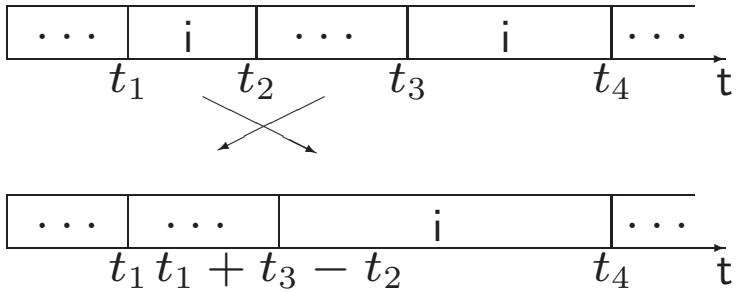
Einmaschinenprobleme

Bemerkung: Bei vielen Einmaschinenproblemen bringen Unterbrechungen keine Vorteile, da eine optimale Lösung mit *non-preemptive* Eigenschaft existiert. Dies muss nicht für $r_i \geq 0$ gelten.

Erläuterung: Sei γ ein reguläres Kriterium und gelte $r_i = 0$ für alle $i \in I$. Im optimalen Schedule sei die Bearbeitung von Auftrag A_i unterbrochen. In der folgenden Abbildung wird gezeigt, wie durch Tausch erreicht wird, dass die Bearbeitung von Auftrag A_i ohne Unterbrechung erfolgt.



Übergang zu einem Schedule ohne Unterbrechung

Aufgrund der Regularität des Kriteriums wird durch den Tausch der Zielfunktionswert γ nicht größer. Die Unterbrechung ist entfernt, eine neue entsteht nicht, also existiert eine optimale Lösung ohne Unterbrechungen. Im Fall $r_i \geq 0$ kann dagegen Unzulässigkeit entstehen!

Einmaschinenproblem mit f_{max}-Zielfunktion

Problem: $1 \mid prec \mid f_{max} = \max_i\{f_i(C_i)\}$, alle $f_i(C_i)$ nichtfallend (reguläres Kriterium).

Bezeichnungen: $n(i)$: Anzahl der direkten Nachfolger von Auftrag A_i im $prec$-Graph;
Die Menge I^* enthält die Indizes aller noch anzuordnenden Aufträge.

Algorithmus zur Lösung des Problems $1 \mid prec \mid f_{max}$

Eingabe: Vorranggraph durch Adjazenzmatrix A, $\forall\, i \in I$: $p_i,\ f_i(C_i)$;
Ausgabe: $\forall\, k \in I$: $\pi(k)$ (Index vom Auftrag an Position k)

BEGIN FOR $i := 1$ **TO** n **DO** $n(i) := \sum_{j=1}^{n} a_{ij}$; $I^* = \{1, \ldots, n\}$;

FOR $k := n$ **DOWNTO** 1 **DO**
BEGIN Bestimme i^* mit $n(i^*) = 0$ und $f_{i^*}(\sum_{i \in I^*} p_i) = \min_{i' \in I^*}\{f_{i'}(\sum_{i \in I^*} p_i)\}$;
$I^* := I^* \backslash \{i^*\}$; $n(i^*) := \infty$; $\pi(k) := i^*$;
FOR $i := 1$ **TO** n **DO IF** $a_{ii^*} = 1$ **THEN** $n(i) := n(i) - 1$; **END**;
END.

Einmaschinenprobleme mit L_{max}-Zielfunktion

Satz 1. *Das Problem* $1 \mid r_i \geq 0 \mid L_{max}$ *gehört zur Klasse* $\mathbb{NP}$*-hard.*

Bemerkung: Folgende Spezialfälle sind in $O(n \log n)$ Schritten lösbar:

1. Das Problem $1 \mid r_i = r \mid L_{max}$ wird durch die EDD-Regel gelöst werden.
2. Das Problem $1 \mid r_i \geq 0 \mid L_{max}$ mit $d_i = d$ für alle Aufträge wird durch folgende Regel gelöst: 'Ordne die Aufträge nach nichtfallenden Bereitstellungsterminen!'
3. Das Problem $1 \mid r_i \geq 0, p_i = 1 \mid L_{max}$ wird durch die folgende Regel gelöst: 'Teile jedem Zeitpunkt einen verfügbaren Auftrag mit kleinstem Fälligkeitstermin zu!'

Der Beweis erfolgt durch Vertauschungsargumente.

Hinweis: Wenn die Knoten des Vorranggraphen topologisch sortiert sind und die Bereitstellungszeiten bzw. die Due Dates bezüglich des Vorranggraphen modifiziert werden, dann lösen die obigen drei Regeln auch die Probleme: $1 \mid prec, r_i = r \mid L_{max}$, $1 \mid prec, r_i \geq 0 \mid L_{max}$ mit $d_i = d$ und $1 \mid prec, r_i \geq 0, p_i = 1 \mid L_{max}$.

Definition 1: Die Knoten des Vorranggraphen sind topologisch sortiert, falls: $i \rightarrow k \;\Rightarrow\; i < k$.

Topologische Sortierung der Knoten eines azyklischen Digraphen

Algorithmus Topologische Sortierung der Knoten eines Digraphen

Eingabe: $G = (V, E)$;
Ausgabe: $\forall\, v \in V:\ \alpha(v)$ ist die neue Knotennummer, falls der Graph zyklenfrei ist.
BEGIN
  $k := 1$;
  WHILE $\exists\, v \in V$ ohne Vorgänger **DO**
    BEGIN
      $\alpha(v) = k;\ \ V := V \setminus \{v\}$;
      Streiche alle Kanten (v, w) aus G;
      $k := k + 1$;
    END;
  IF $V \neq \emptyset$ **THEN** G hat Zyklen, d. h. es existiert keine topologische Sortierung;
END.

Modifikation der Bereitstellungszeiten

Ausgang: Topologisch sortierter Vorranggraph mit Adjazenzmatrix A;

Ziel: Modifikation der Bereitstellungszeiten so, dass für jede Vorrangbedingung $A_i \rightarrow A_k$ gilt: $r_k \geq r_i + p_i$.

Algorithmus Modifikation der Bereitstellungszeiten $\mathbf{r_i}$

Eingabe: Adjazenzmatrix A des $prec$-Graphen, $\forall\, i \in I:\ r_i$;
Ausgabe: Modifizierte Bereitstellungszeiten r_i.
BEGIN
  FOR $i := 1$ **TO** $n-1$ **DO**
    FOR $k := i+1$ **TO** n **DO**
      IF $a_{ik} = 1$ **THEN** $r_k := \max\{r_k,\ r_i + p_i\}$;
END.

Modifikation der Fälligkeitstermine

Ausgang: Topologisch sortierter Vorranggraph mit Adjazenzmatrix A;

Ziel: Modifikation der Fälligkeitstermine so, dass für jede Vorrangbedingung $A_i \rightarrow A_k$ gilt: $d_i \leq d_k - p_k$.

Algorithmus Modifikation der Fälligkeitstermine $\mathrm{d_i}$

Eingabe: Adjazenzmatrix A des $prec$-Graphen, $\forall\, i \in I :\ d_i$;
Ausgabe: Modifizierte Due Dates d_i.
BEGIN
  FOR $k := n$ **DOWNTO** 2 **DO**
    FOR $i := k - 1$ **DOWNTO** 1 **DO**
      IF $a_{ik} = 1$ **THEN** $d_i := \min\{d_i,\ d_k - p_k\}$;
END.

Lösung des Problems $1 \mid prec, pmtn, r_i \geq 0 \mid L_{max}$

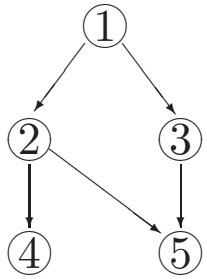
Satz 2. *Das Problem* $1 \mid prec, pmtn, r_i \geq 0 \mid L_{max}$ *wird durch die Anwendung folgender Regel in* $O(n^2)$ *Schritten optimal gelöst:*

- *Beginne mit dem Auftrag mit kleinstem* r_i*-Wert.*
- *Zu jedem Zeitpunkt, der entweder ein Bereitstellungs- oder ein Fertigstellungszeitpunkt ist, ordne den Auftrag mit kleinstem modifizierten Fälligkeitstermin an, dessen Vorgänger schon fertig sind.*

Beweis: kann als Übungsaufgabe geführt werden

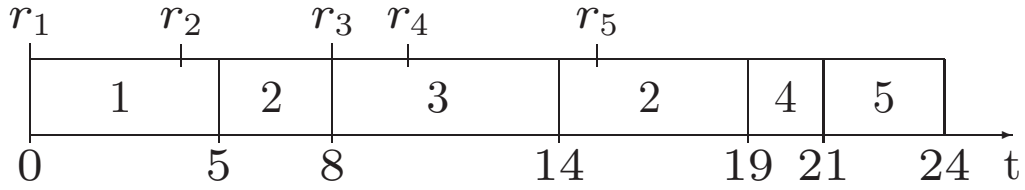
Beispiel 1. Betrachtet wird ein Problem mit fünf Aufträgen und den folgenden Werten r_i, p_i, d_i. Die Werte d_i' ergeben sich durch Modifikation bei Vorrangbedingungen:

Beispiel für Problem: $1 \mid prec, pmtn, r_i \geq 0 \mid L_{max}$



i	1	2	3	4	5
r_i	0	4	8	10	15
p_i	5	8	6	2	3
d_i	6	20	15	18	22
d'_i	6	16	15	18	22

Eingangsdaten und modifizierte Due Dates



Die Anwendung der Regel liefert den obigen optimalen Schedule. Neue Entscheidungen für die Bearbeitungsreihenfolge der Aufträge müssen an den Zeitpunkten 0, 4, 5, 8, 10, 14, 15, 19, 21 getroffen werden. Es gilt $C' = (5, 19, 14, 21, 24)$ und $L_{max} = 3$.

Lösung des Problems $1 \mid\mid \sum U_i$

Bezeichnungen: I: Menge der Indizes aller noch anzuordnenden Aufträge, I^*: Menge der Indizes aller angeordneten Aufträge, I^v: Menge der Indizes aller verspäteten Aufträge.

Algorithmus zur Lösung des Problems $1 \mid\mid \sum U_i$

Eingabe: n, $\forall\, i \in I:\ p_i, d_i$ mit $d_1 \leq \ldots \leq d_n$;
Ausgabe: $\forall\, k \in I:\ \pi(k)$.
BEGIN $I^* := \emptyset;\ \ I^v := \emptyset;\ \ I := \{1, \ldots, n\};$
  FOR $i = 1$ **TO** n **DO**
    BEGIN $I^* := I^* \cup \{i\};\ \ I := I \backslash \{i\}$; **IF** $\sum_{i \in I^*} p_i > d_i$ **THEN**
      BEGIN Bestimme Auftrag A_l mit $p_l = \max_{i \in I^*}\{p_i\}$; $I^v := I^v \cup \{l\};\ \ I^* := I^* \backslash \{l\}$; **END;**
    END;
  $k := 1;\ \ r := |I^*| + 1;$
  FOR $i := 1$ **TO** n **DO**
    IF $i \in I^*$ **THEN** $(\pi(k) := i\ \wedge\ k := k + 1)$ **ELSE** $(\pi(r) := i\ \wedge\ r := r + 1);$
END.

Das Problem $J2 \mid n_i \leq 2 \mid C_{max}$

- n_i bezeichnet die Anzahl der Operationen für Auftrag A_i.
- I_1 bzw. I_2 enthalten alle Aufträge, die nur auf M_1 bzw. M_2 bearbeitet werden,
- I_{12} enthält alle Aufträge mit $M_1 \rightarrow M_2$,
- I_{21} enthält alle Aufträge mit $M_2 \rightarrow M_1$.

Lösung des Problems $\mathbf{J2 \mid n_i \leq 2 \mid C_{max}}$

Eingabe: Matrizen P und MO;
Ausgabe: Organisatorischen Reihenfolgen des optimalen Schedules.
BEGIN
S1: Berechne die optimale Reihenfolge π_{12} für das Flow-Shop Problem mit der Auftragsmenge I_{12}.
S2: Berechne die optimale Reihenfolge π_{21} für das Flow-Shop Problem mit der Auftragsmenge I_{21}.
S3: Auf M_1: Bearbeite zuerst alle Aufträge aus I_{12} entsprechend π_{12}, dann alle Aufträge aus I_1 und schließlich alle Auftrage aus I_{21} entsprechend π_{21}.
S4: Auf M_2: Bearbeite zuerst alle Aufträge aus I_{21} entsprechend π_{21}, dann alle Aufträge aus I_2 und schließlich alle Aufträge aus I_{12} entsprechend π_{12}.
END.

Aktive Schedules für das Job-Shop Problem

Enumeration der aktiven Schedules für das Job-Shop Problem

Eingabe: Bearbeitungsmatrix P, Matrix der technologischen Reihenfolgen MO;
Ausgabe: Menge aller aktiven Schedules.
BEGIN

Step 1: $SIJ := \{(i,j) \mid mo(ij) = 1\}$; für alle $(i,j) \in SIJ: \; r_{ij} = 0$;

Step 2: Berechne den frühestmöglichen Bearbeitungsendzeitpunkt von allen Operationen aus SIJ: $\quad c_{i^*j^*} := \min\limits_{(i,j)\in SIJ}\{r_{ij} + p_{ij}\}$;

Step 3: Berechne die Menge M^* der Operationen auf M_{j^*}, bei deren Einfügung ein aktiver Plan entsteht: $M^* = \{(i,j^*) \in SIJ \mid r_{ij^*} < c_{i^*j^*}\}$.

Step 4: Verzweige durch Einfügung jeder Operation aus M^* als nächsten Auftrag in der organisatorischen Reihenfolge auf M_{j^*};
Falls noch nicht alle Operationen angeordnet sind, gehe zu Step 5, sonst: ENDE.

Step 5: Für jeden Fall wird die entsprechende Operation aus SIJ entfernt, ihr unmittelbarer Nachfolger in SIJ eingefügt und dessen Head r_{ij} aktualisiert. Gehe zu Step 2;

END.

Entwicklung eines Branch & Bound - Verfahrens

Beispiel 2. Gegeben ist die folgende Instanz des Problems $J \mid\mid C_{max}$:

$$P = \begin{pmatrix} 10 & 8 & 4 & . \\ 3 & 8 & 6 & 5 \\ 4 & 7 & . & 3 \end{pmatrix}, \quad MO = \begin{pmatrix} 1 & 2 & 3 & . \\ 2 & 1 & 4 & 3 \\ 1 & 2 & . & 3 \end{pmatrix}.$$

Die Operationen $\{(1,1),(2,2),(3,1)$ sind Quellen in $G(MO)$. Damit wird $c_{31} = \min\{10,8,4\} = 4$, $j^* = 1$ und $M^* = \{(1,1),(3,1)\}$. Wird die Operation $(1,1)$ als erste angeordnet, werden gleichzeitig zwei der disjunktiven Kanten gerichtet: von $(1,1)$ nach $(2,1)$ und von $(1,1)$ nach $(3,1)$. Der Graph, der aus dem disjunktiven Graphen durch Löschung aller ungerichteten Kanten entsteht, wird mit G^* bezeichnet. Schranken:
(1) Benutze die Länge des kritischen Weges C^*_{max} in G^* als untere Schranke.
(2) Löse Einmaschinenprobleme der Form $1 \mid prec, r_i \geq 0 \mid L_{max}$:
Auf einer festen Maschine M_j, $j = 1, \ldots k$, sollen n Aufträge mit $p_i = p_{ij}$ bearbeitet werden. Dabei sind $r_i = h_{ij}$ (Head der Operation (i,j)), $d_i = -t_{ij}$ (Tail der Operation (i,j)) und der Graph der Vorrangbedingungen aus G^* zu berechnen. Es gilt $LB = L^{opt}_{max}$.

Die Shifting-Bottleneck Heuristik für das Problem $J \mid\mid C_{max}$

Gegeben: P und MO. Für eine Teilmenge der Maschinen $M_j,\ j \in J^* \subseteq J$ liegen die organisatorischen Reihenfolgen bereits fest. Der Graph G^* entstehe aus dem disjunktiven Graphen, indem alle noch ungerichteten disjunktiven Kanten gestrichen werden.
Die Shifting-Bottleneck Heuristik basiert auf folgenden Ideen:

- Löse für jede Maschine $M_j,\ j \in J \setminus J^*$ das Problem $1 \mid prec, r_i \geq 0 \mid L_{max}$ und bestimme die *Bottleneck-Maschine*, d. h. diejenige Maschine M_k, für die L_{max} maximal wird. Lege für diese Maschine die organisatorische Reihenfolge durch die optimale Reihenfolge des entsprechenden Einmaschinenproblems fest.
- Unter Einbeziehung der Maschine $M_l,\ l \in J^*$ wird dann versucht, den Teilplan zu verbessern. Dazu werden aus dem disjunktiven Graphen alle Kanten der organisatorische Reihenfolgen auf Maschine M_l gestrichen, und das entsprechende Einmaschinenproblem bezüglich dieser Maschine gelöst. Liefert die optimale Lösung des Einmaschinenproblems eine bessere Anordnung, so werden die organisatorischen Reihenfolgen aktualisiert.
- Setze $J^* := J^* \cup \{k\}; J := J \setminus \{k\}$ und wiederhole diese beiden Schritte, bis der Plan bestimmt ist (d. h. keine organisatorischen Reihenfolgen ändern sich mehr).

Block-Approach für Problem $J \mid\mid C_{max}$ (1)

Block-Approach ist ein Branch & Bound Verfahren mit einer neuen Verzweigungsidee. Ausgang ist ein Schedule S beschrieben durch C und LR, $G(S)$ sei der zugehörende Sequencegraph und $SIJ(CP)$ bezeichne die Operationenmenge auf einem kritischen Weg CP in $G(S)$.

Definition 2: Eine Menge $B \subseteq SIJ(CP(S))$ von aufeinanderfolgenden Operationen auf einer Maschine in $G(S)$ heißt *Block* B, wenn
– $|B| \geq 2$ gilt und
– B die maximale Anzahl solcher Operationen enthält.

Lemma 1. *Falls für das Problem $J \mid\mid C_{max}$ ein Schedule S^* existiert, für den $C_{max}(S^*) < C_{max}(S)$ gilt, wobei S ein vorgegebener Schedule ist, dann muss S^* mindestens auf einem Block $B \subseteq SIJ(CP(S))$ eine andere organisatorische Reihenfolge als S besitzen, die mindestens eine der folgenden Eigenschaften erfüllt:*
(i) die Operation, die zuerst bearbeitet wird, ist nicht die gleiche;
(ii) die Operation, die zuletzt bearbeitet wird, ist nicht die gleiche.

Branching Regel: Vorbetrachtung

Die **Branching Regel** des Block-Approach: Bestimme ein Teilproblem durch Fixierung einer Menge von Kanten, die ausschließt, dass die Menge der Operationen $SIJ(CP(S))$ wieder auf einem Weg liegt und sichere ab, dass die Menge aller erzeugten Teilprobleme gewährleistet, dass jeder Schedule S^* mit $C_{max}(S^*) < C_{max}(S)$ zulässig für eines der Teilprobleme ist.

Die Blöcke bezüglich $CP(S)$ werden mit $B_1, \ldots, B_p$ bezeichnet, wobei $|B_k| = l_k$ und $l_1 \geq l_2 \geq \ldots \geq l_p$ erfüllt sei. Die Mengen B_k seien geordnete Mengen, d. h. $B_k = \{(i_1, j), (i_2, j), \ldots, (i_{l_k}, j)\}$ mit $jo_{i_\mu j} = \mu$ bezüglich B_k für $\mu = 1, \ldots, l_k$.

Die Menge VB_k enthalte alle Kandidaten von Operationen des Blockes B_k, die **vor** der ersten Operation in Block B_k bearbeitet werden können, um den kritischen Weg zu 'zerstören'.
Die Menge NB_k enthalte alle Kandidaten von Operationen des Blockes B_k, die **nach** der letzten Operation in Block B_k bearbeitet werden können, um den kritischen Weg zu 'zerstören'. Damit gilt: $VB_k = B_k \setminus \{((i_1, j)\}$ und $NB_k = B_k \setminus \{((i_{l_k}, j)\}$.

Die Kantenmengen E_k bzw. E_k^* sichern ab, dass die erste bzw. letzte Operation in B_k fest ist, d. h. $E_k = \{((i_1, j), (i_\mu, j)) \mid (i_\mu, j) \in VB_k\}$ und $E_k^* = \{((i_\mu, j), (i_{l_k}, j)) \mid (i_\mu, j) \in NB_k\}$.

Branching Regel für Block-Approach

Branching Regel für Block-Approach

Eingabe: Schedule S, Blöcke B_k bezüglich $CP(S), VB_k, NB_k, E_k, E_k^*$ für alle $k = 1, \ldots, p$;
Ausgabe: Teilprobleme.
BEGIN FOR $k := 1$ **TO** p **DO**
BEGIN
FOR ALL $(i, j) \in VB_k$ **DO**
Bilde ein Teilproblem durch Fixierung der ersten und der letzten Operationen in allen Blöcken $B_1, \ldots, B_{k-1}$, setze $(i_1, j) = (i, j)$ und fixiere alle Kanten aus E_k;
FOR ALL $(i, j) \in NB_k$ **DO**
Bilde ein Teilproblem durch Fixierung der ersten und der letzten Operationen in allen Blöcken $B_1, \ldots, B_{k-1}$, fixiere die erste Operation in B_k, setze $(i_{l_k}, j) = (i, j)$ und fixiere alle Kanten aus E_k^*;
END
END.

Block-Approach für Problem $J \mid\mid C_{max}$ (2)

Teilprobleme können eliminiert werden, falls:

1. das Teilproblem einen Weg mit der Knotenmenge des kritischen Weges enthält,
2. der entstehenden Graph $G^{MO} \cup G^{JO_l^*}$ Zyklen enthält,
3. die untere Schranke für das Teilproblem größer als die obere Schranke, d. h. der beste bisher ermittelte Zielfunktionswert ist,
4. die Mengen VB_k und NB_k leer sind, dann ist auf diesem Zweig des Branch & Bound Baumes keine Verbesserung möglich.

Berechnung von **unteren Schranken**:

(i) Bestimme für $G^{MO} \cup G^{JO_l^*}$ den kritischen Weg, d. h. $LB = \max\limits_{(i,j) \in SIJ} \{r_{ij} + p_{ij} + t_{ij}\}$.

(ii) Löse Einmaschinenprobleme der Form $1 \mid pmtn, prec, r_i \geq 0 \mid L_{max}$.

Beispiel für Block-Approach mit einfacher Schranke

$$P = \begin{pmatrix} 3 & 3 & 2. \\ 2 & 3 & 3 \\ 2 & 3 & 1 \end{pmatrix}, MO = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 1 & 3 \end{pmatrix}, LR = \begin{pmatrix} 1 & 3* & 4* \\ 3 & 1* & 2 \\ 4 & 2* & 5* \end{pmatrix}, C_{max} = 12\ (UB).$$

Problem 1	Problem 2	Problem 3	Problem 4	Problem 5
$\begin{pmatrix} . & 2 & . \\ . & 2 & . \\ . & 1 & . \end{pmatrix}$	$\begin{pmatrix} . & 1 & . \\ . & 2 & . \\ . & 2 & . \end{pmatrix}$	$\begin{pmatrix} . & 1 & . \\ . & 2 & . \\ . & 1 & . \end{pmatrix}$	$\begin{pmatrix} . & 1 & . \\ . & 1 & . \\ . & 2 & . \end{pmatrix}$	$\begin{pmatrix} . & 3 & 2 \\ . & 1 & . \\ . & 2 & 1 \end{pmatrix}$
$\begin{pmatrix} 3 & 3 & 2 \\ \mathbf{2} & \mathbf{3} & \mathbf{3} \\ 2 & \mathbf{3} & 1 \end{pmatrix}$	$\begin{pmatrix} \mathbf{3} & \mathbf{3} & 2 \\ \mathbf{2} & \mathbf{3} & \mathbf{3} \\ 2 & 3 & 1 \end{pmatrix}$	$\begin{pmatrix} \mathbf{3} & \mathbf{3} & 2 \\ \mathbf{2} & \mathbf{3} & \mathbf{3} \\ 2 & 3 & 1 \end{pmatrix}$	$\begin{pmatrix} \mathbf{3} & \mathbf{3} & 2 \\ 2 & 3 & 3 \\ \mathbf{2} & \mathbf{3} & \mathbf{1} \end{pmatrix}$	$\begin{pmatrix} 3 & \mathbf{3} & \mathbf{2} \\ 2 & \mathbf{3} & 3 \\ 2 & \mathbf{3} & 1 \end{pmatrix}$
LB=11	LB=14	LB= 14	LB=12	LB= 11

Immediate Selection für das Problem $J \mid\mid C_{max}$ (1)

Notwendige Bedingungen für die organisatorischen Reihenfolgen, damit ein besserer Schedule als der bereits bekannte erzeugt werden kann.

Es bezeichne I^* mit $|I^*| \geq 2$ eine Indexmenge von Aufträgen auf M_j, weiter sei $k \notin I^*$, aber $(k, j) \in SIJ$. Außerdem sei UB eine obere Schranke (d.h. ein bekannter Zielfunktionswert). Betrachtet werden die folgenden Ungleichungen:

$$r_{kj} + \sum_{i \in I^* \cup \{k\}} p_{ij} + \min_{i \in I^*}\{t_{ij}\} \geq UB \tag{1}$$

$$\min_{i \in I^*}\{r_{ij}\} + \sum_{i \in I^* \cup \{k\}} p_{ij} + t_{kj} \geq UB \tag{2}$$

$$\min_{i \in I^*}\{r_{ij}\} + \sum_{i \in I^* \cup \{k\}} p_{ij} + \min_{i \in I^*}\{t_{ij}\} \geq UB \tag{3}$$

Immediate Selection für das Problem $J \mid\mid C_{max}$ (2)

Es gilt:

1. Die linke Seite von 1 ist eine untere Schranke dafür, dass Auftrag A_k vor allen Aufträgen aus I^* bearbeitet wird;
2. Die linke Seite von 2 ist eine untere Schranke dafür, dass Auftrag A_k nach allen Aufträgen aus I^* bearbeitet wird;
3. Die linke Seite von 3 ist eine untere Schranke dafür, dass Auftrag A_k weder erster noch letzter in $I^* \cup \{k\}$ ist.

Da ein Schedule mit $C_{max} < UB$ gesucht ist, lässt sich ableiten:

- Falls für k und I^* die Ungleichungen 1 und 3 gelten, werden für alle $i \in I^*$ die disjunktiven Kanten von (i, j) zu (k, j) gerichtet;
- Falls für k und I^* die Ungleichungen 2 und 3 gelten, werden für alle $i \in I^*$ die disjunktiven Kanten von (k, j) zu (i, j) gerichtet.

Job-Shop Probleme mit zwei Aufträgen (1)

(a) Das Problem $J \mid n = 2 \mid C_{max}$

Algorithmus von *Akers & Friedman* (1955):

Formulierung des Problems als **Kürzeste-Wege-Problem mit rechteckigen Hindernissen**

Bearbeitungszeiten der Operationen von J_1 (J_2):

Intervalle auf der x-Achse (y-Achse) gemäß technologischer Reihenfolge

seien

$$0 = (0, 0) \qquad \text{und} \qquad F = (a, b),$$

wobei a (b) die Summe der Bearbeitungszeiten von Auftrag J_1 (J_2) ist

Problem: Bestimme einen kürzesten Weg von $0 = (0, 0)$ nach $F = (a, b)$, der nur horizontale, vertikale und diagonale Segmente enthält, die nicht durch das Innere der verbotenen Regionen verlaufen.

Job-Shop Probleme mit zwei Aufträgen (2)

Konstruktion des **Netzwerks** $N = (V,\ A,\ d)$

Knotenmenge V**:**

– enthält 0, F;
– Menge aller Nordwest-Ecken (NW) der Hindernisse;
– Menge alle Südost-Ecken (SO) der Hindernisse.

Bogenmenge A:

jeder Knoten $i \in V \setminus \{F\}$ besitzt höchstens zwei von i fortführende Bögen, nämlich gehe diagonal von i in NO-Richtung bis

(a) der Rand des durch 0 und F bestimmten Rechtecks erreicht ist
$\Rightarrow$ F ist einziger Nachfolger von i;

(b) der Rand eines Hindernisses erreicht wurde

$\Rightarrow$ es gibt zwei Bögen (i, j) zur NW-Ecke sowie (i, k) zur SO-Ecke des betreffenden Hindernisses.

Job-Shop Probleme mit zwei Aufträgen (3)

Länge $d(i,j)$ von Bogen (i,j):

Länge des vertikalen oder horizontalen Abschnittes plus Länge der Projektion des diagonalen Abschnittes auf die x- (bzw. y-)Achse.

Folgerung: Ein Weg von 0 nach F im Netzwerk N entspricht einem zulässigen Plan mit der Länge C_{max}.

Satz 3. *Ein kürzester Weg von 0 nach F in N entspricht einer optimalen Lösung des kürzeste-Wege-Problems mit Hindernissen (d.h. einer optimalen Lösung des Problems $J \mid n = 2 \mid C_{max}$).*

Komplexität der Konstruktion des Netzwerks N: $O(r \log r)$

seien $D_1 \prec D_2 \prec \ldots \prec D_r$ die *lexikografisch geordneten Hindernisse*

Job-Shop Probleme mit zwei Aufträgen (4)

Algorithmus $J \mid n = 2 \mid C_{max}$

Eingabe: Netzwerk $N = (V, A, d)$, Hindernisse $D_1, D_2, \ldots, D_r$;
Ausgabe: Länge d^* eines kürzesten Weges von 0 nach F.
BEGIN
  FOR alle Knoten $i \in V$ **DO** $d(i) = \infty$;
  FOR alle Nachfolger j von 0 **DO** $d(j) = d(0, j)$;
  FOR $i := 1$ **TO** r **DO**
    BEGIN
      FOR alle Nachfolger j der NW-Ecke k von D_i **DO**
        $d(j) := \min\{d(j),\ d(k) + d(k, j)\}$;
      FOR alle Nachfolger j der SO-Ecke k von D_i **DO**
        $d(j) := \min\{d(j),\ d(k) + d(k, j)\}$;
    END;
  $d^* := d(F)$.
END.

Job-Shop Probleme mit zwei Aufträgen (5)

Bemerkung: Algorithmus $J \mid n = 2 \mid C_{max}$ ist leicht modifizierbar, so dass neben einem kürzesten Weg auch der zugehörige Plan ermittelt wird.

Komplexität von Algorithmus $J \mid n = 2 \mid C_{max}$: $O(r)$

Gesamtkomplexität: $O(r \log r)$ $\quad (r \leq n_1 \cdot n_2)$

(b) Das Problem $J \mid n = 2 \mid f$

f - *reguläres* Kriterium $f(C_1, C_2)$ (monoton nichtfallend in jedem Argument)

$\Rightarrow$ Zielfunktionswert der zu einem Weg gehörenden Lösung hängt von dem zuerst erreichten Randpunkt P des durch 0 und F definierten Rechtecks ab:

Job-Shop Probleme mit zwei Aufträgen (6)

P **auf nördlichem Rand** $\Rightarrow$ beste Lösung zu einem Weg durch P hat den Zielfunktionswert

$$f(d(P) + d(P, F),\ d(P)),$$

wobei

$d(P)$	–	Länge eines kürzesten Weges von 0 zu P
$d(P, F)$	–	Länge des horizontalen Segments von P nach F

P **auf östlichem Rand** $\Rightarrow$ beste Lösung zu einem Weg durch P hat den Zielfunktionswert

$$f(d(P),\ d(P) + d(P, F))$$

$\Rightarrow$ modifiziere Algorithmus $J \mid n = 2 \mid C_{max}$ entsprechend