



Branch and Bound – *Examples and Extensions*

Jesper Larsen

jesla@man.dtu.dk

Department of Management Engineering
Technical University of Denmark

Jesper Larsen

1



Some definitions

- A branching is a division of the original feasible set S into subsets $S_1, \dots, S_K$.
- A branching is **valid** if
 - ▶ $\bigcup_{i=1}^K S_i = S$
 - ▶ It is possible to determine and optimize over each set S_i
- A branching is called **partitive** if the sets S_i are disjoint.

Jesper Larsen

2



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 . If all original constraints are satisfied we have an **incumbent**.
- Change the objective function f to g . Here $g(x) \geq f(x)$ for all $x \in S$. Optimality of g does not automatically guarantee optimality of f .
- Use the two above at the same time.

Jesper Larsen

3



An iteration of Branch and Bound

One iteration of Branch and Bound consists of the following four steps:

1. Selection of the node to be processed.
2. Bound calculation.
3. Pruning?
4. Branching.

Jesper Larsen

4





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.

Jesper Larsen

5



Mostly mentioned strategies

- **DFS**: Depth First Search strategy: select among the active nodes one of those with the largest level.
- **BFS**: Breadth First Search strategy: select among the active nodes one of those with the lowest level.
- **BeFS**: Best First Search strategy: select among the active nodes one of those with the lowest bound. (sometime also called *global best node selection*).

Jesper Larsen

7



Strategy for selecting the next subproblem

- In choosing a search strategy we might consider two different goals:
 - ▶ Minimizing overall solution time.
 - ▶ Finding a good feasible solution to our original problem.

Jesper Larsen

6



The Best First Search strategy

- One way to minimize overall solution time is to try to minimize the size of the search tree.
- A subproblem (B&B node) is **critical** if the bound exceeds the value of the optimal solution.
- A critical node must always be examined.
- Only nodes with values larger than or equal to the optimum will be searched.
- No superfluous calculations take place in Best First Search after the optimum has been found.

Jesper Larsen

8





Branch and Bound for ATSP

- Input:** A distance matrix D for a *digraph* $G = (V, E)$ with n vertices (so d_{ij} may be different from d_{ji}). If the edge (i, j) is in E then d_{ij} equals the distance from i to j , otherwise d_{ij} equals ∞ .
- Output:** The length $d(C)$ of a shortest directed circuit C in G visiting each vertex exactly once, and an n -vector of edges $((v_1, v_2), \dots, (v_{n-1}, v_n))$.
- Method:** B&B: Bounding function, search and branching strategy is described below.

Jesper Larsen

9



Mathematical formulation of the ATSP

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

Jesper Larsen

10



ATSP – Example

$$\begin{pmatrix}
 - & 3 & 93 & 13 & 33 & 9 \\
 4 & - & 77 & 42 & 21 & 16 \\
 45 & 17 & - & 36 & 16 & 28 \\
 39 & 90 & 80 & - & 56 & 7 \\
 28 & 46 & 88 & 33 & - & 25 \\
 3 & 88 & 18 & 46 & 92 & -
 \end{pmatrix}$$

Jesper Larsen

11



ATSP – Lower bound calculation

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

Jesper Larsen

12





ATSP - B&B Components I

- Bounding function:** Perform row and column reductions as for the assignment problem – subtract the smallest element in each row from all row elements and then the smallest element in each column of the resulting matrix from all column elements. The sum of the subtracted elements constitutes a lower bound for the length of the optimal ATSP tour.

Jesper Larsen

13



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 ∞),
 - and one in which (i, j) is excluded from the solution (set d_{ij} equal to ∞).

Jesper Larsen

14



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.

Jesper Larsen

15



ATSP – Branching on $(6, 3)$

“1-branch”

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

“0-branch”

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

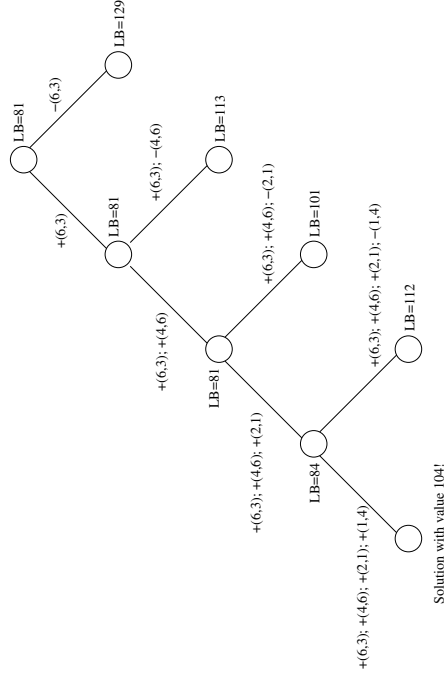
Jesper Larsen

16





ATSP – Enumeration tree with solution



Bounds

- One way to identify a bound for the TSP is by relaxing constraints. This could be to allow subtours. This bound is although known to be rather weak.
- An alternative is the **1-tree relaxation**.



The Symmetric TSP

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



The 1-tree bound

- Identify a special vertex **1** (this can be any vertex of the graph).
- 1** and all edges incident with **1** are removed from G .
- For the remaining graph determine the minimum spanning tree T .
- Now the two smallest edges e_1 and e_2 incident with **1** are added to T producing T_1 (called a **1-tree**)



Why is T_1 a bound?

We need to convince ourselves that the total cost of T_1 is a lower bound of the value of an optimal tour.

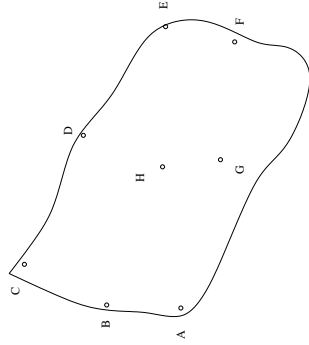
- Note that a Hamiltonian tour can be divided into two edges e'_1 and e'_2 that are incident with 1 and the rest of the tour (let us call it T').
- So the set of Hamiltonian tours is a subset of 1-trees of G .
- Since e_1, e_2 are the two smallest edges incident to 1 $d_{e_1} + d_{e_2} \leq d_{e'_1} + d_{e'_2}$. Furthermore as T' is a tree $d(T) \leq d(T')$.



- So the cost of T_1 is less than or equal to the cost of any Hamiltonian tour.
- In the case T_1 is a tour we have found the optimal solution and can prune by bounding.
- otherwise we need to bound.



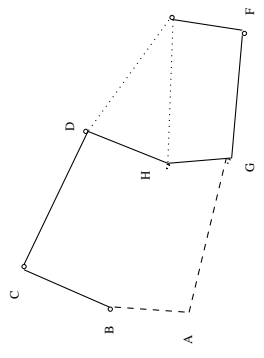
TSP of Bornholm



	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



1-tree bound of Bornholm



Tree in rest of G

Edge left out by Kruskal's MST algorithm

1-tree edge

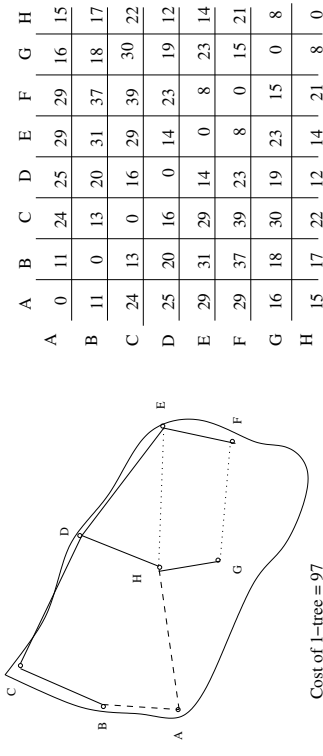
Cost of 1-tree = 97

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 π_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 $d'_{ij} = d_{ij} + \pi_i + \pi_j$.

Strengthen the bound

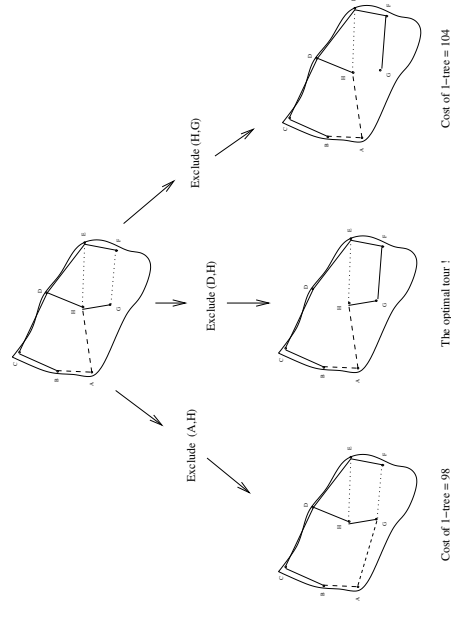
Modified distance matrix:



How do we branch?

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

Branching on Bornholm





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.

Jesper Larsen

29



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:** Useful technique if memory is limited. Set an upper limit α on active nodes that are stored. If this limit is reached only keep the α best.

Jesper Larsen

30



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.

Jesper Larsen

31

