3. Ablaufplanung

Zur Vorlesung

Embedded Systems

WS 14/15

Reiner Kolla

Allgemeine Klassifikation

Schedulingprobleme können nach folgenden 6 Kriterien unterschieden werden:

- statisch versus dynamisch
  - Statische Planungsprobleme treten dann auf, wenn zur Übersetzungs- bzw. Synthesezeit ein Ablaufplan zu erstellen ist. Sie treten in der Regel bei Hardware- oder hardwarenahen Implementierungstechniken auf.
  - Dynamische Ablaufplanung findet zur Laufzeit des ES statt. Sie treten meist bei Softwareimplementierungen von ES auf.
- präemptiv versus nicht präemptiv
  - Bei präemptiven Ablaufplanungsproblemen geht man davon aus, dass die Laufzeit zu planender Aufgaben sehr viel höher ist, als der Zeitaufwand, eine Aufgabe in ihrer Bearbeitung zu unterbrechen, und zu einem späteren Zeitpunkt weiter zu verarbeiten. Sie treten im allgemeinen nur bei Softwareimplementierungen auf.
  - Ist die Bearbeitungszeit der Aufgaben relativ kurz, oder die Bearbeitung gar nicht unterbrechbar, dann betrachtet man diese als atomar und nutzt nicht präemptive Planung.

Allgemeine Klassifikation -- ff

- Abhängigkeitsbedingungen
  - Bei Planungsproblemen mit Abhängigkeitsbedingungen (i.Allg. Datenabhängigkeiten) unterliegt die Bearbeitungsreihenfolge zeitlichen Bedingungen, die durch einen DAG gegeben sind. Planungsprobleme ohne Abhängigkeiten sind eigentlich selten. Wir betrachten sie als Spezialfall.
- Ressourcenbeschränkungen
  - Ablaufplanungsprobleme können zusätzlich Ressourcen-beschränkungen (Processing Units, Busse, Speicher) als Nebenbedingung haben. Wir werden sehen, dass unter Ressourcen-beschränkungen Ablaufplanungsprobleme sehr viel schwieriger sind als ohne Ressourcenbeschränkung.

Allgemeine Klassifikation -- ff

- Zeitbeschränkungen
  - Ablaufplanungsprobleme können auch zusätzlich Zeitbeschränkungen unterliegen. Man unterscheidet dazu weiter
  - absolute Zeitbeschränkungen
    - Deadlines (späteste zugelassene Endzeitpunkte)
    - Release times (früheste zugelassene Startzeitpunkte)
  - relative Zeitbeschränkungen: Bedingungen zwischen den Startzeitpunkten von Paaren von Aufgaben ($\tau(v)-\tau(u) \geq t$)
- periodisch versus aperiodisch
  - Periodische Ablaufplanungsprobleme kommen im Zusammenhang mit Iterationen vor. Jede Aufgabe wird periodisch zu bestimmten Iterationen eingeplant. Solche Aufgaben treten bei Schleifenoptimierungen in der Softwaredomäne oder bei DSP-Anwendungen auf.

3.1 Statische Ablaufplanung

Zur Vorlesung

<u>E</u>mbedded <u>S</u>ystems

WS 14/15

Reiner Kolla

3.1.1 Statische Ablaufplanung ohne Ressourcenbeschränkung

Problem: Statische Ablaufplanung ohne Ressourcenbeschränkung

Gegeben: Spezifikation $(G_S=(V,E),\ G_R=(V_R,E_R))$ G_S azyklisch und
$\forall\, v \in V_T : \alpha(v) = \infty$

Gesucht: Gültiger Ablaufplan $\tau : V_S \rightarrow \mathbf{N}_0$,

d.h. $\forall (u,v) \in E\colon\ \tau(v) - \tau(u) \geq w(u)$

mit minimaler Latenz

$$L(\tau) := \max \{\tau(v_i) + w(v_i);\ v_i \in V_S\} - \min \{\tau(v_j);\ v_j \in V_S\}$$

Dieses Problem kann mit einfachen Graphalgorithmen in polynomieller Laufzeit gelöst werden:

Der <u>A</u>s <u>S</u>oon <u>A</u>s <u>P</u>ossible - Algorithmus

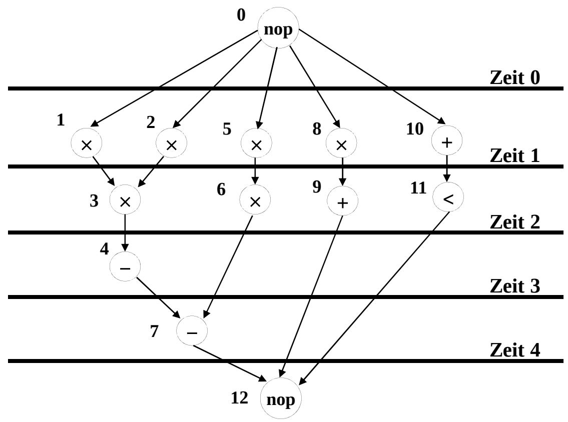
Idee: Starte jeden Knoten so früh wie möglich

Der Algorithmus
-- o.B.d.A. gebe es für jede Aufgabe nur einen Ressourcentyp

ASAP (G_S,w)
{ Berechne topologische Sortierung der Knoten aus V_S;
Sei v die Quelle von G_S , setze $\tau^{ASAP}(v)$:=0;
REPEAT
{ Sei v der nächste Knoten in der topologischen Sortierung;
$\tau^{ASAP}(v) := \max\{ \tau^{ASAP}(v_j)+w(v_j);\ \forall (v_j,v) \in E \}$;
} UNTIL alle Knoten aus V_S sind verplant;
RETURN τ^{ASAP};
}

Komplexität: $O(|V|+|E|)$

Ablaufplan mittel ASAP: Beispiel



Untere Schranke für die Latenz

Lemma

ASAP(G_S,G_R,w) liefert einen Ablaufplan mit minimaler Latenz und somit eine untere Schranke für die Latenz für Ablaufpläne mit Ressourcen- und Zeitbeschränkungen.

Beweis:

Man zeigt einfach durch Induktion nach der Tiefe eines Knotens v, dass in jedem gültigen Ablaufplan τ gilt: $\tau^{ASAP}(v) \leq \tau(v)$.

Da ferner für einen gültigen Ablaufplan τ o.E. an die Latenz $\min_v \tau(v) = 0$ annehmen kann, ist

$L^{ASAP} = \max\{\tau^{ASAP}(v_i) + w(v_i);\ v_i \in V_S\} \leq \max\{\tau(v_i) + w(v_i);\ v_i \in V_S\} = L(\tau)$

eine untere Schranke.

Diese gilt nun auch unter Hinzunahme weiterer Bedingungen, da diese die Latenz nur erhöhen können.

Der As Late As Possible -Algorithmus

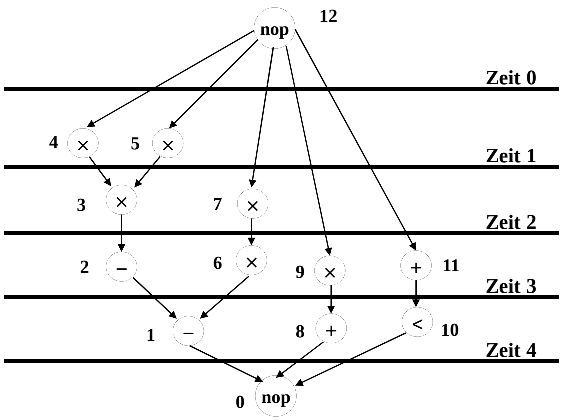
Idee: Ist eine Latenzschranke $\underline{L}$ vorgegeben, so besteht ein anderes Verfahren darin, jeden Knoten so spät wie möglich zu starten

Der Algorithmus
-- o.B.d.A. gebe es für jede Aufgabe nur einen Ressourcetyp

```
ALAP (G_S,w,L)
{    Berechne eine umgekehrte topologische Sortierung der
     Knoten aus V_S; Sei v die Senke von G_S, setze τ^ALAP(v):= L ;
     REPEAT
     {  Sei v der nächste Knoten in der umgekehrten topologischen
             Sortierung;
        τ^ALAP(v):=min{ τ^ALAP(v_j); ∀(v,v_j)∈ E } – w(v);
     } UNTIL alle Knoten aus V_S sind verplant;
     RETURN τ^ALAP;
}
```

Komplexität: $O(|V|+|E|)$

Ablaufplan mittels ALAP: Beispiel



3.1.2 Ablaufplanung mit Zeitbeschränkungen

Erinnerung

- Absolute Zeitbeschränkungen
  - Deadline: späteste Endzeitpunkte von Operationen
  - Releasezeiten: früheste Startzeitpunkte von Operationen
- Relative Zeitbeschränkungen
  - Zeitliche Relationen zwischen Paaren von Operationen

Beobachtung

Absolute Zeitbeschränkungen sind relative Zeitbeschränkungen zum Startknoten des Sequenzgraphen.

Wir können uns also auf die Betrachtung relativer Zeitbeschränkungen zurückziehen.

Relative Zeitbeschränkungen

Wir drücken relative Zeitbeschränkungen zwischen zwei Knoten v_i, v_j eines Sequenzgraphen G_S durch Zahlen $l_{ij} \in \boldsymbol{N}$ wie folgt aus:

- Minimumsbeschränkung $\tau(v_j) \geq \tau(v_i)+l_{ij}$, gibt an, dass die Aufgabe v_j frühestens l_{ij} Zeiteinheiten nach der Aufgabe v_i gestartet werden darf.
- Maximumsbeschränkung $\tau(v_j) \leq \tau(v_i)+l_{ij}$, gibt an, dass die Aufgabe v_j spätestens l_{ij} Zeiteinheiten nach der Aufgabe v_i gestartet werden muss.

Constraintgraph

Definition

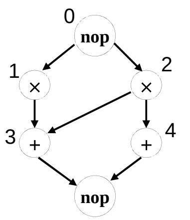
Gegeben sei ein Sequenzgraph G_S=(V,E), ein Ressourcegraph G_R, Ausführungszeiten w:$V_S \rightarrow N_0$ und eine Menge $\{l_{ij}\}$ von n relativen Zeitbeschränkungen.

Ein Constraintgraph $G_C=(V_C,E_C,d)$ ist ein kantengewichteter, gerichteter Graph mit einer Gewichtsfunktion d:$E_C \rightarrow \boldsymbol{Z}$, der sich wie folgt aus dem Sequenzgraph ergibt:

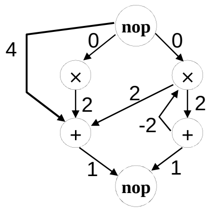
- $V_C=V$; $E_C=E \cup E\,'$ mit $|E\,'|=n$, wobei $E\,'$ eine Kante pro relativer Zeitbeschränkung enthält.
- Pro Minimumsbeschränkung l_{ij} gibt es eine Kante $(v_i,v_j) \in E\,'$ mit $d(v_i,v_j)=l_{ij}$
- Pro Maximumsbeschränkung l_{ij} gibt es eine Kante $(v_j,v_i) \in E\,'$ mit $d(v_j,v_i)=-l_{ij}$
- $\forall (v,u) \in E$ mit v ist nicht der Startknoten von G_S sei $d(v,u):=w(v)$
- Für alle restlichen Kanten e aus E_C sei $d(e)=0$

Illustration

Sequenzgraph G_S



Constraintgraph G_C



w(1)=w(2)=2
w(3)=w(4)=1

Minimumsbedingung: l_{03}=4 -- 3 startet frühestens nach 4 Ticks
Maximumsbedingung: l_{24}=2 -- 4 startet spätestens 2 Ticks nach 2

Algorithmus

Lemma

Die Existenz eines gültigen Ablaufplans bei vorgegebenen Zeitbeschränkungen, aber ohne Ressourcenbeschränkungen kann in polynomieller Zeit entschieden, und ein latenzminimaler, gültiger Ablaufplan τ, falls existent, berechnet werden.

Beweis:

Man kann die Berechnung eines latenzminimalen, gültigen Ablaufplanes auf die Berechnung des längsten Wegs von der Quelle zu allen Knoten zurückführen:

Sei *dist(u)* die Länge eines längsten Weges von der Quelle *s* nach *u* im Constraintgraphen. Dann erfüllt $\tau(u) = dist(u)$ alle Constraints.

Denn: für jede Kante (u,v) in G_C gilt sicher: $dist(u) + d(u,v) \leq dist(v)$.

damit gilt $\tau(u) + w(u) \leq \tau(v)$ für jede Kante aus E_S

$\tau(u) + l_{u,v} \leq \tau(v)$ für jede Minimumsbeschränkung, und

$\tau(v) - l_{u,v} \leq \tau(u) \Leftrightarrow \tau(u) + l_{u,v} \geq \tau(v)$ für jede Maximumsbeschränkung

Hat G_C positive Zyklen, existiert kein gültiger Ablaufplan.

Komplexität $O(|V_S| \times |E_C|)$ mit dem Bellman Ford Algorithmus (RGL WS0708)

3.1.3 Ablaufplanung mit Ressourcenbeschränkung

Optimierungsprobleme

- **Latenzminimierung mit Ressourcenbeschränkungen**:
  Gesucht ist bei gegebener Allokation α eine Implementierung, d.h. Ablaufplan τ und eine Bindung β, mit minimaler Latenz.
- **Kostenminimierung unter Latenzbeschränkung**:
  Gesucht ist bei Vorgabe einer Latenzschranke L eine Implementierung, d.h. Allokation α, Ablaufplan τ und eine Bindung β, die minimale Kosten hat.
- **Zulässiges Ablaufplanungsproblem**:
  Gesucht ist eine Implementierung bei einer gegebenen Latenzschranke L und einer gegebenen Allokation α.
- **Gewichtete Minimierung von Latenz und Kosten** unter Latenz- und Ressourcenbeschränkungen: Gesucht ist eine Implementierung, die bei gegebener Latenzschranke L und Allokation α eine Zielfunktion minimiert, die Kosten und Latenz gewichtet.

3.1.3.1 List Scheduling

List Scheduling ist eine Weiterentwicklung des ASAP-Verfahrens

- Aufgaben werden Schritt für Schritt beginnend beim ersten Zeitschritt geplant. Zu jedem Zeitschritt hält man für jeden Ressourcetypen r eine Liste K_r von Knoten, deren Vorgänger alle beendet, und die selbst noch nicht geplant sind, sowie eine Liste G_r von noch laufenden Operationen auf einer Ressource vom Typ r.
- Wähle nun zu jedem Ressourcetypen r eine maximale Menge $S_r \subseteq K_r$ so aus, dass $|S_r| + |G_r| \leq \alpha(r)$. Ist S_r eine echte Teilmenge von K_r, dann wähle die Elemente zu S_r in absteigender Priorität.
- Plane alle Knoten $v \in S_r$ zu jeder Ressource r mit $\tau(v) = t$,
  - nehme sie mit Beendigungszeit $t + w(v)$ nach G_r
  - Setze die Zeit $t = t + 1$ und nehme dann alle Knoten u mit $\tau(u) + w(u) \leq t$ aus den G_r heraus, nehme alle Nachfolger v von u, die keine laufenden oder ungeplanten Vorgänger mehr haben nach $K_{\beta(v)}$ hinzu und berechne ihre Priorität.

List Scheduling -- ff

Die Güte des berechneten Ablaufplans kann mit der Wahl der zu planenden Knoten beeinflusst werden. Im List Scheduling sind gebräuchlich:

Prioritätskriterien

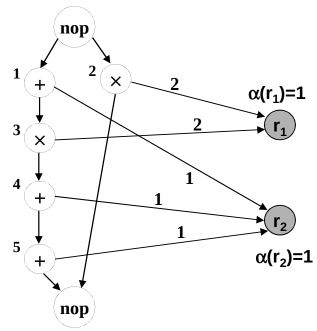
- Anzahl der Nachfolgeaufgaben der Aufgabe
- Länge des längsten Pfades von der Aufgabe zu einer Senke
- Mobilität der Aufgabe als Differenz ihrer Startzeiten nach ALAP- und ASAP-Planung ($slack(v) := \tau^{ALAP}(v) - \tau^{ASAP}(v)$)

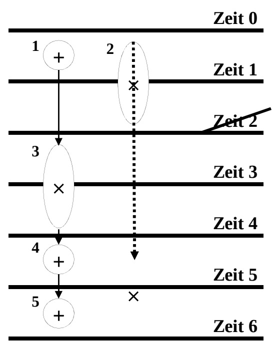
Komplexität: $O((|V_S| + |E_S|) * p)$

wo p der Zeitaufwand für Update der Listen und die Auswahl nach Prioritäten ist. ($p = O(1) \ldots O(\log |V_S|)$)

Vorsicht: List Scheduling ist (nur) eine Heuristik!

List Scheduling ist nur eine Heuristik!





3.1.3.2 Force Directed Scheduling [Paulin'89]

Wir betrachten nun ein Kräftemodell, das versucht Operationen auf geeignete Ausführungszeitpunkte zu schieben.

Eigenschaften

- dynamische Änderung des Modells nach Planung von Aufgaben
- Berücksichtigung direkter Nachbarschaften im Sequenzgraphen
- $|V_S|$ Updates des Kräftemodells, wobei nach jeder Planung einer Aufgabe das Modell aktualisiert wird.

Komplexität: $\Omega((|V_S|+|E_S|)^2)$

Force Directed Scheduling ff

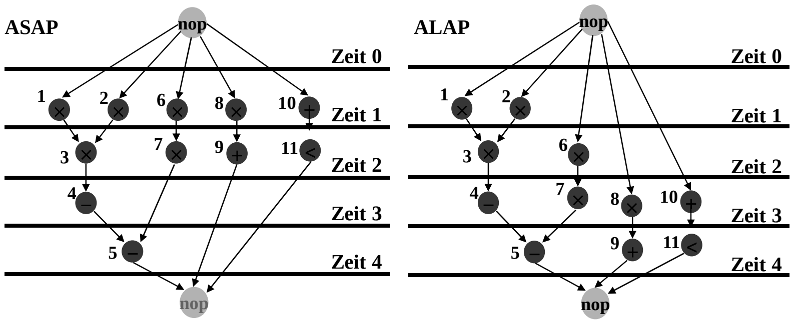
Begriffe:

- Wir betrachten zu jeder Ressource *k* die erwartete Auslastung $q_{k,t}$ zum Zeitpunkt *t*.
- Diese wird zu Beginn wie folgt initialisiert:
  - Zeitrahmen einer Aufgabe $v \in V_S$: $[\tau^{ASAP}(v), \tau^{ALAP}(v)]$,
  - Mobilität $slack(v) = \mu(v) = \tau^{ALAP}(v) - \tau^{ASAP}(v)$
  - Ausführungswahrscheinlichkeit $p_{v,t}$, von $v \in V_S$ in einem Zeitschritt *t*

$$p_{v,t} := \begin{cases} \dfrac{1}{slack(v)+1}, & \text{falls } t \in [\tau^{ASAP}(v) : \tau^{ALAP}(v)] \\ 0 & \text{sonst} \end{cases}$$

$$\boxed{q_{k,t} := \sum_{v \text{ mit } (v,k) \in E_R} p_{v,t}}$$

Zeitrahmen, Ausführungswahrscheinlichkeiten und Auslastung der Resourcen am Beispiel:



	m1	m2	m3	a4	a5	m6	m7	m8	a9	a10	a11	MUL	ALU
t1	1	1				1/2		1/3		1/3		2,83	0,33
t2			1			1/2	1/2	1/3	1/3	1/3	1/3	2,33	1,00
t3				1			1/2	1/3	1/3	1/3	1/3	0,83	2,00
t4					1				1/3		1/3	0,00	1,66

Das Kräftemodell

Wir ordnen nun den Aufgaben *v* **Anziehungskräfte** $F_{v,t}$ zu den Zeitpunkten *t* zu. Je höher die Kraft $F_{v,t}$, umso besser sei die Planung von *v* zum Zeitpunkt *t*.

Selbstkraft:
Offensichtlich ist es gut, eine Aufgabe *v* zu einem Zeitpunkt auszuführen, zu dem eine relativ geringe Auslastung der Ressource $\beta(v)$ zu erwarten ist. Wir lassen daher auf jede Aufgabe *v* für jeden Zeitpunkt *t* in ihrem Zeitrahmen zunächst eine Selbstkraft

$$F^S_{v,t} := \underbrace{\frac{1}{slack(v)+1} \cdot \sum_{i=\tau_0(v)}^{\tau_1(v)} q_{\beta(v),i}}_{:= \overline{q}_v} - q_{\beta(v),t}$$

wirken. Diese ist positiv, wenn die Auslastung der Ressource $\beta(v)$ zum Zeitpunkt *t* niedriger ist, als der Mittelwert der Auslastung über den für *v* noch verfügbaren Zeitrahmen $[\tau_0(v):\tau_1(v)]$.

Das Kräftemodell ff

Selbstkräfte alleine berücksichtigen aber noch nicht die Abhängigkeit der Operationen untereinander. Dieser tragen wir zumindest für die direkte Nachbarschaft noch Rechnung:

Nachbarschaftskräfte:
Wenn wir eine Aufgabe *v* zum Zeitpunkt *t* planen, ändern sich ggf. die Zeitrahmen für die Vorgänger und Nachfolger von *v*, und damit auch die Auslastung ihrer Ressourcen. Es scheint sinnvoll zu sein, *v* zum Zeitpunkt *t* zu planen, wenn die mittlere Auslastung der Ressourcen seiner Nachbarn über deren Zeitrahmen sinkt. Also setzen wir

$$F_{v,t}^{u} := \underbrace{\frac{1}{slack(u)+1} \cdot \sum_{i=\tau_0(u)}^{\tau_1(u)} q_{\beta(u),i}}_{:=\overline{q}_u} - \underbrace{\frac{1}{slack'(u)+1} \cdot \sum_{i=\tau'_0(u)}^{\tau'_1(u)} q'_{\beta(u),i}}_{:=\overline{q}'_u}$$

für jeden Nachfolger und Vorgänger *u* als zusätzliche Kraft ein. *slack'*, τ'_i und *q'* seien die Mobilitäten, Zeitrahmen und Auslastungen, die sich durch Planen von *v* zum Zeitpunkt *t* ergeben würden.

Das Kräftemodell ff

Wir drücken nun die Affinität einer Aufgabe *v* zu einem Zeitpunkt *t* durch die Kraft

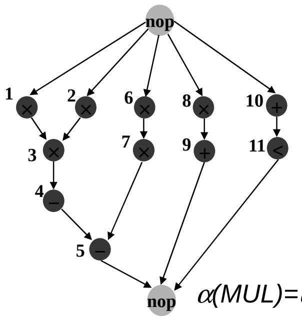
$$\boxed{F_{v,t} := F_{v,t}^{S} + \sum_{(u,v)\in E} F_{v,t}^{u} + \sum_{(v,w)\in E} F_{v,t}^{w}}$$

aus.
Mit diesem Kräftemodell hat man nun zwei Möglichkeiten:

❶ Benutze Kräfte als Priorität beim List Scheduling, d.h. wähle aus der Menge der Kandidaten zum Zeitpunkt *t* eine Teilmenge *S* mit den höchsten Kräften zu *t*.
  ☞ Minimiere Latenz unter Ressourcenbeschränkung

❷ Benutze Kräfte, um Knoten für Knoten einen Zeitpunkt zuzuordnen
  ☞ Minimiere Ressourcenbedarf unter Latenzschranke <u>*L*</u>

Beispiel zu ❶: Listscheduling mit Kräften



$\alpha(MUL)=\alpha(ALU)=2$

i	$\tau_0(i)$	$\tau_1(i)$	$\mu(i)+1$	$\overline{q}_i$
1	1	1	1	2.83
2	1	1	1	2.83
3	2	2	1	2.33
4	3	3	1	2.00
5	4	4	1	1.66
6	1	2	2	2.58
7	2	3	2	1.58
8	1	3	3	2.00
9	2	4	3	1.55
10	1	3	3	1.11
11	2	4	3	1.55

$q_{j,t}$	*MUL*	*ALU*
1	2.83	0.33
2	2.33	1.00
3	0.83	2.00
4	0.0	1.66

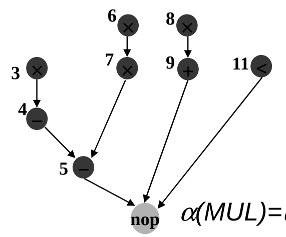
$F_{i,t}^{S}$	t=1
1	0
2	0
6	-0.25
8	-0.83
10	0.78

$F_{1,1}^{3}=0$ $F_{2,1}^{3}=0$ $F_{6,1}^{7}=0$ $F_{8,1}^{9}=0$ $F_{10,1}^{11}=0$

$F_{i,t}$	t=1
1	0
2	0
6	-0.25
8	-0.83
10	0.78

Wähle *1,2,10*

Beispiel zu ❶: Listscheduling mit Kräften ff



$\alpha(MUL)=\alpha(ALU)=2$

i	$\tau_0(i)$	$\tau_1(i)$	$\mu(i)+1$	$\overline{q}_i$
3	2	2	1	2.50
4	3	3	1	1.50
5	4	4	1	1.83
6	2	2	1	2.50
7	3	3	1	1.50
8	2	3	2	2.00
9	3	4	2	1.83
11	2	4	3	1.33

$q_{j,t}$	*MUL*	*ALU*
2	2.50	0.33
3	1.50	1.83
4	0.0	1.83

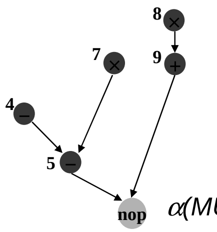
$F_{i,t}^{S}$	t=2
3	0
6	0
8	-0.50
11	1.00

$F_{3,2}^{4}=0$ $F_{6,2}^{7}=0$ $F_{8,2}^{9}=0$

$F_{i,t}$	t=2
3	0
6	0
8	-0.50
11	1.00

Wähle *3,6,11*

Beispiel zu ❶ : Listscheduling mit Kräften -- ff



$\alpha(MUL)=\alpha(ALU)=2$

i	$\tau_0(i)$	$\tau_1(i)$	$\mu(i)+1$	$\overline{q}_i$
4	3	3	1	1.00
5	4	4	1	2.00
7	3	3	1	2.00
8	3	3	1	2.00
9	4	4	1	2.00

$q_{j,t}$	MUL	ALU
3	2.00	1.00
4	0.0	2.00

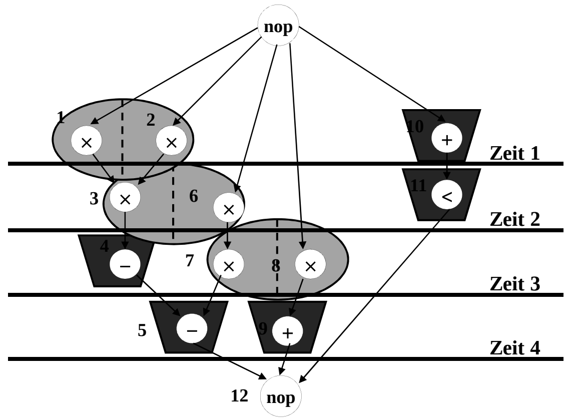
$F^S_{i,t}$	t=3
4	0
7	0
8	0

$F^5_{4,3}=0$ $F^5_{7,3}=0$ $F^9_{8,3}=0$

$F_{i,t}$	t=3
4	0
7	0
8	0

Wähle ***4,7,8***
danach ***5,9***

Beispiel zu ❶ : Listscheduling -- das Resultat



Minimierung der Ressourcen unter Latenzschranke

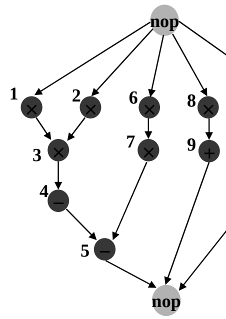
Force directed scheduling (G_S,G_R,$\underline{L}$)

- Initialisiere für die gegebene Latenzschranke $\underline{L}$ alle Zeitrahmen, Auslastungen, etc.
- Solange noch ein Knoten ungeplant ist
  - Wähle *(v,t)* so, dass die Kraft $F_{v,t}$ maximal unter allen *(v,t)*
  - Plane *v* zum Zeitpunkt *t,*
  - Berechne die Auswirkungen auf alle Kräfte

Das Verfahren berechnet einen korrekten Ablaufplan, da eine Aufgabe stets in dem ihr noch verfügbaren Zeitrahmen eingeplant wird.

Die Auslastung der Ressourcen wird durch das Kräftemodell heuristisch minimiert.

Beispiel zu ❷ : Resourcenminimierung Kräften



$\underline{L}=4$

Wir erhalten den gleichen Ablaufplan wie eben.

siehe Excel File **“FDS.xls”**

Verallgemeinerung auf unterschiedliche Ausführungsdauern

- Im Allgemeinen kann die Ausführungsdauer einer Aufgabe durch eine Ressource länger als ein Zeitschritt dauern.

➔ Bestimmung auf den Zeitrahmen

➔ Berechnung der Ausführungswahrscheinlichkeiten

Beispiel: Aufgabe v_i mit $(v_i,\text{MUL}) \in E_R$ und d(MUL)=2

- Wird v_i in Zeitschritt t geplant, so wird $p_{i,t}$ und $p_{i,t+1}$ auf 1 gesetzt
- Ist der Zeitrahmen von v_i durch $[t,t+1]$ gegeben, so kann v_i entweder in Zeitschritt t als auch in Zeitschritt t+1 gestartet werden, so dass gilt:
  - $p_{i,t}$=1/2,
  - $p_{i,t+1}$ =1 und
  - $p_{i,t+2}$ =1/2

3.1.3.3 Ablaufplanung mittels ILP

- Ablaufpläne können auch mit **Integer Linear Programming** (**ILP**), also ganzzahliger linearer Programmierung berechnet werden.
- Gibt es einen gültigen Ablaufplan (d.h. Ablaufplan und Bindung), so berechnet **ILP** auch einen solchen.

Bemerkung:

Ganzzahlige lineare Programme können mittels Branch&Bound gelöst werden. Die Lösung der linearen Relaxation wird dabei als untere Schranke für die beste ganzzahlige Lösung benutzt. Aufspaltung erfolgt durch Hinzufügen von linearen Ungleichungen für Variablen, die in der linearen Relaxation noch nicht ganzzahlig sind.

(vgl. Rechnergestütztes Layout WS07/08)

Das Integer Programm

Sei im folgenden $l_i = \tau^{ASAP}(v_i)$ und $h_i = \tau^{ALAP}(v_i)$.

- $x_{i,t} \in \{0,1\}$ $\qquad \forall v_i \in V_S\ \forall t$ mit $l_i \le t \le h_i$;
- $\sum_{t=l_i}^{h_i} x_{i,t} = 1 \qquad \forall v_i \in V_S$
- $\sum_{t=l_j}^{h_j} t \cdot x_{j,t} - \sum_{t=l_i}^{h_i} t \cdot x_{i,t} \ge w_i \qquad \forall (v_i, v_j) \in E_S$
- $\sum_{i \text{ mit } (v_i,k) \in V_R} \; \sum_{p=max\{0,t-h_i\}}^{min\{w_i-1,t-l_i\}} x_{i,t-p} \le \alpha(k)$

$\forall k \in V_T\ \forall t$ mit $1 \le t \le \max\{h_i;\ v_i \in V_S\}$

Erläuterung des linearen Programms

- Die binäre Variable $x_{i,t}$ drückt die Ablaufplanung aus. Sie ist genau dann 1, wenn Aufgabe v_i zum Zeitpunkt t gestartet wird, also wenn $\tau(v_i)=t$ gilt.
- Aufgabe v_i darf nur genau einmal geplant werden
- Es gilt $\sum_{t=l_i}^{h_i} t \cdot x_{i,t} = \tau(v_i)$ und somit sagt die Bedingung aus, dass Aufgabe v_j frühestens w_i Zeitschritte später als Aufgabe v_i geplant werden darf, wenn es eine Datenabhängigkeit zwischen Aufgabe v_i und Aufgabe v_j gibt.
- **Ressourcenbeschränkung:** Man überlege sich, dass die Ressource $\beta(v_i)$ nur durch v_i belegt sein kann, wenn $\tau(v_i) \le t \le \tau(v_i)+w_i-1$ gilt.