

Knapsack: branch-and-bound on the critical item

Outline

Why a branch-and-bound algorithm for the KP?

The algorithm for the KP

- Solving the LP_{KP}^k problem of a node

- Branching-variable selection strategy

- What changes with respect to the general algorithm

A worked example

Typical mistakes

In summary

Learning objectives

At the end of this chapter you will be able to:

- ▶ write the problem LP_{KP}^k of a node, and solve it with the formulas of Dantzig, without the simplex algorithm;
- ▶ branch on the variable of the split item, unselecting or selecting that item;
- ▶ execute the branch-and-bound algorithm for the KP by hand, with the FIFO or the LIFO rule;
- ▶ explain why the algorithm is simpler than the general one, and what stays the same.

Prerequisites

- ▶ From the module *Integer linear programming*, the chapter «[Branch-and-bound](#)»: branching and bounding operations, the problem list, the pruning criteria, the lower bound LB and the upper bound UB. The general algorithm is applied here to the 0/1 knapsack problem.
- ▶ From this module, the chapter «[Formulations of knapsack problems](#)»: the models ILP_{KP} and LP_{KP} , the efficiency of an item, the split item s , the residual capacity rc and the bound of Dantzig.

The problem of this chapter

The problem of this chapter is the
0/1 knapsack problem (KP):
its definition and its assumptions are on its page.

Why a branch-and-bound algorithm for the KP?



Introduction: Why a branch-and-bound algorithm for the KP?

- ▶ The general branch-and-bound algorithm solves an LP problem in every node of the tree
- ▶ For the KP these LP problems have a very special structure
- ▶ We start with a small challenge, and then we see what the general algorithm becomes for the KP

A challenge

A knapsack of capacity 9 and six items with the following profits and weights:

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$
p_j	6	5	8	9	6	7
w_j	2	3	6	7	5	9

- ▶ Find the optimal value of the LP relaxation *without* the simplex algorithm
- ▶ Hint: think of the efficiency $\frac{p_j}{w_j}$ of each item

From the general algorithm to the KP

- ▶ The general algorithm solves the LP problem of every node with the dual simplex algorithm
- ▶ For the KP this is not needed: the LP problem of every node is again a knapsack LP problem, solved by sorting the items by efficiency
- ▶ This chapter shows what the general algorithm becomes in this case



The challenge:

- ▶ the LP relaxation of a KP instance is solved by looking at the efficiencies $\frac{p_j}{w_j}$, without the simplex algorithm

Branch-and-bound for the KP:

- ▶ the LP problem of every node is again a knapsack LP problem
- ▶ it is solved by sorting the items by efficiency, not by the dual simplex algorithm

The algorithm for the KP

Introduction: The algorithm for the KP

- ▶ In this section we describe a node of the branch-and-bound tree for the ILP_{KP} model, and its LP problem LP_{KP}^k
- ▶ We see that LP_{KP}^k is again a knapsack LP problem, solved by the formulas of Dantzig
- ▶ Then we choose the branching variable, and we list what changes with respect to the general algorithm

Definition 1: branching node and LP_{KP}^k problem (1/2)

In the branch-and-bound algorithm for the ILP_{KP} model, the branching constraints on a variable x_j , $j \in \mathcal{J}$, are $x_j \leq 0$ and $x_j \geq 1$. In a branching node k of the branch-and-bound tree, we have:

1. *a subset of **unselected items** $\rightsquigarrow$ denoted $\mathcal{J}_{out}^k$*
2. *a subset of **selected items** $\rightsquigarrow$ denoted $\mathcal{J}_{in}^k$*
3. *a subset of **remaining items** $\rightsquigarrow$ denoted*
$$\mathcal{J}^k := \mathcal{J} \setminus \{ \mathcal{J}_{out}^k \cup \mathcal{J}_{in}^k \}$$

Definition: branching node and LP_{KP}^k problem (2/2)

The LP_{KP} problem associated to node k , called the LP_{KP}^k problem, reads as follows:

$$\begin{aligned} \max \quad & \sum_{j \in \mathcal{J}} p_j x_j \\ \text{subject to} \quad & \sum_{j \in \mathcal{J}} w_j x_j \leq c, \\ & x_j \geq 1, \quad \forall j \in \mathcal{J}_{\text{in}}^k, \\ & x_j \leq 0, \quad \forall j \in \mathcal{J}_{\text{out}}^k, \\ & x_j \leq 1, \quad \forall j \in \mathcal{J}, \\ & x_j \geq 0, \quad \forall j \in \mathcal{J}. \end{aligned}$$

Unselecting and selecting items

- ▶ The branching constraints $x_j \leq 0$ and $x_j \geq 1$ have the effect of **unselecting** or **selecting** an item.
- ▶ The selected items contribute a constant profit and use part of the capacity: we can move them out of the problem

The LP_{KP}^k problem rewritten

A LP_{KP}^k problem of a branching node k can be equivalently re-written as:

$$\underbrace{\sum_{j \in \mathcal{J}_{\text{in}}^k} p_j}_{=p(k)} + \max \sum_{j \in \mathcal{J}^k} p_j x_j$$

$$\text{subject to } \sum_{j \in \mathcal{J}^k} w_j x_j \leq c - \underbrace{\sum_{j \in \mathcal{J}_{\text{in}}^k} w_j}_{=c(k)},$$

$$x_j \leq 1, \quad \forall j \in \mathcal{J}^k,$$

$$x_j \geq 0, \quad \forall j \in \mathcal{J}^k.$$

A smaller knapsack LP problem

A LP_{KP}^k problem of a branching node k is **structurally equivalent** to the LP_{KP} problem but:

1. the set of items is $\mathcal{I}^k$ instead of $\mathcal{I}$
2. the capacity is reduced to $c(k) := c - \sum_{j \in \mathcal{I}_{\text{in}}^k} w_j$
3. there is an objective-function constant $p(k) := \sum_{j \in \mathcal{I}_{\text{in}}^k} p_j$

Solving the LP_{KP}^k problem of a node (1/3)

- ▶ The problem LP_{KP}^k is a knapsack LP problem on the items of $\mathcal{J}^k$, with capacity $c(k)$: it is solved by the formulas of Dantzig, without any simplex iteration
- ▶ The items of $\mathcal{J}^k$ are kept in the order of non-increasing efficiency
- ▶ If $c(k) < 0$ the problem is **infeasible**
- ▶ If $\sum_{j \in \mathcal{J}^k} w_j \leq c(k)$ all the items of $\mathcal{J}^k$ fit, and the optimal solution is $\tilde{x}_j = 1$ for all $j \in \mathcal{J}^k$: it is integer

Solving the LP_{KP}^k problem of a node (2/3)

- ▶ Otherwise the **split item** s of the node, also called the **critical item**, is the first item of $\mathcal{J}^k$ that does not fit entirely
- ▶ The **residual capacity** is the capacity left by the items that precede it:

$$\sum_{j \in \mathcal{J}^k, j < s} w_j \leq c(k) < \sum_{j \in \mathcal{J}^k, j \leq s} w_j, \quad rc = c(k) - \sum_{j \in \mathcal{J}^k, j < s} w_j.$$

Solving the LP_{KP}^k problem of a node (3/3)

- ▶ The optimal solution is:
 - ▶ $\tilde{x}_j = 1$ for the items of $\mathcal{J}^k$ before s
 - ▶ $\tilde{x}_s = \frac{rc}{w_s}$
 - ▶ $\tilde{x}_j = 0$ for the items after s

- ▶ The optimal value is

$$z(LP_{KP}^k) = p(k) + \sum_{j \in \mathcal{J}^k, j < s} p_j + p_s \frac{rc}{w_s}.$$

One scan of the items per node

In every node at most one variable is fractional, the one of the split item, and the value of the node is the Dantzig bound of a smaller knapsack:

the node is solved by a scan of the items.

Branching-variable selection strategy

- ▶ The optimal solution of an LP_{KP}^k problem given by the formulas above has at most one fractional variable, the one associated to the split item
- ▶ **Branching can be performed on the variable associated to the split item:** this branching-variable selection strategy is called split-item strategy
- ▶ The optimal solution of an LP_{KP} problem is not unique when some items have equal efficiency: the other optimal solutions can have more fractional variables, for instance $x = (\frac{1}{2}, \frac{1}{2})$ for $p = (2, 2)$, $w = (2, 2)$, $c = 2$
- ▶ Here, as in the general algorithm, we always use the solution given by the formulas

What changes with respect to the general algorithm (1/3)

- ▶ The algorithm is the LP-based branch-and-bound algorithm of the module *Integer linear programming* ([chapter «Branch-and-bound»](#)), with its problem list, its pruning criteria and its optimality gap
- ▶ What changes are the three operations: **bounding**, **branching** and **node selection**

What changes with respect to the general algorithm (2/3)

Bounding

- ▶ The problem of a node is solved by the formulas above
- ▶ All the profits are integer, so the bound of a node is $\lfloor z(\text{LP}_{\text{KP}}^k) \rfloor$
- ▶ The node is pruned:
 - ▶ by bound if $\lfloor z(\text{LP}_{\text{KP}}^k) \rfloor \leq \text{LB}$
 - ▶ by infeasibility if $c(k) < 0$
 - ▶ by integrality if its solution is integer (then LB is updated when the value is larger)
- ▶ An integer node is pruned by integrality even when its value does not exceed LB

What changes with respect to the general algorithm (3/3)

Branching

- ▶ The branching variable is the one of the split item s : the children are $x_s \leq 0$ (down, item unselected) and $x_s \geq 1$ (up, item selected)
- ▶ The node is not solved with a simplex tableau but described by the sets $\mathcal{J}_{\text{in}}^k$, $\mathcal{J}_{\text{out}}^k$ and $\mathcal{J}^k$

Node selection

- ▶ The node of the problem list is extracted with the FIFO or the LIFO rule
- ▶ The down-branching node is solved before the up-branching node; the nodes are numbered in the order of resolution
- ▶ The lower bound written in a node is the value at the moment of its resolution

Summary (1/2)

Nodes:

- ▶ a node k is described by the sets $\mathcal{I}_{\text{in}}^k$, $\mathcal{I}_{\text{out}}^k$ and $\mathcal{I}^k$
- ▶ its problem LP_{KP}^k is a knapsack LP problem on $\mathcal{I}^k$, with capacity $c(k)$ and constant $p(k)$

Solving a node:

- ▶ infeasible if $c(k) < 0$, integer if all the items of $\mathcal{I}^k$ fit
- ▶ otherwise the formulas of Dantzig with the split item s and the residual capacity rc

Summary (2/2)

Branching:

- ▶ on the variable of the split item: $x_s \leq 0$ (unselected) and $x_s \geq 1$ (selected)

The rest of the general algorithm:

- ▶ the bound of a node is $\lfloor z(LP_{KP}^k) \rfloor$
- ▶ problem list, pruning criteria and optimality gap stay the same
- ▶ FIFO or LIFO rule, down-branching node first

A worked example

Introduction: A worked example

- ▶ In this section we execute the branch-and-bound algorithm for the KP by hand, on the instance of the challenge
- ▶ In every node we write the sets of items, the capacity $c(k)$, the constant $p(k)$, the split item and the residual capacity
- ▶ We draw the branch-and-bound tree at the end of each iteration

Example: the ILP_{KP} model (1/3)

We consider a knapsack of capacity $c = 9$ and a set $\mathcal{I} = \{1, 2, 3, 4, 5, 6\}$ of 6 items ($n = 6$), with the following profits and weights:

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$
p_j	6	5	8	9	6	7
w_j	2	3	6	7	5	9

Example: the ILP_{KP} model (2/3)

The ILP_{KP} model for this KP instance is:

$$\begin{aligned} \max \quad & 6x_1 + 5x_2 + 8x_3 + 9x_4 + 6x_5 + 7x_6 \\ \text{subject to} \quad & 2x_1 + 3x_2 + 6x_3 + 7x_4 + 5x_5 + 9x_6 \leq 9 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \in \{0, 1\} \end{aligned}$$

Example: the ILP_{KP} model (3/3)

An optimal solution $\tilde{\mathbf{x}}(\text{ILP}_{\text{KP}})$ is:

$$\tilde{x}_1 = 1, \quad \tilde{x}_2 = 0, \quad \tilde{x}_3 = 0, \quad \tilde{x}_4 = 1, \quad \tilde{x}_5 = 0, \quad \tilde{x}_6 = 0$$

The optimal value $z(\text{ILP}_{\text{KP}})$ is 15. The associated optimal solution of the KP is $\tilde{S} = \{1, 4\}$.

The items are sorted by non-increasing efficiency, since:

$$\frac{6}{2} \geq \frac{5}{3} \geq \frac{8}{6} \geq \frac{9}{7} \geq \frac{6}{5} \geq \frac{7}{9}$$

Example: branch-and-bound for the ILP_{KP} model

A branch-and-bound algorithm for the ILP_{KP} model and the KP instance above using:

1. the split-item strategy as branching-variable selection strategy;
2. the FIFO rule as branching-node selection strategy;
3. solve first the down-branching node and then the up-branching node;
4. set the incumbent solution $\bar{\mathbf{x}}(\text{ILP}_{\text{KP}})$ to the following heuristic solution:

$$\bar{x}_1 = 1, \quad \bar{x}_2 = 0, \quad \bar{x}_3 = 0, \quad \bar{x}_4 = 1, \quad \bar{x}_5 = 0, \quad \bar{x}_6 = 0$$

and set the lower bound LB accordingly.

Example: initialization (1/2)

- ▶ For the LP_{KP}^1 problem, the split item is $s = 3$ and the residual capacity is $rc = 4$
- ▶ An optimal solution $\tilde{x}(LP_{KP}^1)$ is:

$$\tilde{x}_1 = 1, \quad \tilde{x}_2 = 1, \quad \tilde{x}_3 = \frac{2}{3}, \quad \tilde{x}_4 = 0, \quad \tilde{x}_5 = 0, \quad \tilde{x}_6 = 0$$

- ▶ The optimal value is:

$$z(LP_{KP}^1) = 6 \cdot 1 + 5 \cdot 1 + 8 \cdot \frac{2}{3} = \frac{49}{3}$$

Example: initialization (2/2)

- ▶ The optimal solution $\tilde{\mathbf{x}}(\text{LP}_{\text{KP}}^1)$ is **fractional** and the LP_{KP}^1 problem is **feasible**
- ▶ The LP_{KP}^1 problem is inserted in the problem list L and the lower bound LB is set to 15 ($\text{LB}(\tilde{\mathbf{x}}) = 15$)
- ▶ Due to the current lower bound LB and upper bound UB, we have:

$$15 \leq z(\text{ILP}_{\text{KP}}) \leq \underbrace{\left\lceil \frac{49}{3} \right\rceil}_{=16}$$

Example: iteration 1 (1/5)

- ▶ The LP_{KP}^1 problem is extracted and deleted from the problem list L
- ▶ The branching variable is x_3
- ▶ The LP_{KP}^2 and LP_{KP}^3 problems are created by unselecting ($x_3 \leq 0$) and selecting ($x_3 \geq 1$) item 3

Example: iteration 1 (2/5)

Solution of the LP_{KP}^2 problem

- ▶ We have

$$\mathcal{J}^2 = \{1, 2, 4, 5, 6\}, \quad \mathcal{J}_{\text{out}}^2 = \{3\}, \quad \mathcal{J}_{\text{in}}^2 = \emptyset$$

- ▶ The capacity is $c(2) = 9$ and the objective-function constant is $p(2) = 0$
- ▶ The split item is $s = 4$ and the residual capacity is $rc = 4$
- ▶ An optimal solution $\tilde{x}(LP_{KP}^2)$ is:

$$\tilde{x}_1 = 1, \quad \tilde{x}_2 = 1, \quad \tilde{x}_4 = \frac{4}{7}, \quad \tilde{x}_5 = 0, \quad \tilde{x}_6 = 0$$

and

$$z(LP_{KP}^2) = 6 \cdot 1 + 5 \cdot 1 + 9 \cdot \frac{4}{7} = \frac{113}{7}$$

Example: iteration 1 (3/5)

Solution of the LP_{KP}^2 problem

- ▶ The LP_{KP}^2 problem is **feasible** and we have:

$$\lfloor z(LP_{KP}^2) \rfloor > LB \quad \text{since} \quad \underbrace{\left\lfloor \frac{113}{7} \right\rfloor}_{=16} > 15$$

- ▶ The optimal solution $\tilde{x}(LP_{KP}^2)$ is **fractional**, accordingly, the LP_{KP}^2 problem is **inserted in the problem list L**

Example: iteration 1 (4/5)

Solution of the LP_{KP}^3 problem

- ▶ We have

$$\mathcal{J}^3 = \{1, 2, 4, 5, 6\}, \quad \mathcal{J}_{\text{out}}^3 = \emptyset, \quad \mathcal{J}_{\text{in}}^3 = \{3\}$$

- ▶ The capacity is $c(3) = 3$ and the objective-function constant is $p(3) = 8$
- ▶ The split item is $s = 2$ and the residual capacity is $rc = 1$
- ▶ An optimal solution $\tilde{x}(LP_{KP}^3)$ is:

$$\tilde{x}_1 = 1, \quad \tilde{x}_2 = \frac{1}{3}, \quad \tilde{x}_4 = 0, \quad \tilde{x}_5 = 0, \quad \tilde{x}_6 = 0$$

and

$$z(LP_{KP}^3) = 8 + 6 \cdot 1 + 5 \cdot \frac{1}{3} = \frac{47}{3}$$

Example: iteration 1 (5/5)

Solution of the LP_{KP}^3 problem

- ▶ The LP_{KP}^3 problem is **feasible** and we have:

$$\lfloor z(LP_{KP}^3) \rfloor \leq LB \quad \text{since} \quad \underbrace{\left\lfloor \frac{47}{3} \right\rfloor}_{=15} \leq 15$$

- ▶ Accordingly, the branching node is pruned by upper bound

Example: the tree after iteration 1

- ▶ At the end of the first iteration, Node 2 is inserted in the problem list L , while Node 3 is pruned by upper bound
- ▶ The corresponding branch-and-bound tree has three nodes: each node box contains the LP problem of the node, its optimal solution and the bounds
- ▶ The node boxes are large: we look at them one at a time, in the next three slides

Example: the tree after iteration 1 (1/3)

Node 1

$$\max 6x_1 + 5x_2 + 8x_3 + 9x_4 + 6x_5 + 7x_6$$

$$2x_1 + 3x_2 + 6x_3 + 7x_4 + 5x_5 + 9x_6 \leq 9$$

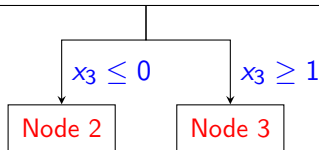
$$x_1, x_2, x_3, x_4, x_5, x_6 \leq 1$$

$$x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$$

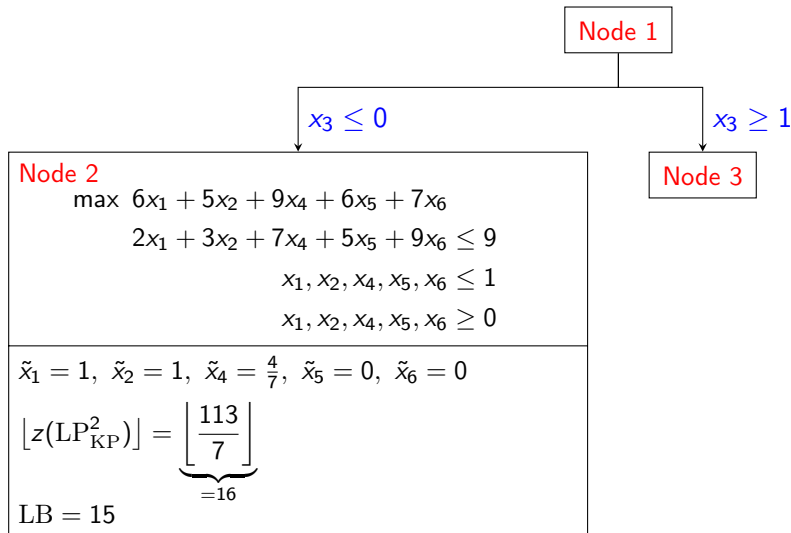
$$\tilde{x}_1 = 1, \tilde{x}_2 = 1, \tilde{x}_3 = \frac{2}{3}, \tilde{x}_4 = 0, \tilde{x}_5 = 0, \tilde{x}_6 = 0$$

$$\lfloor z(\text{LP}_{\text{KP}}^1) \rfloor = \underbrace{\left\lfloor \frac{49}{3} \right\rfloor}_{=16}$$

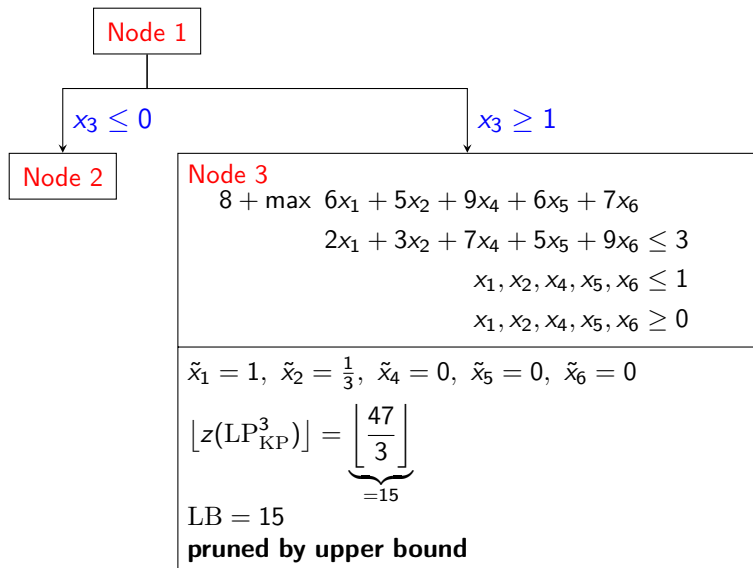
$$\text{LB} = 15$$



Example: the tree after iteration 1 (2/3)



Example: the tree after iteration 1 (3/3)



Example: bounds after iteration 1

- ▶ Node 2 is the only node of the problem list L
- ▶ Due to the current lower bound LB and upper bound UB, we have:

$$15 \leq z(\text{ILP}_{\text{KP}}) \leq \underbrace{\left\lfloor \frac{113}{7} \right\rfloor}_{=16}$$

Example: iteration 2 (1/5)

- ▶ The LP_{KP}^2 problem is extracted and deleted from the problem list L
- ▶ The branching variable is x_4
- ▶ The LP_{KP}^4 and LP_{KP}^5 problems are created by unselecting ($x_4 \leq 0$) and selecting ($x_4 \geq 1$) item 4

Example: iteration 2 (2/5)

Solution of the LP_{KP}^4 problem

- ▶ We have

$$\mathcal{I}^4 = \{1, 2, 5, 6\}, \quad \mathcal{I}_{\text{out}}^4 = \{3, 4\}, \quad \mathcal{I}_{\text{in}}^4 = \emptyset$$

- ▶ The capacity is $c(4) = 9$ and the objective-function constant is $p(4) = 0$
- ▶ The split item is $s = 5$ and the residual capacity is $rc = 4$
- ▶ An optimal solution $\tilde{x}(LP_{KP}^4)$ is:

$$\tilde{x}_1 = 1, \quad \tilde{x}_2 = 1, \quad \tilde{x}_5 = \frac{4}{5}, \quad \tilde{x}_6 = 0$$

and

$$z(LP_{KP}^4) = 6 \cdot 1 + 5 \cdot 1 + 6 \cdot \frac{4}{5} = \frac{79}{5}$$

Example: iteration 2 (3/5)

Solution of the LP_{KP}^4 problem

- ▶ The LP_{KP}^4 problem is **feasible** and we have:

$$\lfloor z(LP_{KP}^4) \rfloor \leq LB \quad \text{since} \quad \underbrace{\left\lfloor \frac{79}{5} \right\rfloor}_{=15} \leq 15$$

- ▶ Accordingly, the branching node is pruned by upper bound

Example: iteration 2 (4/5)

Solution of the LP_{KP}^5 problem

- ▶ We have

$$\mathcal{J}^5 = \{1, 2, 5, 6\}, \quad \mathcal{J}_{\text{out}}^5 = \{3\}, \quad \mathcal{J}_{\text{in}}^5 = \{4\}$$

- ▶ The capacity is $c(5) = 2$ and the objective-function constant is $p(5) = 9$
- ▶ The split item is $s = 2$ and the residual capacity is $rc = 0$
- ▶ An optimal solution $\tilde{\mathbf{x}}(LP_{KP}^5)$ is:

$$\tilde{x}_1 = 1, \quad \tilde{x}_2 = 0, \quad \tilde{x}_5 = 0, \quad \tilde{x}_6 = 0$$

and

$$z(LP_{KP}^5) = 9 + 6 \cdot 1 = 15$$

Example: iteration 2 (5/5)

Solution of the LP_{KP}^5 problem

- ▶ The LP_{KP}^5 problem is **feasible** and its optimal solution $\tilde{x}(LP_{KP}^5)$ is **integer**
- ▶ Its value does not improve the incumbent solution, since $z(LP_{KP}^5) = 15 \leq 15 = LB$, so the lower bound LB stays 15
- ▶ Accordingly, the branching node is pruned by integrality: when a node could be pruned both by integrality and by bound, it is pruned by integrality

Example: the tree after iteration 2

- ▶ At the end of the second iteration, Node 4 is pruned by upper bound and Node 5 is pruned by integrality
- ▶ The corresponding branch-and-bound tree is the subtree of Node 2: as before, we look at its three node boxes one at a time

Example: the tree after iteration 2 (1/3)

Node 2

$$\max 6x_1 + 5x_2 + 9x_4 + 6x_5 + 7x_6$$

$$2x_1 + 3x_2 + 7x_4 + 5x_5 + 9x_6 \leq 9$$

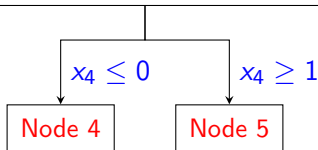
$$x_1, x_2, x_4, x_5, x_6 \leq 1$$

$$x_1, x_2, x_4, x_5, x_6 \geq 0$$

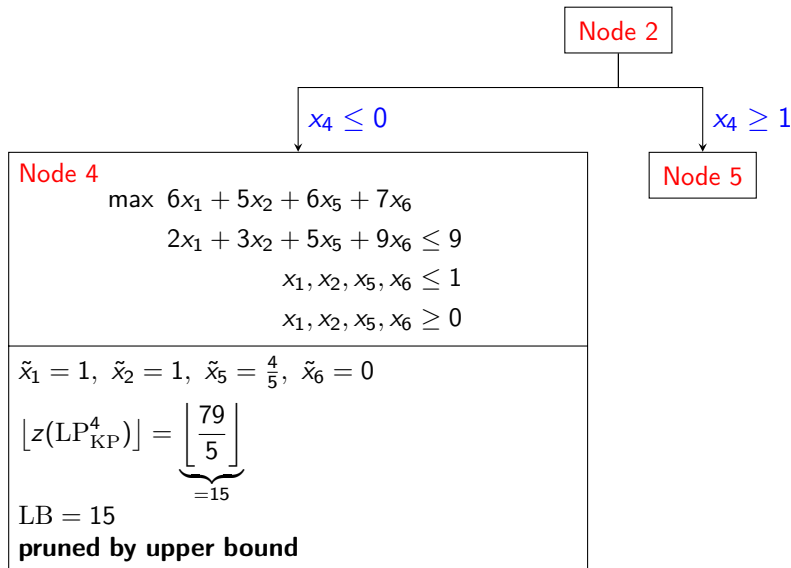
$$\tilde{x}_1 = 1, \tilde{x}_2 = 1, \tilde{x}_4 = \frac{4}{7}, \tilde{x}_5 = 0, \tilde{x}_6 = 0$$

$$\lfloor z(\text{LP}_{\text{KP}}^2) \rfloor = \underbrace{\left\lfloor \frac{113}{7} \right\rfloor}_{=16}$$

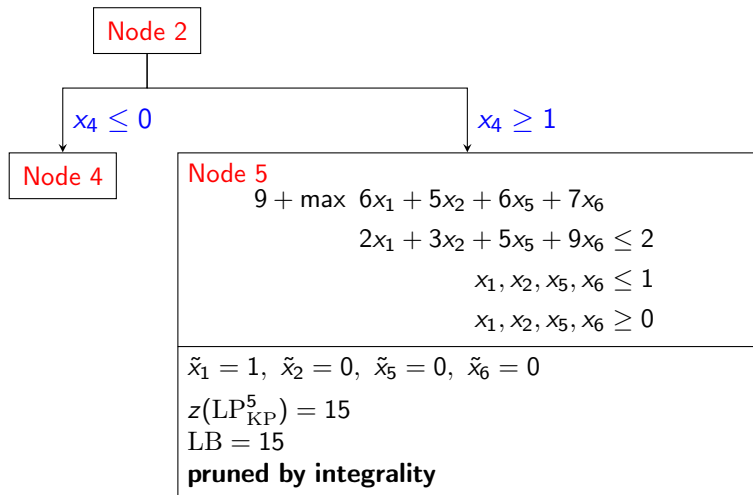
$$\text{LB} = 15$$



Example: the tree after iteration 2 (2/3)



Example: the tree after iteration 2 (3/3)



Example: termination

- ▶ Since the problem list L is empty, the ILP_{KP} problem is solved to optimality
- ▶ The optimal value is $z(\text{ILP}_{\text{KP}}) = 15$ (the value of the lower bound LB)
- ▶ An optimal solution $\tilde{\mathbf{x}}(\text{ILP}_{\text{KP}})$ is

$$\tilde{x}_1 = 1, \tilde{x}_2 = 0, \tilde{x}_3 = 0, \tilde{x}_4 = 1, \tilde{x}_5 = 0, \tilde{x}_6 = 0$$

(the incumbent solution $\bar{\mathbf{x}}(\text{ILP}_{\text{KP}})$)

Example: the branch-and-bound tree

This is the branch-and-bound tree of the previous execution of the branch-and-bound algorithm, showing for each branching node:

the optimal value of the associated LP_{KP} problem,
the lower bound LB ,
and the pruning criteria.

Node 1

$$\lfloor z(\text{LP}_{\text{KP}}^1) \rfloor = \underbrace{\left\lfloor \frac{49}{3} \right\rfloor}_{=16}$$

$$\text{LB} = 15$$

Node 2

$$x_3 \leq 0$$

$$\lfloor z(\text{LP}_{\text{KP}}^2) \rfloor = \underbrace{\left\lfloor \frac{113}{7} \right\rfloor}_{=16}$$

$$\text{LB} = 15$$

$$x_3 \geq 1$$

Node 3

$$\lfloor z(\text{LP}_{\text{KP}}^3) \rfloor = \underbrace{\left\lfloor \frac{47}{3} \right\rfloor}_{=15}$$

$$\text{LB} = 15$$

pruned by upper bound

Node 4

$$x_4 \leq 0$$

$$\lfloor z(\text{LP}_{\text{KP}}^4) \rfloor = \underbrace{\left\lfloor \frac{79}{5} \right\rfloor}_{=15}$$

$$\text{LB} = 15$$

pruned by upper bound

$$x_4 \geq 1$$

Node 5

$$z(\text{LP}_{\text{KP}}^5) = 15$$

$$\text{LB} = 15$$

pruned by integrality

Summary (1/2)

The instance:

- ▶ $c = 9$, six items already sorted by non-increasing efficiency
- ▶ the heuristic solution $\{1, 4\}$ gives $LB = 15$; the root gives $\lfloor \frac{49}{3} \rfloor = 16$

Iteration 1:

- ▶ branching on the split item 3
- ▶ Node 2 ($x_3 \leq 0$) goes in the list, Node 3 ($x_3 \geq 1$) is pruned by upper bound

Summary (2/2)

Iteration 2:

- ▶ branching on the split item 4 of Node 2
- ▶ Node 4 ($x_4 \leq 0$) is pruned by upper bound, Node 5 ($x_4 \geq 1$) by integrality

Termination:

- ▶ the list is empty after 5 nodes
- ▶ the incumbent solution $\{1, 4\}$ is optimal, $z(\text{ILP}_{\text{KP}}) = 15$

Typical mistakes

Introduction: Typical mistakes

- ▶ We collect the mistakes that are made most often when the branch-and-bound algorithm for the KP is executed by hand
- ▶ They concern the LP problem of a node, its value, the lower bound and the numbering of the nodes
- ▶ Check them before solving the exercises

Observation 1: Typical mistakes (1/2)

- ▶ **Forgetting the upper bounds $x_j \leq 1$.** Without them the LP problem of a node would put all the capacity on the item of best efficiency: the node boxes must contain $x_j \leq 1$.
- ▶ **Computing the split item on the original instance.** In a node, the split item and the residual capacity are computed on the items of $\mathcal{I}^k$ with the capacity $c(k)$, after fixing the items of $\mathcal{I}_{\text{in}}^k$ and $\mathcal{I}_{\text{out}}^k$.
- ▶ **Forgetting the constant $p(k)$.** The value of the node is $p(k)$ plus the value of the items of $\mathcal{I}^k$: the profit of the selected items counts.

Observation: Typical mistakes (2/2)

- ▶ *Using the old lower bound.* The lower bound is updated as soon as a node has an integer solution of larger value; the nodes of the list that become dominated are discarded.
- ▶ *Numbering the skipped children.* A child that is not solved does not receive a node index, exactly as in the general algorithm.

Summary

The LP problem of a node:

- ▶ it contains the upper bounds $x_j \leq 1$
- ▶ split item and residual capacity are computed on $\mathcal{I}^k$ with capacity $c(k)$
- ▶ its value includes the constant $p(k)$

The algorithm:

- ▶ update LB as soon as a better integer solution is found
- ▶ a child that is not solved does not receive a node index

In summary

Introduction: In summary

- ▶ We collect the key ideas of the chapter on the branch-and-bound algorithm for the KP
- ▶ The dashboard gathers the bounds found on the example of the chapter

Summary (1/2)

- ▶ The branch-and-bound algorithm for the KP is the LP-based algorithm with two simplifications: the problem of every node is solved by the formulas of Dantzig, and the branching variable is the one of the split item.
- ▶ A node is described by the sets $\mathcal{I}_{\text{in}}^k$, $\mathcal{I}_{\text{out}}^k$ and $\mathcal{I}^k$, by the capacity $c(k)$ and by the constant $p(k)$.
- ▶ The bound of a node is $\lfloor z(\text{LP}_{\text{KP}}^k) \rfloor$; when the problem list is empty the lower and the upper bound coincide and the incumbent solution is optimal.

Summary (2/2)

Dashboard of the KP instance of the example ($c = 9$, $p = (6, 5, 8, 9, 6, 7)$, $w = (2, 3, 6, 7, 5, 9)$). The bounds on $z(\text{ILP}_{\text{KP}})$ found so far:

Source	Lower bound	Upper bound	From
Dantzig bound	–	16	$\lfloor \frac{49}{3} \rfloor$
This chapter, initialization	15	16	heuristic solution $\{1, 4\}$
This chapter, iteration 1	15	16	node 2 in the list, $\lfloor \frac{113}{7} \rfloor$
This chapter, termination	15	15	empty list, 5 nodes