E0 318: Topic in Geometric Algorithms October 22, 2024

Approximate TSP using Quadtrees

Instructor: Abhiram *Scribe: Abhiram*

1 Introduction

This lecture will present an algorithm which provides a PTAS for the Euclidean Travelling Salesman Problem in time $O\left(n\left(\frac{1}{\varepsilon}log(n)\right)^{\frac{1}{\varepsilon}}\right)$ as given by [Aro96]. The structure of the lecture will (loosely) follow Chapter 13 of [HP11].

2 A first Approach: Divide and Conquer

First we provide an exact algorithm that sets the stage for the techniques and notation we shall use.

2.1 Definitions

Consider n **points**, $P \subset \mathbb{R}^2$. A **tour** in $\mathbb{R}^2$ is a continuous and piece-wise differentiable map $p : [0,1] \to \mathbb{R}^2$ with $p(0) = p(1)$. The **length of a tour** $||p|| = \int_0^1 |p'(t)|dt$. A TSP **tour** is a tour π in $\mathbb{R}^2$ that traverses every node in P. The problem statement is to find a TSP path of **minimum length**. That is, find a TSP path π_{opt} such that for every TSP path π, $||\pi|| \geq ||\pi_{opt}||$.
To solve this using divide and conquer, we shall also need to define what our sub problems will look like. Consider a half open **rectangle** $R = [a,b) \times [c,d)$, a set $K \subset \partial R$ of **points on the boundary** of the rectangle, $|K| \leq t$, even for some constant t, and $\sigma : [|K|] \to K$ a **permutation** of K. Let us use K_i^σ to denote $\sigma(i)$, omitting σ where it is not ambiguous. A subproblem TSP^* consists of R, K, σ and the goal is to find $\frac{|K|}{2}$ paths $\vec{\pi}$ where π_i begins at K_{2i-1} and ends at K_{2i} such that $K \subset \bigcup_{i=1}^{\frac{|K|}{2}} \pi_i$ and $\sum_{i=1}^{\frac{|K|}{2}} ||\pi_i||$ is **minimised**.
For a point $p \in \mathbb{R}^2$, $V(p)$ is the **vertical line** passing through p and $H(p)$ the **horizontal line** passing through p. A rectangle can be **cut** by an axis-aligned line L to two rectangles, on either side of the line which we will denote R_1^L and R_2^L, dropping the L where unambiguous.

Definition 1 (Cutting Lines)**.** *The set $\mathcal{L}_P = \{V(p) \mid p \in P\} \cup \{H(p) \mid p \in P\}$ of all axis aligned lines crossing points in P is called the set of Cutting Lines.*

A simple heuristic is that since we are in the Euclidean metric, the shortest tour has to be straight lines between the points.

Definition 2 (Possible Edges)**.** *$E = \bigcup_{i,j \in P} E_{i,j}$ is the set of possible edges, where $E_{i,j}$ is the straight line from i to j.*

Definition 3 (Rectilinear Binary Space Partition)**.** *It is a family of rectangles $\mathcal{R}$ such that:*

- *$R, S \in \mathcal{R}, R \cap S \neq \phi \Rightarrow R \subset S$ or $S \subset R$.*
- *$R \cap P \neq \phi \ \forall R \in \mathcal{R}$.*
- *If $R, S \in \mathcal{R}$, $R \subset S$, then $\exists \mathcal{S} \subset \mathcal{R}$ such that $S \setminus R = \bigsqcup_{r \in \mathcal{S}} r$.*
- *$\forall\ p \in P\ \exists\ R_p \in \mathcal{R}$ such that $R_p \cap P = \{p\}$.*

Definition 4 (t-friendly)**.** *A RBSP $\mathcal{R}$ is said to be t-friendly with an optimal TSP tour π_{opt} if every rectangle $R \in \mathcal{R}$ intersects π_{opt} at most t times.*

We have the following result provided by the algorithm to come.

Theorem 5. *For a constant t, if a point set P admits a minimum TSP tour π_{opt} and a RBSP $\mathcal{R}$ that is t-friendly with π_{opt}, then we can find π_{opt} in time $n^{O(t)}$.*

2.2 Algorithm

First, we provide the algorithm to solve a subproblem.

Algorithm 1: Recursive Algorithm (AlgSub) for a Subproblem

Input: R, K, σ
Output: $\vec{\pi}$
if $|P \cap R| = 1$ **then**
  Find shortest by brute force
end
Initialise $best \leftarrow \infty$
Initialise *path*
for $L \in \mathcal{L}_{P \cap R}$, $K_{new} \subset E \cap L$, σ_1, σ_2 *permutations of* R_1, R_2 **do**
  Verify compatibility of the subproblems generated.
  if *compatible* **then**
    $sum \leftarrow ||AlgSub(R_1, (K \cap R_1) \cup K_{new}, \sigma_1)|| + ||AlgSub(R_2, (K \cap R_2) \cup K_{new}, \sigma_2)||$
    if $sum < best$ **then**
      $best \leftarrow sum$, $path \leftarrow AlgSub(R_1, (K \cap R_1) \cup K_{new}, \sigma_1) \cup AlgSub(R_2, (K \cap R_2) \cup K_{new}, \sigma_2)$
    end
  end
end
return path

We assume verifying compatibility is simple, and use it as a black box. The more skeptical reading can refer to Appendix A.

Next, to generate the first two sub-problems, we have to make a small change to the algorithm to account for the output being a tour instead of a collection of paths. We consider the frame F of P, the smallest rectangle containing it, and for every $L \in \mathcal{L}_P$, we look at subsets of $E \cap L$ of size at most t and generate compatible permutations σ_1 and σ_2 (again, Appendix A). Now we again take the minimum sum of solutions to the subproblems and report the union of the paths given as the tour π_{alg}.

2.3 Correctness

Note that the set of all rectangles considered during a recursion of Alg 1 form a RBSP. Thus, by different recursions, it considers different RBSPs. The correctness comes from the fact that the algorithm covers every RBSP, so if any of them were t-friendly, we would have found it.

Lemma 6. *Algorithm 1 provides the correct set of paths $\pi_{opt} \cap R$ for the subproblems in the RBSP that is t-friendly with π_{opt}.*

Proof: Let's assume for the sake of contradiction that the algorithm is incorrect for some subproblem. Let R, K, σ be a minimal subproblem for which we get an incorrect answer. The Algorithm returns some $\vec{\pi}_{alg} \neq \pi_{opt} \cap R$. If $||\vec{\pi}_{alg}|| < ||\pi_{opt} \cap R||$, then replacing that section of π_{opt} with $\vec{\pi}_{alg}$ will give a shorter TSP tour, contradicting optimality of π_{opt}. Alternatively, if $||\pi_{opt} \cap R||$ is lesser, then since there exists a

t-friendly RBSP, there is a line $L \in \mathcal{L}_{P \cap R}$ which splits R into 2 valid subproblems for $K_{new} = \pi_{opt} \cap R \cap L$ with the orders σ_1, σ_2 induced by π_{opt}. Since they are valid subproblems, they have to be considered while solving the subproblem for R. This contradicts that the Algorithm chooses the lowest value among all the subproblems. Thus, by contradiction, the algorithm will find π_{opt}. □

2.4 Runtime

We will use memoisation so that we only need to solve each subproblem once. Thus, the total runtime is the number of subproblems multiplied by the time taken to solve a sub problem given the solution to each smaller subproblem.

Time to solve a Subproblem

For each rectangle R we consider, A line segment can intersect a rectangle at most twice. E has ${}^nC_2$ edges, so $|E \cap R| \leq 2\ {}^nC_2$ points, out of which we choose at most t points for K. Ignoring "compatibility" to get a generous upper bound, we assume all $|K|!$ permutations can be taken. Then, the number of subproblems for each rectangle R is at most

$$\sum_{i=0}^{t} 2\ ({}^nC_2)^i\ i! \leq 2n^{2t}t!$$

Thus, for each of the $2n$ lines in $\mathcal{L}_{P \cap R}$ we consider as many subproblems for both the rectangles generated and find the minimum of the sum of the answers. Thus, we can update a cell in time $n^{O(t)}$. Remember that the base case is just an enumeration on $t+1$ points and can be solved by brute force in time $t^{O(t)}$.

Number of Subproblems

This leaves the last matter to address, a matter that would have the moderately invested reader on the edge of their seat, eating their hat, wondering why (We say "moderately" because we expect the heavily invested reader to have noticed this, work it out, and found the answer for themselves). The matter is the following: Why do we only consider lines from $\mathcal{L}_{R \cap P}$ for generating subrectangles of R. The answer is simple, considering *every* line would leave us with a lot of redundancy. Consider two lines L_1, L_2 such that the points in the two subrectangles are the same. Let us name the subrectangles generated by L_1 to be R_1^1 and R_1^2 and those generate by L_2 to be R_2^1 and R_2^2. Then, An edge $e \in E$ intersects $L_1 \Leftrightarrow e = ab, a \in P \cap R_1^1, b \in R_1^2$. But $P \cap R_1^1 = P \cap R_2^1$ and $P \cap R_1^2 = P \cap R_2^2$. Which means e intersects L_2 as well. By a symmetric argument, we conclude that there is a 1-1 correspondence between $E \cap L_1$ and $E \cap L_2$. Through this correspondence, we can translate every pair of subproblems generated by L_1 to a pair of subproblems generated by L_2 and vice versa. Thus, for every set of lines with the same $P \cap R^1$, we need only one representative line to be considered. $\mathcal{L}_{P,R}$ has a representive line for each set.
I go through the trouble of this entire justification for the simple result this gives: Any rectangle of a subproblem must be made of 2 vertical and 2 horizontal lines from $\mathbf{L}_P$. Thus, there are only n^4 rectangles that need to be considered. This coupled with the number of subproblems per rectangle shown above tells us that the total number of subproblems is $\leq 2n^{4+2t}t! = n^{O(t)}$. Multiplying this with the time taken to solve a subproblem, we get the total runtime to be $n^{O(t)}$. □

If we are able to comment on the existence of t-friendly RBSPs, we will be able to get a polynomial exact algorithm. Alternatively, if we can show that there is a TSP tour π_t which has a t-friendly RBSP, and $||\pi_t|| \leq (1 + \frac{1}{t})||\pi_{opt}||$, then we can set $t = \theta(\frac{1}{\varepsilon})$ and get a PTAS. However, both these questions have proven hard to answer. By making slight improvements to the algorithm, we can accomplish the second objective. We will present that approach in the next section.

3 A Randomised PTAS using Sliding Quad Trees and Portals

Now we present the main algorithm of the lecture. In place of the the RBSP, we shall use a hierarchical quadtree, in place of E, we will use fixed "portals" on the sides of each square. We will then show the analysis that by choice of the number of portals, we can bring a multiplicative error within $(1+\varepsilon)$ taking only time $O\left(n\left(\frac{1}{\varepsilon}log(n)\right)^{\frac{1}{\varepsilon}}\right)$.

3.1 Definitions

Definition 7 (Snapping)**.** *Given a point set P and a grid G, the snapping function $\rho_{P,G} : P \to G$ maps each point of P to its closest grid point in G. The snapped point set P_G is the range $\{g \in G \mid g = \rho_{P,G}(p)$ for some $p \in P\}$ of $\rho_{P,G}$.*

Definition 8 (Hierarchical Quad Tree)**.** *A collection of concentric grids $\vec{T}$ of decreasing breadth. T_i has breadth 2^{-i}. More precisely, $T_i = \{V((2^{-i}j, 0)) \mid j \in \mathbb{Z}\} \cup \{H((0, 2^{-i}j)) \mid j \in \mathbb{Z}\}$. Each cell of a grid is split into 4 in the next grid. A **shift** of a HQT by a vector $\vec{z} \in \mathbb{R}^2$ is the new set of grids $T'_i = \{V((z_1 + 2^{-i}j, z_2)) \mid j \in \mathbb{Z}\} \cup \{H((z_1, z_2 + 2^{-i}j)) \mid j \in \mathbb{Z}\}$. That is, the origin of the concentric grids is shifted to $\vec{z}$.*

3.2 Algorithm

We begin by picking a constant $\varepsilon > 0$.
First we scale and shift the point set P to fit in $[0,1)^2$, while having diameter $> \frac{1}{2}$. We perform an affine scaling, with both axes scaling by the same ratio, so that distances in the scaled set are proportional to distances in the original set. Thus, we can now work with the scaled set. Let us abuse notation a bit, and call the scaled set P, suggesting that P was scaled to begin with.
Next, we snap P to a very fine grid. let $\mathcal{E} = \left\lceil \frac{32}{\varepsilon} \right\rceil$. Let G be the grid of width $\frac{1}{n\mathcal{E}}$. We snap P to G and find π_{snap}, a TSP tour on P_G (Details on how are coming right up). Then, from each $p \in P_G$, we go to each point in $\rho^{-1}_{P,G}(p)$ and back to give a TSP tour of P.
Now for portals, for each grid cell in T_i, we place m equidistant portals on each side of the cell. The exact value of m will be disclosed in the analysis where the reader will be able to appreciate the choice of the value. For now, it suffices to say that $m = \theta(\frac{1}{\varepsilon}log(n))$.

Generating the Initial Subproblems

For a change, in this section let us handle this case first and then move to solving the subproblems. We consider the new edges of T_1 not in $[0,1)^2$, that is the plus in the middle, and pick portals out of the 4m+5 portals on this plus such that no individual square has more than t portals, for a constant t. From our conversation above, you will remember that $t = \theta(\frac{1}{\varepsilon})$. Again, the exact value will be disclosed later. Pick 4 orderings σ_1 to σ_4 for each of the squares, and check compatibility (Appendix A). Moving over all choices of portals and compatible orders, and finding the minimum sum of the four subproblems, we return the final path.

Solving a Subproblem

Given a grid cell R of T_i, and K the multiset of portals to be used in order σ, if $|R \cap P| == 0$ or 1, we find the shortest path by brute force, else we divide the cell into 4, by considering the cells of T_{i+1}. We consider more portals from the 4 new edges added, this gives $4m + 5$ new portals out of which we choose compatible portals such that no edge uses more than t portals. Taking the minimum over the value of the 4 subproblems thus generated, the union of the 4 paths is returned as the answer to the subproblem.

3.3 Runtime

Again, we use memoisation to avoid resolving subproblems. Let's begin with a lemma whose corollary limits the number of times a portal is used. But remember this lemma till later since the stronger statement we prove will be used in analysing the approximation ratio.

Lemma 9 (Patching Lemma)**.** *Given a tour π and a line segment s, if π crosses s more than twice, we can construct a new tour π^* such that:*

1. $\pi^* \setminus s = \pi \setminus s$
2. *π^* crosses s at most twice*
3. $||\pi^*|| \leq ||\pi|| + 3||s||$

What this tells us is that if π cuts a line segment many times, we can reduce the crossings by moving up and down the line segment without crossing, and the total distance we move in this manner is at most thrice the length of the line segment. We shall put off the proof to Appendix B since the proof comes from Eulerian tours of graphs and does not provide insight any more to this problem than the statement does. We will however note the following corollary.

Corollary 10. *Given a tour π which only uses portals on grid cells of $\vec{T}$ there is a tour π^* which only uses portals on grid cells of $\vec{T}$ of length $\leq ||\pi||$ which uses the same portals as π, using each portal at most twice, and is the same as π away from the portals.*

Proof: Use the patching lemma taking each portal as a s, $||s|| = 0$ so $||\pi^*|| \leq ||\pi||$ □

This tells us that when considering all possible combinations of portals to use while generating subproblems, we only need to choose each portal at most twice. Consider them as two separate instances of each portal, then from the $8m + 10$ choices, we choose up to t of them. As for the ordering of the portals, let us be generous and forget compatibility, to allow all $|K|!$ permutations to be allowed, to get an upper bound on the number of sub problems we can generate from a given subproblem. It comes out to be

$$\sum_{i=0}^{8t} {}^{8m+10}C_i \cdot i! \leq 8k(8m+10)^{8k}$$

which, by our choice of m and t, it $O\left((\frac{1}{\varepsilon}log(n))^{\frac{1}{\varepsilon}}\right)$. Then, in as much time, we can find the sum of all such subproblems and find the minimum. Thus, this is the time to solve a subproblem given all smaller subproblems. Now we find the total number of subproblems.
Due to our snapping, the minimum distance between any two distinct points is $\geq \frac{\varepsilon}{33n}$ so the maximum depth of the tree H required is given by $2^{-H} < \frac{\varepsilon}{33n}$ or $H = O(\log(n) + \frac{1}{\varepsilon})$. At each depth, we only split cells if they contain 2 or more points of P. Thus at most $2n$ subproblems need to be solved at each depth. So the total running time is given by $2nH \cdot O\left((\frac{1}{\varepsilon}log(n))^{\frac{1}{\varepsilon}}\right) = O\left(n(\frac{1}{\varepsilon}log(n))^{\frac{1}{\varepsilon}}\right)$, while matches the claim at the beginning of this lecture.

3.4 Approximation Ratio

This algorithm gives us π_{alg}, the best TSP tour on the snapped point set which uses only the portals, for which the Quad Tree is t-friendly. We will deduce the approximation ratio in 3 steps, listed as the 3 theorems below.

Theorem 11 (The cost of Portals)**.** *Let $\pi_{snap,t}$ be the best TSP tour on the snapped point set for which the quad tree is t-friendly. Then $\mathbb{E}\left[\frac{||\pi_{alg}||}{||\pi_{snap,t}||}\right] \leq (1 + \frac{\sqrt{2}H}{m+1})$*

Theorem 12 (The cost of t-friendliness)**.** $\mathbb{E}\left[\frac{||\pi_{snap,t}||}{||\pi_{snap}||}\right] \leq (1 + \frac{3\sqrt{2}}{t-2})$

Theorem 13 (The cost of Snapping)**.** $||\pi_{snap}|| \leq (1 + \frac{\varepsilon}{2}) \cdot ||\pi_{opt}||$

Then, combining these 3 theorems, choosing $t = \frac{50}{\varepsilon}$ and $m = \frac{20H}{\varepsilon}$, we get the required

$$\mathbb{E}\left[||\pi_{alg}||\right] \leq (1 + \varepsilon)||\pi_{opt}||$$

To prove those theorems, we need some properties of the Shifting Hierarchical Quad Tree.

Lemma 14 (Properties of SHQTs)**.** *1. For axis aligned line segments s, $\boldsymbol{P}[s \cap T_i \neq \phi] = 2^i||s||$ if $||s|| \leq 2^{-i}$ and 1 otherwise. $\mathbb{E}[|s \cap T_i|] = 2^i||s||$*

2. For any line segment s, $\boldsymbol{P}[s \cap T_i \neq \phi] \leq \sqrt{2} \cdot 2^i||s||$ and $2^i||s|| \leq \mathbb{E}[|s \cap T_i|] \leq \sqrt{2} \cdot 2^i||s||$

3. The above bounds hold for the union of any finite collection of line segments and in particular for polygonal curves.

4. Let E_i be the set of open edges (discluding the vertices) of T_i. Then, for an edge e, $\boldsymbol{P}[e \in E_{i-1} \mid e \in E_i] = \frac{1}{2}$

These results arise from basic probability and I shall not belabour the reader with the proofs here. Now we set out proving the theorems.

Proof: **(Theorem 11)** Let Z_i be the number of intersections of $\pi_{snap,t}$ with T_i. We construct a TSP tour on P_G for which the SHQT is t-friendly and it only crosses at the portals, which has at most the desired error. Then, since π_{alg} is the shortest such tour, the error is within the desired error and so we prove the result. For the construction, every time $pi_{snap,t}$ intersects an edge of T_i, it moves along the edge to the nearest portal, crosses there, and moves back to the original point of intersection, before continuing along its path. Since each edge is of length 2^{-i} and is has m evenly spaced portals on it, the nearest portal is at most $\frac{2^{-i}}{2m+2}$ distance away. So the total error accumulated in the detour is at most $\frac{2^{-i}}{m+1}$. Summing this for all Z_i and over all depths i, Error $= \sum_i^H \frac{Z_i}{2^i(m+1)}$. Now, $\pi_{snap,t}$ is a polygonal curve, so $\mathbb{E}[Z_i] \leq \sqrt{2} \cdot 2^i||\pi_{snap,t}||$. Substituting, we get the desired expected error. □

Proof: **(Theorem 12)** Consider an optimal TSP tour π_{snap} on P_G. Starting from H, going to 1, for each depth, if the tour intersects an edge $e \in E_i$ more than t times, we patch it using Lemma 9 incurring an additive error of $\frac{3}{2^i}$ and removing at least $(t-2)$ intersections. Let π_i be the tour patched up to tree T_i and $\pi_{H+1} = \pi_{snap}$. For notation, let Y_i be the number of times π_{i+1} intersects T_i. After patching, let n_i be the number of times π_i intersects T_i. Let F_i be the number of times the patching lemma is invoked at depth i. Then $n_i \leq Y_i - (t-2)F_i$. Each intersection has a probability $\frac{1}{2}$ of being an intersection of T_{i-1}. $\mathbb{E}[Y_{i-1}] = \frac{\mathbb{E}[n_i]}{2} \leq \mathbb{E}\left[\frac{Y_i-(t-2)F_i}{2}\right]$. Rearranging, $\mathbb{E}[F_i] \leq \frac{1}{t-2}(\mathbb{E}[Y_i] - 2\,\mathbb{E}[Y_{i-1}])$.
The total error is the sum of errors accumulated at each depth. $Error = \sum_1^H \frac{3}{2^i}F_i$

$$\mathbb{E}[Error] \leq \frac{3}{t-2}\sum_{i=1}^{H}\frac{\mathbb{E}[Y_{i+1}] - 2\,\mathbb{E}[Y_i]}{2^i} = \frac{3}{t-2}\left(\frac{\mathbb{E}[Y_H]}{2^H} - \mathbb{E}[Y_0]\right) \leq \frac{3}{t-2}\frac{\mathbb{E}[Y_H]}{2^H}$$

$\mathbb{E}[Y_H]$ is the expected number of intersections of polygonal curve π_{snap} with T_H. By Lemma 14, this is $\leq \sqrt{2} \cdot 2^H||\pi_{snap}||$. Substituting this above, we get the required result. □

Proof: **(Theorem 13)** Along the lines of the proof of Theorem 11 above, we construct a TSP tour of P_G that is not too much worse than π_{opt}, which serves as an upper bound for the error of π_{snap}, after which we add the error from extending π_{snap} to a TSP tour of P.
To construct the TSP tour of P_G, consider π_{opt} and from each point in P, make a detour to $\rho_{P,G}(p)$. Since the width of the grid is $\leq \frac{\varepsilon}{32n}$, the maximum distance of each detour is $\leq 2 \cdot \frac{\sqrt{2}\varepsilon}{32n}$, and with n such detours, the total length added is $\leq \frac{\varepsilon}{8\sqrt{2}}$. Thus, π_{snap} is lesser than this. Again, to extend π_{snap} to a tour of P, we gain the same error. However, these are additive error. To convert it to multiplicative error, we make use of the fact that the diameter is $> \frac{1}{2}$. Thus, the error is $\leq \frac{\varepsilon}{2}||\pi_{opt}||$, giving us the required result. □

References

[Aro96] S. Arora. Polynomial time approximation schemes for euclidean tsp and other geometric problems. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 2–11, 1996.

[HP11] Sariel Har-Peled. *Geometric approximation algorithms*. Number 173. American Mathematical Soc., 2011.

A Verifying Compatibility

The main aim of verifying compatibility is to ensure that we get a meaningful solution on taking the union of the paths. For the problem, we want the tour to be connected. For a subproblem (R, K, σ), we want to maintain that the orders σ_1 and σ_2 when we take the union give us the same order σ on the portals K. Thus, verification can be done in $O(|K|)$ by ignoring the points of P and just following along the portals. We check that rectangles are always entered on odd numbered portals K_{2j-1} and exit on the next even numbered portal K_{2j}. If the exit portal is in the main problem, we check that the number of that portal in σ is even, and enter from the next portal according to σ. If it goes to the other subrectangle, we check if it is odd in σ_2 and continue there. For the main problem, we also check that before looping back to any point, we cover every single portal. That is, this process gives one big connected tour.

B Patching Lemma

Proof: **(Lemma 9)** Consider a line segment s and a tour π. The tour is allowed to touch s and go back to the same side it approached from, we only consider it an intersection if it approaches s from one side and crosses over, leaving from the other side. We can visualise this by thinking of s as a thin rectangle instead of a line segment. A crossing is when the tour enters the rectangle from one side and exits from the other. Let $K = \{K_i\}$ be the points where π crosses s, ordered along the length of s. $|K| = m > t$. Let C be the components of $\pi \setminus K$. The paths in C connect points in K to each other, with each K_i being a limit point of two $c \in C$, one on either side. Taking this idea of sides further, we construct sets $L = \{L_i\}$ and $R = \{R_i\}$ to be the left and right sides of K_i for each i. Then, each L_i and R_i are incident to exactly one $c \in C$.
Now, we construct a graph where the vertices are $L \cup R$. We need to construct edges such that each vertex has even degree, and the graph is connected. Then, we can construct an Eulerian tour on the graph, and report this tour as the required π'. Then, the tour needs the properties that outside s, it is the same as π. This means that we need one edge corresponding to each $c \in C$ and cannot have any other edges (L_i, R_j), $i \neq j$. Also, we need that π' crosses s at most twice. This means that we can include (L_i, R_i) for at most two i. The remaining edges we add should be of the form (L_i, L_j) or (R_i, R_j), $i \neq j$ and the sum of lengths of these edges should be $\leq 3||s||$. For each $1 \leq i \leq m-1$, add edges (L_i, L_{i+1}) and (R_i, R_{i+1}). Now L and R are connected. And the length of these edges is $\leq 2||s||$, once along L and once along R. This is because the sets are ordered along the length of s, so the union of the edges is a subset of s, with no overlapping. To decide the remaining edges of size $||s||$ and the crossings, we consider 2 cases.
m is even:
$m = 2k$ for some k. Consider edges $E_1 = \{(L_{2i-1}, L_{2i})\}_{1 \leq i \leq k} \cup \{(R_{2i-1}, R_{2i})\}_{1 \leq i \leq k} \cup \{(L_1, R_1), (L_m, R_m)\}$ and $E_2 = \{(L_{2i}, L_{2i+1})\}_{1 \leq i \leq k-1} \cup \{(R_{2i}, R_{2i+1})\}_{1 \leq i \leq k-1} \cup \{(L_1, R_1), (L_1, R_1)\}$
m is odd:
$m = 2k+1$ for some k. Consider edges $E_1 = \{(L_{2i-1}, L_{2i})\}_{1 \leq i \leq k} \cup \{(R_{2i-1}, R_{2i})\}_{1 \leq i \leq k} \cup \{(L_1, R_1)\}$ and $E_2 = \{(L_{2i}, L_{2i+1})\}_{1 \leq i \leq k} \cup \{(R_{2i}, R_{2i+1})\}_{1 \leq i \leq k} \cup \{(L_m, R_m)\}$

In both cases, the union of E_1 and E_2 form the set of all edges (L_i, L_{i+1}) and (R_i, R_{i+1}), the length of which we have established is $2||s||$. Then, the length of the lower of the two has to be lower than $||s||$. So we include the lower of the two. I shall leave it to the reader to verify that in both cases, both E_1 and E_2 satisfy that every node has even degree.