Pragmatic Programming

Session 12 - The Traveling Salesman in $\mathbb{R}^2$

Max Nilsson

February 16th

The Euclidean Traveling Salesman

The general Traveling Salesman Problem is *Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?*

- In other words, this is the weighted Hamiltonian cycle problem. It is in **NP**.
- We will consider n cities in $\mathbb{R}^2$. The important properties is that the graph is complete and that the distances between the distances between the nodes satisfy the triangle inequality.
- The solution cycle will form a simple polygon, i.e., it will have no self-intersections (except the case when $n = 2$?).
- For some collection of edges E' let $d(E')$ represent the total distance of all edges in E'. Let the cycle C be the solution to the problem.
- We will look at one handsome approximation and one beautiful approximation.

The Salesman's Lament - Minimum Spanning Trees

Before the salesman will begin his venture he will have to pack his knapsack. The cursed items of old reminds him of late evenings writing slides to some outdated course in pragmatic programming.

- Recall from lecture 2 that a minimum spanning tree is found by sorting the edges, iterate through them and perform union-find by rank. The bottleneck is the sorting, which in our case gives $\mathcal{O}(n^2 \log n)$.
- In a Euclidean graph, such as this one, there is a better way. First perform Delaunay triangulation (with triangles being "small" in the sense that their respective circumcircles do not contain any nodes in their interiors). With a divide and conquer technique this can be done in $\mathcal{O}(n \log n)$. This reduced the number of edges to $\mathcal{O}(n)$ and the final spanning tree is actually minimum. In total $\mathcal{O}(n \log n)$.
- Let T be this minimum spanning tree. By the triangular inequality

$$d(C) \leq 2d(T) \leq 2d(C) \qquad \text{(Handsome)}$$

since if we remove one edge from C we get a spanning tree. We can also easily construct a cycle within this 2-bound.

The Salesman's Mistake - Euler Cycles

If only the spanning tree were traversable, like some Eulerian cycle, passing through its edges exactly once. This reminded the Salesman of a mistake in lecture 4. He daydreams of correcting it:

- **Proposition:** If every vertex of a connected graph G has even degree, then it contains a Euler cycle.
- *Proof:* Perform a depth first search from any root node which terminates when it reaches the root node. This gives a cycle C only passing through edges at most once (called a *circuit*).
- If we remove C from the edges of the graph, we will be left with possibly disconnected components, but all have even degrees. By induction, these components all have a Euler cycle.
- Now form a final Euler cycle of G by inserting Euler cycles of the sub-components into C the first time a vertex of that sub-component is visited in C. *End of daydream and proof.*
- This proof in fact gives a recursive algorithm in $\mathcal{O}(m)$, where m is the number of edges, called *Hierholzer's algorithm* (1873).

The Salesman's Promise - Edmund's Blossom

Unfortunately, a Euler cycle cannot be found in T, since it contains some nodes with odd degrees. By the Handshake Lemma, there are an even amount of such nodes. Given the complete induced subgraph from these nodes, the Salesman contemplates its minimum weight maximum matching. The main idea is the Blossom Algorithm, which was promised in lecture 9.

- First consider the unweighted case. Let M be a matching of a graph G.
- A node is **exposed** if no edge of M is incident with it. A path is an **alternating path** if its edges are alternately not in M and in M (or in M and not in M). An **augmenting path** is an alternating path that starts and ends at two distinct exposed nodes.
- One can show that a matching M is maximum if and only if there exists no augmenting paths with respect to M.
- Edmund's Blossom algorithm repeatedly augments the matching (i.e. switching the matching in an augment path). To make this efficient, the graph is **contracted** and **lifted** with the help of **blossoms**.

Blossoms

A blossom is a cycle in G consisting of $2k+1$ edges of which exactly k belong to M, and where one of the vertices of the cycle has an alternating path of even length from it to an exposed vertex.

- Define the contracted graph G' as the graph obtained from G by contracting every edge of the blossom, and define the contracted matching M' as the matching of G' corresponding to M.
- Since the blossom has cycle of odd length, one can show that G' has an augmenting path if and only if G has an augmenting path. If the augmenting path goes through a contracted blossom, then one of its two directions yields an alternating path.
- This contraction creates a recursive call to finding an augmenting path to G' which we can then lift up to G and perform augmentation.
- The details are quite messy, but this can be extended to the weighted case (similar to how it is done in network flow) and in the end you get a $\mathcal{O}(n^4)$ algorithm which Gabow (1974) improved to $\mathcal{O}(n^3)$.

The Salesman's Conclusion - Christofide's Algorithm

Let us recap the algorithm so far and give the final step. We are given a complete Euclidean graph G and we want to bound the solution C.

1. Find the minimum spanning tree T of G in $\mathcal{O}(n \log n)$
2. Find the nodes U of odd degrees in $\mathcal{O}(n)$
3. Compute the minimum weight matching M of U in $\mathcal{O}(n^3)$
4. Find a Euler cycle of $T \cup M$ with Hierholzer's in $\mathcal{O}(n)$
5. Finally: remove duplicate nodes from the Euler cycle to create a Hamiltonian cycle $\tilde{C}$ of G in $\mathcal{O}(n)$

- First note that $d(C) \leq d(\tilde{C}) = d(T) + d(M) \leq d(C) + d(M)$. So it remains to bound M.
- The trick is to consider the cycle C as it passes over the nodes U. It does so in order $u_1, \ldots, u_r$ which induces two matchings $M_1 = \{(u_1, u_2), (u_3, u_4) \ldots\}$, $M_2 = \{(u_2, u_3), (u_4, u_5) \ldots\}$.
- We get $2d(M) \leq d(M_1) + d(M_2) \leq d(C)$ and so

$$d(C) \leq d(\tilde{C}) \leq \frac{3}{2} d(C). \qquad \text{(Beautiful)}$$

Improvements to the Euclidean Traveling Salesman

- In 2020, Karlin, Klein, and Gharan managed to improve the 1.5 approximation in general to a

  $$1.4999999999999999999999999999999999$$ approximation.

- If the problem is embedded in $\mathbb{R}^d$ then Arora and Mitchell gave in 2010 a polynomial-time approximation scheme with bound $1 + c^{-1}$ for any $c > 0$ with time complexity

  $$\mathcal{O}\left(n(\log n)^{\mathcal{O}(c\sqrt{d})^{d-1}}\right).$$

- They were awarded the Gödel Prize.

This and Next Week

- This week's problem set will be the last one.
- There will be 12 (one for each session) randomly picked problems between 3 and 6.5 in Kattis difficulty. I will not sort the problems this week.
- There will be a 13th and final session next week 13:15-15:00. Come prepared for something completely different.
- There will also be a 13th problem on this final problem set. It is optional and there is really no way to "solve" it.
- If you solve 6 out of the 12 random problems and have fulfilled each week's requirement, contact me and I will let Pontus know after the final session.