Optimum Scheduling, Allocation, and Resource binding (OSCAR)

Peter Marwedel
TU Dortmund
Informatik 12

2008/06/08

Überwindung der Trennung in 3 Phasen

Trennung in 3 Phasen lässt potenziell die optimale Lösung verpassen

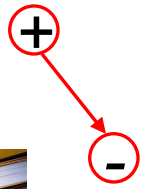
☞ Wir koppeln *scheduling, allocation* und *assignment.*

Integration von Scheduling, Allokation und Assignment in OSCAR

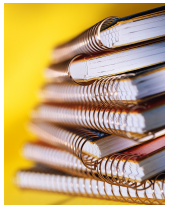
OSCAR = optimum scheduling, allocation, and resource binding

Gegeben:

- Datenfluss-Graph mit Operationen



- Bausteinbibliothek



Quellen:

- B. Landwehr, P. Marwedel, R. Dömer: OSCAR: Optimum Simultaneous Scheduling, Allocation and Resource Binding Based on Integer Programming, Proc. Euro-DAC, 1994, S. 90-95
- B. Landwehr, P. Marwedel: A New Optimization Technique for Improving Resource Exploitation and Critical Path Minimization, 10th International Symposium on System Synthesis (ISSS), Antwerpen, 1997

Entscheidungsvariablen im IP-Modell

Zentrale Entscheidungsvariable:

$$x_{i,j,k} = \begin{cases} 1, \text{wenn Operation } j \text{ im Kontrollschritt } i \text{ auf Exemplar } k \text{ startet} \\ 0, \text{sonst} \end{cases}$$

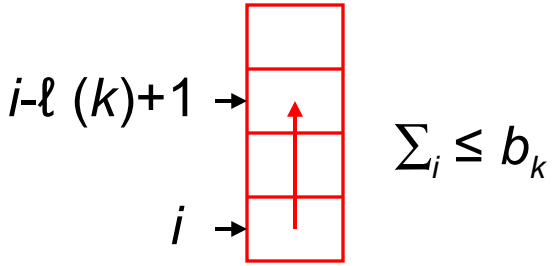
Hilfsvariable:

$$b_k = \begin{cases} 1, \text{falls die Bausteininstanz } k \text{ benötigt wird} \\ 0, \text{sonst} \end{cases}$$

Beschränkungen (*constraints*) (1)

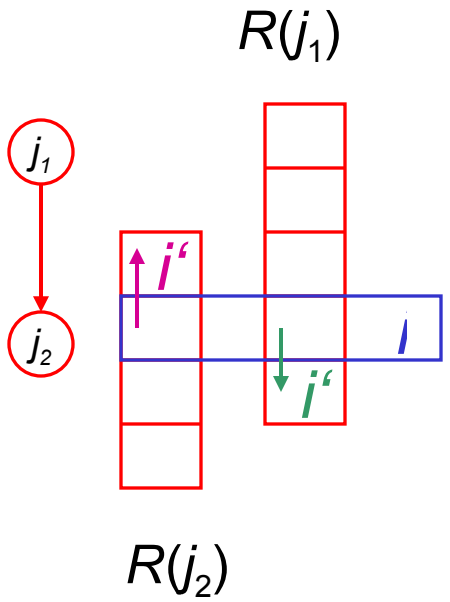
- ***Operation assignment constraints***:
  Jede Operation an exakt einen Kontrollschritt und an einen Hardware-Baustein zu binden:
  $\forall\, j \in J: \sum_{i \in R(j)} \sum_{k \in K,\, j\ executable\_on\ k} x_{i,j,k} = 1$

- ***Resource assignment constraints***:
  Jede Komponente k kann höchstens alle $\ell(k)$ Kontrollschritte eine neue Operation starten.
  $\forall\, k \in K: \forall i \in I: \sum_{j \in J} \sum_{i' \in [i-\ell(k)+1\,..\,i]} x_{i',j,k} \leq b_k$



Beschränkungen (*constraints*) (2)

- ***Precedence constraints***: Zuerst müssen die Argumente berechnet werden.
  Sei j_2 datenabhängig von Operation j_1.



$R(j_1)$, $R(j_2)$: mögliche Kontrollschritt-Bereiche.

$\forall\ i \in R(j_1) \cap R(j_2)$: j_2 darf nicht vor oder in i
und j_1 nicht nach oder in i ausgeführt werden.

Für alle Paare (j_1, j_2) daten-abhängiger Operationen:

$$\forall i \in R(j_1) \cap R(j_2): \quad \sum_{k} \sum_{\substack{i' \le i \\ i' \in R(j_2)}} x_{i',j_2,k} + \sum_{k} \sum_{\substack{i' \ge i \\ i' \in R(j_1)}} x_{i',j_1,k} \le 1$$

Reservierung von Indices

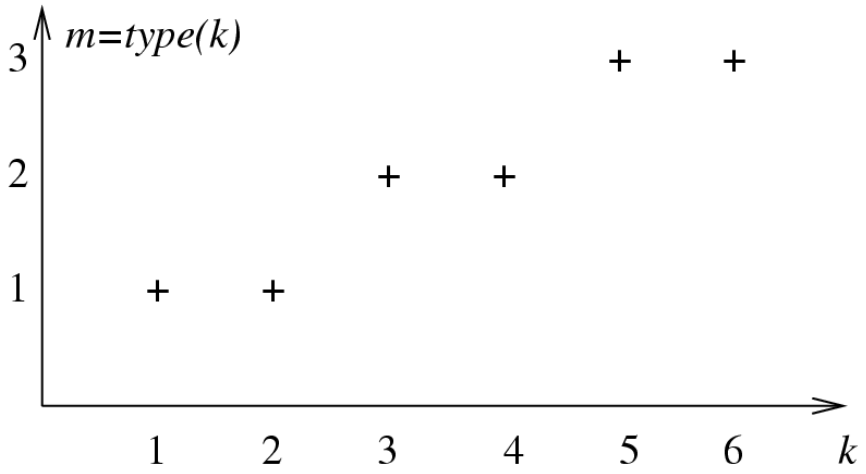
Reservierung von Indices für potenzielle Bausteine *k.*

Strukturierte Methode:

Treppenfunktion für die Funktion *type*.

Beispiel:

Reservierung von max. 2 Indices pro Bausteintyp:

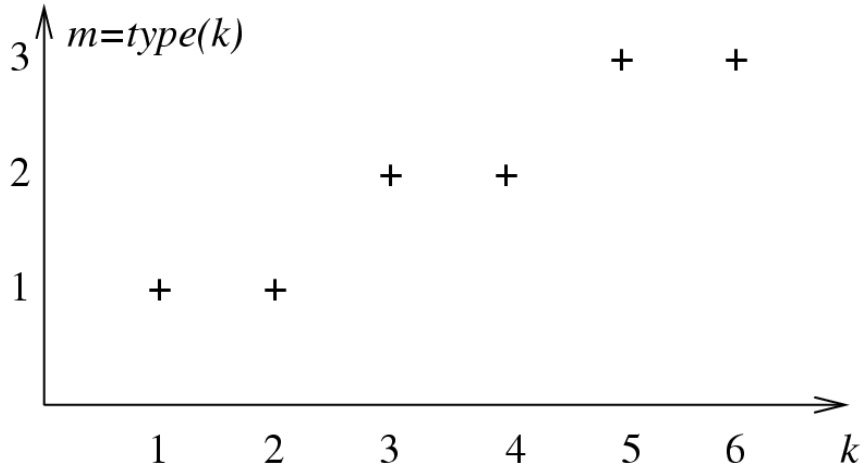


Vermeidung isomorpher Lösungen

IP-Verfahren benötigen ggf. viel Zeit, um Optimalität einer Lösung nachzuweisen

☞ Isomorphe* Lösungen möglichst vermeiden!

☞ Zusätzliche Constraints: Bausteine mit kleinerem *k* immer zuerst benutzen.



Beispiel:
$b_{k+1} \leq b_k$ für $type(k)=type(k+1)$

B. Landwehr, P. Marwedel, I. Markhof, R. Dömer: Exploiting Isomorphism for Speeding-Up Instance-Binding in an Integrated Scheduling, Allocation and Assignment Approach to Architectural Synthesis, *CHDL-97*

*Lösungen, die sich nur durch Umbenennung der Bausteine unterscheiden

Kostenfunktion

Def.

- c_m: Kosten des Bausteintyps m
- $c_{k1,k2}$: Kosten für Verbindung von $k1$ nach $k2$

$$w_{k1,k2} = \begin{cases} 1, \text{falls Baustein } k1 \text{ mit } k2 \text{ verbunden ist} \\ 0, \text{sonst} \end{cases}$$

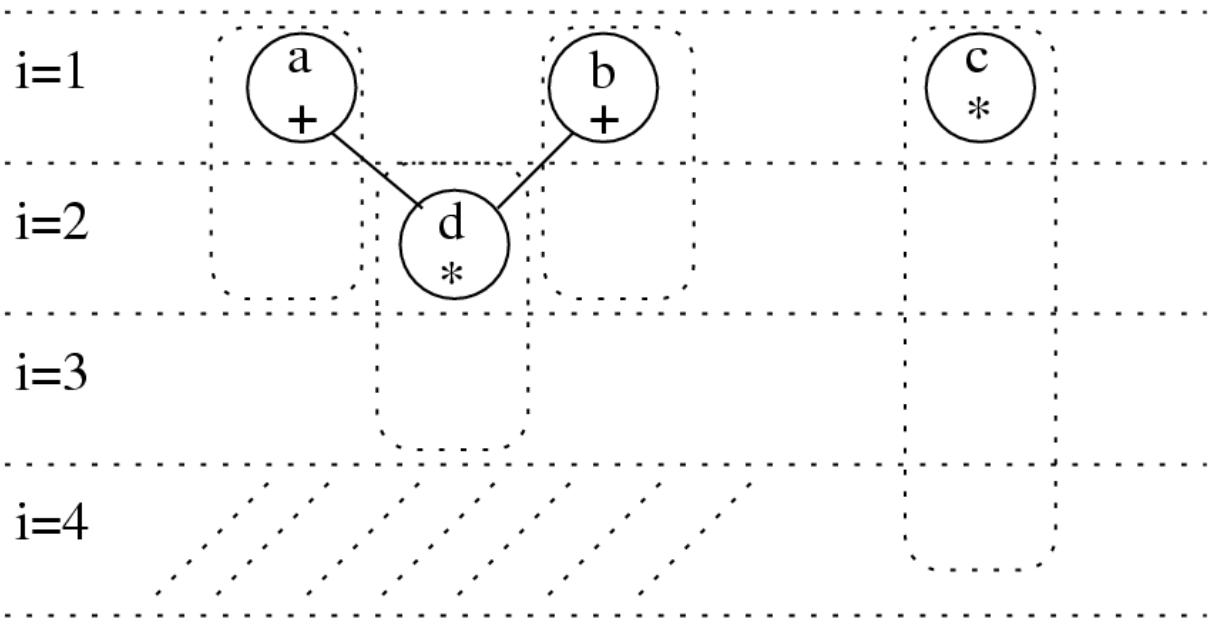
Dann ergeben sich die Gesamtkosten als:

$$C = \sum_{m \in M} \left(c_m * \sum_{\substack{k \in M \\ type(k) = m}} b_k \right) + \sum_{k1,k2} c_{k1,k2} * w_{k1,k2}$$

Ein einfaches Beispiel

Gegeben: `s := (u+v) * (w+x); t:=y*z;`

Für die erste Zuweisung seien drei, für die zweite vier Kontrollschritte zulässig.



ASAP/ALAP-Kontrollschritt-Bereiche:
$R(a) = [1,2]$; $R(b) = [1,2]$;
$R(c) = [1,4]$; $R(d) = [2,3]$;

Bausteinbibliothek:

Typ	Operationen
1	+
2	*
3	+,*

Ergebnisse in einem Kontrollschritt verfügbar: $\forall\, k$: $\ell(k)=1$.

Gleiche Kosten aller Bausteine: $\forall\, m$: $c_m=1$.

Zielfunktion für Maximierung

Zielfunktion für Maximierung:

$$C = -b_1\ -b_2\ -b_3\ -b_4\ -b_5\ -b_6$$

$$-w_{1,3}\ -w_{1,4}\ -w_{1,5}\ -w_{1,6}\ -w_{2,3}\ -w_{2,4}\ -w_{2,5}\ -w_{2,6}\ -w_{5,3}\ -w_{5,4}\ -w_{5,6}$$

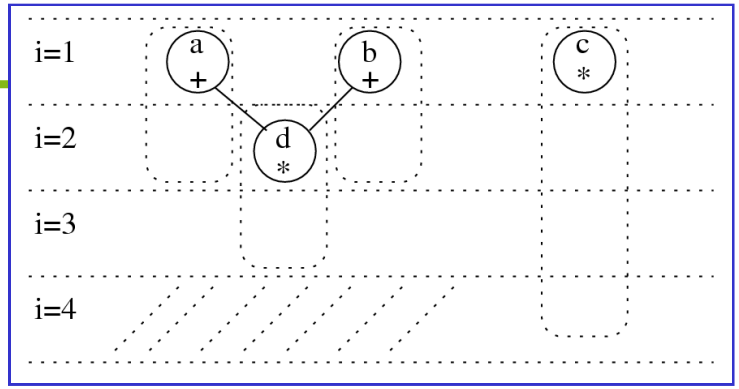
$$-w_{6,4}\ -w_{6,5}$$

$w_{i,j}$ nur für *chaining** () benötigt;

Annahme: kein *chaining* (immer Register zwischen arithmetischen Einheiten).

* Hintereinander geschaltete arithmetische Einheiten, die in demselben Kontrollschritt arbeiten

Operation assignment constraints (1)



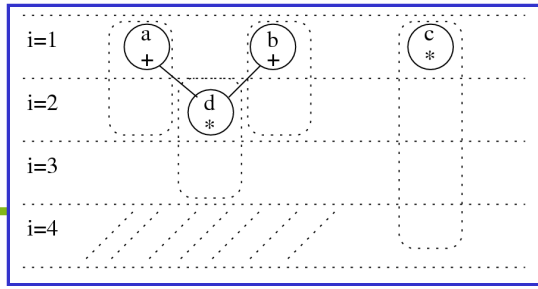
$$\forall j \in J: \sum_{i \in R(j)} \sum_{k \in K,\, j \text{ executable\_on } k} x_{i,j,k} = 1$$

1. $x_{1a1}+x_{1a2}+x_{1a5}+x_{1a6}+x_{2a1}+x_{2a2}+x_{2a5}+x_{2a6}=1$

   (die Operation a muss in Schritt 1 oder in Schritt 2 ausgeführt werden und an einen zur Addition fähigen Baustein gebunden werden);

2. Wie 1, jedoch für Operation b;

Operation assignment constraints (2)

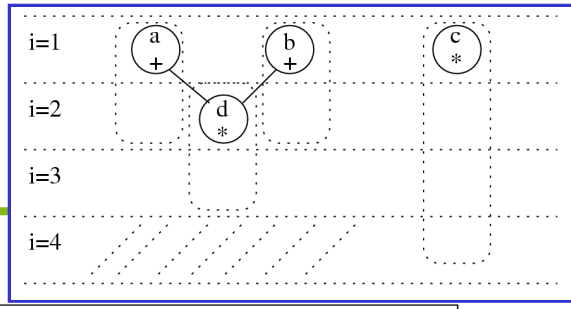


$\forall j \in J: \sum_{i \in R(j)} \sum_{k \in K,\, j \text{ executable\_on } k} x_{i,j,k} = 1$

3. $x_{2d3} + x_{2d4} + x_{2d5} + x_{2d6} + x_{3d3} + x_{3d4} + x_{3d5} + x_{3d6} = 1$
   (die Operation d muss in Schritt 2 oder in Schritt 3 ausgeführt werden und an einen zur Multiplikation fähigen Baustein gebunden werden);

4. $x_{1c3} + x_{1c4} + x_{1c5} + x_{1c6} + x_{2c3} + x_{2c4} + x_{2c5} + x_{2c6} + x_{3c3} + x_{3c4} + x_{3c5} + x_{3c6} + x_{4c3} + x_{4d4} + x_{4c5} + x_{4c6} = 1$
   (die Operation c kann in den Schritten 1 bis 4 ausgeführt werden und muss an einen zur Multiplikation fähigen Baustein gebunden werden).

Resource assignment constraints



$\forall\ k \in K$: $\forall i \in I$: $\sum_{j \in J} \sum_{i' \in [i-\ell(k)+1..i]} x_{i',j,k} \leq b_k$

Für k=1,2 (Addierer)

$$\begin{array}{|l|l|l|}\hline i=1 & x_{1a1} + x_{1b1} \leq b_1 & x_{1a2} + x_{1b2} \leq b_2 \\ i=2 & x_{2a1} + x_{2b1} \leq b_1 & x_{2a2} + x_{2b2} \leq b_2 \\ \hline \end{array}$$

Für k=3,4 (Mult.)

$$\begin{array}{|l|l|l|}\hline i=1 & x_{1c3} \leq b_3 & x_{1c4} \leq b_4 \\ i=2 & x_{2c3} + x_{2d3} \leq b_3 & x_{2c4} + x_{2d4} \leq b_4 \\ i=3 & x_{3c3} + x_{3d3} \leq b_3 & x_{3c4} + x_{3d4} \leq b_4 \\ i=4 & x_{4c3} \leq b_3 & x_{4c4} \leq b_4 \\ \hline \end{array}$$

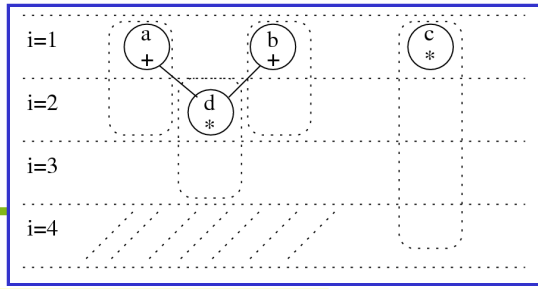
Für k=5 (Kombin.)

$$\begin{array}{|l|l|}\hline i=1 & x_{1a5} + x_{1b5} + x_{1c5} \leq b_5 \\ i=2 & x_{2a5} + x_{2b5} + x_{2c5} + x_{2d5} \leq b_5 \\ i=3 & x_{3c5} + x_{3d5} \leq b_5 \\ i=4 & x_{4c5} \leq b_5 \\ \hline \end{array}$$

Für k=6 (Kombin.)

$$\begin{array}{|l|l|}\hline i=1 & x_{1a6} + x_{1b6} + x_{1c6} \leq b_6 \\ i=2 & x_{2a6} + x_{2b6} + x_{2c6} + x_{2d5} \leq b_6 \\ i=3 & x_{3c6} + x_{3d6} \leq b_6 \\ i=4 & x_{4c6} \leq b_6 \\ \hline \end{array}$$

Precedence constraints

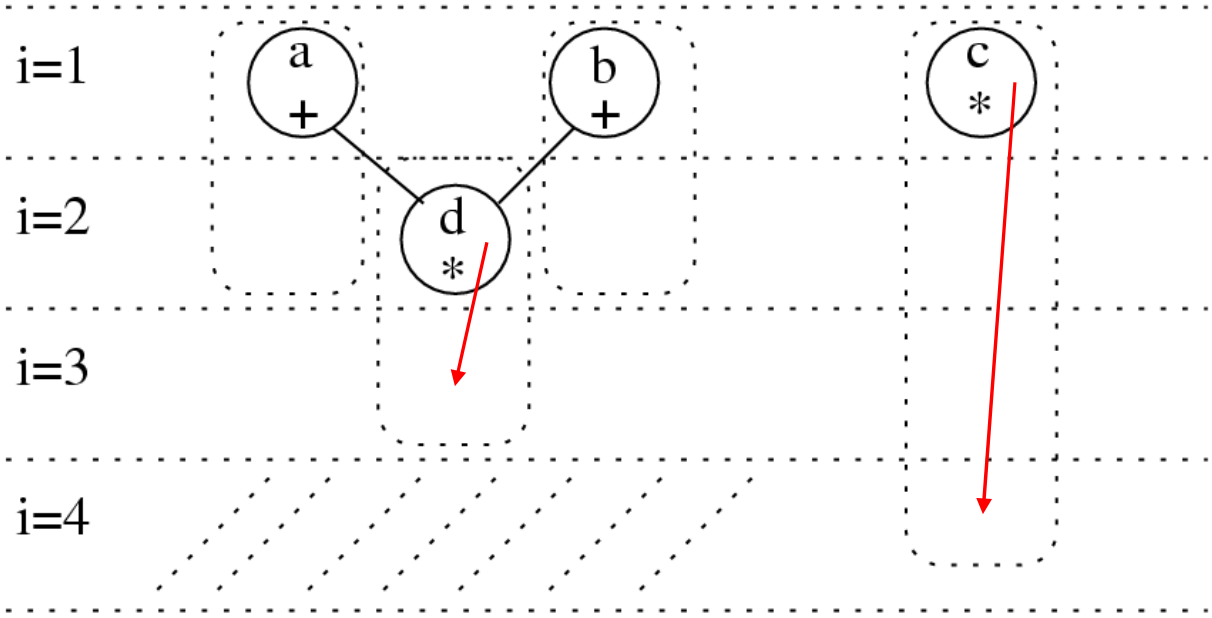


$\forall i \in R(j_1) \cap R(j_2)$: $\sum_k \sum_{i' \le i,\ i' \in R(j2)} x_{i,j2,k} + \sum_k \sum_{i' \ge i,\ i' \in R(j1)} x_{i,j1,k} \le 1$

1. $x_{2d3} + x_{2d4} + x_{2d5} + x_{2d6} + x_{2a1} + x_{2a2} + x_{2a5} + x_{2a6} \le 1$
   (Schritt 2 darf nicht a und d enthalten);
2. $x_{2d3} + x_{2d4} + x_{2d5} + x_{2d6} + x_{2b1} + x_{2b2} + x_{2b5} + x_{2b6} \le 1$
   (Schritt 2 darf nicht b und d enthalten).

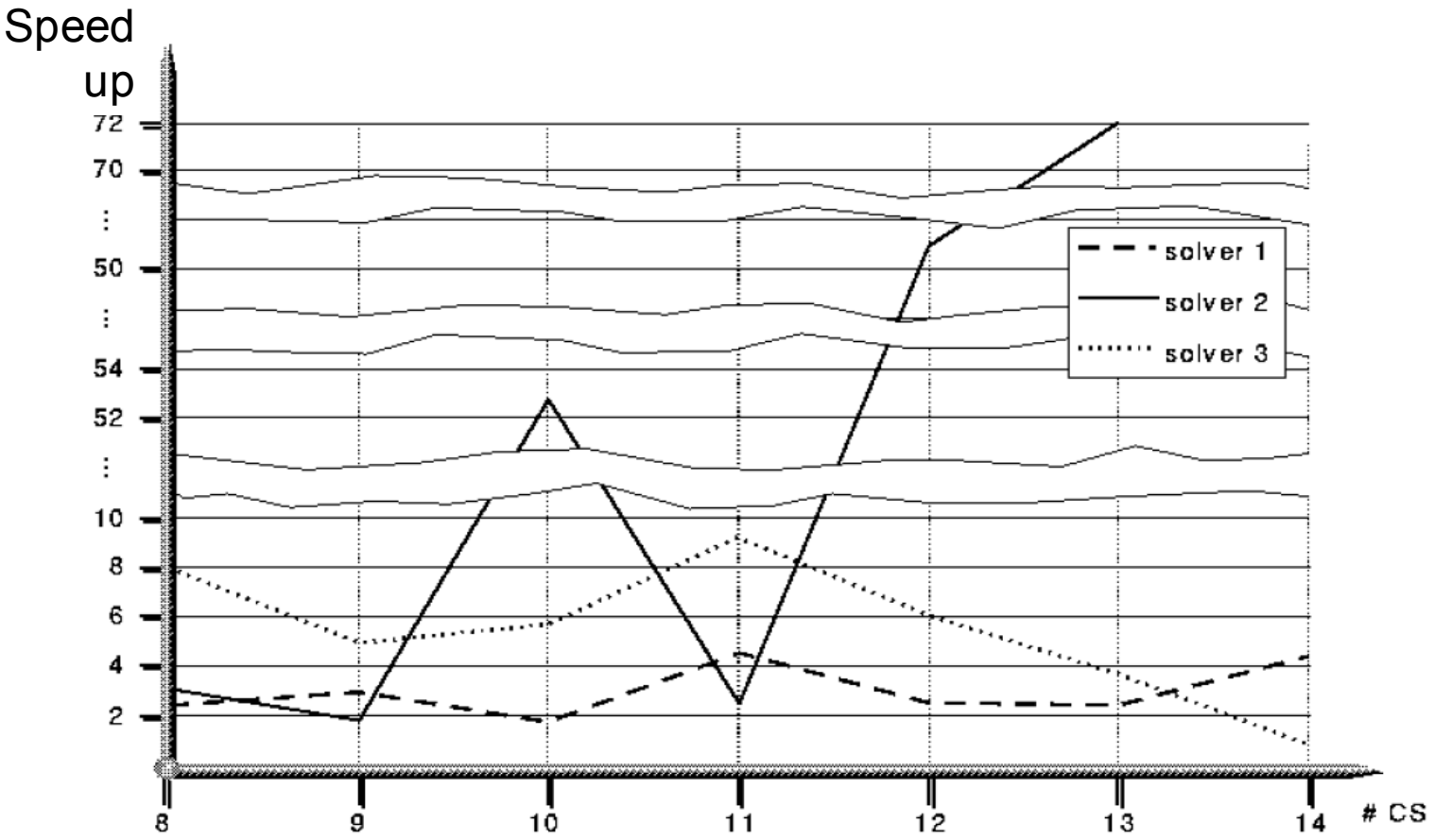
Lösung

Lösung:
Operation c: Schritt 4
Operation d: Schritt 3
Operation a und b: Schritte 1 und 2.



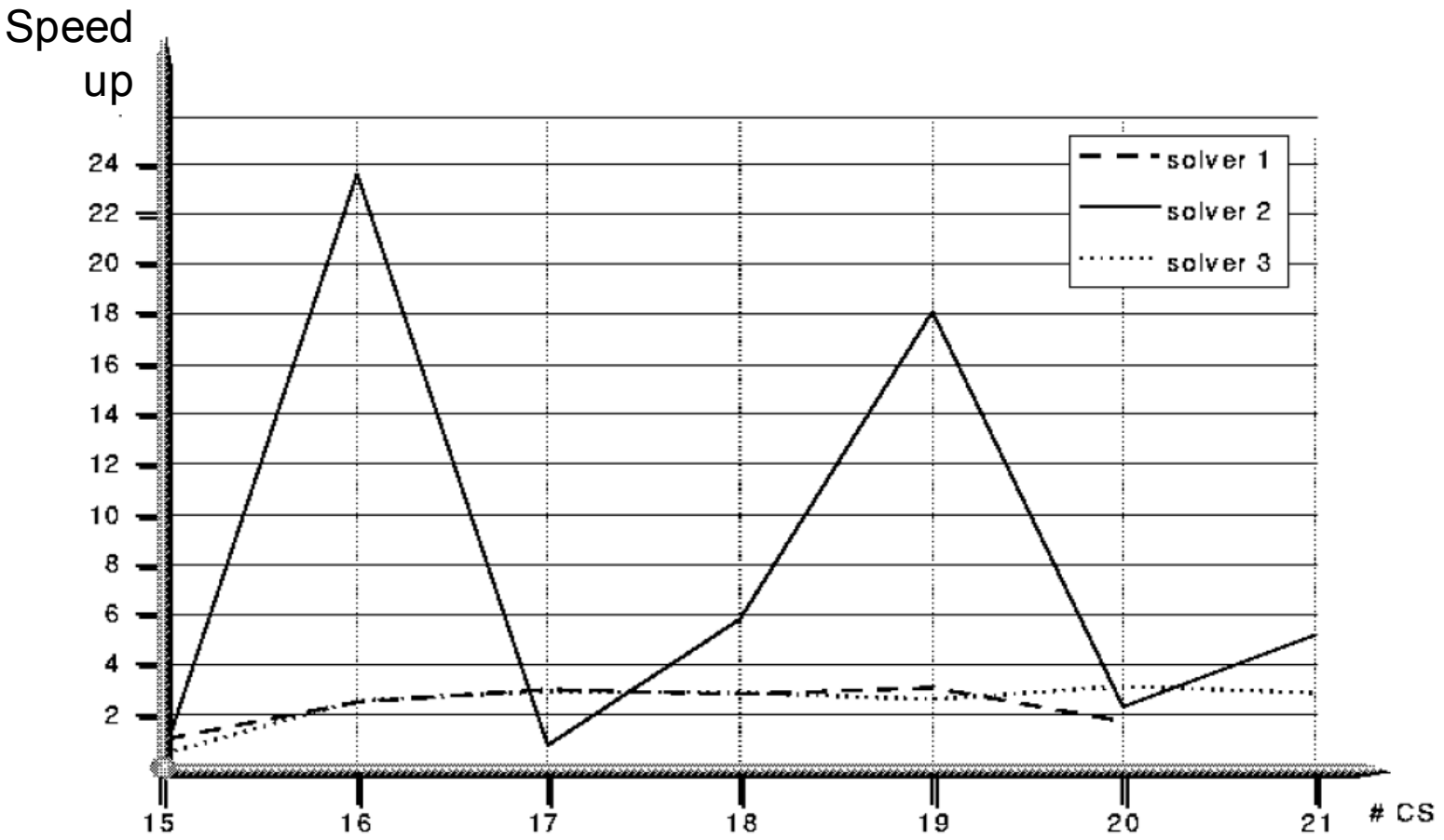
Beschleunigung durch Vermeidung von Isomorphie (2)

Beispiel: 3x3 Determinanten



Beschleunigung durch Vermeidung von Isomorphie

Beispiel: *Elliptical Wave Filter*



Ergebnis der Nutzung algebraischer Regeln (*Elliptical Wave Filter*)

Kontroll-schritte	Ohne Opti-mierung	Ersetzung von * durch Schiebe-operationen	Ausnutzung der Assoziativität	Ausnutzung der Assoziativität & von Schiebeoperationen
9	-	-	-	4a
10	-	-	-	4a
11	-	4a(ddierer)	-	3a
12	-	3a	3m, 3a	3a
13	-	3a	2m, 3a	3a
14	2m(ult.), 3a	3a	1m, 3a	2a
15	1m, 3a	2a	1m, 2a	2a

(Ohne algebraische Regeln) „optimale“ Lösung : 14 Kontrollschritte. Ausnutzung der algebraischen Regeln erlaubt die Reduktion des Hardware-Aufwandes bei gleicher Kontrollschritt-Zahl und die Reduktion der Kontrollschritt-Zahl.

Charakteristische Eigenschaften von IP-basierten Verfahren

- Es kann Optimalität bezüglich des gewählten Kostenmodells garantiert werden,
- es liegt ein präzises mathematisches Modell der zu lösenden Aufgabe vor,
- es kann formal nachgewiesen werden, welche Eigenschaften eine synthetisierte Implementierung besitzt,
- zusätzliche Bedingungen können verhältnismäßig leicht integriert werden.
- IP ist NP-vollständig, für große Beispiele Heuristiken erforderlich.
- IP-Modell gut als Ausgangsmodell für andere Methoden geeignet.

Charakteristische Eigenschaften von OSCAR

- Kopplung an kommerzielle Software.
- nutzt komplexe Baustein-Bibliotheken aus,
- unterstützt Bausteine mit unterschiedlichen Geschwindigkeiten.
- entscheidet, ob *chaining* sinnvoll ist,
- erlaubt die Angabe von Zeitbedingungen,
- nutzt algebraische Regeln aus.
- benutzt genetischen Algorithmus zur Wahl einer Kombination von Regeln.

Zusammenfassung

OSCAR

- Integration von *scheduling, allocation* und *assignment*
- Modellierung als Modell der Ganzzahligen Programmierung
- Nutzung algebraischer Regeln