Travelling Salesman Problem

An implementation of a branch and bound algorithm to solve the Travelling Salesman Problem (TSP).

Course: Communication Networks and Ambient Intelligence
Miniproject 1: Graph Theory
October 2012

Group: 12gr721
Students:
• Egon Kidmose
• Mads Holdgaard Vejlø
• Niels Fristrup Andersen
• Stefan Almind Jensen
The problem

- A list of cities to be visited
- Pairwise distances between the cities
- Must return to the home/starting city

How to find the shortest route, to visit all cities?
The problem, example

Example: Visit a list of cities in Denmark

Map: http://geology.com/world/denmark-map.gif
Interpretation of the problem

- Cities could be thought of as → Vertices
- Distances could be described as → Edges (weighted)

→ Then we have reformulated the problem into something, on which we could apply Graph Theory.

→ Then we look at the example map again...
The problem, example

Map: http://geology.com/world/denmark-map.gif
Solution

- The problem is categorized as a NP-hard problem.
- Several solution approaches exist, fx:
  - Excessive search / brute force
    - LOTS of resources necessary
- To optimize this approach, a branch and bound (B&B) algorithm could be applied...
Solution: Branch and bound (B&B)

- Systematic analysis of possible solutions
- Candidate solutions are progressively rated by its cost
- The cost of the first possible solution is stored
- As the algorithm analyses other candidate solutions, those that will cost more than the found solution, will be discarded (known as pruning).
- This could lower the 'search-space' an thereby also lower the computation time.
Analysis of the solutions

- To keep track of which solutions to discard and which to select → the solutions are represented as a binary tree.

![Binary Tree Diagram]
Analysis of the solutions

- Each node in the tree contains a lower bound cost → leaf nodes holds a cost of a final (candidate) solution.
Pruning

- If a non-leaf node in the tree costs more than an earlier found solution, then this node (and its children!) is pruned.
The work: Algorithm iteration 1

- We started the work, by drawing an initial flowchart of the algorithm
The work: Algorithm iteration 2

- After another review...
The work: Algorithm iteration 3

- And again...
The work: Algorithm iteration 3

- ...and the 3rd iteration interpreted as a state machine
The work: Algorithm iteration 4

- Discovered that the state machine could be simplified. State 3, 4 and 5 could be made into simply one state.
The work: Algorithm iteration 4

- Rearrange the flowchart with the new state discover...
The implementation

- After the design process, the algorithm was implemented in C.
- Git was used as version control on the code.
• The algorithm was tested with the following graph.
  - 5 vertices
  - 10 edges

Graph: [Data Structures and Algorithms, by Alfred V. Aho et. al. p.333]
Results, without pruning

- A full binary tree (without the root) is expected to include the sum of nodes from level 1, level 2, ..., level 10. That is:

\[ \sum_{i=1}^{10} 2^i = 2046 \]

where \( i \) denotes the number of edges.

- By inspection of the log, the algorithm was tested and verified to expand such a full tree (i.e. without pruning).
Results, with pruning

Pruning is based on visiting this node (Pruned at level 2), so all nodes below this one, is discarded.

Unvisited nodes in branch of binary tree: \( \sum_{i=1}^{Q} 2^i \)

where \( Q = N - P \).

\( N \) is the number of edges in the graph.

\( P \) is the pruning level

\( Q = 10 - 2 = 8 \) \( \Rightarrow \)

\[ \sum_{i=1}^{8} 2^i = 510 \]
Results, with pruning

- NB! Not all paths in the full binary tree makes sense!! These are marked as 'dead' (dead-points).
- By inspecting the log, the total amount of pruning-points resulted in 9, and dead-points was 21.
- By calculating the sum of subsequent nodes of the pruning-points (shown in previous slide) and the dead-points, the amount of unvisited treenodes sums up to 1982.
- This results in a 'visitratio' of $64/2046 \Rightarrow \approx 3\%$
- No. of visited leafs: 3 (12 without pruning)