PD per il problema del lot sizing capacitato

Sapienza Università di Roma

Indice

Notazione, modello e soluzioni ammissibili

Un algoritmo di PD per il CLS
- Passo 1 e Passo 2
- Numero di passi
- Il caso a capacità costante

Errori tipici

In sintesi

Obiettivi

Al termine di questo capitolo saremo in grado di:

- spiegare su un esempio perché la proprietà zero-inventory ordering non vale con le capacità;
- scrivere la ricorrenza $v(t, k)$ sulle coppie (periodo, scorta finale), con i suoi casi base;
- eseguire a mano l'algoritmo di PD, riempire le matrici v e r, e ricostruire un piano ottimo;
- contare i passi dell'algoritmo di PD, e spiegare perché è pseudo-polinomiale;
- enunciare la proprietà più debole che vale quando la capacità è la stessa in tutti i periodi.

Prerequisiti (1/2)

- Da questo modulo, il capitolo «PD per il problema del lot sizing senza capacità»: le domande totali $D_{j,t}$, la codifica di un piano di produzione e la sua rete, la proprietà zero-inventory ordering, e l'istanza del suo primo esempio.

- Dal modulo *Problemi di ottimizzazione combinatoria*, i capitoli «Problemi di lot sizing» e «Formulazioni dei problemi di lot sizing»: il problema capacitato, la sua istanza di riferimento, e il suo modello PLI con le variabili x_t, s_t, y_t.

Prerequisiti (2/2)

- Da questo modulo, il capitolo «Principi della programmazione dinamica»: ricorrenza, casi base, tabella di lookup e ricostruzione di una soluzione.
- Dal modulo *Algoritmi*: lo pseudocodice, e la dimensione dell'input.

Dove eravamo

- **Come problema e come modello**: il problema di lot sizing capacitato e le sue formulazioni PLI (*Problemi di ottimizzazione combinatoria*, capitoli «Problemi di lot sizing» e «Formulazioni dei problemi di lot sizing»).
- **Come tabella, senza capacità**: l'algoritmo di PD di Wagner e Whitin del capitolo «PD per il problema del lot sizing senza capacità», sui lotti di produzione, in $\Theta(m^2)$ passi.
- **Ora, con le capacità**: un algoritmo di PD sulle coppie (periodo, scorta), che è pseudo-polinomiale.

Il problema di questo capitolo

Il problema di questo capitolo è il

problema di lot sizing capacitato (CLS):

la sua definizione e le sue ipotesi sono nella sua pagina.

Notazione, modello e soluzioni ammissibili

Introduzione: notazione, modello e soluzioni ammissibili

- Richiamiamo la notazione dell'algoritmo di PD senza capacità: le domande totali $D_{j,t}$
- Richiamiamo il modello PLI del CLS e come si codifica una soluzione ammissibile
- Vediamo perché la proprietà zero-inventory ordering, su cui si basa l'algoritmo di PD senza capacità, non vale con le capacità
- Fissiamo l'istanza del CLS che usiamo in tutti gli esempi del capitolo

Domande totali

- Come nel capitolo «PD per il problema del lot sizing senza capacità», per ogni $j, t \in T$ indichiamo con

$$D_{j,t} = \sum_{k=j}^{t} d_k$$

  la domanda totale dei periodi $j, j+1, \ldots, t$

- Per convenzione, $D_{j,t} = 0$ se $j > t$

Il modello PLI

- Il modello PLI è il modello $\mathrm{ILP}_{\mathrm{CLSP}}$ del capitolo «Formulazioni dei problemi di lot sizing», con il valore di big-M

$$\min\{cap_t, D_{t,m}\}$$

del periodo t

- È il modello $\mathrm{ILP}_{\mathrm{ULS}}$ con i vincoli aggiuntivi

$$x_t \leq cap_t \qquad \forall\, t \in T$$

Soluzioni ammissibili

- Una soluzione ammissibile si codifica con lo stesso vettore $(x_1, x_2, \ldots, x_m)$ dell'ULS, con le scorte

$$s_t = \sum_{j=1}^{t} x_j - D_{1,t}$$

- Con il requisito aggiuntivo

$$x_t \leq cap_t \qquad \forall\, t \in T$$

Niente zero-inventory ordering con le capacità

- La **proprietà zero-inventory ordering non vale** per il CLS
- Un periodo può dover produrre mentre ci sono ancora unità in scorta, perché la capacità non permette di produrre tutto nel periodo in cui sarebbe più economico
- La proprietà, e l'algoritmo di PD sui lotti di produzione che si basa su di essa, sono quelli del capitolo «PD per il problema del lot sizing senza capacità»

Esempio: lot sizing capacitato (1/2)

Consideriamo l'istanza senza capacità del primo esempio del capitolo «PD per il problema del lot sizing senza capacità», misurata in unità di 10 prodotti (quindi il costo di mantenimento è 10 per unità), con una capacità costante di 4 unità per periodo:

t	1	2	3	4	5
d_t	2	1	3	4	1
sc_t	50	50	50	50	50
vc_t	0	0	0	0	0
hc_t	10	10	10	10	10
cap_t	4	4	4	4	4

Esempio: lot sizing capacitato (2/2)

- Chiamiamo questa istanza del CLS **l'istanza dell'esempio**: è l'istanza di riferimento del capitolo «Problemi di lot sizing» di *Problemi di ottimizzazione combinatoria*

- La soluzione ottima dell'istanza senza capacità, in unità di 10, è il vettore

$$(3, 0, 8, 0, 0),$$

  di costo 170

- Non è ammissibile per l'istanza dell'esempio, poiché il periodo 3 produce più di 4 unità

Riepilogo

Notazione e modello:

- $D_{j,t}$ è la domanda totale dei periodi $j, \ldots, t$
- Il modello $\mathrm{ILP}_{\mathrm{CLSP}}$ è $\mathrm{ILP}_{\mathrm{ULS}}$ con i vincoli $x_t \leq cap_t$
- Una soluzione ammissibile è un vettore $(x_1, \ldots, x_m)$ con $x_t \leq cap_t$ per ogni $t \in T$

Zero-inventory ordering:

- Non vale con le capacità: un periodo può produrre mentre ci sono unità in scorta
- Nell'istanza dell'esempio ($cap_t = 4$), l'ottimo senza capacità $(3, 0, 8, 0, 0)$ non è ammissibile

Un algoritmo di PD per il CLS

Introduzione: un algoritmo di PD per il CLS

- Senza capacità l'algoritmo di PD lavora sui lotti di produzione; con le capacità serve uno stato diverso: la coppia (periodo, scorta finale)
- Scriviamo la ricorrenza $v(t, k)$, verifichiamo la sottostruttura ottima e i sottoproblemi sovrapposti, e scriviamo l'algoritmo di PD con l'approccio bottom-up
- Lo eseguiamo a mano sull'istanza dell'esempio e ricostruiamo un piano di produzione ottimo
- Contiamo i suoi passi, che lo rendono pseudo-polinomiale, e guardiamo il caso a capacità costante

Passo 1: calcolo del valore ottimo

- Per ogni periodo $t \in \{1, \ldots, m\}$, la **funzione di costo di produzione** del periodo t è

$$f_t(x) = \begin{cases} 0 & \text{se } x = 0, \\ sc_t + vc_t\, x & \text{se } x \in \{1, 2, \ldots, cap_t\}. \end{cases}$$

- La scorta alla fine del periodo t non supera mai la domanda $D_{t+1,m}$ dei periodi successivi, poiché la scorta deve essere vuota alla fine del periodo m

Le sottoistanze

Per ogni $t \in \{1, 2, \ldots, m\}$ e $k \in \{0, 1, \ldots, D_{t+1,m}\}$, sia $v(t, k)$ il valore ottimo della sottoistanza del CLS data da:

1. un insieme $\{1, 2, \ldots, t\}$ di t periodi di tempo,
2. per ogni periodo $j \in \{1, 2, \ldots, t\}$, una domanda d_j,
3. per ogni periodo $j \in \{1, 2, \ldots, t\}$, un costo di setup sc_j, un costo variabile di produzione vc_j e un costo di mantenimento a scorta hc_j,
4. per ogni periodo $j \in \{1, 2, \ldots, t\}$, una capacità cap_j,
5. una scorta uguale a k alla fine del periodo t.

La ricorrenza

La formula ricorsiva per calcolare $v(t,k)$ è la seguente:

$$v(t,k) = \begin{cases} \min\Big\{ v(t-1, k+d_t-x) \quad \text{se } 1 \le t \le m, \\ \qquad + f_t(x) : x \in X_t(k) \Big\} \\ + hc_t\, k \\ 0 \qquad\qquad \text{se } t = 0 \text{ e } k = 0, \\ +\infty \qquad\qquad \text{se } t = 0 \text{ e } k > 0, \end{cases}$$

dove

$$X_t(k) = \{0, 1, \ldots, \min\{cap_t,\ k + d_t\}\}$$

è l'insieme delle possibili produzioni del periodo t.

Stati e sottoproblemi

- La coppia (t, k) è lo **stato**: l'ultimo periodo della sottoistanza e la scorta alla sua fine
- Per $t = m$ e $k = 0$ otteniamo l'istanza del CLS stessa, quindi il suo valore ottimo è $v(m, 0)$
- Il processo di decomposizione ricorsiva della ricorrenza genera al più $cap_t + 1$ **sottoproblemi indipendenti**, ciascuno corrispondente a una possibile scelta della produzione x dell'ultimo periodo t

Il sottoproblema residuo

- Se si producono x unità nel periodo t e alla sua fine restano k unità, la scorta alla fine del periodo $t-1$ è $k + d_t - x$
- Il sottoproblema residuo è la sottoistanza del CLS con i periodi $\{1, \ldots, t-1\}$ e scorta finale $k + d_t - x$
- La produzione x è al più cap_t, e al più $k + d_t$ perché la scorta alla fine del periodo $t-1$ non può essere negativa
- Inoltre $k + d_t - x \leq D_{t+1,m} + d_t = D_{t,m}$, quindi il sottoproblema residuo è uno stato

Casi base

- Il caso base è $v(0, 0) = 0$: nessun periodo, nessuna scorta, nessun costo
- Per $k > 0$ poniamo $v(0, k) = +\infty$: la scorta iniziale è vuota, quindi nessuna soluzione ammissibile ha una scorta positiva prima del periodo 1
- Il valore $+\infty$ esclude questo caso, e si propaga a ogni stato che non si può raggiungere entro le capacità, con la convenzione $+\infty + a = +\infty$

Sottostruttura ottima

- Il valore ottimo $v(t, k)$ si ottiene provando tutte le possibili produzioni x del periodo t e scegliendo quella che minimizza la somma

$$v(t-1, k+d_t-x)+f_t(x)$$

  del valore ottimo del sottoproblema residuo e del costo di produzione del periodo t
- Il costo di mantenimento $hc_t\,k$ della scorta finale non dipende da x
- Quindi, una soluzione ottima della sottoistanza dello stato (t, k) si costruisce combinando la produzione scelta del periodo t con una soluzione ottima del sottoproblema residuo dello stato $(t-1, k+d_t-x)$

$\rightsquigarrow$ **Sottostruttura ottima verificata!**

Sottoproblemi sovrapposti

- Sequenze diverse di decisioni di produzione possono portare allo stesso stato
- Per esempio, nell'istanza dell'esempio, partendo dallo stato $(5, 0)$:
  - le decisioni $x_5 = 0$ e $x_4 = 4$ portano allo stato $(3, 1)$
  - le decisioni $x_5 = 1$ e $x_4 = 3$ portano allo stesso stato $(3, 1)$
- Quindi lo stesso valore $v(t, k)$ può dover essere calcolato più volte durante la valutazione ricorsiva della ricorrenza

$\rightsquigarrow$ **Sottoproblemi sovrapposti verificati!**

Primo metodo: l'approccio top-down

- Un **primo metodo** per calcolare il valore ottimo $v(m, 0)$ dell'istanza del CLS è l'**approccio top-down**
- Il metodo parte dalla chiamata con $(t, k) = (m, 0)$ e valuta ricorsivamente la ricorrenza
- Ogni chiamata per uno stato (t, k) genera ulteriori chiamate per gli stati $(t - 1, k + d_t - x)$, per ogni $x \in \{0, \ldots, \min\{cap_t, k + d_t\}\}$, finché non si raggiunge uno dei casi base $v(0, k)$

La tabella di lookup

- Il valore $v(t, k)$ si può calcolare una volta noti i valori $v(t-1, k')$ per ogni $k' \in \{0, \ldots, D_{t,m}\}$
- Possiamo evitare di risolvere ripetutamente gli stessi sottoproblemi memorizzando il valore di ciascun sottoproblema non appena viene calcolato, in una **tabella di lookup**: la matrice

$$v[0..m][0..D_{1,m}]$$

- La cella $v[t][k]$ contiene $v(t, k)$ per $k \leq D_{t+1,m}$ (le altre celle non si usano)

Secondo metodo: l'approccio bottom-up

- Un **secondo metodo** per calcolare il valore ottimo $v(m, 0)$ dell'istanza del CLS è l'**approccio bottom-up**
- Il metodo parte dai casi base $v(0, 0) = 0$ e $v(0, k) = +\infty$ per ogni $k \in \{1, \ldots, D_{1,m}\}$
- Valuta la ricorrenza per valori crescenti di $t \in \{1, \ldots, m\}$ e, per ogni t fissato, per $k \in \{0, 1, \ldots, D_{t+1,m}\}$: ogni valore $v(t, k)$ usa valori già calcolati della riga $t - 1$, finché non si ottiene $v(m, 0)$

Ora: lo pseudocodice dell'**algoritmo di PD con l'approccio bottom-up**.

Algorithm: DP_CLS (1/2)

Input : Un'istanza del CLS data dal numero m di periodi e dai vettori $d[1..m]$, $sc[1..m]$, $vc[1..m]$, $hc[1..m]$, $cap[1..m]$

Output : Il valore ottimo

```
1  function DP_CLS(m, d, sc, vc, hc, cap)
2      R := new array[0..m]
3      R[m] ← 0
4      for t := m − 1 downto 0 do
5          R[t] ← R[t + 1] + d[t + 1]
6      end
7      v := new array[0..m][0..R[0]]
8      v[0][0] ← 0
9      for k := 1 to R[0] do
10         v[0][k] ← +∞
11     end
```

Algorithm: DP_CLS (2/2)

function *DP_CLS*(m, d, sc, vc, hc, cap) (continua)

```
12      for t := 1 to m do
13          for k := 0 to R[t] do
14              b := +∞
15              for x := 0 to min{cap[t], k + d[t]} do
16                  if v[t − 1][k + d[t] − x] + f_t(x) + hc[t] · k < b
                    then
17                      b ← v[t − 1][k + d[t] − x] + f_t(x) + hc[t] · k
18                  end
19              end
20              v[t][k] ← b
21          end
22      end
23      return v[m][0]
24  end
```

Come funziona DP_CLS (1/2)

- Il vettore $R[0..m]$ contiene le domande residue $R[t] = D_{t+1,m}$, calcolate all'indietro a partire da $R[m] = 0$: gli stati del periodo t sono le coppie (t, k) con $k \leq R[t]$
- La matrice v è la **tabella di lookup**. L'inizializzazione pone $v[0][0] = 0$ e $v[0][k] = +\infty$ per ogni $k \in \{1, \ldots, R[0]\}$, poiché la scorta iniziale è vuota
- Il ciclo esterno, su $t \in \{1, 2, \ldots, m\}$, considera i periodi in ordine crescente, e il secondo ciclo, su $k \in \{0, \ldots, R[t]\}$, considera tutte le scorte finali del periodo t

Come funziona DP_CLS (2/2)

- Il ciclo interno, sulla produzione x del periodo t, mantiene in b il valore minimo

$$v[t-1][k+d[t]-x]+f_t(x)+hc[t]\cdot k$$

- Il valore $f_t(x)$ è 0 per $x=0$ e $sc[t]+vc[t]\cdot x$ altrimenti
- Se tutti i candidati valgono $+\infty$, il confronto non è mai vero e $v[t][k]=+\infty$: lo stato (t,k) non si può raggiungere entro le capacità
- Infine, viene restituito il valore ottimo $v[m][0]$ dell'istanza del CLS

Esempio: esecuzione dell'algoritmo di PD (1/11)

Consideriamo l'istanza del CLS dell'esempio:

t	1	2	3	4	5
d_t	2	1	3	4	1
sc_t	50	50	50	50	50
vc_t	0	0	0	0	0
hc_t	10	10	10	10	10
cap_t	4	4	4	4	4

Le domande residue sono $(R[0], \ldots, R[5]) = (11, 9, 8, 5, 1, 0)$.

Per ogni passo dell'algoritmo di PD, la tabella nelle slide seguenti riporta i valori di t e k, la cella $v[t][k]$ e l'operazione eseguita. I candidati del minimo sono elencati per $x = 0, 1, \ldots, \min\{cap_t, k + d_t\}$.

Esempio: esecuzione dell'algoritmo di PD (2/11)

t	k	$v[t][k]$	Azione
0	0	0	$v[0][0] \leftarrow 0$
0	1	$+\infty$	$v[0][1] \leftarrow +\infty$
0	2	$+\infty$	$v[0][2] \leftarrow +\infty$
0	3	$+\infty$	$v[0][3] \leftarrow +\infty$
0	4	$+\infty$	$v[0][4] \leftarrow +\infty$
0	5	$+\infty$	$v[0][5] \leftarrow +\infty$
0	6	$+\infty$	$v[0][6] \leftarrow +\infty$
0	7	$+\infty$	$v[0][7] \leftarrow +\infty$
0	8	$+\infty$	$v[0][8] \leftarrow +\infty$
0	9	$+\infty$	$v[0][9] \leftarrow +\infty$
0	10	$+\infty$	$v[0][10] \leftarrow +\infty$
0	11	$+\infty$	$v[0][11] \leftarrow +\infty$

Esempio: esecuzione dell'algoritmo di PD (3/11)

t	k	$v[t][k]$	Azione
1	0	50	$\min\{v[0][2]+f_1(0), v[0][1]+f_1(1),$ $v[0][0]+f_1(2)\}+hc_1 \cdot 0$ $= \min\{+\infty, +\infty, 0+50\}+10 \cdot 0 = 50$
1	1	60	$\min\{v[0][3]+f_1(0), v[0][2]+f_1(1), v[0][1]+f_1(2),$ $v[0][0]+f_1(3)\}+hc_1 \cdot 1$ $= \min\{+\infty, +\infty, +\infty, 0+50\}+10 \cdot 1 = 60$
1	2	70	$\min\{v[0][4]+f_1(0), v[0][3]+f_1(1), v[0][2]+f_1(2),$ $v[0][1]+f_1(3), v[0][0]+f_1(4)\}+hc_1 \cdot 2$ $= \min\{+\infty, +\infty, +\infty, +\infty, 0+50\}+10 \cdot 2 = 70$
1	3	$+\infty$	i candidati usano le celle $v[0][1], \ldots, v[0][5]$, che valgono tutte $+\infty$: $v[1][3] \leftarrow +\infty$
1	4	$+\infty$	i candidati usano le celle $v[0][2], \ldots, v[0][6]$, che valgono tutte $+\infty$: $v[1][4] \leftarrow +\infty$

Esempio: esecuzione dell'algoritmo di PD (4/11)

t	k	$v[t][k]$	Azione
1	5	$+\infty$	i candidati usano le celle $v[0][3], \ldots, v[0][7]$, che valgono tutte $+\infty$: $v[1][5] \leftarrow +\infty$
1	6	$+\infty$	i candidati usano le celle $v[0][4], \ldots, v[0][8]$, che valgono tutte $+\infty$: $v[1][6] \leftarrow +\infty$
1	7	$+\infty$	i candidati usano le celle $v[0][5], \ldots, v[0][9]$, che valgono tutte $+\infty$: $v[1][7] \leftarrow +\infty$
1	8	$+\infty$	i candidati usano le celle $v[0][6], \ldots, v[0][10]$, che valgono tutte $+\infty$: $v[1][8] \leftarrow +\infty$
1	9	$+\infty$	i candidati usano le celle $v[0][7], \ldots, v[0][11]$, che valgono tutte $+\infty$: $v[1][9] \leftarrow +\infty$

Esempio: esecuzione dell'algoritmo di PD (5/11)

t	k	$v[t][k]$	Azione
2	0	60	$\min\{v[1][1]+f_2(0), v[1][0]+f_2(1)\}+hc_2 \cdot 0$ $= \min\{60+0, 50+50\}+10 \cdot 0 = 60$
2	1	80	$\min\{v[1][2]+f_2(0), v[1][1]+f_2(1),$ $v[1][0]+f_2(2)\}+hc_2 \cdot 1$ $= \min\{70+0, 60+50, 50+50\}+10 \cdot 1 = 80$
2	2	120	$\min\{v[1][3]+f_2(0), v[1][2]+f_2(1), v[1][1]+f_2(2),$ $v[1][0]+f_2(3)\}+hc_2 \cdot 2$ $= \min\{+\infty, 70+50, 60+50, 50+50\}+10 \cdot 2 = 120$
2	3	130	$\min\{v[1][4]+f_2(0), v[1][3]+f_2(1), v[1][2]+f_2(2),$ $v[1][1]+f_2(3), v[1][0]+f_2(4)\}+hc_2 \cdot 3$ $= \min\{+\infty, +\infty, 70+50, 60+50,$ $50+50\}+10 \cdot 3 = 130$

Esempio: esecuzione dell'algoritmo di PD (6/11)

t	k	$v[t][k]$	Azione
2	4	150	$\min\{v[1][5]+f_2(0), v[1][4]+f_2(1), v[1][3]+f_2(2),$ $v[1][2]+f_2(3), v[1][1]+f_2(4)\}+hc_2 \cdot 4$ $= \min\{+\infty, +\infty, +\infty, 70+50, 60+50\}+10 \cdot 4 = 150$
2	5	170	$\min\{v[1][6]+f_2(0), v[1][5]+f_2(1), v[1][4]+f_2(2),$ $v[1][3]+f_2(3), v[1][2]+f_2(4)\}+hc_2 \cdot 5$ $= \min\{+\infty, +\infty, +\infty, +\infty, 70+50\}+10 \cdot 5 = 170$
2	6	$+\infty$	i candidati usano le celle $v[1][3], \ldots, v[1][7]$, che valgono tutte $+\infty$: $v[2][6] \leftarrow +\infty$
2	7	$+\infty$	i candidati usano le celle $v[1][4], \ldots, v[1][8]$, che valgono tutte $+\infty$: $v[2][7] \leftarrow +\infty$
2	8	$+\infty$	i candidati usano le celle $v[1][5], \ldots, v[1][9]$, che valgono tutte $+\infty$: $v[2][8] \leftarrow +\infty$

Esempio: esecuzione dell'algoritmo di PD (7/11)

t	k	$v[t][k]$	Azione
3	0	110	$\min\{v[2][3]+f_3(0), v[2][2]+f_3(1), v[2][1]+f_3(2), v[2][0]+f_3(3)\}+hc_3 \cdot 0$ $= \min\{130+0, 120+50, 80+50, 60+50\}+10 \cdot 0 = 110$
3	1	120	$\min\{v[2][4]+f_3(0), v[2][3]+f_3(1), v[2][2]+f_3(2), v[2][1]+f_3(3), v[2][0]+f_3(4)\}+hc_3 \cdot 1$ $= \min\{150+0, 130+50, 120+50, 80+50, 60+50\}+10 \cdot 1 = 120$
3	2	150	$\min\{v[2][5]+f_3(0), v[2][4]+f_3(1), v[2][3]+f_3(2), v[2][2]+f_3(3), v[2][1]+f_3(4)\}+hc_3 \cdot 2$ $= \min\{170+0, 150+50, 130+50, 120+50, 80+50\}+10 \cdot 2 = 150$

Esempio: esecuzione dell'algoritmo di PD (8/11)

t	k	$v[t][k]$	Azione
3	3	200	$\min\{v[2][6]+f_3(0), v[2][5]+f_3(1), v[2][4]+f_3(2),$ $v[2][3]+f_3(3), v[2][2]+f_3(4)\}+hc_3 \cdot 3$ $= \min\{+\infty, 170+50, 150+50, 130+50,$ $120+50\}+10 \cdot 3 = 200$
3	4	220	$\min\{v[2][7]+f_3(0), v[2][6]+f_3(1), v[2][5]+f_3(2),$ $v[2][4]+f_3(3), v[2][3]+f_3(4)\}+hc_3 \cdot 4$ $= \min\{+\infty, +\infty, 170+50, 150+50,$ $130+50\}+10 \cdot 4 = 220$
3	5	250	$\min\{v[2][8]+f_3(0), v[2][7]+f_3(1), v[2][6]+f_3(2),$ $v[2][5]+f_3(3), v[2][4]+f_3(4)\}+hc_3 \cdot 5$ $= \min\{+\infty, +\infty, +\infty, 170+50,$ $150+50\}+10 \cdot 5 = 250$

Esempio: esecuzione dell'algoritmo di PD (9/11)

t	k	$v[t][k]$	Azione
4	0	160	$\min\{v[3][4]+f_4(0), v[3][3]+f_4(1), v[3][2]+f_4(2),$ $v[3][1]+f_4(3), v[3][0]+f_4(4)\}+hc_4 \cdot 0$ $= \min\{220+0, 200+50, 150+50, 120+50,$ $110+50\}+10 \cdot 0 = 160$
4	1	180	$\min\{v[3][5]+f_4(0), v[3][4]+f_4(1), v[3][3]+f_4(2),$ $v[3][2]+f_4(3), v[3][1]+f_4(4)\}+hc_4 \cdot 1$ $= \min\{250+0, 220+50, 200+50, 150+50,$ $120+50\}+10 \cdot 1 = 180$
5	0	180	$\min\{v[4][1]+f_5(0), v[4][0]+f_5(1)\}+hc_5 \cdot 0$ $= \min\{180+0, 160+50\}+10 \cdot 0 = 180$

Esempio: esecuzione dell'algoritmo di PD (10/11)

Questa è la matrice v dopo l'esecuzione dell'algoritmo di PD (un punto indica una coppia (t, k) con $k > D_{t+1,m}$, che non è uno stato), colonne $k = 0, \ldots, 5$:

$v[t][k]$	0	1	2	3	4	5
$t = 0$	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
$t = 1$	50	60	70	$+\infty$	$+\infty$	$+\infty$
$t = 2$	60	80	120	130	150	170
$t = 3$	110	120	150	200	220	250
$t = 4$	160	180	.	.	.	.
$t = 5$	180	.	.	.	.	.

Esempio: esecuzione dell'algoritmo di PD (11/11)

La matrice v dopo l'esecuzione dell'algoritmo di PD, colonne $k = 6, \ldots, 11$:

$v[t][k]$	6	7	8	9	10	11
$t = 0$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
$t = 1$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	.	.
$t = 2$	$+\infty$	$+\infty$	$+\infty$	.	.	.
$t = 3$	.	.	.	.	.	.
$t = 4$	.	.	.	.	.	.
$t = 5$	.	.	.	.	.	.

Il valore ottimo dell'istanza del CLS è $v[5][0] = 180$.

Passo 2: calcolo di una soluzione ottima

- Dopo l'esecuzione dell'algoritmo di PD, la matrice v contiene il valore ottimo $v[t][k]$ di ogni stato (t, k)
- Per calcolare una soluzione ottima è necessario memorizzare alcune informazioni aggiuntive
- Per ogni stato (t, k), oltre al valore ottimo $v[t][k]$, memorizziamo $r[t][k]$, la produzione del periodo t in una soluzione ottima della sottoistanza dello stato (t, k); per gli stati con valore $+\infty$ memorizziamo 0

Lettura della matrice r

- Partendo da $(t, k) = (m, 0)$, il valore $r[t][k]$ dà la produzione del periodo t
- La scorta alla fine del periodo $t - 1$ è allora $k + d_t - r[t][k]$
- La stessa regola si può applicare di nuovo al periodo $t - 1$, e così via, fino a $t = 0$

Ora: lo pseudocodice dell'**algoritmo di PD con l'approccio bottom-up**,
con la memorizzazione delle informazioni necessarie per calcolare una soluzione ottima.

Algorithm: DP_CLS_InfoSol (1/4)

Input : Un'istanza del CLS data dal numero m di periodi e dai vettori $d[1..m]$, $sc[1..m]$, $vc[1..m]$, $hc[1..m]$, $cap[1..m]$

Output : Il valore ottimo e la matrice delle informazioni $r[0..m][0..D_{1,m}]$

```
1 function DP_CLS_InfoSol(m, d, sc, vc, hc, cap)
2 |   R := new array[0..m]
3 |   R[m] ← 0
4 |   for t := m − 1 downto 0 do
5 |   |   R[t] ← R[t + 1] + d[t + 1]
6 |   end
7 |   v := new array[0..m][0..R[0]]
8 |   r := new array[0..m][0..R[0]]
```

Algorithm: DP_CLS_InfoSol (2/4)

function *DP_CLS_InfoSol*(m, d, sc, vc, hc, cap) (continua)

```
9   |   v[0][0] ← 0
10  |   r[0][0] ← 0
11  |   for k := 1 to R[0] do
12  |   |   v[0][k] ← +∞
13  |   |   r[0][k] ← 0
14  |   end
```

Algorithm: DP_CLS_InfoSol (3/4)

function *DP_CLS_InfoSol*(m, d, sc, vc, hc, cap) (continua)

```
15      for t := 1 to m do
16          for k := 0 to R[t] do
17              b := +∞
18              a := 0
19              for x := 0 to min{cap[t], k + d[t]} do
20                  if v[t − 1][k + d[t] − x] + f_t(x) + hc[t] · k < b
                    then
21                      b ← v[t − 1][k + d[t] − x] + f_t(x) + hc[t] · k
22                      a ← x
23                  end
24              end
```

Algorithm: DP_CLS_InfoSol (4/4)

```
function DP_CLS_InfoSol(m, d, sc, vc, hc, cap) (continua)
    for t := 1 to m (continua) do
        for k := 0 to R[t] (continua) do
25          v[t][k] ← b
26          r[t][k] ← a
27      end
28  end
29  return v[m][0], r
30 end
```

Memorizzazione delle decisioni

- L'algoritmo di PD usa la variabile temporanea aggiuntiva a per tenere traccia della decisione presa quando b viene aggiornato
- Se viene scelta la produzione x, $a \leftarrow x$ memorizza questa decisione
- Alla fine del ciclo interno $r[t][k] \leftarrow a$ la memorizza, insieme al valore ottimo $v[t][k] \leftarrow b$
- A causa del confronto stretto, in caso di pareggio viene memorizzata la produzione più piccola

Esempio: calcolo delle informazioni aggiuntive (1/8)

Consideriamo l'istanza del CLS dell'esempio:

t	1	2	3	4	5
d_t	2	1	3	4	1
sc_t	50	50	50	50	50
vc_t	0	0	0	0	0
hc_t	10	10	10	10	10
cap_t	4	4	4	4	4

Per ogni passo dell'algoritmo di PD, la tabella nelle slide seguenti riporta i valori di t e k, la cella $r[t][k]$ e la produzione x per cui il minimo della riga corrispondente dell'esempio precedente viene raggiunto per primo.

Esempio: calcolo delle informazioni aggiuntive (2/8)

t	k	$r[t][k]$	Azione
0	0	0	$r[0][0] \leftarrow 0$
0	1	0	$r[0][1] \leftarrow 0$
0	2	0	$r[0][2] \leftarrow 0$
0	3	0	$r[0][3] \leftarrow 0$
0	4	0	$r[0][4] \leftarrow 0$
0	5	0	$r[0][5] \leftarrow 0$
0	6	0	$r[0][6] \leftarrow 0$
0	7	0	$r[0][7] \leftarrow 0$
0	8	0	$r[0][8] \leftarrow 0$
0	9	0	$r[0][9] \leftarrow 0$
0	10	0	$r[0][10] \leftarrow 0$
0	11	0	$r[0][11] \leftarrow 0$

Esempio: calcolo delle informazioni aggiuntive (3/8)

t	k	$r[t][k]$	Azione
1	0	2	primo minimo in $x = 2$, ossia $v[0][0]+f_1(2)+hc_1 \cdot 0 = 0+50+0 = 50$: $r[1][0] \leftarrow 2$
1	1	3	primo minimo in $x = 3$, ossia $v[0][0]+f_1(3)+hc_1 \cdot 1 = 0+50+10 = 60$: $r[1][1] \leftarrow 3$
1	2	4	primo minimo in $x = 4$, ossia $v[0][0]+f_1(4)+hc_1 \cdot 2 = 0+50+20 = 70$: $r[1][2] \leftarrow 4$
1	3	0	tutti i candidati valgono $+\infty$: $r[1][3] \leftarrow 0$
1	4	0	tutti i candidati valgono $+\infty$: $r[1][4] \leftarrow 0$
1	5	0	tutti i candidati valgono $+\infty$: $r[1][5] \leftarrow 0$
1	6	0	tutti i candidati valgono $+\infty$: $r[1][6] \leftarrow 0$
1	7	0	tutti i candidati valgono $+\infty$: $r[1][7] \leftarrow 0$
1	8	0	tutti i candidati valgono $+\infty$: $r[1][8] \leftarrow 0$
1	9	0	tutti i candidati valgono $+\infty$: $r[1][9] \leftarrow 0$

Esempio: calcolo delle informazioni aggiuntive (4/8)

t	k	$r[t][k]$	Azione
2	0	0	primo minimo in $x = 0$, ossia $v[1][1]+f_2(0)+hc_2 \cdot 0 = 60+0+0 = 60$: $r[2][0] \leftarrow 0$
2	1	0	primo minimo in $x = 0$, ossia $v[1][2]+f_2(0)+hc_2 \cdot 1 = 70+0+10 = 80$: $r[2][1] \leftarrow 0$
2	2	3	primo minimo in $x = 3$, ossia $v[1][0]+f_2(3)+hc_2 \cdot 2 = 50+50+20 = 120$: $r[2][2] \leftarrow 3$
2	3	4	primo minimo in $x = 4$, ossia $v[1][0]+f_2(4)+hc_2 \cdot 3 = 50+50+30 = 130$: $r[2][3] \leftarrow 4$

Esempio: calcolo delle informazioni aggiuntive (5/8)

t	k	$r[t][k]$	Azione
2	4	4	primo minimo in $x = 4$, ossia $v[1][1]+f_2(4)+hc_2 \cdot 4 = 60+50+40 = 150$: $r[2][4] \leftarrow 4$
2	5	4	primo minimo in $x = 4$, ossia $v[1][2]+f_2(4)+hc_2 \cdot 5 = 70+50+50 = 170$: $r[2][5] \leftarrow 4$
2	6	0	tutti i candidati valgono $+\infty$: $r[2][6] \leftarrow 0$
2	7	0	tutti i candidati valgono $+\infty$: $r[2][7] \leftarrow 0$
2	8	0	tutti i candidati valgono $+\infty$: $r[2][8] \leftarrow 0$

Esempio: calcolo delle informazioni aggiuntive (6/8)

t	k	$r[t][k]$	Azione
3	0	3	primo minimo in $x = 3$, ossia $v[2][0]+f_3(3)+hc_3 \cdot 0 = 60+50+0 = 110$: $r[3][0] \leftarrow 3$
3	1	4	primo minimo in $x = 4$, ossia $v[2][0]+f_3(4)+hc_3 \cdot 1 = 60+50+10 = 120$: $r[3][1] \leftarrow 4$
3	2	4	primo minimo in $x = 4$, ossia $v[2][1]+f_3(4)+hc_3 \cdot 2 = 80+50+20 = 150$: $r[3][2] \leftarrow 4$
3	3	4	primo minimo in $x = 4$, ossia $v[2][2]+f_3(4)+hc_3 \cdot 3 = 120+50+30 = 200$: $r[3][3] \leftarrow 4$

Esempio: calcolo delle informazioni aggiuntive (7/8)

t	k	$r[t][k]$	Azione
3	4	4	primo minimo in $x = 4$, ossia $v[2][3]+f_3(4)+hc_3 \cdot 4 = 130+50+40 = 220$: $r[3][4] \leftarrow 4$
3	5	4	primo minimo in $x = 4$, ossia $v[2][4]+f_3(4)+hc_3 \cdot 5 = 150+50+50 = 250$: $r[3][5] \leftarrow 4$
4	0	4	primo minimo in $x = 4$, ossia $v[3][0]+f_4(4)+hc_4 \cdot 0 = 110+50+0 = 160$: $r[4][0] \leftarrow 4$
4	1	4	primo minimo in $x = 4$, ossia $v[3][1]+f_4(4)+hc_4 \cdot 1 = 120+50+10 = 180$: $r[4][1] \leftarrow 4$
5	0	0	primo minimo in $x = 0$, ossia $v[4][1]+f_5(0)+hc_5 \cdot 0 = 180+0+0 = 180$: $r[5][0] \leftarrow 0$

Esempio: calcolo delle informazioni aggiuntive (8/8)

Questa è la matrice r dopo l'esecuzione dell'algoritmo di PD:

$r[t][k]$	0	1	2	3	4	5	6	7	8	9	10	11
$t = 0$	0	0	0	0	0	0	0	0	0	0	0	0
$t = 1$	2	3	4	0	0	0	0	0	0	0	·	·
$t = 2$	0	0	3	4	4	4	0	0	0	·	·	·
$t = 3$	3	4	4	4	4	4	·	·	·	·	·	·
$t = 4$	4	4	·	·	·	·	·	·	·	·	·	·
$t = 5$	0	·	·	·	·	·	·	·	·	·	·	·

Ricostruzione di una soluzione ottima

- Una volta calcolata la matrice r, si può calcolare una soluzione ottima dell'istanza del CLS applicando la regola seguente per $t = m, m-1, \ldots, 1$, a partire da $k = 0$:

$$x_t \leftarrow r[t][k], \qquad k \leftarrow k + d_t - x_t.$$

- Si ottiene il vettore $(x_1, x_2, \ldots, x_m)$, in cui ogni componente x_t è il numero di unità prodotte nel periodo t in una soluzione ottima

- Il seguente pseudocodice implementa il calcolo di una soluzione ottima, data la matrice r

Algorithm: DP_CLS_SolComputation

Input : Il numero m di periodi e le domande $d[1..m]$ dell'istanza del CLS, e la matrice $r[0..m][0..D_{1,m}]$

Output: Una soluzione ottima dell'istanza del CLS

```
1 function DP_CLS_SolComputation(m, d, r)
2     x := new array[1..m]
3     k := 0
4     for t := m downto 1 do
5         x[t] ← r[t][k]
6         k ← k + d[t] − x[t]
7     end
8     return x
9 end
```

Esempio: calcolo di una soluzione ottima (1/3)

Dopo l'esecuzione dell'algoritmo di PD sull'istanza dell'esempio, abbiamo la matrice r dell'esempio precedente. Possiamo calcolare una soluzione ottima come segue:

Iterazione	t	k	$r[t][k]$	$x[1]$	$x[2]$	$x[3]$	$x[4]$	$x[5]$	$k \leftarrow k + d[t] - x[t]$
–	–	–	–	0	0	0	0	0	$t = 5,\ k = 0$
1	5	0	0	0	0	0	0	0	$k = 0 + 1 - 0 = 1$
2	4	1	4	0	0	0	4	0	$k = 1 + 4 - 4 = 1$
3	3	1	4	0	0	4	4	0	$k = 1 + 3 - 4 = 0$
4	2	0	0	0	0	4	4	0	$k = 0 + 1 - 0 = 1$
5	1	1	3	3	0	4	4	0	$k = 1 + 2 - 3 = 0$

Esempio: calcolo di una soluzione ottima (2/3)

Una soluzione ottima è il vettore:

$$(3, 0, 4, 4, 0)$$

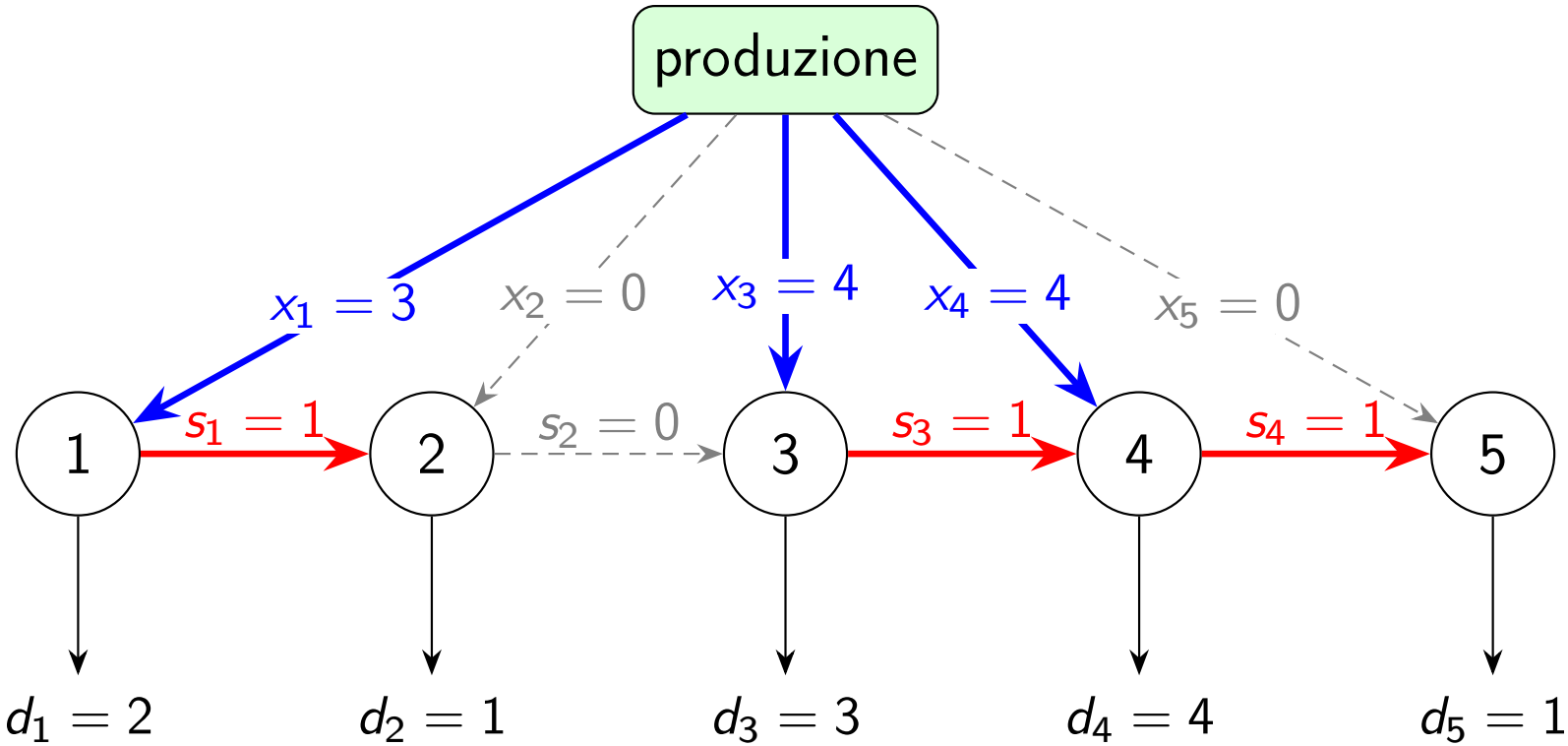
I setup sono nei periodi 1, 3, 4, le scorte sono $(s_1, \ldots, s_5) = (1, 0, 1, 1, 0)$, e il costo totale è (non c'è costo variabile di produzione)

$$\underbrace{50 + 50 + 50}_{\text{setup}} + \underbrace{10 \cdot 1 + 10 \cdot 0 + 10 \cdot 1 + 10 \cdot 1 + 10 \cdot 0}_{\text{mantenimento}}$$
$$= 150 + 30 = 180.$$

Il valore ottimo corrispondente è $v[5][0] = 180$, lo stesso valore dato da Gurobi sul modello $\mathrm{ILP}_{\mathrm{CLSP}}$; la capacità costa $180 - 170$ rispetto all'istanza senza capacità.

Esempio: calcolo di una soluzione ottima (3/3)

La rete della soluzione è:



Il periodo 4 produce anche se alla fine del periodo 3 c'è un'unità in scorta: la soluzione non ha la proprietà zero-inventory ordering (ed è l'unica soluzione ottima con quantità intere).

Numero di passi

- Indichiamo con $S(m, D, C)$ il numero di passi eseguiti da DP_CLS nel caso peggiore, dove:
  - m è il numero di periodi,
  - $D = D_{1,m}$ è la domanda totale,
  - $C = \max_{t \in T} cap_t$ è la capacità più grande

- Per ogni $t \geq 1$ abbiamo $R[t] + 1 \leq D$ (poiché $d_1 \geq 1$), e il ciclo interno ha al più $C + 1$ iterazioni

- Nel caso peggiore, la condizione della riga 16 è sempre vera

Conteggio delle esecuzioni (1/2)

Il numero di esecuzioni delle righe è:

- Le righe 2, 3, 7 e 8 vengono eseguite 1 volta ciascuna.
- Le righe 4 e 5 vengono eseguite m volte ciascuna.
- Le righe 9 e 10 vengono eseguite D volte ciascuna.
- La riga 12 viene eseguita m volte.

Conteggio delle esecuzioni (2/2)

Il numero di esecuzioni delle righe è:

- Le righe 13, 14 e 20 vengono eseguite $\sum_{t=1}^{m}(R[t]+1) \leq mD$ volte ciascuna.
- Le righe 15, 16 e 17 vengono eseguite al più $mD(C+1)$ volte ciascuna.
- La riga 23 viene eseguita 1 volta.

Il numero totale di passi

Quindi,

$$\begin{aligned} S(m, D, C) &\leq 5 + 3m + 2D + 3mD + 3mD(C + 1) \\ &= 5 + 3m + 2D + 6mD + 3mDC, \end{aligned}$$

e di conseguenza

$$S(m, D, C) = O(mDC).$$

Numero di passi dell'algoritmo di PD

L'algoritmo di PD esegue

$$S(m, D, C) = O(m\,D\,C)$$

passi nel caso peggiore.

- Il numero di passi dipende dai valori D e C, non solo dal numero m di periodi: l'algoritmo di PD è **pseudo-polinomiale**
- DP_CLS_InfoSol aggiunge solo un numero costante di operazioni dentro gli stessi cicli, quindi il suo numero di passi ha lo stesso ordine di crescita

Esempio: un'istanza con 6 periodi

- Consideriamo l'istanza capacitata con 6 periodi, domande

$$(50, 30, 10, 10, 40, 10),$$

  costo di setup 40, costo di mantenimento 1 e capacità 80
- Misurata in unità di 10, ha 35 stati con $t \geq 1$

- L'algoritmo di PD dà il valore ottimo 160, con il piano di produzione

$$(50, 50, 0, 0, 50, 0)$$

  (nelle unità originali): lo stesso valore dato da Gurobi sul modello $\mathrm{ILP}_{\mathrm{CLSP}}$

Capacità generali e capacità costante

- Con capacità cap_t generali il CLS è NP-difficile, quindi non ci si aspetta un algoritmo polinomiale
- L'algoritmo di PD pseudo-polinomiale è un metodo esatto naturale per piccoli valori di D e C
- Quando la capacità è la stessa in tutti i periodi, $cap_t = C$ per ogni $t \in T$, il CLS si può risolvere in tempo polinomiale, perché la proprietà zero-inventory ordering è sostituita da una più debole

Una proprietà più debole

Se $cap_t = C$ per ogni $t \in T$, esiste una soluzione ottima in cui, fra due periodi consecutivi con scorta vuota, al più un periodo produce una quantità strettamente compresa fra 0 e C; tutti gli altri periodi producono 0 o C.

- Il CLS a capacità costante ha la sua pagina nel catalogo dei problemi

Esempio: la proprietà sull'istanza dell'esempio

- L'istanza dell'esempio ha capacità costante $C = 4$
- La sua soluzione ottima $(3, 0, 4, 4, 0)$ ha scorte $(1, 0, 1, 1, 0)$, vuote alla fine dei periodi 0, 2 e 5
- Nei periodi 1–2 solo il periodo 1 produce una quantità strettamente compresa fra 0 e 4
- Nei periodi 3–5 ogni periodo produce 0 o 4, come afferma la proprietà

Riepilogo (1/2)

Passo 1: il valore ottimo:

- Lo stato è (t, k), con $k \leq D_{t+1,m}$; la ricorrenza prova ogni produzione $x \in X_t(k)$ e aggiunge $hc_t\, k$ fuori dal minimo
- Casi base $v(0, 0) = 0$ e $v(0, k) = +\infty$ per $k > 0$
- DP_CLS riempie la matrice v riga per riga: nell'esempio, $v[5][0] = 180$

Passo 2: una soluzione ottima:

- $r[t][k]$ memorizza la produzione scelta (la più piccola in caso di pareggio)
- All'indietro da $(m, 0)$: $x_t \leftarrow r[t][k]$, $k \leftarrow k + d_t - x_t$; nell'esempio, $(3, 0, 4, 4, 0)$

Riepilogo (2/2)

Numero di passi:

- $S(m, D, C) \leq 5 + 3m + 2D + 6mD + 3mDC$, quindi $S(m, D, C) = O(mDC)$
- L'algoritmo di PD è pseudo-polinomiale

Capacità costante:

- Con capacità generali il CLS è NP-difficile
- Con $cap_t = C$, fra due periodi consecutivi con scorta vuota al più un periodo produce una quantità strettamente compresa fra 0 e C, e il CLS è polinomiale

Errori tipici

Introduzione: errori tipici

- Elenchiamo gli errori che si fanno più spesso con l'algoritmo di PD per il CLS
- Riguardano la proprietà zero-inventory ordering, i casi base, l'intervallo delle scorte, il costo di mantenimento e il numero di passi
- Conoscerli aiuta a eseguire correttamente l'algoritmo a mano negli esercizi

Osservazione 1: Errori tipici (1/2)

- ***Applicare la proprietà zero-inventory ordering con le capacità.*** *Con le capacità un periodo può produrre mentre ci sono unità in scorta: nell'istanza dell'esempio l'unico piano ottimo viola la proprietà.*
- ***Dimenticare i casi base con valore*** $+\infty$***.*** *La scorta iniziale è vuota, quindi* $v(0, k) = +\infty$ *per* $k > 0$*; questo valore marca ogni stato che non si può raggiungere entro le capacità.*
- ***Ammettere una scorta troppo grande.*** *Alla fine del periodo* t *la scorta è al più* $D_{t+1,m}$*, poiché deve essere vuota alla fine del periodo* m*: le altre coppie* (t, k) *non sono stati.*

Osservazione: Errori tipici (2/2)

- ***Mettere il costo di mantenimento dentro la scelta.*** *Il costo $hc_t\, k$ della scorta finale si paga nello stato (t, k) qualunque sia la produzione x: si aggiunge fuori dal minimo.*
- ***Chiamare polinomiale*** $O(mDC)$***.*** *Il tempo di calcolo contiene la domanda totale D e la capacità più grande C, che sono numeri dell'input: l'algoritmo di PD è pseudo-polinomiale.*

Riepilogo

Errori tipici:

- Con le capacità la proprietà zero-inventory ordering non vale
- $v(0, k) = +\infty$ per $k > 0$ marca gli stati che non si possono raggiungere
- Alla fine del periodo t la scorta è al più $D_{t+1,m}$
- Il costo di mantenimento $hc_t\, k$ si aggiunge fuori dal minimo
- $O(mDC)$ è pseudo-polinomiale, non polinomiale

In sintesi

Introduzione: in sintesi

- Raccogliamo le idee chiave del capitolo sull'algoritmo di PD per il CLS
- Chiudiamo con il quadro riassuntivo dell'istanza dell'esempio: i bound trovati in questo capitolo

Riepilogo (1/2)

- Con le capacità la proprietà zero-inventory ordering non vale: un periodo può produrre mentre ci sono unità in scorta, e l'algoritmo di PD sui lotti di produzione non si applica.
- Lo stato è la coppia (t, k) del periodo e della scorta finale, con $k \leq D_{t+1,m}$; la ricorrenza $v(t, k)$ sceglie la produzione x del periodo t, e il valore ottimo è $v(m, 0)$.
- La matrice r delle produzioni scelte, letta all'indietro a partire dallo stato $(m, 0)$, dà un piano ottimo.
- L'algoritmo di PD richiede $O(mDC)$ passi, quindi è pseudo-polinomiale.
- Con una capacità costante vale una proprietà più debole, e il problema è polinomiale; con capacità generali è NP-difficile.

Riepilogo (2/2)

Quadro riassuntivo dell'esempio.

Istanza	Lower bound	Upper bound	Origine
L'istanza dell'esempio (capacitata, $C = 4$)	180	180	PD su (periodo, scorta)