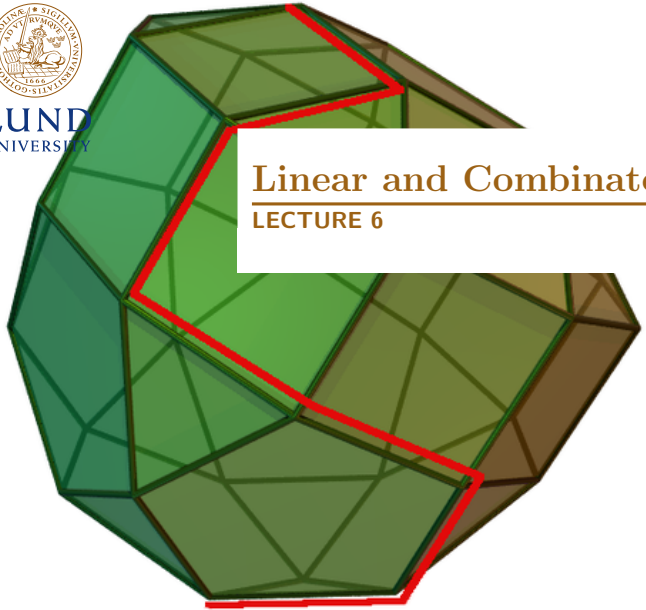
LUND UNIVERSITY



Linear and Combinatorial Optimization 2020

LECTURE 6

Overview

LUND UNIVERSITY

The branch and bound method

- In today's lecture, we present another method for solving ILP problems, called the branch and bound method.
- We will solve an example and present the method at the same time.
- The computations needed are done with Matlab, and you can find the matlab script `linearbranchandbound.m` that were used on the course homepage.

LUND UNIVERSITY

The branch and bound method for ILP problems

Example (p. 281)

- *The ILP problem that we will solve is*

$$\begin{aligned} &\textit{maximize} && z = 7x_1 + 3x_2, \\ &\textit{subject to} && \begin{cases} 2x_1 + 5x_2 \leq 30, \\ 8x_1 + 3x_2 \leq 48, \\ x_1, x_2 \geq 0 \textit{ integers.} \end{cases} \end{aligned}$$

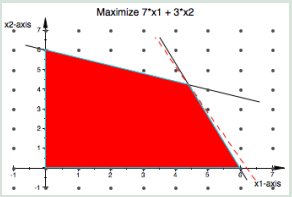
- *Relax the problem by removing the integer constraints* x_1, $x_2 \in \mathbb{Z}$.
- *Introduce slack variables* x_3, x_4 *and put the problem into canonical form:*

LUND UNIVERSITY

The branch and bound method for ILP problems

Example

$$
\begin{aligned}
&\textit{maximize} && z = 7x_1 + 3x_2, \\
&\textit{subject to} && \begin{cases} 2x_1 + 5x_2 + x_3 = 30, \\ 8x_1 + 3x_2 + x_4 = 48, , \\ x_1, \ldots, x_4 \geq 0. \end{cases}
\end{aligned}
$$

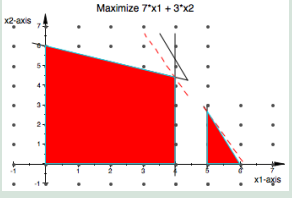


The first LP solution

The branch and bound method for ILP problems

Example (Cont.)

- *Solve using your file* `checkbasic1.m`. *We get the noninteger solution* $x_1 \approx 4.41$, $x_2 \approx 4.23$ *and* $z \approx 43.23$.
- *Now the feasible set is split into two parts,* $x_1 \leq 4$ *or* $x_1 \geq 5$. x_1 *is called a* branch variable. *The actual optimum of ILP must be in one of these subsets.*

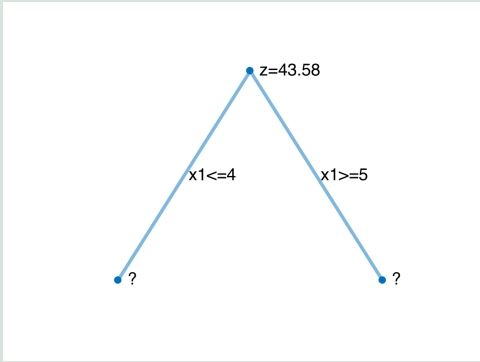


The optimum for ILP must be in one of the two subsets.

The branch and bound method for ILP problems

Example (Cont.)

- *We get a tree structure with the first LP problem at the stem of the tree.*



The stem and the first two branches of the tree, one for each of the two subdomains.

The branch and bound method for ILP problems

Example (Cont.)

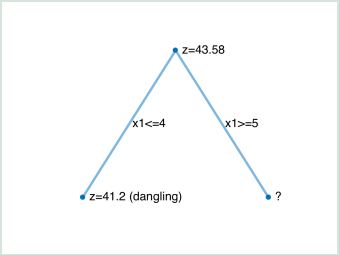
- *We will solve the two new LP problems with the simplex method (using* `checkbasic1.m`*).*
- *We start with the left branch, and add the constraint* $x_1 \leq 4$ *to the previous constraints, and start with the basic solution which was the optimum in the previous step.*
- *Note that this basic solution is not feasible now that the optimum is outside of the feasible set.*
- *We avoid the two phase method by using the dual as we did for the cutting plane method.*

LUND UNIVERSITY

The branch and bound method for ILP problems

Example (Cont.)

- *Using* `checkbasic1`, *we quickly find an optimum with optimal value* 41.2.
- *We leave this node for now (it is called a dangling node), and investigate the other branch corresponding to* $x_1 \geq 5$.



We leave the dangling node for now and investigate the right branch.

The branch and bound method for ILP problems

Example (Cont.)

- *We replace the constraint* $x_1 \leq 4$ *with* $x_1 \geq 5$*, and obtain a new (infeasible) basic solution.*
- *Again, we solve the problem using the dual (with the script* `dualproblem.m`*.*
- *After a few iterations, we reach the optimal solution with* $z = 43$*, which is noninteger since e.g.* $x_2 \approx 2.6667$*.*
- *Since* $43 > 41.2$*, the right branch seems more promising than the left one, and we choose to continue with this branch.*

LUND UNIVERSITY

The branch and bound method for ILP problems

Example (Cont.)

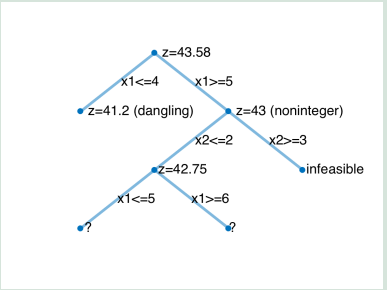
- *We use x_2 as a branching variable, and split the problem into two cases: $x_2 \leq 2$ or $x_2 \geq 3$.*
- *We start with the left branch, and solve using the dual problem. We get a noninteger optimum solution with $z = 42.75$ and $x_1 = 5.25$. This node is left as a dangling node.*
- *Next, we investigate the right branch, and so we replace the constraing $x_2 \leq 2$ with $x_2 \geq 3$. This dual problem is unbounded, and so the primal problem is infeasible. There are no solutions in this branch.*

LUND UNIVERSITY

The branch and bound method for ILP problems

Example (Cont.)

- *Next, we go back to the dangling node corresponding to* $x2 \leq 2$. *The optimal solution had* $x_1 = 5.25$, *and so we get two branches corresponding to* $x_1 \leq 5$ *and* $x_1 \geq 6$, *respectively.*



Two more branches to investigate

The branch and bound method for ILP problems

Example (Cont.)

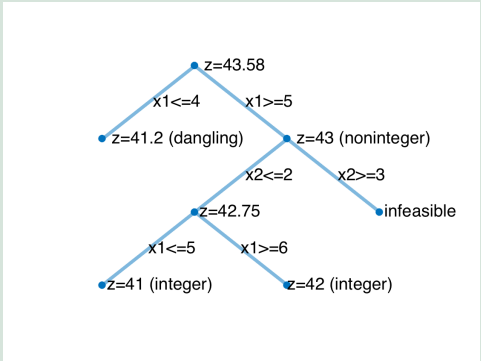
- *Solving with the simplex method for the dual problem of the left branch, we get an integer optimal solution with* $z = 41$. *We still need to investigate the other branches.*
- *Doing the same for the right branch, we find an integer optimal solution with* $z = 42$.
- *Compare the two integer optima that we have found and notice that* $z = 42$ *from the right branch has the highest value.*
- *There is another dangling node which we have to compare, but note that the LP optimum for that node was* 41.2, *and so the optimum for ILP in that branch cannot be higher than* 41. *Therefore, we don't need to investigate that branch any further.*

LUND UNIVERSITY

The branch and bound method for ILP problems

Example (Cont.)

The final tree is shown in the figure below.



The final tree structure

Branch and bound algorithms in general

- In the branch and bound method that we just saw, we obtained a tree structure.
- We didn't need to explore all the branches of the tree.
- The simplex method gave an upper bound for the value of the ILP in each subset (each branch of the tree).
- When an integer optimum solution was found, we found an optimum value for the ILP problem for that branch.
- The simplex method was only needed for finding an upper bound in the subtrees (branches). If we have another way of getting such bounds, then the same idea can be used for other types of (discrete) optimization problems as well.
- Hence also nonlinear problems can be solved with branch and bound methods.

LUND UNIVERSITY

General branch and bound algorithms

What do we need?

- *A procedure* branch, *which splits each subset* D^k *into smaller parts* $D^{k,1}, \ldots, D^{k,n}$.
- *A procedure* bound, *which for each subset* D^k,
  - always *finds an upper bound* $f_U^k \geq \max_{\mathbf{x} \in D^k} f(\mathbf{x})$,
  - sometimes *finds a lower bound* $f_L^k \leq \max_{\mathbf{x} \in D^k} f(\mathbf{x})$,
  - sometimes *finds the optimum* $f_L^k = \max_{x \in D^k} f(\mathbf{x})$

Note: The above is for maximization problems. For a minimization we need to modify the scheme slightly (e.g. by replacing max by min and reversing some inequalities).

LUND UNIVERSITY

General branch and bound algorithms

Algorithm idea:

1. Use the procedure *bound* to find upper and lower bounds for the whole problem.
2. Save the node D together with its upper bound in a list $L = \{(D, f_U)\}$. Keep a note of f_L if a lower bound was obtained. Otherwise, let $f_L = -\infty$.

LUND UNIVERSITY

General branch and bound algorithms

3. Take the most promising node (for example the one with the largest f_U and *branch* the corresponding subset into smaller subsets. Bound the maximum for each subset. If any of the new lower bounds are greater than f_L, then replace f_L with (the largest of the) new f_L.
   - If a node has $f_U^k \leq f_L$, then the optimum cannot be found in that subdomain. This branch can be cut off.
   - If we find the optimum in a subdomain, then this domain need not be split any further. The branch ends here.
4. The nodes where an optimal solution is not found $f_U^k > f_L$ need to be examined further. Put them in a list and repeat from 2.

LUND UNIVERSITY

General branch and bound algorithms

- Note that in the ILP example, the simplex method was used to find the upper bounds.
- Lower bounds were only found in the case where there was an integer optimal solution of the subproblem.

LUND UNIVERSITY

The travelling salesman problem

Example (The travelling salesman problem)

- *A salesman has to visit n cities $C_1, \ldots, C_n$, starting from his home city C_1 and then visiting the other $n-1$ cities in some order and finally returning to his home city C_1.*
- *Let c_{ij} be the distance between city i and j. In which order should he visit the cities in order to minimize the total distance?*
- *There are many different algorithms for this problem, but it is a known hard problem to solve if the number of n is large.*

LUND UNIVERSITY

The travelling salesman problem (Cont.)

Example (Travelling salesman problem, Cont.)

- *The problem can be formulated as an ILP problem using the variables*

$$x_{ij} = \begin{cases} 1 & \text{if the salesman travels directly from} \\ & \quad \text{city } i \text{ to city } j \text{ the route,} \\ 0 & \text{otherwise.} \end{cases}$$

  as well as artificial variables u_i, $i = 2, \ldots, n$.
- *The artificial variables* u_i *will prevent subcycles from forming, as we will soon see.*

LUND UNIVERSITY

The travelling salesman problem (Cont.)

Example (Travelling salesman problem, Cont.)

The LP problem is

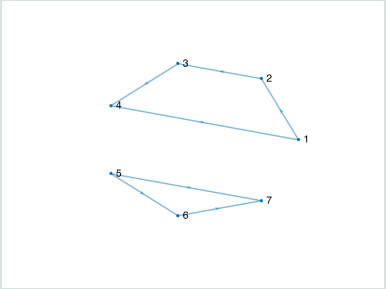
$$
\begin{aligned}
&\textit{minimize} && z = \textstyle\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}, \\
&\textit{subject to} && \begin{cases}
\sum_{i=1}^{n} x_{ij} = 1 & \text{for all } j, \\
\sum_{j=1}^{n} x_{ij} = 1 & \text{for all } i, \\
u_i - u_j + n x_{ij} \leq n - 1, & i, j = 2, \ldots, n \text{ and } i \neq j, \\
u_i \in \mathbb{Z}, & \text{for all } i, \\
u_i \geq 0, & \text{for all } i.
\end{cases}
\end{aligned}
$$

LUND UNIVERSITY

The travelling salesman problem (Cont.)

Example (Travelling salesman, Cont.)

- *The objective function is just the total length of the route.*
- *The first two constraints ensures that the salesman arrives at and leaves each city exactly once.*
- *The third constraint prevents subcycles from forming like in the figure below.*



The travelling salesman problem (Cont.)

Example (Travelling salesman problem, Cont.)

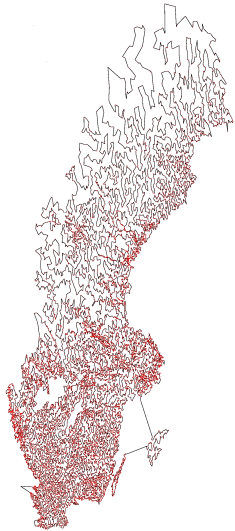
- *The problem can be solved as an ILP with the cutting plane method or branch and bound as last time* or *directly with a branch and bound algorithm without the simplex method as you will do in handin 2.*
- *It is hard to solve this in a reasonable time when the number of cities is large, and it will be interesting to see how big problems you can solve with your algorithm.*

LUND UNIVERSITY

The travelling salesman problem (Cont.)

Example (Travelling salesman problem, Cont.)

- *In 2004, the travelling salesman problem for Sweden's towns was solved. The problem contains 24,978 towns. At the time, it was the largest TSP that had been solved. Read more about the project.*
- *The current record is an 85,900-city tour that arose in a chip-design application.*



TSP for Sweden's 24,978 cities

The travelling salesman problem (Cont.)

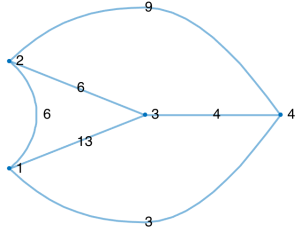
Example (Travelling salesman as in handin 2)

- *In handin exercise 2, you will work on a branch and bound algorithm for the TSP problem.*
- *Note that TSP is a minimization problem, so the algorithm has to be constructed with this in mind.*
- *I will show you how the algorithm should work in a small example based on the data given on the next slide.*

LUND UNIVERSITY

Travelling salesman as in handin 2

Example (Cont.)



TSP example

The table contains min *and* max *distances from each city (except 1) to some other city.*

	Min	*Max*
2	6	9
3	4	13
4	3	9

Travelling salesman as in handin 2

Example (Cont.)

- *Start in city 1. There are three possibilities, depending on which city we choose as number 2. We get a tree structure.*
- *How to calculate upper (u.b.) and lower bounds (l.b.): If the salesman travels from city* 1 *to city* 2*, we get the following quick upper and lower bounds for the route length, based on the table on the previous page:*

$$6 + \begin{cases} 6 + 4 + 3 = 19, \\ 9 + 13 + 9 = 37. \end{cases}$$

LUND UNIVERSITY

TSP as in handin 2

Example (Cont.)

- *Likewise, if he travels to city* 2 *or city* 3 *from city* 1*, then we get the bounds*

$$13 + \begin{cases} 6+4+3 = 26, \\ 9+13+9 = 44, \end{cases} \quad \textit{and}$$

$$3 + \begin{cases} 6+4+3 = 16, \\ 9+13+9 = 34, \end{cases} \quad \textit{respectively.}$$

LUND UNIVERSITY

Travelling salesman (Cont.)

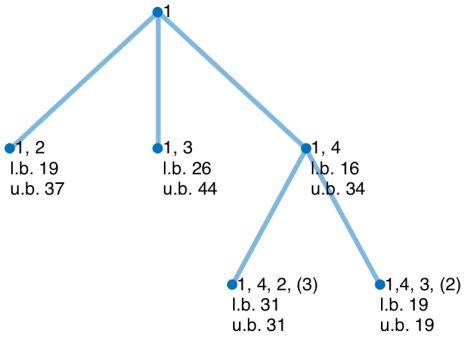
Example (Cont.)

- *From the lower bounds, we find that the node* $1, 4$ *looks most promising, and we need the other two nodes dangling.*
- *We branch the node* $1, 4$ *into the two nodes* $1, 4, 2$ *and* $1, 4, 3$*, for which the tour is known (since the fourth city has to be the* 3 *and* 2*, respectively). We get optimal values* 31 *and* 19*, respectively.*
- *See a figure of the tree structure on the next slide.*

LUND UNIVERSITY

Travelling salesman (Cont.)

Example (Cont.)



Travelling salesman (Cont.)

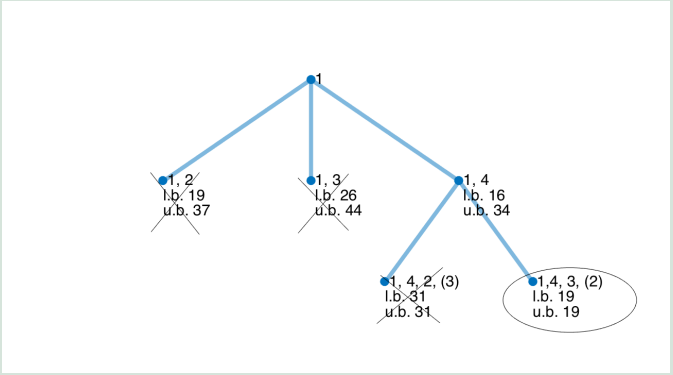
Example (Cont.)

- *If we are content with finding one solution, we can cut the node* $1, 2$ *since the lower bound* $19 \geq 19$ *(the upper bound/optimum in a side branch). If we are looking for all solutions, we need to investigate this node further.*
- *We cut the note* $1, 3$ *and* $1, 4, 2$ *since* $26 > 19$ *and* $31 > 19$*, respectively.*

LUND UNIVERSITY

Example (Cont.)

The final tree is shown below:



The final tree structure for the TSP problem