6.891 Approximation Algorithms	March 17, 2000
Lecture 7	
Lecturer: David P. Williamson	*Scribe: David P. Williamson*

1 A PTAS for Euclidean TSP

Today, we give a polynomial-time approximation scheme for Euclidean TSP due to Arora (1996, 1997) which finds a tour with cost no more than $(1+\epsilon)OPT$ in $O(n \log^{O(\frac{1}{\epsilon})} n)$ time. Recall that Euclidean TSP is a special case of TSP in which vertices correspond to points in the plane, and the cost of an edge (i, j) is the Euclidean distance between the corresponding points. It turns out that this technique for Euclidean TSP also works for Euclidean Steiner trees, perfect matchings, problems called k-MST and k-TSP (in which one must find the min-cost MST and TSP on k out of the n points), problems in $\Re^d$, and more.

First, some interesting history behind this result. Mitchell (1996) independently discovered a similar PTAS soon after Arora had made a preliminary announcement of his result. Also, Arora is better known for his work showing that for many problems, no PTAS can exist unless $P = NP$. He was trying to prove the same for Euclidean TSP, when he realized it wouldn't work, and figured out a PTAS.

The basic strategy for obtaining the PTAS is (as it has been for most of the PTAS's we've seen) to apply dynamic programming. The main difficulty is getting the problem to a point where we can apply dynamic programming. Our overall strategy for doing so is the following:

1. We perturb the instance to get nice properties, increasing OPT by at most ϵOPT.
2. We subdivide the plane randomly, in a way to be defined later.
3. We show that with probability at least $\frac{1}{2}$, a highly structured tour of cost no more than $(1+\epsilon)OPT$ exists for the perturbed instance with respect to the random subdivision.
4. We use dynamic programming to find cheapest tour with the structure of part 3.

1.1 Perturbing the Problem Instance

We want to perturb the problem instance so that the following properties hold:

1. All points have integer coordinates.

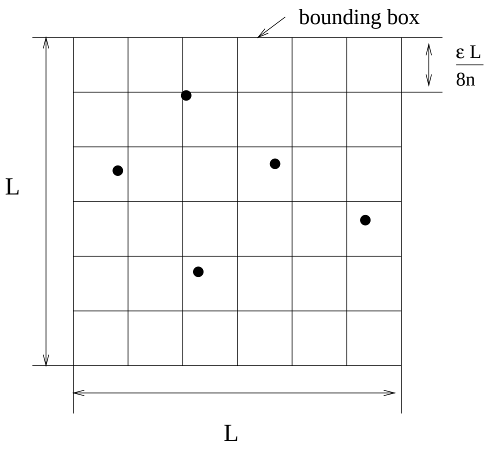

Figure 1: Illustration of bounding box and grid.

2. The minimum nonzero distance is at least 8.

3. The maximum distance is $O(n)$.

Let L denote the length of the "bounding box" around the points in the instance (see Figure 1). Observe that $L \leq OPT$. To get a perturbed instance with the properties we want, we first put down a grid of spacing $\frac{\epsilon L}{8n}$ (see Figure 1), and move the points to nearest grid points. How does this affect the length of the optimal tour? Well, each edge increases by at most $\frac{2\epsilon L}{8n}$, which implies that the total increase is at most $n\frac{2\epsilon L}{8n} = \frac{\epsilon L}{4} \leq \frac{\epsilon}{4} OPT$. Thus an optimal tour in this instance has length at most $(1+\frac{\epsilon}{4})OPT$.

We now blow up the grid spacing by a factor of $\frac{64n}{\epsilon L}$. This implies that every point is now at some multiple of $\frac{\epsilon L}{8n}(\frac{64n}{\epsilon L}) = 8$, which gives us desired properties 1 and 2. Furthermore, the maximum distance is now $O(L)\frac{64n}{\epsilon L} = O(\frac{64n}{\epsilon}) = O(n)$, assuming ϵ is fixed. Clearly if we find a near-optimal tour in the "blown up" instance, the same tour will also be near optimal in the non-blown-up instance.

1.2 Subdividing the Plane

To apply dynamic programming, we need some structure on the problem whereby we can build up a solution from the solution to smaller problems. To do this, we are going to subdivide the plane in a random way. First we will consider the subdivision without randomness, and then we will introduce the randomization.

To create the subdivision, we divide the bounding box into four equally-sized boxes, and then recursively divide the boxes into four boxes, and so on (See Figure 2). We will consider two different types of subdivisions, which differ only in when we stop the recursion. For a *quadtree*, we stop when there is at most 1 node per square.

For a *dissection*, we stop when each box has side length 1. The subdivision in Figure 2 is a quadtree.

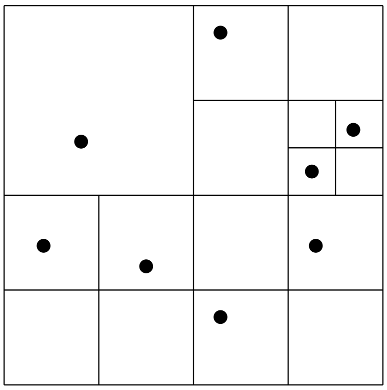

Figure 2: Example of a quadtree.

We will need to talk about the different levels of recursion in subdividing the instance. We will say that the bounding box has level 0, and the boxes in the subdivision of a level i box are at level $i+1$. Since the maximum distance between points is $O(n)$ and points at non-zero distance are at least distance 8 apart, the level of smallest box is $O(\log n)$. Thus it is not difficult to see that the total number of boxes in quadtree is $O(n \log n)$.

We introduce randomness into the subdivision by "shifting" the center of dissection. If the side length of the bounding box is L, the original center of the dissection is $(\frac{L}{2}, \frac{L}{2})$. To shift the dissection, we pick a, b from $[0, L)$ randomly, and start dissection at center $(\frac{L}{2} + a (\text{mod } L), \frac{L}{2} + b (\text{mod } L))$. We call dissections and quadtrees created in this way (a,b)-dissections and (a,b)-quadtrees. See Figure 3 for an example.

1.3 The Structure Theorem

Given the subdivisions of the plane, we can now begin discussing how to use them to find a tour that is near optimal. To do this, for each box in the (a,b)-quadtree, we put m regularly spaced points on each side and one at each corner. We call these points *portals.* The main idea that enables us to carry out dynamic programming is that we will look for tours that cross the sides of boxes *only* through portals.

Let's define such a tour. We say we have a *salesman path* if we have a connected set of edges such that degree at portals is even (possibly 0), and degree at nodes is 2. We know from previous discussion of the TSP that if we find a short salesman path, then by shortcutting the portals we can find a tour of cost no greater. We define an (m,r)-*light salesman path* to be a salesman path that crosses each side of each box at most r times, and crosses the the sides of boxes only at portals.

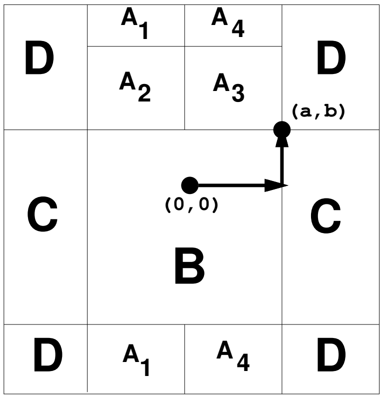

Figure 3: Example of the beginning of an (a,b)-dissection. Areas labelled with the same letter are conceptually the same box.

We then have the following theorem about (m,r)-light salesman paths. We defer the proof for the time being.

Theorem 1 (Structure Theorem, Arora '97) *Pick $a, b \in [0, L)$ at random. Then with probability at least $\frac{1}{2}$, the (a,b)-dissection has an (m,r)-light salesman path of cost no more than $(1+\epsilon)OPT$, where $m = O(\frac{1}{\epsilon}\log L)$ and $r = O(\frac{1}{\epsilon})$.*

1.4 Applying Dynamic Programming

Since we know by the Structure Theorem that there is an (m,r)-light salesman path of cost no more than $(1+\epsilon)OPT$ with probability at least $1/2$, we now apply dynamic programming to find the least cost (m,r)-light salesman path, then convert this into a tour of no greater cost. Then with probability at least $1/2$ we will have found a tour of cost at most $(1+\epsilon)OPT$.

Now to define the dynamic program. By the definition of an (m,r)-light salesman path, we know for every box that the path crosses each side of it at most r times. Given some Eulerian tour of the salesman path, we can think of the tour entering the box, visiting some of the nodes inside it, then exiting, visiting points elsewhere, re-entering, and so on. Thus for each square, we know that the salesman path

- Uses some even number of portals (possibly repeated), no more than $4r$.
- These portals are paired into entry/exit pairs.

Inside each box, given the portals used and the entry/exit pairing, the cheapest (m,r)-light salesman path uses cheapest paths in the box to visit all nodes in that box.

Thus our dynamic program is as follows: for each box in the (a,b)-quadtree, for each choice of up to r portals per side, for each pairing of these portals into entry/exit pairs, we find the cheapest way to visit all the nodes in the box. We build up a table containing an entry for each box, choice of portals, and pairing of portals, and use some of the entries to help us find the solutions for other entries.

First we calculate the size of the table we will need. There are $O(n \log n)$ boxes in the quadree, $(4m+4+1)^{4r}$ choices of up to r portals on each side of the box (including not choosing any portals) for each box, and $(4r)!$ possible pairings of the portals into entry/exit pairs. So the table has

$$O(n \log n) \times (4m+4+1)^{4r} \times (4r)! = O(n \log^{O(\frac{1}{\epsilon})} n) \text{ entries.}$$

Now we discuss how to build up the table. Our base case considers the boxes at the "lowest" level of the quadtree (i.e. boxes that do not contain any boxes inside of them). This case is fairly simple: for each choice of portals and pairing of portals, we find the shortest paths that enter/exit the box in the designated way and visit the 1 node inside the box. The inductive case builds up a solution for a box B from the solutions for the four boxes $b_1, \ldots, b_4$ it contains. Note that the paths visiting the nodes in B may use portals on the four boundaries of the b_i that are not also a boundary of B. Call these boundaries the "internal" sides of the b_i. To combine the solutions from the b_i to get a solution for box B, we enumerate over all the portals on the internal sides that the paths might have used, and the order in which these portals were visited by the paths, and pick the best solution we find. Notice that specifying a set of portals used on the internal sides and an order in which they are used implies for each b_i a set of portals which are used, in addition to an entry/exit pairing on the portals. Thus we can simply look up the best solution for this subproblem for each b_i from our table and combine them to get a solution for B. We can pick up to r portals from each of the four internal sides (which is no more than $(m+5)^{4r}$ possibilities) specify which of the $2r$ paths for B on which these portals lie (which is no more than $(2r)^{4r}$ possibilities), and the ordering of these portals on the pahts (no more than $(4r)!$ possibilities). Thus it takes at most $O((m+5)^{4r}(2r)^{4r}(4r)!) = O(\log^{O(\frac{1}{\epsilon})} n)$ time to compute the answer for each entry in the table. Therefore, the overall time taken is $O(n \log^{O(\frac{1}{\epsilon})} n)$.

The overall solution to our problem is found in the table entry corresponding to the bounding box with no portals chosen.

1.5 Proving the Structure Theorem

We now turn to the proof of Theorem 1. The basic idea is to modify the optimal tour to be (m,r)-light salesman path. We will show that modifications don't cost much. To prove this, we first need to show that if the tour crosses a line too much, we can change it at a small increase in cost so that it doesn't cross the line very much.

Lemma 2 (Patching Lemma) *Given a line segment S of length ℓ, if a tour crosses S three or more times, we can add line segments on S of length no more than $6l$ to get Eulerian tour that contains the previous tour, and crosses S at most twice.*

Proof: We take the tour, and break it at the points at which it crosses line S. We put new points just to the "left" and "right" of S where the tour crossed S; see Figure 4.

We now add a tour and a matching to the points on the left side of the line, and a tour and matching to the points on the right side of the line, plus edges connecting the last one or two pairs of points. See Figure 5 for an example. This gives an Eulerian graph which contains the previous tour and crosses S at most twice. Each tour added has cost at most $2l$ and each matching has cost l, for an overall cost of $6l$.

□

We now apply the patching lemma to the various "grid lines" in the dissection (i.e. the lines parallel with the sides of the bounding box) in order to make the tour into an (m, r)-light salesman path. We need to do this in a particular order, which we give in the following lemma.

Lemma 3 *Given line ℓ from random (a, b)-dissection, the expected cost of making tour (m, r)-light on line ℓ is $\frac{13t(\ell)}{r}$, where $t(\ell)$ is number of times that the optimal tour crosses line ℓ.*

Proof: Define the *level* of line ℓ to be the minimum level over all boxes such that ℓ contains a side of the box. Then observe that since we chose the center of the dissection uniformly at random that this implies that

$$\Pr[\text{level of } \ell \text{ is } i] = \frac{2^i}{L},$$

that is, the probability it has level 0 is $1/L$, etc.

The first thing we need to do is to move all the points at which the optimal tour crosses ℓ to a portal. A box at level i has portals at distance $\frac{L}{2^i m}$, so that the expected increase in the tour for moving the crossings to the nearest portal is

$$\begin{aligned}
\sum_{i \geq i} \Pr[\text{level of } \ell \text{ is } i] \cdot t(\ell) \cdot \frac{L}{2^i m} &= \sum_{i \geq 1} \frac{2^i}{L} \cdot t(\ell) \cdot \frac{L}{2^i m} \\
&= \frac{t(\ell) \log L}{m} \\
&\leq \frac{t(\ell)}{r-1},
\end{aligned}$$

for $m = O(r \log L)$.

We now invoke the patching lemma to make sure that for every side of every box which ℓ contains, it is not crossed more than r times. We consider a vertical line ℓ

(the horizontal case is similar), and apply the following procedure to modify ℓ, given that it has level i and we have an (a,b)-dissection:

Modify(ℓ, i, b)

For $j \leftarrow \log L$ downto i
For $p \leftarrow 0$ to $2^j - 1$
If segment of ℓ from $b + p \cdot \frac{L}{2^j} \bmod L$ to $b + (p+1) \cdot \frac{L}{2^j} \bmod L$
is crossed more than r times
Apply patching lemma to cross segment twice

Let c_j denote the number of times the patching lemma is applied in iteration j. Then $\sum_{j \geq 1} c_j \leq t(\ell)/(r-1)$, since each time the patching lemma is invoked, it replaces at least $r+1$ crossings with at most 2. Then the total expected increase in the cost of the tour due to the patching of ℓ is

$$\begin{aligned}
&\sum_{i\geq 1} \Pr[\text{level of } \ell \text{ is } i] \cdot \text{Increase in cost due to } \mathbf{Modify}(\ell, i, b) \\
&\leq \sum_{i \geq 1} \frac{2^i}{L} \sum_{j \geq i} c_j \cdot 6 \cdot \frac{L}{2^j} \\
&= 6 \sum_{j \geq 1} \frac{c_j}{2^j} \sum_{i \leq j} 2^i \\
&= 6 \sum_{j \geq 1} 2c_j \\
&= \frac{12t(\ell)}{r-1}.
\end{aligned}$$

Thus the total expected increase in cost for making the tour (m,r)-light on line ℓ is

$$\frac{t(\ell)}{r-1} + \frac{12t(\ell)}{r-1} \leq \frac{13t(\ell)}{r-1}.$$

□

Let T be the sum of $t(\ell)$ over all horizontal and vertical grid lines ℓ. The above lemma tells us that the overall expected increase caused by transforming the optimal tour to an (m,r)-light salesman path is at most $\frac{13}{r-1}T$. If we set $r \geq \frac{52}{\epsilon} + 1$, then with probability at least $1/2$, the cost is at most $\frac{\epsilon}{2}T$. The final lemma shows that $T \leq 2OPT$, which completes the proof of the Structure Theorem.

Lemma 4 $T \leq 2OPT$.

Proof: Consider an edge e in the optimal tour from point (x_1, y_1) to (x_2, y_2). The edge contributes at most $|x_1 - x_2| + |y_1 - y_2| + 2$ to T, and has length $s = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. But

$$|x_1 - x_2| + |y_1 - y_2| + 2 \leq \sqrt{2[(x_1 - x_2)^2 + (y_1 - y_2)^2]} + 2$$

$$
\begin{aligned}
&\leq \sqrt{2s^2} + 2 \\
&\leq 2s,
\end{aligned}
$$

since (by our perturbation of the instance) $s \geq 4$. Thus, summing over all edges in the optimal tour, $T \leq 2OPT$. $\square$

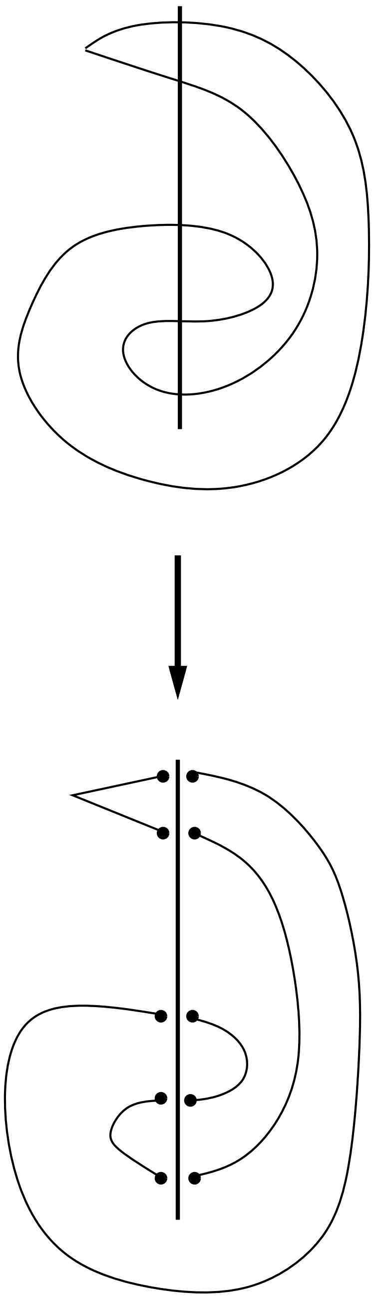

Figure 4: Breaking the tour where it crosses the line.

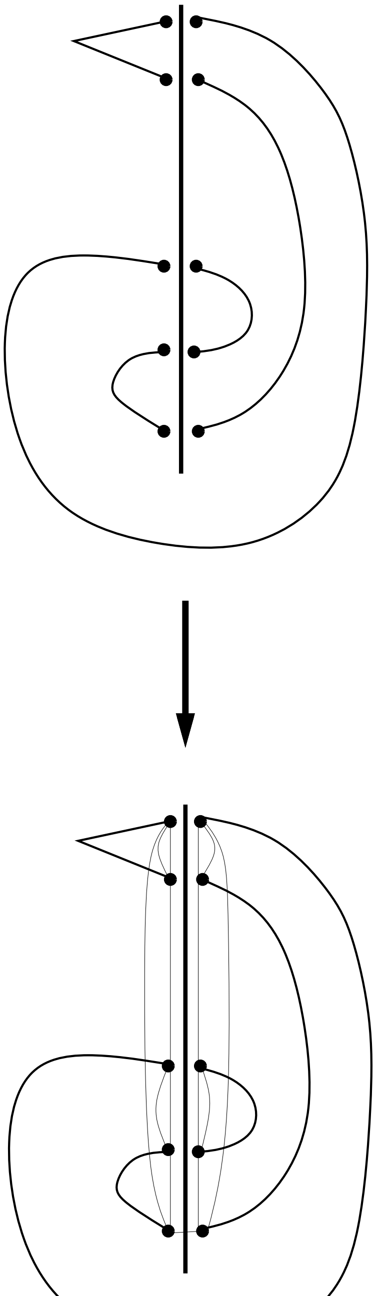

Figure 5: Adding tours and matchings to obtain Eulerian tour.