

## Production planning problem

Equations (19)–(22)

$$\min \sum_{i \in P} sc_{it} y_{it} + \sum_{i \in P} \sum_{k=t}^{m-1} (cv_{ik} - \pi_{it} + \pi_{i,k+1}) z_{vik} + \sum_{i \in P} (cv_{im} - \pi_{it}) z_{v_{im}} - \mu_t$$

s.t.

$$\begin{aligned} \sum_{i \in P} st_{it} y_{it} + \sum_{i \in P} \sum_{k=t}^m vt_{it} sd_{ik} z_{vik} &\leq \text{cap}_t \\ \sum_{k=t}^m z_{vik} &\leq y_{it} \quad \forall i \in P \\ y_{it} \in \{0, 1\}, z_{vik} &\geq 0 \quad \forall i \in P, \forall k \in T, k \geq t \end{aligned}$$

## Production planning problem

If we LP-relax problem, setup constraint is equality  
Eliminate setup variable  $y$ .

$$\min \sum_{i \in P} \sum_{k=t}^{m-1} (cv_{it} - sc_{ik} - \pi_{it} + \pi_{i,k+1}) z_{vik} + \sum_{i \in P} (sc_{it} + cv_{im} - \pi_{it}) z_{v_{im}}$$

s.t.

$$\begin{aligned} \sum_{i \in P} \sum_{k=t}^m (st_{it} + vt_{it} sd_{ik}) z_{vik} &\leq \text{cap}_t \\ \sum_{k=t}^m z_{vik} &\leq 1 \quad \forall i \in P \\ 0 \leq z_{vik} &\leq 1 \quad \forall i \in P, \forall k \in T, k \geq t \end{aligned}$$

1

## Multiple-choice knapsack problem

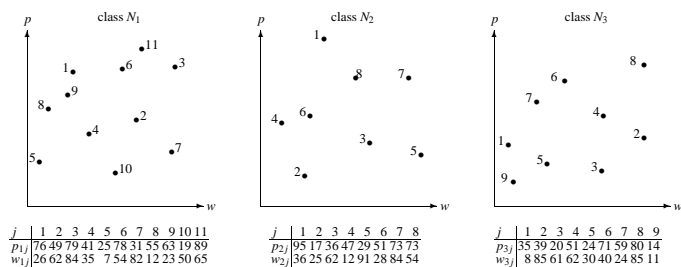
$$\min \sum_{i \in P} \sum_{k=t}^m p_{ik} z_{vik}$$

s.t.

$$\begin{aligned} \sum_{i \in P} \sum_{k=t}^m w_{ik} z_{vik} &\leq \text{cap}_t \\ \sum_{k=t}^m z_{vik} &\leq 1 \quad \forall i \in P \\ 0 \leq z_{vik} &\leq 1 \quad \forall i \in P, \forall k \in T, k \geq t \end{aligned}$$

2

## Multiple-choice knapsack problem



$$\max \quad z = \sum_{i=1}^k \sum_{j \in N_i} p_{ij} x_{ij}$$

$$\text{subject to} \quad \sum_{i=1}^k \sum_{j \in N_i} w_{ij} x_{ij} \leq c,$$

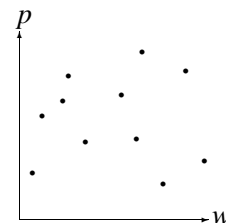
$$\sum_{j \in N_i} x_{ij} = 1, \quad i = 1, \dots, k,$$

$$x_{ij} \in \{0, 1\}, \quad i = 1, \dots, k, j \in N_i.$$

- $k$  classes  $N_1, \dots, N_k$  of items
- class  $N_i$  size  $n_i$ , total size  $n = \sum_{i=1}^k n_i$ .
- minimization equivalent to maximization

3

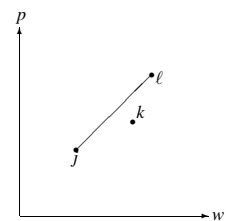
## Dominance



$k$  is dominated by item  $j$

$$w_{ij} \leq w_{ik} \quad \text{and} \quad p_{ij} \geq p_{ik}$$

## LP-dominance



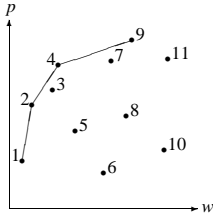
$k$  is LP-dominance by items  $j$  and  $\ell$

$$\frac{p_{i\ell} - p_{ik}}{w_{i\ell} - w_{ik}} \geq \frac{p_{ik} - p_{ij}}{w_{ik} - w_{ij}},$$

4

## Undominated items

LP-undominated items  $R_i$  (black) form the upper convex boundary of  $N_i$



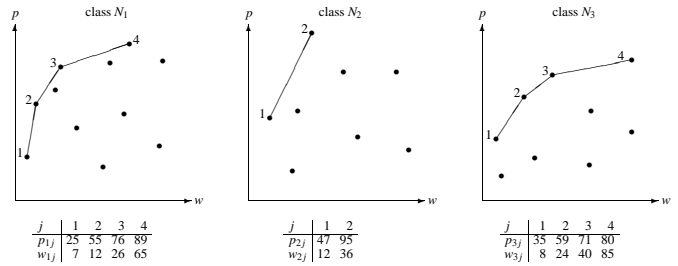
The set of undominated items is  $\{1, 2, 3, 4, 7, 9\}$ .  
The set of LP-extreme items is  $R_i = \{1, 2, 4, 9\}$ .  
 $R_i$  forms the upper left convex hull of  $N_i$ .

## Time complexity

Sort items increasing weight (trivial)  
 $O(n \log n)$

5

## Greedy LP



- remove LP-dominated
- $R_1 = \{1, 2, 3, 4\}$ ,  $R_2 = \{1, 2\}$ ,  $R_3 = \{1, 2, 3, 4\}$ .
- incremental efficiencies

$$e_{ij} = \frac{p_{ij} - p_{i,j-1}}{w_{ij} - w_{i,j-1}}$$

$$\begin{aligned} e_{12} &= 30/5 = 6.00 \\ e_{22} &= 48/24 = 2.00 \\ e_{13} &= 21/14 = 1.50 \\ e_{32} &= 24/16 = 1.50 \\ e_{33} &= 12/16 = 0.75 \\ e_{14} &= 13/39 = 0.33 \\ e_{34} &= 9/45 = 0.20 \end{aligned}$$

Optimal solution if  $c = 100$

$$x_{13} = 1, x_{22} = 1, x_{33} = \frac{14}{16}, x_{32} = \frac{2}{16}$$

6

## Time complexity of LP-solution

- Sort items in each class:  $\sum_{i=1}^k n_i \log n_i$ .
- Remove dominated  $\sum_{i=1}^k n_i = O(n)$ .
- Calculate incremental efficiencies  $O(n)$ .
- Sort incremental efficiencies  $O(n \log n)$

## Properties of LP-solution

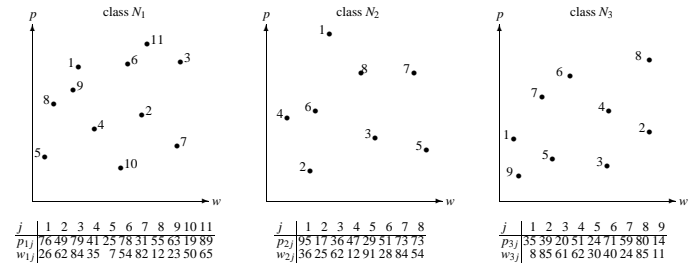
Optimal solution  $x^{\text{LP}}$  to LP-relaxed problem

- 1)  $x^{\text{LP}}$  has at most two fractional variables.
- 2) If  $x^{\text{LP}}$  has two fractional variables they must be adjacent variables in the (sorted) class  $R_s$ .
- 3) If  $x^{\text{LP}}$  has no fractional variables, then  $x^{\text{LP}}$  is an optimal solution to MCKP.

7

## If we knew optimal "slope"

Let  $\alpha$  be last  $e_{ij}$  considered



If we knew  $\alpha$  we could construct  $x^{\text{LP}}$  in  $O(n)$  time

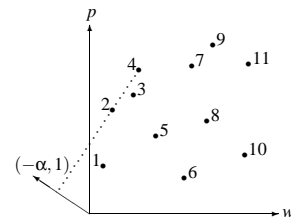


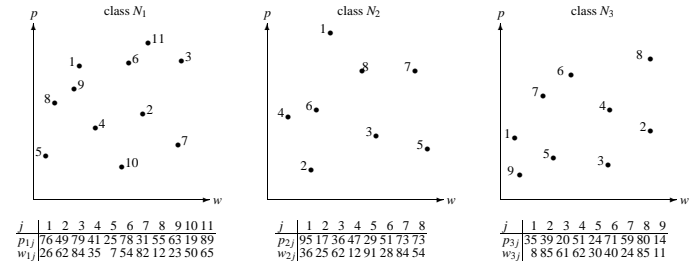
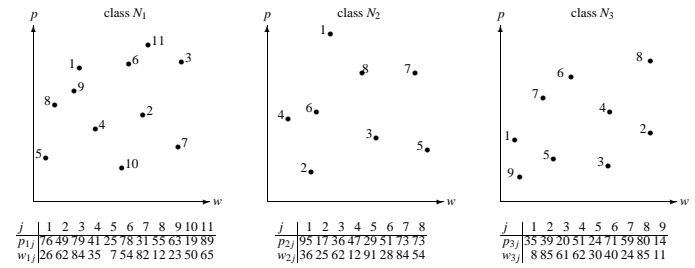
Figure 1: Projection of  $(w_{ij}, p_{ij}) \in N_i$  on  $(-\alpha, 1)$ . Here we have  $M_i(\alpha) = \{2, 4\}$  and  $a_i = 2, b_i = 4$ .

8

repeat forever

1. for all classes  $N_i$   
 Pair the items two by two as  $(ij, ik)$ .  
 Check for dominance  
 Continue until all items have been paired
2. for all classes  $N_i$   
 if the class has only one item  $j$  left then  
 Decrease the capacity  $c := c - w_{ij}$ , delete  $N_i$ .
3. for all pairs  $(ij, ik)$   
 Derive the slope  $\alpha_{ijk} := \frac{p_{ik} - p_{ij}}{w_{ik} - w_{ij}}$ .  
 Let  $\alpha$  be the median of the slopes  $\{\alpha_{ijk}\}$ .
4. for  $i = 1, \dots, m$   
 Derive  $M_i(\alpha)$  and  $a_i, b_i$
5. if  $\sum_{i=1}^m w_{ia_i} \leq c < \sum_{i=1}^m w_{ib_i}$ , then  
 $\alpha$  is the optimal slope  $\alpha^*$ . Stop.
6. if  $\sum_{i=1}^m w_{ia_i} \geq c$  then  
 for all pairs  $(ij, ik)$  with  $\alpha_{ijk} \leq \alpha$  delete item  $k$ .
7. if  $\sum_{i=1}^m w_{ib_i} < c$  then  
 for all pairs with  $\alpha_{ijk} \geq \alpha$  delete item  $j$ .

Dyer - Zemel (1980).



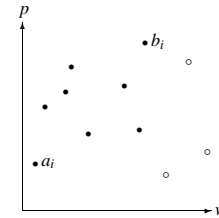
Example

Capacity  $c = 100$ .

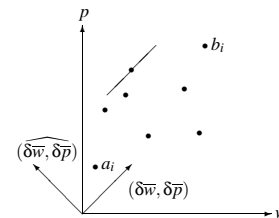
- Pair the items in the classes, in order we meet them
- In  $N_1$  we pair items (1,2) but 2 is dominated so we pair (1,3). Next we pair (4,5) then (6,7) but since 7 is dominated we pair (6,8). Finally we pair (9,10) noticing that 10 is dominated and hence pairing (9,11).
- In  $N_2$  we pair (1,2), then (3,4) but since 3 is dominated we try (4,5) but again 5 is dominated and hence we end with (4,6). Finally (7,8) are paired, but 7 is dominated by 8, hence we get the singleton 8.
- In  $N_3$  we pair (1,2), (3,4), (5,6) and finally (7,8). In this set we have the singleton 9.
- The slopes are  $\alpha_{1,1,3} = \frac{3}{38}$ ,  $\alpha_{1,5,4} = \frac{16}{38}$ ,  $\alpha_{1,8,6} = \frac{23}{42}$ ,  $\alpha_{1,9,11} = \frac{26}{42}$ ,  $\alpha_{2,1,2} = \frac{78}{11}$ ,  $\alpha_{2,4,6} = \frac{4}{16}$ ,  $\alpha_{3,1,2} = \frac{4}{77}$ ,  $\alpha_{3,3,4} = \frac{31}{11}$ ,  $\alpha_{3,5,6} = \frac{47}{10}$ ,  $\alpha_{3,7,8} = \frac{21}{61}$ . The median of these is  $\alpha = \frac{16}{38}$ .
- We find  $M_1(\alpha) = 1$ ,  $M_2(\alpha) = 1$ ,  $M_3(\alpha) = 6$ .
- Since the weight sum is  $26 + 36 + 40 \geq c$  we delete items (1,3), (1,4), (1,6), (2,6), (3,2), (3,8) and repeat the process.

Figures

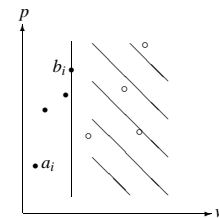
Preprocessing of  $N_i$ . White nodes are dominated by  $b_i$ .



Conclusion of  $N_i$ .



Partition set  $N_i$



## Theorem

The time complexity of the DyerZemel algorithm is  $O(n)$ .

## Proof

All items and all classes are lists (delete in  $O(1)$ ).

There are  $\sum_{i=1}^m \lfloor n_i/2 \rfloor$  pairs of items  $(ij, ik)$ . Since  $\alpha$  is the median of  $\{\alpha_{ijk}\}$ , half of the pairs will satisfy the criteria in Steps 6 or 7, and thus one item from these pairs will be deleted, i.e. at least  $\frac{1}{2} \sum_{i=1}^m \lfloor n_i/2 \rfloor$  items are deleted out of  $n = \sum_{i=1}^m n_i \leq \sum_{i=1}^m (2\lfloor n_i/2 \rfloor + 1)$ . Since  $\lfloor n_i/2 \rfloor \geq 1$ , each iteration deletes at least

$$\frac{\frac{1}{2} \sum_{i=1}^m \lfloor n_i/2 \rfloor}{\sum_{i=1}^m (2\lfloor n_i/2 \rfloor + 1)} \geq \frac{\sum_{i=1}^m \lfloor n_i/2 \rfloor}{2m + 4 \sum_{i=1}^m \lfloor n_i/2 \rfloor} \geq \frac{1}{6},$$

of the items. The running time now becomes

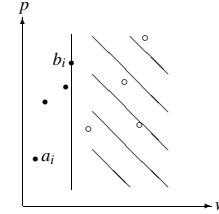
$$O\left(n + \frac{5}{6}n + \left(\frac{5}{6}\right)^2 n + \left(\frac{5}{6}\right)^3 n + \dots\right) = O\left(\frac{n}{1 - \frac{5}{6}}\right) = O(n),$$

13

## Further improvements

Pisinger (1995) proposed to improve the above algorithm

- in Step 6 delete in  $N_i$  all items with  $w_{ij} \geq w_{ia_i}$  since the current slope  $\alpha$  should be increased, meaning that items with  $w_{ij} > w_{ia_i}$  will not be considered.
- in Step 7, delete in  $N_i$  all items with  $p_{ij} \leq p_{ib_i}$  since the current slope  $\alpha$  needs to be decreased, and hence items with larger profits  $p_{ij} > p_{ib_i}$  will enter the LP-solution.



14

## Dynamic Programming

$z_\ell(d)$  optimal solution value to MCKP defined on first  $\ell$  classes, capacity  $d$

$$z_\ell(d) := \max \left\{ \sum_{i=1}^{\ell} \sum_{j \in N_i} p_{ij} x_{ij} \mid \begin{array}{l} \sum_{i=1}^{\ell} \sum_{j \in N_i} w_{ij} x_{ij} \leq d, \\ \sum_{j \in N_i} x_{ij} = 1, \quad i = 1, \dots, \ell, \\ x_{ij} \in \{0, 1\}, \end{array} \right\}$$

Initially:

$$z_0(d) := 0 \text{ for all } d = 0, \dots, c$$

Recursion:

$$z_\ell(d) = \max \begin{cases} z_{\ell-1}(d - w_{\ell 1}) + p_{\ell 1} & \text{if } 0 \leq d - w_{\ell 1}, \\ z_{\ell-1}(d - w_{\ell 2}) + p_{\ell 2} & \text{if } 0 \leq d - w_{\ell 2}, \\ \vdots \\ z_{\ell-1}(d - w_{\ell n_\ell}) + p_{\ell n_\ell} & \text{if } 0 \leq d - w_{\ell n_\ell}, \end{cases}$$

Assume  $\max\{\emptyset\} = -\infty$ .

15

## Dynamic Programming

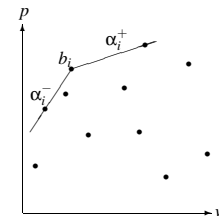
Disadvantages:

- We need to run recursion for all items  $\sum_{i=1}^k n_i c = O(nc)$
- We need to reduce all dominated items  $O(n \log n)$

But: LP-optimal choice is often also IP-optimal choice

## Gradients

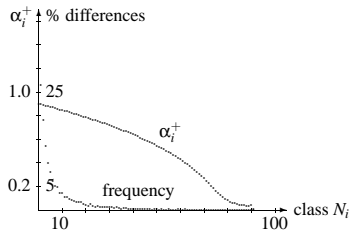
Gradients  $\alpha_i^+, \alpha_i^-$  in class  $N_i$



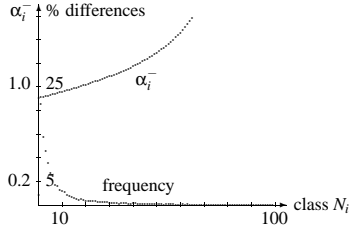
16

## Verification of assumption

Frequency of classes  $N_i$  where IP-optimal choice differ from LP-optimal choice, compared to gradient  $\alpha_i^+$



Frequency of classes  $N_i$  where IP-optimal choice differ from LP-optimal choice, compared to gradient  $\alpha_i^-$ .



17

## Better dynamic programming algorithm

- Find LP-solution in  $O(n)$  time
- Find gradients in  $O(n)$  time
- Run dynamic programming from LP-optimal solution
- Consider classes in order according to gradients
- Gradients used as upper bounds

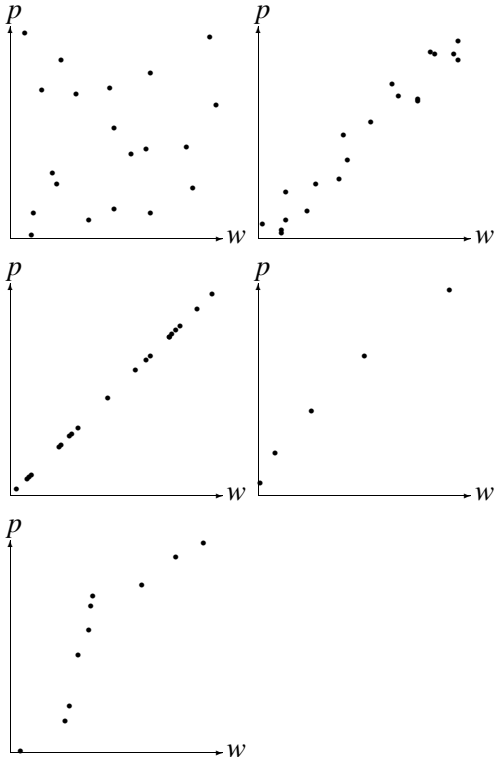
Lazy principle

- Gradients are sorted in lazy way
- Dominated items in classes are only removed when class is considered

Observed running time for many instances  $O(n)$ .

18

## Instances



19

## Solution times

$m$	$n_i$	Uncorrelated		Weakly corr.		Strongly corr.		Subset sum		Monotone	
		$R_1$	$R_2$	$R_1$	$R_2$	$R_1$	$R_2$	$R_1$	$R_2$	$R_1$	$R_2$
10	10	0	0	6	50	447	649	2	9	0	1
100	10	1	1	11	—	—	—	1	21	4	6
1000	10	9	21	8	—	—	—	3	19	12	113
10000	10	76	307	81	165	—	—	35	35	94	585
10	100	0	0	5	—	—	—	3	34	0	1
100	100	3	1	16	97	—	—	3	220	8	13
1000	100	55	159	69	1804	—	—	33	206	80	256
10	1000	7	4	8	—	—	—	9	199	7	34
100	1000	47	47	61	784	2176	—	49	49	51	66

Total computing time, DyerKayalWalker, in milliseconds (Intel Pentium 4, 1.5 GHz).  
Averages of 100 instances.

$m$	$n_i$	Uncorrelated		Weakly corr.		Strongly corr.		Subset sum		Monotone	
		$R_1$	$R_2$	$R_1$	$R_2$	$R_1$	$R_2$	$R_1$	$R_2$	$R_1$	$R_2$
10	10	0	0	2	16	4	32	2	21	0	0
100	10	1	1	8	144	337	3214	3	26	4	5
1000	10	14	18	48	190	11321	—	18	41	61	96
10000	10	612	878	1082	1766	—	—	204	281	1863	2520
10	100	0	0	6	111	—	—	6	171	5	7
100	100	5	5	16	133	—	—	15	177	43	110
1000	100	54	63	174	391	—	—	114	336	655	1125
10	1000	4	4	14	301	1059	—	57	1020	30	229
100	1000	46	48	82	647	14931	—	134	1139	224	2547

Total computing time, DyerRihaWalker, in milliseconds (Intel Pentium 4, 1.5 GHz).  
Averages of 100 instances.

$m$	$n_i$	Uncorrelated		Weakly corr.		Strongly corr.		Subset sum		Monotone	
		$R_1$	$R_2$	$R_1$	$R_2$	$R_1$	$R_2$	$R_1$	$R_2$	$R_1$	$R_2$
10	10	0	0	1	9	2	14	1	20	0	0
100	10	0	0	2	31	61	561	1	14	1	1
1000	10	3	3	4	30	964	9432	1	12	5	8
10000	10	26	34	28	44	18440	152166	15	14	38	48
10	100	0	0	6	83	4	29	10	134	2	3
100	100	1	1	5	81	52	828	1	89	6	10
1000	100	11	12	22	64	1419	19484	10	11	32	40
10	1000	2	2	19	378	335	20	2	1138	33	136
100	1000	8	10	25	155	15835	458	12	15	60	411

Total computing time, MCKNAP, in milliseconds (Intel Pentium 4, 1.5 GHz).  
Averages of 100 instances.

20