Advanced Algorithms and Data Structures
Hand-in 2

Philip Pickering
pgpick@gmx.at

Thomas Bracht Laumann Jespersen
ntl316@alumni.ku.dk

Prove that 1-tree is a lower bound

Consider the optimal tour C^* in a graph G and a vertex u. Included in the tour are two edges connecting u to the rest of the graph, e_1 and e_2. Let G' be the graph formed from G by removing u and all its incident edges from G. Then we can find a spanning tree $T^{G'}$ from the optimal solution, by removing e_1 and e_2 from C^*. The following holds:

$$c(C^*) = c(e_1) + c(e_2) + c(T^{G'}). \tag{1}$$

Next consider a minimum spanning tree M found in G', and the two lowest-cost edges e'_1, e'_2 connecting u to G'. It is trivial to see that:

$$c(e'_1) + c(e'_2) \leq c(e_1) + c(e_2), \tag{2}$$

and we can also state that:

$$c(M) \leq c(T^{G'}), \tag{3}$$

since M and $T^{G'}$ are both spanning trees in G'.

Combining equations (2) and (3) gives:

$$c(e'_1) + c(e'_2) + c(M) \leq c(e_1) + c(e_2) + c(T^{G'}) \overset{(1)}{=} c(C^*), \tag{4}$$

which proves that the 1-tree is indeed a lower bound on the optimal tour.

Zone LP

Given a graph $G = (V, E)$ with cost/distance function $c : V \times V \to \mathbb{R}$, we can write the following linear program to maximize the radii of non-overlapping circles drawn around each vertex (the vertices being the origin):

$$\begin{array}{lrcll} \text{max.:} & \sum_{i=1}^{|V|} r_i & & & \\ \text{s.t.:} & r_i + r_j & \leq & c(v_i, v_j), & \forall i, j \in \{1, \ldots, |V|\}, v_i, v_j \in V, i \neq j \\ & r_i & \geq & 0, & \forall i \in \{1, \ldots, |V|\} \end{array}$$

Zone lower bound

As is described in the exercise a tour visiting each city has to travel from somewhere on the circumference of each circle, to its center and back.

Let $r_1^*, \ldots, r_2^*$ be the optimal radii found by solving the liner program presented in the previous section. Then a lower bound on the optimal TSP solution is given by:

$$2 \sum_{i=0}^{|V|} r_i^*.$$

Because the circles are non-overlapping, the optimal solution C^* may have to travel outside the circles, this is indeed a lower bound.

ILP formulation of TSP

To formulate TSP as an *integer linear program* (ILP), we introduce decision variables for each edge x_{ij}, where $i, j \in \{1, \ldots, |V|\}$ and $i < j$, and demand that $x_{ij} \in \{0, 1\}$, indicating by zero that the edge is not included in a tour, and one if it is.

$$\text{min.:} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_{ij}$$

$$\begin{aligned}
\text{s.t.:} \quad \sum_{k=1}^{i-1} x_{ki} + \sum_{k=i+1}^{n} x_{ik} &= 2, && i \in \{1, \ldots, |V|\} \\
\sum_{i,j \in Z} x_{ij} &< |Z|, && \varnothing \subset Z \subset V \\
x_{ij} &\in \{0, 1\}, && i, j \in \{1, \ldots, |V|\}
\end{aligned}$$

The first constraint specifies that for every vertex i, exactly two edges incident to i must be included in the tour.

The second constraint is concerned with subtour elimination, demanding that for any non-empty $Z \subset V$, the number of edges that are entirely in Z (both endpoints in Z) must be strictly less than $|Z|$. Say we have five vertices in a subset Z. In order to form a tour amongst themselves, we would need use six edges, which is disallowed by this constraint.

The third constraint is our *integrality constraint*, demanding that we either include an edge completely or not at all.

In order to relax these constraints, we could start by removing the integrality constraint on x_{ij}, and let $0 \leq x_{ij} \leq 1$. The we definitely have a linear program, which could be solved much more efficiently. The optimal solution to this relaxed form offers a lower bound on the optimal value.

But could we do better? We have an number of constraints to rule out subtours exponential in the size of $|V|$. If we remove these our problem becomes a lot easier to solve by a linear program — we only have $|V|$ constraints.

By removing the integrality constraint and the subtour constraint, we have reduced our problem to a form of the assignment problem, which is promised to have an integral solution, because the constraint matrix is totally unimodular.

1-tree After implementing the main idea of 1-tree by modifying the graph using additional BnBnodes, the challenge was to choose a node at each iteration, which could lead to a good bounding result. As can be seen in the different versions, various possibilities have been tested. The version used in the end sorted the vertices once by their two least costing edges and chose accordingly.

Zones We implemented the zone lower bound as described above wih one small difference: First we formulate a linear program to compute the maximal radii of the zones. Then, for a given branch-and-bound node, we collect all the edges that *must* be included in the tour (by this node), and attempt to add a little more to our lower bound.

Consider an included edge (i, j): we can add $k = c(i, j) - r_i^* - r_j^*$ to our lower bound. Because the circles are non-overlapping, $k \geq 0$, so we can only improve our lower bound.

ILP Relaxation The implementation of ILP is rather straight-forward and generates at every branch-and-bound node a new linear program and computes a lower bound as described above.

For a given branch-and-bound node, a few more constraints are added to the linear program generated: Say a given edge (i, j) must be *included*, we simply add the constraint "$x_{ij} = 1$" and if the edge must be *excluded* we demand "$x_{ij} = 0$."

	Instance 1		Instance 2		Instance 3	
none	8.649	233902	19.030	1659236	'inf'	'inf'
1-tree (V6)	8.649	12919	19.030	137	26.753	1250972
Zones	8.649	23740	19.030	86364	'inf'	'inf'
ILP Relaxation	8.649	3770	19.030	744	'inf'	'inf'

Table 1: Results for the various algorithms

1-tree The version used in the end - V6 - performed very good compared to the former versions, a lot of possible improvements have already been implemented. Choosing the vertex for one-tree was one of the main leverages for improvement, as described above. Instance 1 and 2 show the different results depending on the graph density, a sparse graph needs a lot less generated nodes. Even instance 3 could be solved in reasonable time compared to the brute force approach. A not implemented, but possibly interesting improvement would be to sort the vertices for chosing the one-tree-vertex dynamically, depending on the possible minimum cost edges.

Zones Of the three lower bound methods, the zones seems to fare the worst, but we didn't perform all the optimizations possible either. We didn't consider *excluded* edges when improving our lower bound, which is a little more involved.

Say the edge (i, j) for a certain branch-and-bound node is excluded we can inspect $c(i, j)$ and the radii r_i^* and r_j^*. If $r_i^* + r_j^* = c(i, j)$, then the edge (i, j) is constraining our radii. But since we cannot include it in the tour (for this step in the branch-and-bound algorithm), we could have ignored it (or pretended that $c(i, j) = \infty$).

That means another edge would have constrained the radii r_i^* and r_j^*. If we for node i step through its incident edges, we select, among the edges available to us (disregarding (i, j) and other excluded edges), the edge that constrains us the most, say (i, j'), and we can add $k = c(i, j) - r_i^* - r_{j'}^*$ to our lower bound. The symmetrical argument holds for j.

ILP Relaxation The interesting observation about the LP relaxation of TSP is that the lower bound is much stronger than any of the other two bounds. But this advantage is more or less lost on computation time, because we haven't been smart about the treatment of linear programs. We simply generate a new linear program per branch-and-bound node, which is a waste.

Ideally, we should construct an initial LP and add and remove constraints to it and "re-solve" it. This would save both time and space, as adding just a single constraint and searching for the solution from a near-optimal point is bound to be more efficient than reconstructing the entire linear program plus the extra constraint.