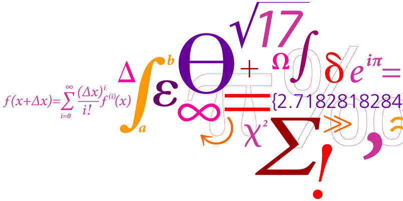
# Branch and Bound II

Jesper Larsen, David Pisinger[1]

[1]Department of Management Engineering
Technical University of Denmark



DTU Management Engineering
Department of Management Engineering

## Learning objectives

After today's lecture you should:

- Understand that branch and bound is more general than the LP-based branch and bound.
- Know different schemes for seaching through the branch and bound tree.
- Know different examples of general branch and bound.

## An iteration of Branch and Bound

**4 Steps of a B&B iteration**

For the current node, do the following:

1. Generate bound
2. Can we prune?
3. If not, branch
4. Select next node, repeat

## Bounding

How do we get ourselves a bounding function? Relaxation.

- **Leave out some constraints.** Keep the same objective function $f$. So now we maximize $f$ over a larger solution space $P$.
- **Change the objective function** $f$ to $g$, where:
  $g(x) \geq f(x)$ for all $x \in S$ (maximization problem)
  $g(x) \leq f(x)$ for all $x \in S$ (minimization problem)
  Optimality of $g$ does not automaticly guarantee optimality of $f$.
- Use the two above at the same time.

## Pruning

The incumbent is the best feasible solution found so far — the best primal bound in the current tree.

For the current node we generate a **dual bound**

and maybe a **primal bound** (MIP heuristics).

| Pruning by | **minimize** | **maximize** |
|---|---|---|
| optimality: | dual bound $=$ primal bound | dual bound $=$ primal bound |
| bound: | dual bound $\geq$ incumbent value | dual bound $\leq$ incumbent value |
| infeasibility: | set dual bound $:= \infty$ | set dual bound $:= -\infty$ |

## Strategy for branching

In many situations there is not only one variable/candidate for branching. The question is which to choose.

- Branch on most **fractional** variable.
- Branch on least **fractional** variable.
- "Estimate the cost of forcing $x_j$ to become integer."

These are all "heuristic" strategies, there is no guaranteed best strategy.

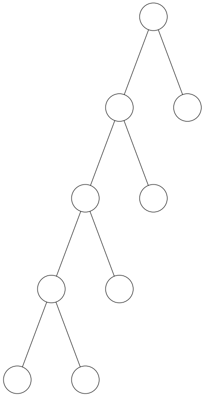
# Build-in strategies for branching in CPLEX

## Settings for Branching Variable Choice

| | |
|---|---|
| MININFEAS | The minimum infeasibility rule chooses the variable with the value closest to an integer but still fractional. The minimum infeasibility rule may lead more quickly to a first integer feasible solution, but is usually slower overall to reach the optimal integer solution. |
| MAXINFEAS | The maximum infeasibility rule chooses the variable with the value furtherest from an integer. The maximum infeasibility rule forces larger changes earlier in the tree. |
| DEFAULT | ILOG CPLEX decides each branch direction. |
| PSEUDO | Use pseudo costs, which derives an estimate about the effect of each proposed branch from duality information. |
| STRONG | Strong branching causes variable selection based on partially solving a number of subproblems with tentative branches to see which branch is the most promising. This strategy can be effective on large, difficult MIP problems |
| PSEUDOREDUCED | Use pseudo reduced costs, which is a computationally less-intensive form of pseudo costs. |

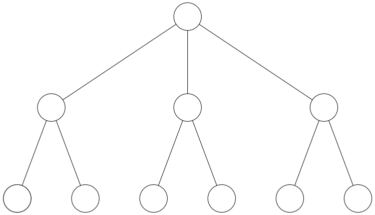
## Strategy for selecting the next subproblem

- **DFS**: Depth First Search strategy: select among the active nodes one of those that are the deepest down in the tree.



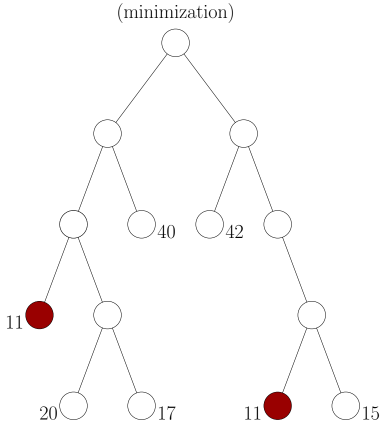
## Strategy for selecting the next subproblem

- **BFS**: Breadth First Search strategy: select among the active nodes one of those that are highest up in the tree.



## Strategy for selecting the next subproblem

- **BeFS**: Best First Search strategy: select among the active nodes one of those with the best dual bound. (sometime also called global best node selection).



## Build-in strategies in CPLEX

### Settings for Node Search Type

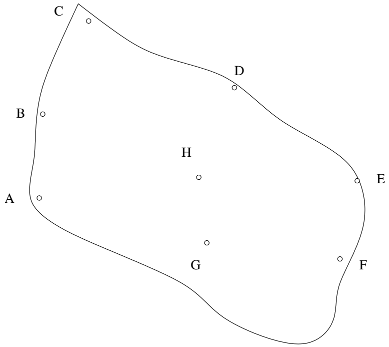
| | |
|---|---|
| DFS | **Depth First search** will be conducted. In many cases this amounts to a brute force strategy for solving the combinatorial problem, gaining a small amount of tactical efficiency due to a variety of reasons, and it is rare that it offers any advantage over other settings. |
| BESTBOUND | **Best Bound search**, which means that the node with the best objective function will be selected, generally near the top of the tree. |
| BESTEST | **Best Estimate search**, whereby ILOG CPLEX will use an estimate of a given node's progress toward integer feasibility relative to its degradation of the objective function. This setting can be useful in cases where there is difficulty in finding feasible solutions or in cases where a proof of optimality is not crucial. |
| BESTEST_ALT | **Alternative Best Estimate search.** |

## The Symmetric TSP

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in E} d_{ij}x_{ij} & \\
\text{s.t.} \quad & \sum_{j} x_{ij} = 2 & i \in \{1,2,\ldots,n\} \\
& \sum_{i,j\in S} x_{ij} \le |S| - 1 & \emptyset \subset S \subset V \\
& x_{ij} \in \{0,1\} & (i,j) \in E
\end{aligned}$$

(Note: $x_{ij} = x_{ji}$, so we only use variables $i < j$)

# TSP of Bornholm



|   | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| A | 0 | 11 | 24 | 25 | 30 | 29 | 15 | 15 |
| B | 11 | 0 | 13 | 20 | 32 | 37 | 17 | 17 |
| C | 24 | 13 | 0 | 16 | 30 | 39 | 29 | 22 |
| D | 25 | 20 | 16 | 0 | 15 | 23 | 18 | 12 |
| E | 30 | 32 | 30 | 15 | 0 | 9 | 23 | 15 |
| F | 29 | 37 | 39 | 23 | 9 | 0 | 14 | 21 |
| G | 15 | 17 | 29 | 18 | 23 | 14 | 0 | 7 |
| H | 15 | 17 | 22 | 12 | 15 | 21 | 7 | 0 |

# Bounds

- One way to identify a bound for the TSP is by relaxing constraints, e.g. to allow subtours. This bound is, however, rather weak.

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in E} d_{ij}x_{ij} & \\
\text{s.t.} \quad & \sum_{j} x_{ij} = 2 & i \in \{1,2,\ldots,n\} \\
& \xcancel{\sum_{i,j\in S} x_{ij} \leq |S|-1} & \xcancel{\varnothing \subset S \subset V} \\
& x_{ij} \in \{0,1\} & (i,j) \in E
\end{aligned}
$$

- An alternative is the **1-tree relaxation**.

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in E} d_{ij}x_{ij} & \\
\text{s.t.} \quad & \xcancel{\sum_{j} x_{ij} = 2} & \xcancel{i \in \{1,2,\ldots,n\}} \\
& \sum_{i,j\in S} x_{ij} \leq |S|-1 & \varnothing \subset S \subset V \\
& x_{ij} \in \{0,1\} & (i,j) \in E
\end{aligned}
$$

where we relax the constraint to $\sum_j x_{ij} \geq 1$

## What is a Minimum Spanning Tree?

Let $G = (V, E)$ be a graph, then

- A **tree** $H$ is a connected subgraph with no cycles (circuits) of $G$.
- Spanning: **all vertices** are in the $H$

### The Minimum Spanning Tree problem

Furthermore given costs $w_e$ for each arc $e$ the **minimum (weight) spanning tree** problem consists of finding a spanning tree of minimum cost.

## Kruskals Algorithm
– for finding a Minimum Spanning Tree in a graph

```
$F \leftarrow \emptyset$ ;
sort edges by nondecreasing cost (denote the list $\mathcal{L}$);
for each $(u, v) \in \mathcal{L}$ do
    if a path from $u$ to $v$ in $F$ does not exist then
        $F \leftarrow F \cup (u, v)$;
    end
end
```

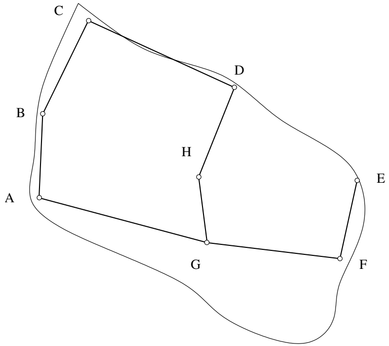
**Note**

The algorithm can terminate as soon as $n - 1$ edges have been included in $F$ (where $n$ is the number of nodes).

## The 1-tree bound
– for finding a lower bound for the STSP

1. Identify a special vertex **1** (this can be any vertex of the graph)
2. Remove **1** and all incident edges from $G$
3. For the remaining graph determine the minimum spanning tree $T$
4. Add two lightest edges $e_1$ and $e_2$ incident with **1** to $T$ producing $T_1$ (called a **1-tree**)

## 1-tree bound of Bornholm



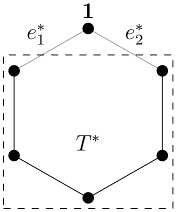
| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| A | 0 | 11 | 24 | 25 | 30 | 29 | 15 | 15 |
| B | 11 | 0 | 13 | 20 | 32 | 37 | 17 | 17 |
| C | 24 | 13 | 0 | 16 | 30 | 39 | 29 | 22 |
| D | 25 | 20 | 16 | 0 | 15 | 23 | 18 | 12 |
| E | 30 | 32 | 30 | 15 | 0 | 9 | 23 | 15 |
| F | 29 | 37 | 39 | 23 | 9 | 0 | 14 | 21 |
| G | 15 | 17 | 29 | 18 | 23 | 14 | 0 | 7 |
| H | 15 | 17 | 22 | 12 | 15 | 21 | 7 | 0 |

$$\text{Cost} = (7 + 9 + 12 + 13 + 14 + 16) + (11 + 15) = 97$$

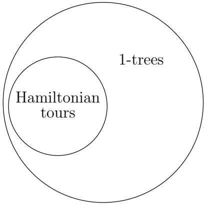
## Why is $T_1$ a bound?

We need to convince ourselves that the total cost of a **minimum cost 1-tree** $T_1$ is a lower bound of the value of an optimal tour.

- Note that the optimal Hamiltonian tour can be divided into two edges $e_1^*$ and $e_2^*$ that are incident with **1** and the rest of the tour (let us call it $T^*$).



- So the set of Hamiltonian tours is a subset of 1-trees of $G$.



## Why is $T_1$ a bound?

We need to convince ourselves that the total cost of a **minimum cost 1-tree** $T_1$ is a lower bound of the value of an optimal tour.

- $T_1$ is a minimum cost 1-tree.
- $T^*$ is the optimal Hamiltonian tour.
- Every Hamiltonian tour is a 1-tree.
- Since $T^*$ is also a 1-tree:

$$d(T_1) \leq d(T^*)$$

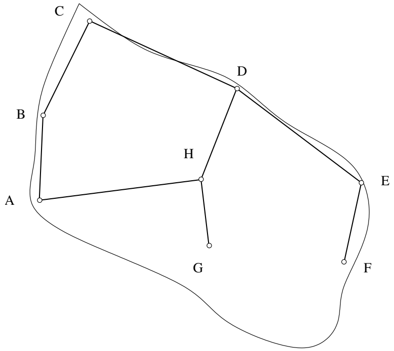
## Strengthening the bound

- Idea: Vertices of $T_1$ with high degree are incident with too many attractive edges. Vertices of degree 1 have on the other hand too many unattractive edges.
- Define $\pi_i$ as the degree of vertex $i$ minus 2.
- Note that $\sum_{i \in V} \pi_i$ equals 0 since $T_1$ has $n$ edges and therefore the degree sum is $2n$.
- For each edge $(i, j) \in E$ we transform the cost to $d'_{ij} = d_{ij} + \pi_i + \pi_j$.
- For a Hamilton cycle the cost is unchanged

$$\sum_{(i,j) \in H} d'_{ij} = \sum_{(i,j) \in H} d_{ij} + \sum_{i \in V} \pi_i + \sum_{j \in V} \pi_j = \sum_{(i,j) \in H} d_{ij}$$

## Strengthening the bound

Add 1 to node $G$, subract 1 from node $E$.



Modified distance matrix:

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| A | 0 | 11 | 24 | 25 | 29 | 29 | 16 | 15 |
| B | 11 | 0 | 13 | 20 | 31 | 37 | 18 | 17 |
| C | 24 | 13 | 0 | 16 | 29 | 39 | 30 | 22 |
| D | 25 | 20 | 16 | 0 | 14 | 23 | 19 | 12 |
| E | 29 | 31 | 29 | 14 | 0 | 8 | 23 | 14 |
| F | 29 | 37 | 39 | 23 | 8 | 0 | 15 | 21 |
| G | 16 | 18 | 30 | 19 | 23 | 15 | 0 | 8 |
| H | 15 | 17 | 22 | 12 | 14 | 21 | 8 | 0 |

$$\text{New cost} = (8+8+12+13+14+16)+(11+15) = 97$$
$$\text{Next iteration cost} = 98$$

## B&B for the Symmetric TSP

### 4 Steps of a B&B iteration

For the current node, do the following:

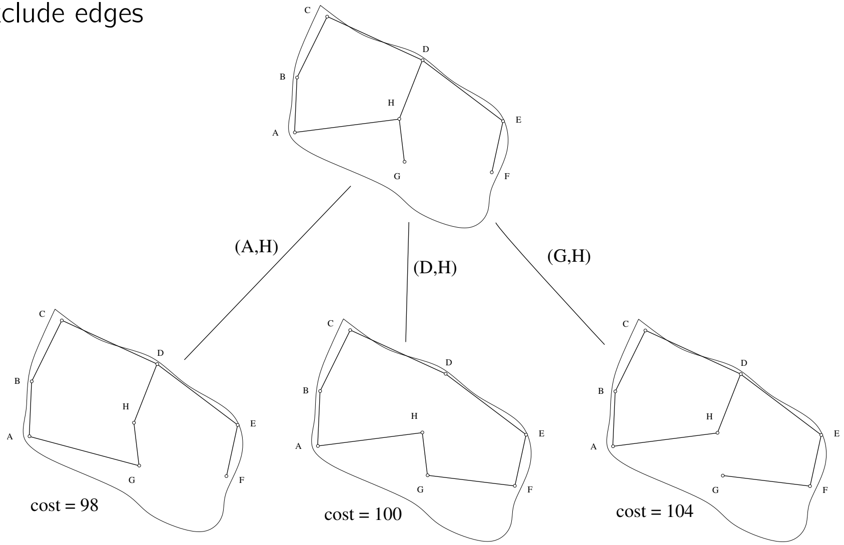
1. Generate bound **The cost of $T_1$ is less that or equal to the cost of any Hamiltonian tour.**
2. Can we prune? **In the case $T_1$ is a Hamilton tour we have found the optimal solution and can prune by bounding.**
3. **If not, branch**
4. Select next node, repeat

## How do we branch?

- Observe that in the case our 1-tree is **not** a Hamilton tour, at least one vertex has degree 3 or more.
- So choose a vertex $v$ with degree 3 or more.
- For each edge $(u, v)$ generate a subproblem where $(u, v)$ is excluded from the set of edges.

# Branching on Bornholm

Exclude edges



## Asymmetric TSP

The graph is now **directed**

$$\begin{pmatrix} - & 3 & 93 & 15 & 46 & 13 \\ 6 & - & 79 & 46 & 36 & 22 \\ 32 & 4 & - & 25 & 16 & 19 \\ 53 & 104 & 92 & - & 83 & 25 \\ 6 & 24 & 66 & 13 & - & 7 \\ 3 & 88 & 18 & 48 & 105 & - \end{pmatrix}$$

Find shortest tour visiting all nodes (Hamilton cycle)

## B&B for the ATSP

### Mathematical formulation of the Assymetric TSP

$$
\begin{array}{llcll}
\min & \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} & & & \\
\text{st.} & \sum_{j=1, i \neq j}^{n} x_{ij} & = & 1 & i \in \{1, \ldots, n\} \\
 & \sum_{i=1, i \neq j}^{n} x_{ij} & = & 1 & j \in \{1, \ldots, n\} \\
 & \sum_{i,j \in S} x_{ij} & \leq & |S| - 1 & \emptyset \subset S \subset V \\
 & x_{ij} \in \{0, 1\} & & & i, j \in \{1, \ldots, n\}
\end{array}
$$

## B&B for the ATSP

Generating lower bounds for the ATSP:

$$
\begin{array}{llll}
\min & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} & & \\
\text{st.} & \displaystyle\sum_{j=1, i\neq j}^{n} x_{ij} = 1 & & i \in \{1,\dots,n\} \\
& \displaystyle\sum_{i=1, i\neq j}^{n} x_{ij} = 1 & & j \in \{1,\dots,n\} \\
& \xcancel{\displaystyle\sum_{i,j\in S} x_{ij} \leq |S|-1 \qquad \emptyset \subset S \subset V} & & \\
& x_{ij} \in \{0,1\} & & i,j \in \{1,\dots,n\}
\end{array}
$$

The **Assignment Problem** (Matching Problem) can be solved efficiently using **Hungarian Algorithm**

## ATSP – Example

$$\begin{pmatrix} - & 3 & 93 & 15 & 46 & 13 \\ 6 & - & 79 & 46 & 36 & 22 \\ 32 & 4 & - & 25 & 16 & 19 \\ 53 & 104 & 92 & - & 83 & 25 \\ 6 & 24 & 66 & 13 & - & 7 \\ 3 & 88 & 18 & 48 & 105 & - \end{pmatrix}$$

**Hungarian Algorithm** makes use of the fact that optimal solution ($x_{ij}$) does not change if we subtract a constant from the objective function

## ATSP – Matrix after row and column reductions

$$\begin{array}{cc}
\begin{pmatrix}
- & 0 & 75 & 2 & 30 & 6 \\
0 & - & 58 & 30 & 17 & 12 \\
29 & 1 & - & 12 & 0 & 12 \\
32 & 83 & 58 & - & 49 & 0 \\
3 & 21 & 48 & 0 & - & 0 \\
0 & 85 & 0 & 35 & 89 & -
\end{pmatrix} & \begin{array}{c} 0 \\ 3 \\ 0 \\ 18 \\ 0 \\ 0 \end{array} \\
\begin{array}{cccccc} 3 & 3 & 18 & 13 & 16 & 7 \end{array} &
\end{array}$$

Lower bound: $(3 + 3 + 18 + 13 + 16 + 7) + (3 + 18) = 81.$

# ATSP – Matrix after row and column reductions

$$\begin{array}{cc}
\left(\begin{array}{cccccc}
- & \mathbf{0} & 75 & 2 & 30 & 6 \\
\mathbf{0} & - & 58 & 30 & 17 & 12 \\
29 & 1 & - & 12 & \mathbf{0} & 12 \\
32 & 83 & 58 & - & 49 & \mathbf{0} \\
3 & 21 & 48 & \mathbf{0} & - & 0 \\
0 & 85 & \mathbf{0} & 35 & 89 & -
\end{array}\right) &
\begin{array}{c} 0 \\ 3 \\ 0 \\ 18 \\ 0 \\ 0 \end{array} \\
\begin{array}{cccccc} 3 & 3 & 18 & 13 & 16 & 7 \end{array} &
\end{array}$$

Find 0-assignment (pick $n$ 0-values so each row/column is covered)

If 0-assignments form a Hamilton cycle, we can use it as upper bound.
In our case we have two syb-cycles (1,2,1) and (6,3,5,4,6)

## ATSP - B&B Components I

- **Bounding function:** Perform column and row reductions as in the Hungarian method for the assignment problem:
  - subtract the smallest element in each row from all row elements
  - subtract the smallest element in each column of the resulting matrix from all column elements
- The sum of the subtracted elements gives a lower bound for the length of the optimal ATSP tour.
- Whether you do first column and then row reductions or first row and then column reductions does not matter. It may result in different matrices but both ways produce lower bounds.

## Estimating the cost of change – branching

- We estimate how much it would cost to eliminate one of the "0-arcs" from the lower bound we have:

$$\begin{aligned}
(1,2) &= 2+1 &&= 3\\
(2,1) &= 12+0 &&= 12\\
(3,5) &= 1+17 &&= 18\\
(4,6) &= 32+0 &&= 32\\
(5,4) &= 2+0 &&= 2\\
(5,6) &= 0+0 &&= 0\\
(6,1) &= 0+0 &&= 0\\
(6,3) &= 0+48 &&= 48
\end{aligned}$$

$$\begin{pmatrix}
- & 0 & 75 & 2 & 30 & 6\\
0 & - & 58 & 30 & 17 & 12\\
29 & 1 & - & 12 & 0 & 12\\
32 & 83 & 58 & - & 49 & 0\\
3 & 21 & 48 & 0 & - & 0\\
0 & 85 & 0 & 35 & 89 & -
\end{pmatrix}$$

## ATSP - B&B Components II

- **Branching:** Find that 0-position $(i, j)$ in the reduced matrix, for which the sum of the next-smallest elements in row $i$ and column $j$ is largest ("the most expensive edge to leave out in a new solution"). Construct two subproblems:
  - one in which $(i, j)$ is forced into the solution (remove row $i$ and column $j$ from the matrix and set $d_{ji}$ to $\infty$),
  - and one in which $(i, j)$ is excluded from the solution (set $d_{ij}$ equal to $\infty$).

## ATSP – Branching on $(6, 3)$

"1-branch"

$$\begin{pmatrix} - & 0 & 2 & 30 & 6 \\ 0 & - & 30 & 17 & 12 \\ 29 & 1 & 12 & 0 & 12 \\ 32 & 83 & - & 49 & 0 \\ 3 & 21 & 0 & - & 0 \end{pmatrix}$$

"0-branch"

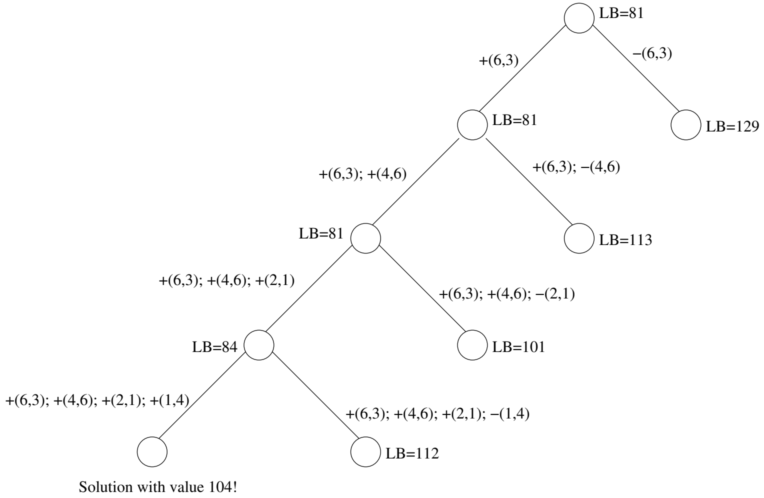
$$\begin{pmatrix} - & 0 & 75 & 2 & 30 & 6 \\ 0 & - & 58 & 30 & 17 & 12 \\ 29 & 1 & - & 12 & 0 & 12 \\ 32 & 83 & 58 & - & 49 & 0 \\ 3 & 21 & 48 & 0 & - & 0 \\ 0 & 85 & - & 35 & 89 & - \end{pmatrix}$$

## ATSP - B&B Components III

- **Search strategy:** Depth first: Examine the subproblem with the included edge first.

  Because $(i, j)$ is most expensive to leave out, there is a fair chance that the other subproblems is pruned in a later iteration.

# ATSP – Enumeration tree with solution



## Using B&B as a heuristic

- From the root-node of the B&B-tree we have a (global) upper bound $u$.
- When we compute an initial solution at root node or bound by optimality we have a global lower bound $l$.
- This gives us the opportunity to estimate the worst-case deviation from the optimum.

## Optimization-based heuristics

- Stop the B&B when the estimate on the deviation is below a certain threshold.
- Don't investigate all branches in the B&B-tree (especially applied within Binary Integer Programming)
- **Beam Search:** Usefull technique if memory is limited. Set an upper limit $\alpha$ on active nodes that are stored. If this limit is reached only keep the $\alpha$ best.

## Keys to success

- Have a good bounding function. If several are available the stronger the better even if it is computationally harder to solve.
- Get a good incumbent early (never underestimate the value of a good initial solution).
- Experiment, experiment, experiment.