

# Multi-Dimensional/ Knapsack Problem

In the original knapsack problem, the value of the contents of the knapsack is maximized subject to a single capacity constraint, for example weight. In the multi-dimensional knapsack problem, additional capacity constraints, such as volume, must be enforced.

Mathematical statement of 2-dimensional problem:

$$\text{Maximize } \sum_{j=1}^n v_j x_j$$

$$\text{subject to } \sum_{j=1}^n a_{1j} x_j \leq b_1$$

$$\sum_{j=1}^n a_{2j} x_j \leq b_2$$

$$x_j \in X_j, \quad j = 1, \dots, n$$

The coefficients  $v_j$  and  $a_{ij}$  are nonnegative real numbers, and the set  $X_j$  may be the binary set  $\{0,1\}$  or the set of all nonnegative integers.

$b_i$  = capacity of knapsack with respect to measure  $i$  (e.g.,  $i=1$ : weight,  $i=2$ : volume)

$a_{ij}$  = measure  $i$  of item  $j$ , e.g., weight and volume

# Dynamic Programming

Let's assume that the coefficients  $a_{ij}$  are nonnegative integers, and the capacity limits  $b_i$  are positive integers.

In the DP model for the 1-dimensional knapsack problem, a **stage** is defined for each of the  $n$  items, and the **state** of the system at stage  $j$  is the unused capacity after items have been added in the previous stages.

In the **multi-dimensional** knapsack problem, the state of the system is a **vector** of dimension  $m$ , one element per capacity constraint, for example ( *available weight, available volume* )

Consider the **two-dimensional** knapsack problem with

data:  $n = \# \text{ items} = 6$

Item $j$	1	2	3	4	5	6
Value $v_j$	2	3	3	4	4	5
Weight $a_{1j}$	1	2	1	3	2	3
Volume $a_{2j}$	2	1	3	2	2	3

Maximum **weight** is 6 and maximum **volume** is 4.

Define **stages**  $j=1, 2, \dots, 6$  where

**decision**  $x_j = 1$  if item  $j$  is to be included in the knapsack,  
else **0**

**state**  $(s_1, s_2)$  where

$s_1 \in \{0, 1, \dots, 6\}$  is the slack in the **weight** constraint, and

$s_2 \in \{0, 1, \dots, 4\}$  is the slack in the **volume** constraint.

Thus the state space contains  $7 \times 5 = 35$  elements.

### Optimal value function

Using a **backward** recursion and imagining that we begin by deciding whether to include item #6 and end by deciding whether to include item #1,

$f_j(s_1, s_2) =$  maximum total value of items  $j, j-1, \dots, 1$  which

can be included if the knapsack if the weight and volume are restricted to  $s_1$  and  $s_2$ , respectively.

We aim, of course, is to determine the value of  $f_6(6, 4)$ .

### Recursion:

$$f_j(s_1, s_2) = \max_{x_j \in \{0, 1\}} \left\{ v_j x_j + f_{j-1}(s_1 - a_{1j} x_j, s_2 - a_{2j} x_j) \right\}, j = 6, 5, \dots, 1$$

$$f_0(s_1, s_2) = 0 \text{ for all nonnegative integers } s_1 \text{ and } s_2$$

## APL definition of optimal value function $f_n$

```

▽ z←F N;t
⌘
⌘ Optimal Value Function for 2-D knapsack problem
⌘
:if N=0 ⌘ Terminal conditions
z←((ρs)ρ0),-BIG ⌘ Big penalty for infeasible states
:else
⌘ recursion
z←MAX (((ρs)ρ0)∘.+V[N]×x)+(F N-1)[TRANSITION s∘.-W[N]×x]
:endif
▽

```

Definition of **state** & **decision** vectors:

```

s←,(0,16)∘.,0,14
x←0 1

```

Definition of **constants**

```

V← 2 3 3 4 4 5
W← (1 2 1 3 2 3),“(2 1 3 2 2 3)

```

### ---Stage 1---

s	x:	0	1	Maximum
0	0	0.00	-99999.99	0.00
0	1	0.00	-99999.99	0.00
0	2	0.00	-99999.99	0.00
0	3	0.00	-99999.99	0.00
0	4	0.00	-99999.99	0.00
1	0	0.00	-99999.99	0.00
1	1	0.00	-99999.99	0.00
1	2	0.00	2.00	2.00
1	3	0.00	2.00	2.00
1	4	0.00	2.00	2.00
2	0	0.00	-99999.99	0.00
2	1	0.00	-99999.99	0.00
2	2	0.00	2.00	2.00
2	3	0.00	2.00	2.00
2	4	0.00	2.00	2.00
3	0	0.00	-99999.99	0.00
3	1	0.00	-99999.99	0.00
3	2	0.00	2.00	2.00
3	3	0.00	2.00	2.00
3	4	0.00	2.00	2.00
4	0	0.00	-99999.99	0.00
4	1	0.00	-99999.99	0.00
4	2	0.00	2.00	2.00
4	3	0.00	2.00	2.00
4	4	0.00	2.00	2.00
5	0	0.00	-99999.99	0.00
5	1	0.00	-99999.99	0.00
5	2	0.00	2.00	2.00
5	3	0.00	2.00	2.00
5	4	0.00	2.00	2.00

We begin with the computation of  $f_1(\cdot)$  at stage 1, i.e.,

we consider that only item #1 remains to be added.

Recall that

Item j	1
Value $v_j$	2
Weight $a_{1j}$	1
Volume $a_{2j}$	2

### ---Stage 2---

s	x:	0	1	Maximum
0	0	0.00	-99999.99	0.00
0	1	0.00	-99999.99	0.00
0	2	0.00	-99999.99	0.00
0	3	0.00	-99999.99	0.00
0	4	0.00	-99999.99	0.00
1	0	0.00	-99999.99	0.00
1	1	0.00	-99999.99	0.00
1	2	2.00	-99999.99	2.00
1	3	2.00	-99999.99	2.00
1	4	2.00	-99999.99	2.00
2	0	0.00	-99999.99	0.00
2	1	0.00	3.00	3.00
2	2	2.00	3.00	3.00
2	3	2.00	3.00	3.00
2	4	2.00	3.00	3.00
3	0	0.00	-99999.99	0.00
3	1	0.00	3.00	3.00
3	2	2.00	3.00	3.00
3	3	2.00	5.00	5.00
3	4	2.00	5.00	5.00
4	0	0.00	-99999.99	0.00
4	1	0.00	3.00	3.00
4	2	2.00	3.00	3.00
4	3	2.00	5.00	5.00
4	4	2.00	5.00	5.00
5	0	0.00	-99999.99	0.00
5	1	0.00	3.00	3.00
5	2	2.00	3.00	3.00
5	3	2.00	5.00	5.00
5	4	2.00	5.00	5.00

Next we imagine that only items #1 & 2 remain to be added to the knapsack, and compute their optimal value,  $f_2(\cdot)$  where

Item j	2
Value $v_j$	3
Weight $a_{1j}$	2
Volume $a_{2j}$	1

### ---Stage 3---

s	x:	0	1	Maximum
0	0	0.00	-99999.99	0.00
0	1	0.00	-99999.99	0.00
0	2	0.00	-99999.99	0.00
0	3	0.00	-99999.99	0.00
0	4	0.00	-99999.99	0.00
1	0	0.00	-99999.99	0.00
1	1	0.00	-99999.99	0.00
1	2	2.00	-99999.99	2.00
1	3	2.00	3.00	3.00
1	4	2.00	3.00	3.00
2	0	0.00	-99999.99	0.00
2	1	3.00	-99999.99	3.00
2	2	3.00	-99999.99	3.00
2	3	3.00	3.00	3.00
2	4	3.00	3.00	3.00
3	0	0.00	-99999.99	0.00
3	1	3.00	-99999.99	3.00
3	2	3.00	-99999.99	3.00
3	3	5.00	3.00	5.00
3	4	5.00	6.00	6.00
4	0	0.00	-99999.99	0.00
4	1	3.00	-99999.99	3.00
4	2	3.00	-99999.99	3.00
4	3	5.00	3.00	5.00
4	4	5.00	6.00	6.00
5	0	0.00	-99999.99	0.00
5	1	3.00	-99999.99	3.00
5	2	3.00	-99999.99	3.00
5	3	5.00	3.00	5.00
5	4	5.00	6.00	6.00

s	x:	0	1	Maximum
6	0	0.00	-99999.99	0.00
6	1	3.00	-99999.99	3.00
6	2	3.00	-99999.99	3.00
6	3	5.00	3.00	5.00
6	4	5.00	6.00	6.00

Item j	3
Value $v_j$	3
Weight $a_{1j}$	1
Volume $a_{2j}$	3

---Stage 4---

s	x:	0	1	Maximum
0	0	0.00	-99999.99	0.00
0	1	0.00	-99999.99	0.00
0	2	0.00	-99999.99	0.00
0	3	0.00	-99999.99	0.00
0	4	0.00	-99999.99	0.00
1	0	0.00	-99999.99	0.00
1	1	0.00	-99999.99	0.00
1	2	2.00	-99999.99	2.00
1	3	3.00	-99999.99	3.00
1	4	3.00	-99999.99	3.00
2	0	0.00	-99999.99	0.00
2	1	3.00	-99999.99	3.00
2	2	3.00	-99999.99	3.00
2	3	3.00	-99999.99	3.00
2	4	3.00	-99999.99	3.00
3	0	0.00	-99999.99	0.00
3	1	3.00	-99999.99	3.00
3	2	3.00	4.00	4.00
3	3	5.00	4.00	5.00
3	4	6.00	4.00	6.00
4	0	0.00	-99999.99	0.00
4	1	3.00	-99999.99	3.00
4	2	3.00	4.00	4.00
4	3	5.00	4.00	5.00
4	4	6.00	6.00	6.00
5	0	0.00	-99999.99	0.00
5	1	3.00	-99999.99	3.00
5	2	3.00	4.00	4.00
5	3	5.00	7.00	7.00
5	4	6.00	7.00	7.00

<b>Item j</b>	<b>4</b>
<b>Value <math>v_j</math></b>	<b>4</b>
<b>Weight <math>a_{1j}</math></b>	<b>3</b>
<b>Volume <math>a_{2j}</math></b>	<b>2</b>

s	x:	0	1	Maximum
6	0	0.00	-99999.99	0.00
6	1	3.00	-99999.99	3.00
6	2	3.00	4.00	4.00
6	3	5.00	7.00	7.00
6	4	6.00	7.00	7.00

---Stage 5---

s	x:	0	1	Maximum
0	0	0.00	-99999.99	0.00
0	1	0.00	-99999.99	0.00
0	2	0.00	-99999.99	0.00
0	3	0.00	-99999.99	0.00
0	4	0.00	-99999.99	0.00
1	0	0.00	-99999.99	0.00
1	1	0.00	-99999.99	0.00
1	2	2.00	-99999.99	2.00
1	3	3.00	-99999.99	3.00
1	4	3.00	-99999.99	3.00
2	0	0.00	-99999.99	0.00
2	1	3.00	-99999.99	3.00
2	2	3.00	4.00	4.00
2	3	3.00	4.00	4.00
2	4	3.00	4.00	4.00
3	0	0.00	-99999.99	0.00
3	1	3.00	-99999.99	3.00
3	2	4.00	4.00	4.00
3	3	5.00	4.00	5.00
3	4	6.00	6.00	6.00
4	0	0.00	-99999.99	0.00
4	1	3.00	-99999.99	3.00
4	2	4.00	4.00	4.00
4	3	5.00	7.00	7.00
4	4	6.00	7.00	7.00
5	0	0.00	-99999.99	0.00
5	1	3.00	-99999.99	3.00
5	2	4.00	4.00	4.00
5	3	7.00	7.00	7.00
5	4	7.00	8.00	8.00

s	x:	0	1	Maximum
6	0	0.00	-99999.99	0.00
6	1	3.00	-99999.99	3.00
6	2	4.00	4.00	4.00
6	3	7.00	7.00	7.00
6	4	7.00	8.00	8.00

<b>Item j</b>	<b>5</b>
<b>Value <math>v_j</math></b>	<b>4</b>
<b>Weight <math>a_{1j}</math></b>	<b>2</b>
<b>Volume <math>a_{2j}</math></b>	<b>2</b>

---Stage 6---

s	x:	0	1	Maximum
0	0	0.00	-99999.99	0.00
0	1	0.00	-99999.99	0.00
0	2	0.00	-99999.99	0.00
0	3	0.00	-99999.99	0.00
0	4	0.00	-99999.99	0.00
1	0	0.00	-99999.99	0.00
1	1	0.00	-99999.99	0.00
1	2	2.00	-99999.99	2.00
1	3	3.00	-99999.99	3.00
1	4	3.00	-99999.99	3.00
2	0	0.00	-99999.99	0.00
2	1	3.00	-99999.99	3.00
2	2	4.00	-99999.99	4.00
2	3	4.00	-99999.99	4.00
2	4	4.00	-99999.99	4.00
3	0	0.00	-99999.99	0.00
3	1	3.00	-99999.99	3.00
3	2	4.00	-99999.99	4.00
3	3	5.00	5.00	5.00
3	4	6.00	5.00	6.00
4	0	0.00	-99999.99	0.00
4	1	3.00	-99999.99	3.00
4	2	4.00	-99999.99	4.00
4	3	7.00	5.00	7.00
4	4	7.00	5.00	7.00
5	0	0.00	-99999.99	0.00
5	1	3.00	-99999.99	3.00
5	2	4.00	-99999.99	4.00
5	3	7.00	5.00	7.00
5	4	8.00	8.00	8.00

s	x:	0	1	Maximum
6	0	0.00	-99999.99	0.00
6	1	3.00	-99999.99	3.00
6	2	4.00	-99999.99	4.00
6	3	7.00	5.00	7.00
6	4	8.00	8.00	8.00

Since we want only the value of  $f_6(6,4)$ , only the last row of this table is necessary!

<b>Item j</b>	<b>6</b>
<b>Value <math>v_j</math></b>	<b>5</b>
<b>Weight <math>a_{1j}</math></b>	<b>3</b>
<b>Volume <math>a_{2j}</math></b>	<b>3</b>

Optimal value is **8.00**  
There are **2 optimal solutions**

Optimal Solution No. **1**

stage	state	decision
6	6 4	Omit
5	6 4	Include
4	4 2	Include
3	1 0	Omit
2	1 0	Omit
1	1 0	Omit
0	1 0	

Weight= 5, Volume = 4

Optimal Solution No. **2**

stage	state	decision
6	6 4	Include
5	3 1	Omit
4	3 1	Omit
3	3 1	Omit
2	3 1	Include
1	1 0	Omit
0	1 0	

Weight= 5, volume= 4

Note that there are  $(1+b_1) \times (1+b_2)$  elements in the state space, or in general,  $\prod_{i=1}^m (1+b_i)$  elements, which for even modest values of  $b_i$  can be quite large -- the so-called "*curse of dimensionality*"-- and make dynamic programming computations prohibitive.

The dimension of the state space can be reduced to 1 by solving a one-dimensional knapsack problem which is a **relaxation** of the original two-dimensional knapsack problem, that is,

*the feasible region of the two-dimensional knapsack problem is contained within the feasible region of the relaxation.*

Two approaches for relaxing constraints are

- **Lagrangian Relaxation**, in which only one constraint is kept (unchanged) and the objective includes a penalty term for violation of the other.
- **Surrogate Relaxation**, in which a nonnegative combination of the original constraints are used, but the objective function is unchanged.

## Lagrangian Relaxation

### Reducing Dimensionality by Lagrangian Relaxation

We **relax**, i.e., no longer enforce, one of the capacity restrictions, and introduce a **Lagrangian multiplier**  $\lambda$  which we will interpret as the value ("*shadow price*") of one unit of the relaxed capacity.

For example, in our two-dimensional knapsack problem, we will no longer impose the volume restriction-- instead, we place a value  $\lambda$  on a unit of volume so that the value of including item  $j$  will be reduced from  $v_j$  to  $(v_j - \lambda a_{2j})$ .

It can easily be shown that the optimal value of the resulting *one-dimensional knapsack*, i.e.,

$$\text{Maximize } \sum_{j=1}^n v_j x_j + \lambda \left( b_2 - \sum_{j=1}^n a_{2j} x_j \right) = \sum_{j=1}^n (v_j - \lambda a_{2j}) x_j + \lambda b_2$$

subject to  $\sum_{j=1}^n a_{1j} x_j \leq b_1$

$$x_j \in X_j, j = 1, \dots, n$$

is an **upper bound** on, i.e., *at least as large* as, the optimal value of the original two-dimensional problem.

The *Lagrangian Dual problem* is to select a value of  $\lambda$  so that this upper bound is as *small* as possible.

**A crude search algorithm for Lagrangian dual:**

**Step 0.** Initialize  $\lambda = 0$ .

**Step 1.** Solve the one-dimensional Lagrangian relaxation to find all optimal solutions  $x^*(\lambda)$  and the associated volumes  $\sum_{j=1}^n a_{2j} x_j^*(\lambda)$ . Let  $m$  &  $M$  be the minimum and maximum volumes, respectively.

**Step 2.** If  $m \leq b_2 \leq M$ , STOP. If the volume  $\sum_{j=1}^n a_{2j} x_j^*(\lambda)$  of a solution is *exactly* that available,  $b_2$ , then that solution  $x^*(\lambda)$  is optimal for the original problem. Otherwise a duality gap exists and none of the solutions  $x^*(\lambda)$  are optimal in the original problem

**Step 3.** If  $b_2 < m$ , i.e., the volume restriction is violated by all optima, increase the "shadow price"  $\lambda$  placed on a unit of volume, while if  $b_2 > M$ , i.e., the volume restriction is "slack", decrease  $\lambda$ . Return to Step 1.

★★★★★★★★★★★★★★★★

**Example:** Relax the volume restriction of the 2-dimensional knapsack problem above, with Lagrange multiplier initial value  $\lambda = 0$ . The results are shown below.

---Stage 1---

s \ x:	0	1	Maximum
0	0.00	-99.99	0.00
1	0.00	2.00	2.00
2	0.00	2.00	2.00
3	0.00	2.00	2.00
4	0.00	2.00	2.00
5	0.00	2.00	2.00
6	0.00	2.00	2.00

---Stage 2---

s \ x:	0	1	Maximum
0	0.00	-99.99	0.00
1	2.00	-99.99	2.00
2	2.00	3.00	3.00
3	2.00	5.00	5.00
4	2.00	5.00	5.00
5	2.00	5.00	5.00
6	2.00	5.00	5.00

---Stage 3---

s \ x:	0	1	Maximum
0	0.00	-99.99	0.00
1	2.00	3.00	3.00
2	3.00	5.00	5.00
3	5.00	6.00	6.00
4	5.00	8.00	8.00
5	5.00	8.00	8.00
6	5.00	8.00	8.00

---Stage 4---

s \ x:	0	1	Maximum
0	0.00	-99.99	0.00
1	3.00	-99.99	3.00
2	5.00	-99.99	5.00
3	6.00	4.00	6.00
4	8.00	7.00	8.00
5	8.00	9.00	9.00
6	8.00	10.00	10.00

---Stage 5---

s \ x:	0	1	Maximum
0	0.00	-99.99	0.00
1	3.00	-99.99	3.00
2	5.00	4.00	5.00
3	6.00	7.00	7.00
4	8.00	9.00	9.00
5	9.00	10.00	10.00
6	10.00	12.00	12.00

---Stage 6---

s \ x:	0	1	Maximum
6	12.00	12.00	12.00

\*\*\* Optimal value is 12.00 \*\*\*

There are 2 optimal solutions

**Optimal Solution No. 1**

stage	state	decision
6	6	Omit
5	6	Include
4	5	Include
3	4	Include
2	3	Include
1	2	Include
0	1	

Volume used by this solution: 10

**Optimal Solution No. 2**

stage	state	decision
6	6	Include
5	5	Include
4	4	Omit
3	4	Include
2	3	Include
1	2	Include
0	1	

Volume used by this solution: 11

Because both solutions exceed the volume capacity (which is only 4 units), the Lagrange multiplier  $\lambda$  must be increased. Suppose we increase the multiplier to the value 1.00, i.e., each unit of volume has a "shadow price" of \$1.00.

Item j	1	2	3	4	5	6
Value $v_j$	2	3	3	4	4	5
$v_j - \lambda a_{2j}$	0	2	0	2	2	2
Weight $a_{1j}$	1	2	1	3	2	3
Volume $a_{2j}$	2	1	3	2	2	3

---Stage 1---

s \ x:	0	1	Maximum
0	0.00	-99.99	0.00
1	0.00	0.00	0.00
2	0.00	0.00	0.00
3	0.00	0.00	0.00
4	0.00	0.00	0.00
5	0.00	0.00	0.00
6	0.00	0.00	0.00

---Stage 4---

s \ x:	0	1	Maximum
0	0.00	-99.99	0.00
1	0.00	-99.99	0.00
2	2.00	-99.99	2.00
3	2.00	2.00	2.00
4	2.00	2.00	2.00
5	2.00	4.00	4.00
6	2.00	4.00	4.00

---Stage 2---

s \ x:	0	1	Maximum
0	0.00	-99.99	0.00
1	0.00	-99.99	0.00
2	0.00	2.00	2.00
3	0.00	2.00	2.00
4	0.00	2.00	2.00
5	0.00	2.00	2.00
6	0.00	2.00	2.00

---Stage 5---

s \ x:	0	1	Maximum
0	0.00	-99.99	0.00
1	0.00	-99.99	0.00
2	2.00	2.00	2.00
3	2.00	2.00	2.00
4	2.00	4.00	4.00
5	4.00	4.00	4.00
6	4.00	4.00	4.00

---Stage 3---

s \ x:	0	1	Maximum
0	0.00	-99.99	0.00
1	0.00	0.00	0.00
2	2.00	0.00	2.00
3	2.00	2.00	2.00
4	2.00	2.00	2.00
5	2.00	2.00	2.00
6	2.00	2.00	2.00

---Stage 6---

s \ x:	0	1	Maximum
6	4.00	4.00	4.00

The result: **seventeen** optimal solutions!

Item	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	1	0	0	1	0	1	0	1	0	0	1	0	0	0	1	0
2	1	1	1	1	1	1	1	0	0	0	1	1	1	0	0	0	0
3	0	0	1	0	0	1	1	0	0	1	0	0	1	0	0	0	1
4	1	1	1	0	0	0	0	1	1	1	0	0	0	1	0	0	0
5	0	0	0	1	1	1	1	1	1	1	0	0	0	0	1	1	1
6	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
Vol.	3	5	6	3	5	6	8	4	6	7	4	6	7	5	5	7	8

The optimal value in each case is \$8, i.e., value of knapsack (\$4) + value of 4 units of volume (\$4).

Notice that, of the 17 solutions of the relaxed problem,

- 2 solutions (#8 & 11) use *exactly* 4 units of volume satisfying a *complementary slackness* condition

$$\lambda \left( b_i - \sum_{j=1}^n a_{ij} x_j \right) = 0$$

- 2 solutions (#1 & 4) use less (and are therefore *feasible*), while
- the remaining 13 solutions exceed the 4 unit volume restriction and are *infeasible*.

The 2 solutions satisfying the complementary slackness condition are optimal in the original problem. The other two feasible solutions (#1 & 4) have value \$7 and are not optimal!

Suppose that we relax the weight restriction (with "shadow price"

$\lambda = 0$ ) and impose the volume restriction:

---Stage 1---			
s	x:0	1	Maximum
0	0	-9999	0
1	0	-9999	0
2	0	2	2
3	0	2	2
4	0	2	2

---Stage 4---			
s	x:0	1	Maximum
0	0	-9999	0
1	3	-9999	3
2	3	4	4
3	5	7	7
4	6	7	7

---Stage 2---			
s	x:0	1	Maximum
0	0	-9999	0
1	0	3	3
2	2	3	3
3	2	5	5
4	2	5	5

---Stage 5---			
s	x:0	1	Maximum
0	0	-9999	0
1	3	-9999	3
2	4	4	4
3	7	7	7
4	7	8	8

---Stage 3---			
s	x:0	1	Maximum
0	0	-9999	0
1	3	-9999	3
2	3	-9999	3
3	5	3	5
4	5	6	6

---Stage 6---			
s	x:0	1	Maximum
0	0	-9999	0
1	3	-9999	3
2	4	-9999	4
3	7	5	7
4	8	8	8

\*\*\* Optimal value is 8 \*\*\*  
 \*\*\* There are 2 optimal solutions \*\*\*

Optimal Solution No. 1

stage	state	decision
6	4	Omit
5	4	Include
4	2	Include
3	0	Omit
2	0	Omit
1	0	Omit
0	0	

Optimal Solution No. 2

stage	state	decision
6	4	Include
5	2	Omit
4	2	Omit
3	0	Omit
2	0	Include
1	0	Omit
0	0	

Both of these solutions use exactly 4 units of volume, and are therefore feasible. In this case, the complementary slackness condition is again satisfied ( $0 \times 0 = 0$ ) and it therefore follows that both must be optimal in the original problem!

### Important!

In the example shown,

- at most one adjustment was required for the Lagrangian multiplier, and
- no duality gap was encountered,

whereas in general

- many such adjustments are required, and
- the optimal solution of the 2-dimensional problem might never be found!



# Surrogate Relaxation

## Reducing Dimensionality by Surrogate Relaxation

Choose nonnegative multipliers  $\mu_1$  and  $\mu_2$ , so that

$$\begin{cases} \sum_{j=1}^n a_{1j}x_j \leq b_1 \\ \sum_{j=1}^n a_{2j}x_j \leq b_2 \end{cases} \Rightarrow \begin{cases} \sum_{j=1}^n \mu_1 a_{1j}x_j \leq \mu_1 b_1 \\ \sum_{j=1}^n \mu_2 a_{2j}x_j \leq \mu_2 b_2 \end{cases} \Rightarrow \sum (\mu_1 a_{1j} + \mu_2 a_{2j})x_j \leq (\mu_1 b_1 + \mu_2 b_2)$$

In general, for an m-dimensional knapsack problem, the **surrogate relaxation** is

$$S(\mu) = \text{Maximum} \sum_{j=1}^n v_j x_j$$

subject to  $\sum_{i=1}^m \sum_{j=1}^n \mu_i a_{ij} x_j \leq \sum_{i=1}^m \mu_i b_i,$   
 $x_j \in X_j, j = 1, \dots, n$

- As was the case with Lagrangian relaxation, for any  $m \geq 0$  the optimal solution of the relaxation (a one-dimensional knapsack problem) may *not* be feasible in the two-dimensional problem, but the optimal value is an **upper bound** on the optimum of the two-dimensional problem.
- The **Surrogate Dual** problem is the problem of finding the surrogate multipliers  $\mu$  which will yield the **least upper bound**.
- Theory is available that guarantees that the surrogate **duality gap** is no larger than, and is often smaller than, the Lagrangian duality gap.