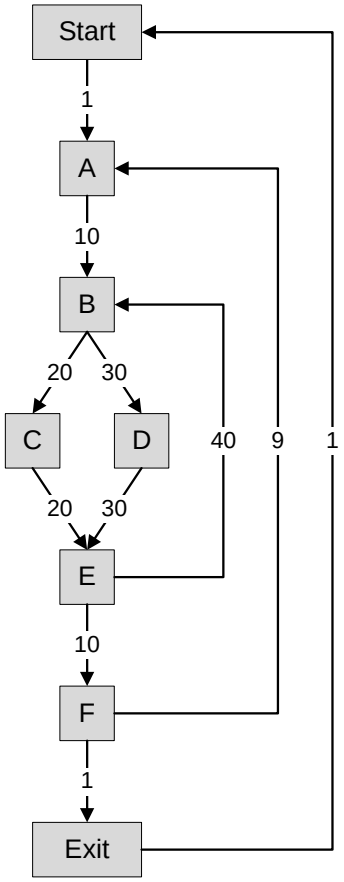
BS:	CS:	Step:
100	100	**Start:** L21
89	89	L21 ⇒L31
87	87	L31 ⇒L32
86	86	L32 ⇒L33
85	85	L33 ⇒L34
81	81	L34 ⇒L44
80	80	L44 ⇒L45
80	80	L45 ⇒L36
79	79	L36 ⇒L26
79	86	L26 ⇒L25
79	85	L25 ⇒L34
79	85	L34 ⇒L43
79	79	L43 ⇒L52
72	72	L52 ⇒L61

Arbeitsphase

- Aufgabe 3.07: Profiling
  - Kanten-Frequenzproblem
- Aufgabe 4.01: Hierarchical Clustering
  - Verschmelzung von Knoten
- Aufgabe 4.0.2: Tabu-Search

Aufgabe 3.07: Profiling

- Gegeben ist folgender Kontrollflussgraph mit Gewichten:



a) Was ist der Unterschied zwischen WCET und Profiling? Welches findet zur Laufzeit bzw. Compilezeit statt?

b) Welches Problem wurde in der Vorlesung bei Profiling gelöst?

c) Bestimmen Sie den Maximal Aufspannenden Baum (MAB) des Kontrollflussgraphen. Sie können in der Zeichnung die Kanten, die zum MAB gehören, mit einem X markieren.

d) Auf welchen Kanten werden jetzt die Zähler für das Profiling platziert? Markieren Sie diese mit einem Kreis.

e) Warum wird in diesem Verfahren der Maximale und nicht der Minimale Aufspannende Baum verwendet?

f) Wie können die restlichen Zählerwerte aus den platzierten Zählern bestimmt werden?

Lösung Aufgabe 3.07: Profiling

a) Was ist der Unterschied zwischen WCET und Profiling? Welches findet zur Laufzeit bzw. Compilezeit statt?

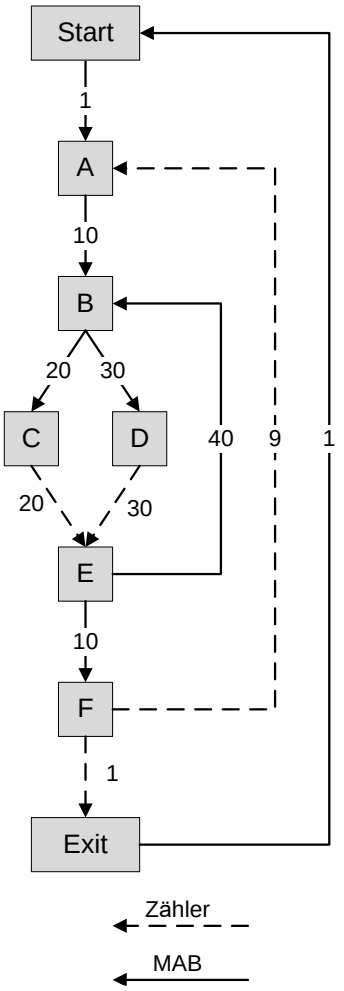
- WCET bestimmt die maximale Ausführungszeit anhand einer statischen Quellcodeanalyse. Profiling hingegen findet zur Laufzeit statt und misst die Verzweigungs- bzw. Ausführungshäufigkeiten jedes Basisblockes.

b) Welches Problem wurde in der Vorlesung bei Profiling gelöst?

- Kanten-Frequenzproblem

c) Bestimmen Sie den Maximal Aufspannenden Baum (MAB) des Kontrollflussgraphen. Sie können in der Zeichnung die Kanten, die zum MAB gehören, mit einem X markieren.

d) Auf welchen Kanten werden jetzt die Zähler für das Profiling platziert? Markieren Sie diese mit einem Kreis.



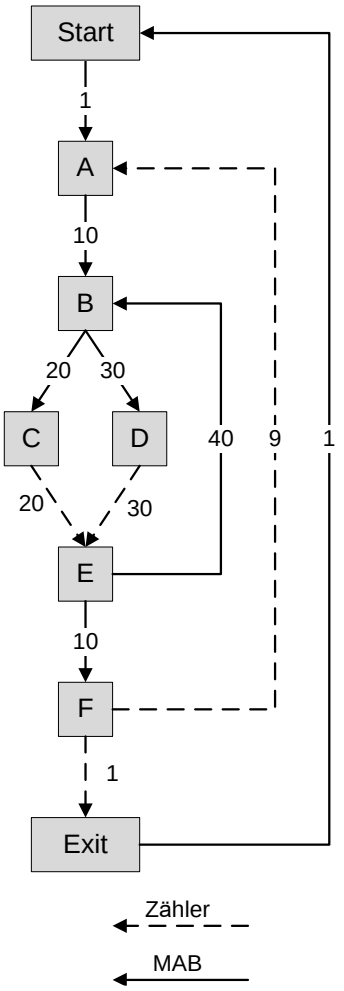
Lösung Aufgabe 3.07: Profiling

e) Warum wird in diesem Verfahren der Maximale und nicht der Minimale Aufspannende Baum verwendet?

- Ziel ist das Platzieren der Zähler auf Kanten mit einer niedrigen Aufrufshäufigkeit um den Overhead für das Profiling möglichst gering zu halten. Da die Zähler auf den Kanten platziert werden, die vom Aufspannenden Baum NICHT verwendet werden, sollten die Kanten des MAB möglichst maximal sein, damit die niedrigen übrig bleiben.

f) Wie können die restlichen Zählerwerte aus den platzierten Zählern bestimmt werden?

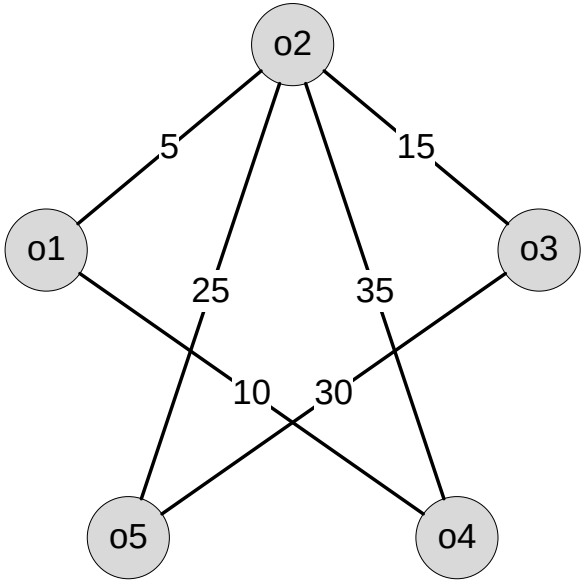
- Die freien Zählerstellen bilden einen Baum. An den Blättern des Baumes sind alle bis auf einen Zählerwert bekannt und durch die Flussgleichungen kann dieser Zählerwert bestimmt werden. Somit ist es sukzessive möglich sämtliche Werte zu bestimmen.



- Profiling
  - Unterschied zu WCET und Tracing?
  - Welches Problem wird gelöst?
  - Warum wird ein MAB verwendet?
  - Welche Eigenschaften hat dieser?
  - Wie wird dieser angewendet?
  - Was bedeuten die Zahlen an den Kanten?

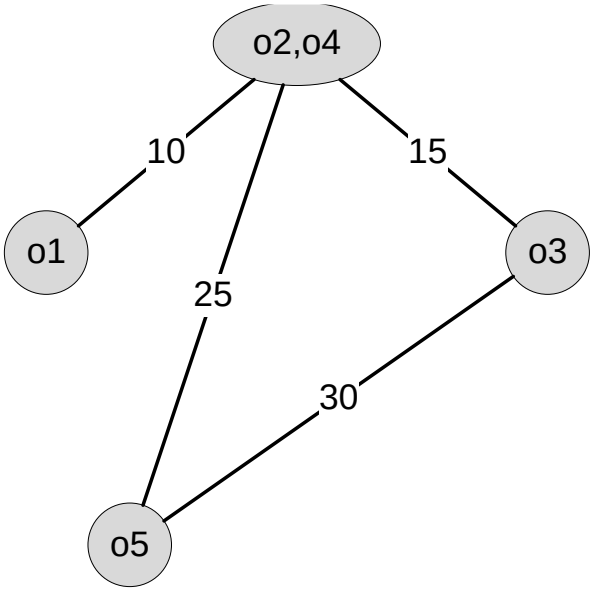
Aufgabe 4.01: Hierarchical Clustering

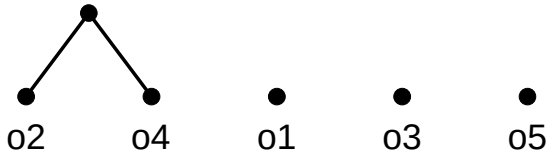
- Führen Sie das Hierarchical Clustering Verfahren anhand des gegebenen Graphen durch. Bei der Verschmelzung der Knoten soll die Maximum-Metrik angewandt werden, d.h. das Kantengewicht einer neuen Kante entspricht dem Maximum der vorherigen Kantengewichte.

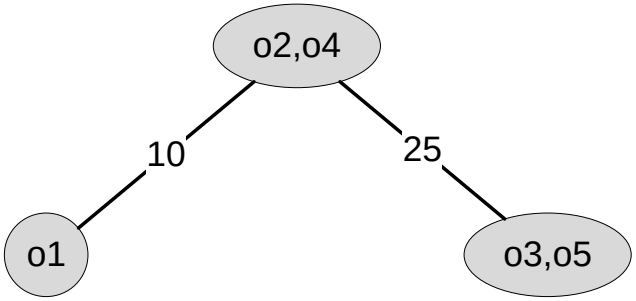


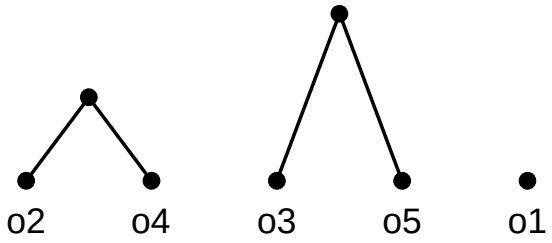
Leonard Masing
Hardware/Software Co-Design

Lösung Aufgabe 4.01: Hierarchical Clustering



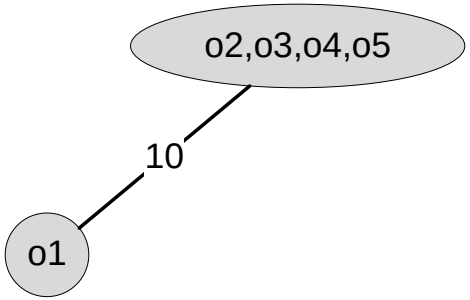


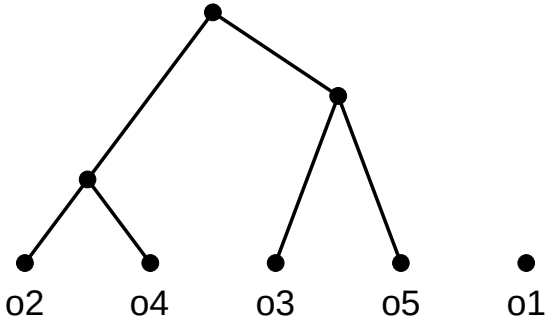


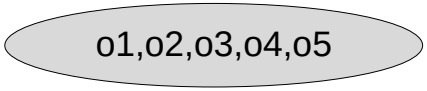


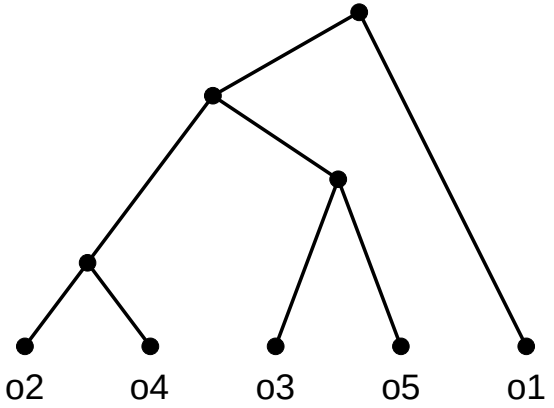
Leonard Masing
Hardware/Software Co-Design

Lösung Aufgabe 4.01: Hierarchical Clustering









Leonard Masing
Hardware/Software Co-Design

- Hierarchical Clustering
  - In welche Klasse von Verfahren gehört Hierarchical Clustering?
  - Wie werden Knoten verschmolzen?
  - Wie kann eine Bi-Partitionierung durchgeführt werden?

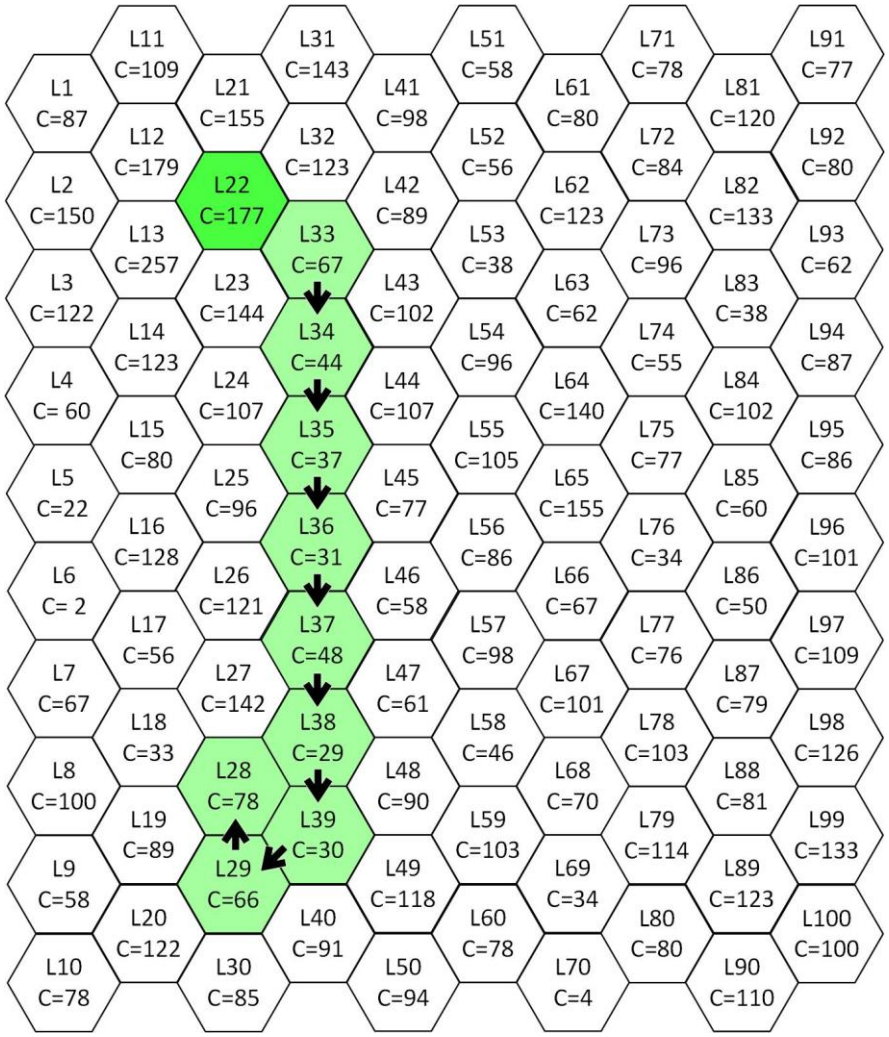
Lösung Aufgabe 4.02: Tabu-Search

a) Führen Sie den Tabu-search Algorithmus auf der in Figure 4.4. gegebenen Lösungsmenge jeweils für die zwei unterschiedlichen Startpunkte „L22“ und „L71“ aus. Der Algorithmus soll mit der Stop-condition „stop_cond < 5“ beendet werden und verfügt über eine Tabu-Liste der Länge 8.

b) Kann das Verfahren hillclimbing?

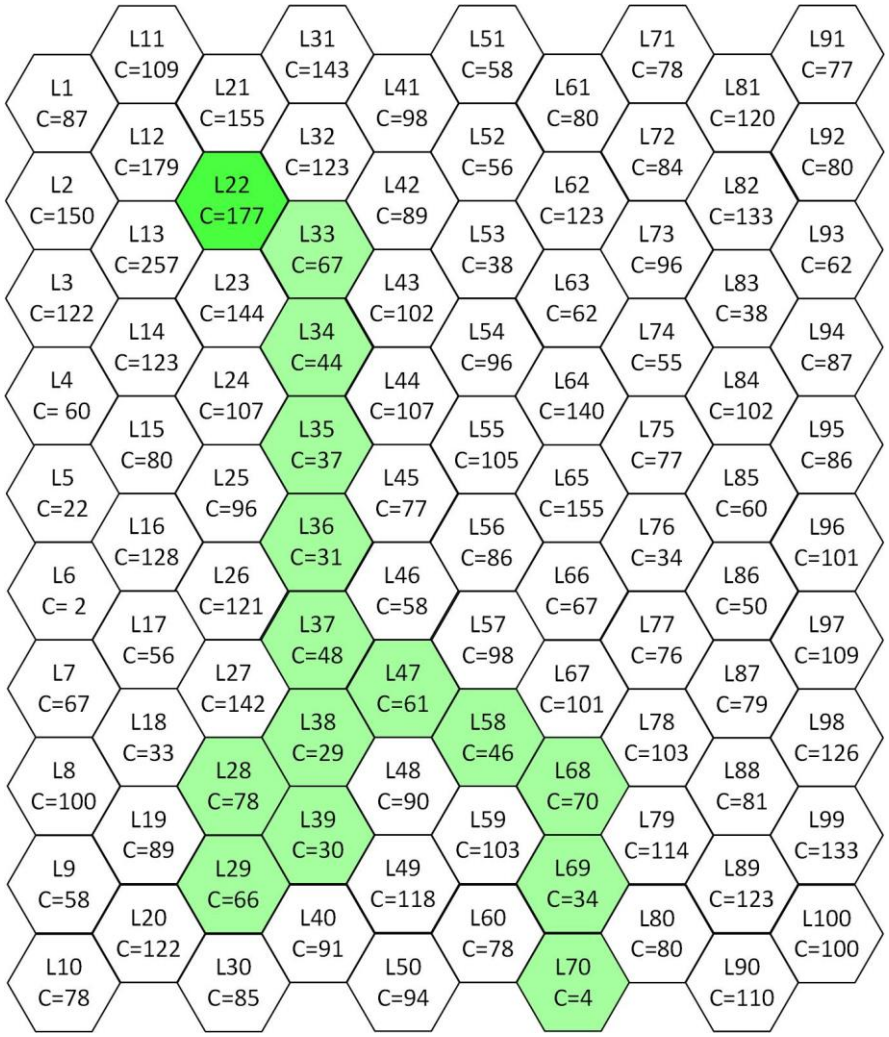
c) Findet das Verfahren die global beste Lösung?

Aufgabe 4.02: Tabu-Search



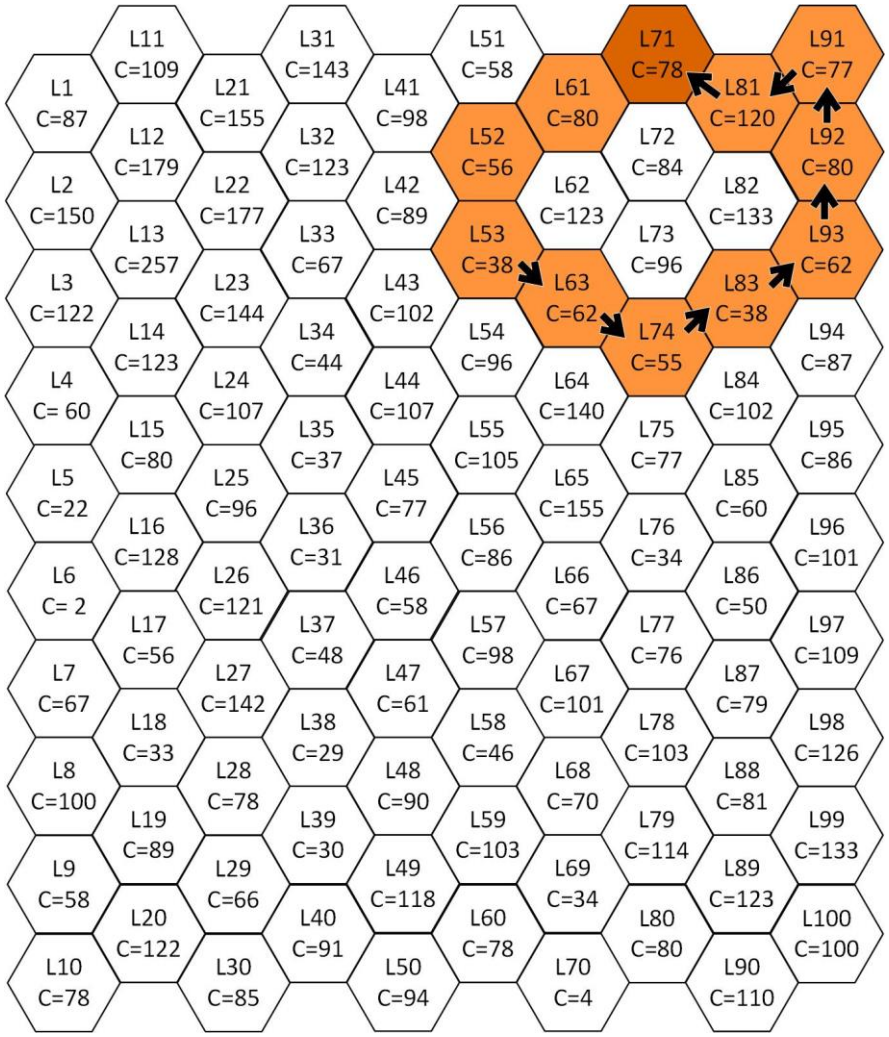
Leonard Masing
Hardware/Software Co-Design

Aufgabe 4.02: Tabu-Search



Leonard Masing
Hardware/Software Co-Design

Aufgabe 4.02: Tabu-Search



Leonard Masing
Hardware/Software Co-Design

Lösung Aufgabe 4.02: Tabu-Search

a) Führen Sie den Tabu-search Algorithmus auf der in Figure 4.4. gegebenen Lösungsmenge jeweils für die zwei unterschiedlichen Startpunkte „L22“ und „L71“ aus. Der Algorithmus soll mit der Stop-condition „stop_cond < 5“ beendet werden und verfügt über eine Tabu-Liste der Länge 8.

b) Kann das Verfahren hillclimbing?

- Grundsätzlich verfügen alle iterative Verfahren über eine hillclimbing Eigenschaft (also die Fähigkeit, auch Verschlechterungen in Kauf zu nehmen um aus lokalen Extrema herauszukommen). In diesem Beispiel wäre dies an mehreren Stellen z.B. bei L36->L37 oder bei L39->L29.

c) Findet das Verfahren die global beste Lösung?

- Dies ist möglich, es kann jedoch nie garantiert werden. Im vorgestellten Beispiel wird das globale Optimum nicht gefunden (dies liegt bei L6). Da man das globale Optimum aber im Voraus nicht kennt, kann man nur den Algorithmus mit unterschiedlichen Startkonfigurationen neu laufen lassen und dann (üblicherweise anhand der stop condition) einen Wert definieren ab welchem man mit dem Ergebnis zufrieden ist.

- Für was kann TS gut eingesetzt werden?
- Wie geht der Algorithmus vor?
- Was ist die Tabu-List?
- Wie wird diese upgedated?
- Welche Komplexität hat der Algorithmus?