Advanced Algorithms and Data Structures
Branch and Bound

Jens Raaby, Mads Hartmann & Ulrik Rasmussen

May 30, 2012

Question 1

Given a graph G a 1-tree bound is defined as follows

- You select a vertex v_1 from G
- Remove all edges incident on v_1 from G producing G'
- Find the minimum spanning tree T of G'
- Add the two smallest edges e_1 and e_2 incident on v_1 to T producing a 1-tree T_1

We're asked to show that the 1-tree is a lower bound for TSP; that is, that the weight of the Hamiltonian tour of G with the lowest weight is equal to or greater than weight of T_1.

If we assume we have a Hamiltonian tour of G that has has a lower weight than the T_1 then it would require that there exists a spanning tree of G' that has a lower wight than T; this is impossible since T is a minimum spanning tree. Here's why: A Hamiltonian tour of G will also need to be a Hamiltonian path of G'. In the best case the Hamiltonian path H and the minimum spanning tree T will be the same. Since we're adding the two edges e_1, e_2 with the lowest weight then the only way for TSP to have a lower weight than T_1 would be if the weight of H was smaller than T; a contradiction.

Question 2

In this part we consider representing each monument as a circle, so that when all circles are maximum size and do not overlap we have a way of travelling to each monument by entering and leaving every circle.

To form a linear program to maximise the radii of these circles, it is a simple matter of representing each monument with a radius variable and adding constraints such that any two circles to not overlap. In the following r_i denotes the radius of the circle placed on vertex i, and $d(u, v)$ is the distance between vertices u and v:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{v \in V} r_v & \\
\text{subject to} \quad & r_u + r_v \leq d(u,v) & \forall (u,v) \in E, \\
& r_v \geq 0 & \forall v \in V.
\end{aligned}
$$

Question 3

How do you compute a lower bound from these circles?

Considering any single monument, its distance from any other monument is at least the radius of that monument's circle. Since visiting all the monuments requires both entering and leaving each of the circles, a lower bound for a TSP problem could be expressed as $2\sum_{v\in V} r_v$, in other words, double the sum of all the radii. This will be a fairly good lower bound if all the monuments are similar distances apart. However, if many of the monuments are located in small areas, then the radii will also be quite small. There may therefore be a much larger distance between these groups of monuments, meaning the actual TSP solution has a much higher cost. We can mitigate this problem using the concept of *moats* as described in [1], where clusters of nodes are chosen from which a common moat can be grown until it hits a circle or another moat. We then define the *width* w of a moat as the distance between its perimeter and the perimeter of the circles that it encloses. Since a valid TSP solution must enter and leave any cluster of nodes, we can add $2w$ to the lower bound.

Assuming we have chosen some clustering of vertices, expressed as a family of disjoint sets $C_i \in \mathcal{C}$ where $\bigcap_i C_i = \emptyset$ and $\bigcup_i C_i = V$, a linear program maximizing circles with moats can be formulated as follows.

$$
\begin{aligned}
\text{maximize} \quad & \sum_{v\in V} r_v + \sum_{C_i\in\mathcal{C}} w_i, \\
\text{subject to} \quad & r_u + r_v \le d(u,v) && \forall (u,v)\in E, u\in C_i, v\in C_i, \\
& r_u + r_v + w_i + w_j \le d(u,v) && \forall (u,v)\in E, u\in C_i, v\in C_j, i\neq j, \\
& r_v \ge 0 && \forall v\in V.
\end{aligned}
$$

We note that the edges that are forced in a given branch in the branch&bound algorithm gives rise to a set of clusters.

Question 4

We can use the mixed integer linear formulation by Miller, Tucker and Zemlin to obtain a linear program of polynomial size with regards to the number of nodes in the TSP problem. In the following, d_{ij} denotes the distance from node i to node j, and x_{ij} is an integer variable in the domain $\{0,1\}$, where $x_{ij}=1$ iff the edge from node i to j is included in the solution. In addition, there are $|V|-1$ auxiliary variables u_i in the domain of the real numbers, which are used to eliminate subtours:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i\in V}\sum_{j\in V} d_{ij}x_{ij} \\
\text{subject to} \quad & \sum_{j\in V} x_{ij} + \sum_{j\in V} x_{ji} = 2 && \forall i\in V \\
& u_i - u_j + |V|\cdot x_{ij} \le |V|-1 && \forall i,j\in V-\{1\} \\
& u_i \ge 0 && \forall i\in V-\{1\} \\
& x_{ij}\in\{0,1\} && \forall i,j\in V
\end{aligned}
$$

The first constraint ensures that each node has exactly one incoming and one outgoing edge that is included in the solution. The second constraint ensures that there is only a single tour in the solution, namely the tour beginning and ending at node 1.

Question 5

We can relax the above mixed integer linear program by allowing the edge variables x_{ij} to take on real values. This yields the following linear program, which does not include integer variables, and hence can be solved efficiently:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i \in V} \sum_{j \in V} d_{ij} x_{ij} & \\
\text{subject to} \quad & \sum_{j \in V} x_{ij} + \sum_{j \in V} x_{ji} = 2 & \forall i \in V \\
& u_i - u_j + |V| \cdot x_{ij} \leq |V| - 1 & \forall i, j \in V - \{1\} \\
& u_i \geq 0 & \forall i \in V - \{1\} \\
& x_{ij} \leq 1 & \forall i, j \in V \\
& x_{ij} \geq 0 & \forall i, j \in V
\end{aligned}
$$

The solution space for the original integer linear program is a subset of the solution space for the relaxed version. Hence an optimal solution to the relaxed program will be a lower bound for the integer solution.

Question 6

We have implemented the three lower bounds based on the Java code that was handed out. The code archive that was turned in with this report should be self-contained. To run the TSP solver with each of the lower bounds, use the provided `Makefile`, by issuing `make` *`bound`* on the terminal, where *bound* is one of "onetree", "radius" or "relax".

For the three graphs, we get the following optimal values for TSP tours:

$$
\begin{aligned}
OPT_1 &= 8.649, \\
OPT_2 &= 19.030, \\
OPT_3 &= 26.753.
\end{aligned}
$$

We evaluate the following number of nodes for each lower bound. Note that the number of nodes are not correlated with the actual time it took to compute the solution, as some of these lower bounds has a considerable overhead in our implementation, due to linear programs being generated in inefficient ways. Thus, *1-tree* completed fastest, followed by *relax*, followed by *radius*.

	Instance 1	Instance 2	Instance 3
Brute-force	233902	1659236	∞
1-Tree	2117	185	482435
Radius	2283	586630	∞
Relax	1377	90	70362

Question 7

1-tree

This lower bound method did not perform very well for the large graph instance. The reason for this could be that the 1-tree does not give any guarantees of the size of the cycle that it containsl, just that it *will* contain a cycle. The 1-tree therefore will not necessarily have any structure in common with the optimal solution, causing it to underestimate the lower bound considerably.

radius

This lower bound method did not perform very well for neither the medium sized or the large graph instance. It seems that the performance of this lower bound depends a lot on the choice of clusters used for calculating the moats.

We got a slight performance increase when ordering the edges in ascending order according to their sizes, as this seems to form clusters of nodes that are close to each other. This heuristic could probably be improved by using a more advanced technique than sorting the edges.

relax

This lower bound method performed fairly well with regards to the number of nodes evaluated.

Bibliography

[1] W. J. Cook. Solving a tsp, tsp control zones. `http://www.tsp.gatech.edu/methods/opt/zone.htm`, January 2007.