18.415/6.854 Advanced Algorithms November 12, 1996

Lecture 15

Lecturer: Michel X. Goemans *Scribe: Salil Vadhan*

1 Arora's $(1+\epsilon)$-Approximation Scheme for the Euclidean TSP

In the last lecture, we saw that, given any n points in $[0,1]^2$, Karp's partitioning algorithm finds a tour whose length is longer than the optimal tour by an additive factor of at most $o(\sqrt{n})$. In this lecture, we will see another partioning-based algorithm, due to Arora [1], which finds a tour of length at most $(1+\epsilon)OPT$ in time $n^{O(1/\epsilon)}$ for any $\epsilon > 0$. A similar result was also obtained by Mitchell.

First, we observe that we can make some simplifying assumptions about the locations of our cities. Given any finite set S of points in the plane, the *bounding box* of S is the smallest rectangle (with sides parallel to the coordinate axes) which contains all of S. The *size* of a rectangle is the length of its longest side.

Claim 1 *Suppose there is an approximation scheme*[1] *for instances of the Euclidean TSP*[2] *in which the n cities lie in a bounding box of size at most n^2 and the distance between any two cities is at least 1, or 0. Then there is an approximation scheme for all instances of the Euclidean TSP.*

Proof: Clearly, we can scale to make the bounding box size n^2 without changing the relative lengths of any tour. Moreover, if the bounding box was the smallest possible, the length OPT of the optimum tour is at least $2n^2$ (since there must be cities on the 2 shortest sides). Now, consider the instance obtained by rounding the coordinates of every city to the nearest integer. In the rounded instance, the distance between any two distinct cities is at least 1. Moreover, in the transformation, every city has moved by at most $\sqrt{2}/2$. For any tour T, let $l(T)$ and $l'(T)$ denote its length in the original and transformed instances respectively. Since we can perform an excursion from the new location to the old location of any city (or vice versa), we derive that

$$|l(T) - l'(T)| \leq \sqrt{2}n. \tag{1}$$

Thus, if we have a $(1+\epsilon')$-approximation algorithm for the rounded instance, we get that the tour produced T has a length satisfying

$$l(T) \;\leq\; l'(T) + \sqrt{2}n \leq (1+\epsilon')OPT' + \sqrt{2}n \leq (1+\epsilon')(OPT + \sqrt{2}n) + \sqrt{2}n$$

[1] By approximation scheme, we mean a $1+\epsilon$-approximation algorithm for any fixed $\epsilon > 0$.

[2] We always mean the Euclidean TSP in *two* dimensions.

$$
\begin{aligned}
&= (1+\epsilon')OPT + (2+\epsilon')\sqrt{2}n \le (1+\epsilon')OPT + (2+\epsilon')\sqrt{2}\frac{OPT}{2n} \\
&= \left(1+\epsilon' + (2+\epsilon')\frac{\sqrt{2}}{2n}\right) OPT \le (1+\epsilon)n,
\end{aligned}
$$

whenever, for example, $\epsilon' = 0.8\epsilon$, $\epsilon < 1$, and $n \geq 10/\epsilon$ (but remember ϵ is fixed so this last condition is not a problem). □

The partitioning in Arora's algorithm works differently than in Karp's algorithm. Arora's algorithm looks at what is called a *"1/3: 2/3-tiling"*. Informally, a 1/3: 2/3-tiling is a recursive partitioning of the bounding box into smaller rectangles by horizontal and vertical lines so that each line partition divides the longest side s of the corresponding rectangle into two pieces whose length is at least 1/3 the length of s. We also require that each vertical (resp. horizontal) line partition goes "slightly" to the left or to the right (resp. slightly above or below) of a city. Slightly, for example, can be understood as off by $\delta/2$ where δ represents the smallest gap between two distinct $x-$ or $y-$coordinates of cities. The goal of this is to allow only a small number of possible $x-$ and $y-$coordinates for line partitions (namely $4n$), while avoiding the problem of having to assign cities that fall right on a line partition to either side of it. However, it may not be possible to have such a line partition and satisfy the 1/3: 2/3 requirement. This is the reason for the somewhat complicated definition that follows.

Definition 1 *Let R be a rectangle in $\mathbb{R}^2$ with sides parallel to the coordinate axes. Let s be the longest side of R and let $|s|$ be its length. A* **line-partition** *of R is a partitioning of R into two smaller rectangles by a line-segment perpendicular to s and at a distance of δ of a city. A line-partition P is* **valid** *iff one of the following conditions holds:*

1. *P divides s into two segments, each of length at least $|s|/3$.*
2. *There are no line-partitions satisfying (1) and P can be moved to the center of the rectangle without crossing over any cities.*

*A 1/3: 2/3-***tiling** *of a rectangle is a recursive partitioning of a rectangle by valid line-partitions until there is at most one city in the interior of any rectangle. (See Figure 1.)*

Figure 1 gives an example of a 1/3: 2/3-tiling. Notice that line partition 3 does not divide the longest side of the corresponding rectangle into pieces of relative length at least 1/3.

How deep can the recursion in a 1/3: 2/3-tiling be? Note that the sizes of the rectangles need not decrease at each stage, because the first partition of a square produces two rectangles of the same size as the original square. Also, if the cities are all near the sides of the rectangle, no valid line-partition will cut the longest side s into pieces of length at least $|s|/3$. However, after at most four levels of recursion,

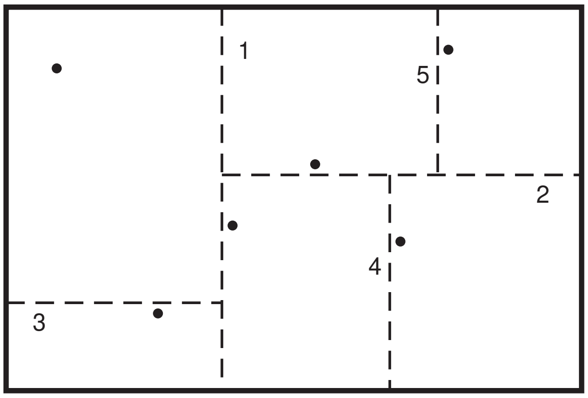

Figure 1: A 1/3: 2/3-tiling

all the remaining rectangles will either have size at most $2|s|/3$ or will have no cities in their interior. Observe that a rectangle of size $1/\sqrt{2}$ has at most one point in its interior. Thus the recursion has depth at most $4\log_{3/2}(\sqrt{2}n^2) + 1 = O(\log n)$.

The key theorem underlying Arora's algorithm will tell us that there is a near-optimal tour that doesn't cross any of the partitions we made too often and that we can find such the best such tour quickly. In order to make this precise, we need to specify the points at which we will allow crossings. Let $m = \lceil c \log n/\epsilon \rceil$ for some constant c to be specified later. Divide each line segment drawn in our partition into m equally-sized segments. We call the midpoints of these segments *portals*. We will look at tours which cross our partition only at portals.

Definition 2 *A tour is called* m-**light** *(with respect to a given tiling) iff the following conditions hold:*

1. *Each line partition is crossed at most m times and only at portals.*
2. *The tour is not self-crossing, but it may meet itself at portals.*

To clarify Part 2 of the above definition, see Figure 2. We allow a tour which goes from city a to city b and from city c to city d via portal p; this is an example of a tour which meets itself at a portal. However, we do not allow a tour which goes from city a to city d and from city b to city c both via portal p; this is a self-crossing. Figure 3 gives an example of a 3-light tour.

The main theorem behind Arora's algorithm is the following:

Theorem 2 *1. There exists a $1/3\colon 2/3$-tiling and an m-light tour whose length is at most $1+\epsilon$ times the length of the optimal tour.*
2. The best m-light tour over all $1/3\colon 2/3$-tilings can be found in time $n^{O(1/\epsilon)}$ by dynamic programming.

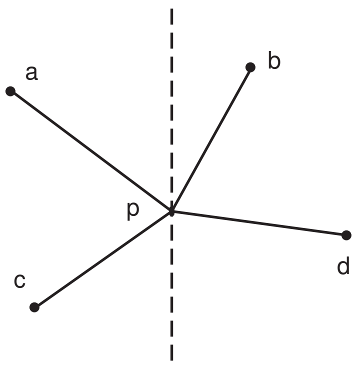

Figure 2: Crossings

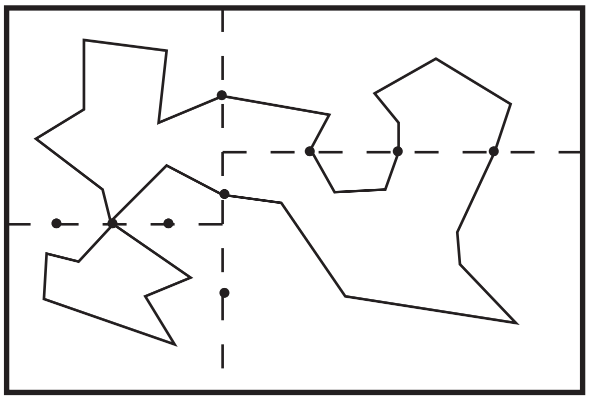

Figure 3: A 3-light tour

Proof: We first prove part 1. We start with the empty-tiling and the optimum tour and calculate how much the length of the tour must increase as we modify it to be m-light at each successive refinement of the partition. Consider a single rectangle R at some stage of our partition. Let T be the total length of the tour within R and let W be the size of R. We treat the problem in two cases:

Case 1: $T \leq mW/3$. Let $\mu = 3T/W \leq m$. Consider a random line segment which cuts the longest side of R into a fraction selected uniformly between 1/3 and 2/3. Then

$$E[\#crossings] \leq \sum_e \frac{l_e}{W/3} = \frac{3T}{W} = \mu$$

where the sum is over all segments e of the tour which lie in R and l_e denotes the length of e. So there exists a line segment with at most $\mu \leq m$ crossings. We may slide this segment to the nearest city on either side without changing the number of crossings, so there exists a valid line-partition with at most μ crossings. Now, moving each crossing to the closest portal, the length of the tour is increased by at most $\mu W/m = 3T/m$. Thus the new tour is longer than the old tour by a factor of at most $1 + 3/m$.

Case 2: $T > mW/3$. Take any valid line partition of R. The tour might cross this partition many times, say k times. However, we can use the same trick as in Karp's partitioning algorithm to reduce the number of crossings to 2 while not increasing the length much: Split the ith crossing point into two points, i and i', one on each side of the boundary. Now connect all these (split) crossing points in a loop that only crosses the line partition between 1 and 1' and between k and k'. Add another k edges which pair up consecutive points in this loop using the shorter of the two possible perfect matchings. (See Figure 4.) Now the tour, along with these additional

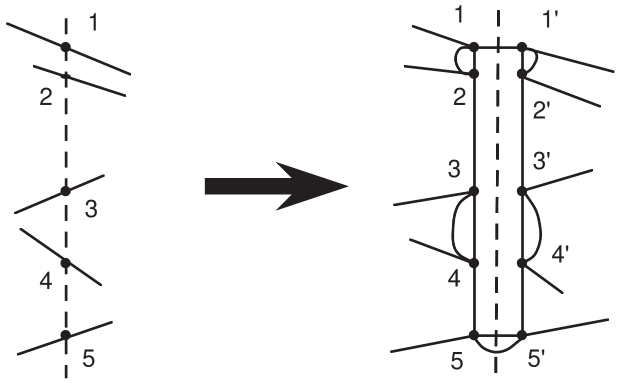

Figure 4: Short-circuiting the crossings

edges forms a connected graph in which every node has even degree. So there is an Eulerian path, *i.e.* a tour which uses every edge exactly once. This is longer than the original tour by at most $3W$ (since the line partition has length at most W) and it crosses the line partition at most four times. These crossings can occur in only two places (between 1 and $1'$ and between k and k'), so, if there are more than two edges crossing, one of these points must be crossed at least twice. Removing two of the crossings at one of these points still leaves a connected Eulerian graph (the degrees of two nodes have each been reduced by two), so we can find a new Eulerian path (at no additional cost), which crosses the line partition at most twice. Now we need to move these crossings to portals, which increases the tour length by at most $2W/m$. Our tour length has increased by at most $3W + 2W/m \le (9 + 6/m)T/m \le 10T/m$ (as long as $m > 6$, but this is okay since $m \to \infty$ as $n \to \infty$.) So the length of the tour within R has increased by a factor of at most $(1 + 10/m)$.

There is one small problem — the tour might now cross itself. First we deal with crossings that occur at points other than portals. We can introduce a fictitious pair of nodes at each self-crossing and short-circuit as above to remove the crossing. For example, if the tour goes from city a to city b and from city c to city d and these two paths cross at point x, then we introduce two fictitious nodes x_1 and x_2 at point x. We then either (1) connect x_1 to a and c and x_2 to b and d, or (2) connect x_2 to a and d and x_2 to b and c. One of these two possibilities must yield a connected Eulerian graph, so there is a new tour which does not cross at x. In the first case, this tour goes from a to x to c and from b to x to d. We no longer need x as an intermediate point, because we can go directly from a to c and from b to d, resulting in a strictly shorter tour. However, this may introduce other crossings, in which case we repeat. Since there are only finitely many possible tours on n nodes and since the length strictly decreases during the short-circuiting, this process must terminate, resulting in a tour that does not meet itself, except at portals.

To deal with crossings at portals, we do the same thing, introducing a fictitious node and short-circuiting. However, we cannot remove the portal as an intermediate point since we may only cross line-partitions at portals. Doing this for all crossings at portals, we obtain a tour which does not cross itself anywhere and meets itself only at portals.

If we perform this procedure on each rectangle at some stage of the refinement process, we increase the total length by a factor of at most $(1 + c_1/m)$ for $c_1 = 10$. Thus the total increase over all levels is a factor of $(1 + c_1/m)^{O(\log n)} \leq e^{\epsilon/2} \leq 1 + \epsilon$, for $\epsilon < 1$, if we choose the constant c in the definition of m appropriately. This proves part 1 of the theorem.

Now we need to show how to find the best m-light tour. This is done by dynamic programming. The idea is to show that we can break the problem of finding the best m-light tour into $n^{O(1/\epsilon)}$ subproblems and that we can solve each subproblem efficiently given solutions to all of its subproblems. First we describe and enumerate all the subproblems.

Every line-partition in a $1/3\colon 2/3$-tiling occurs at an $x \pm \delta$ or $y \pm \delta$ coordinate of a city. Consider all the rectangles defined by these coordinates. There are at most $\binom{2n}{2}^2 = O(n^4)$ such rectangles. Each side of each rectangle comes from some line partition, so there are at most $\binom{2n}{2}$ ways to place portals on any side of a given rectangle. This makes for at most $\binom{2n}{2}^4 = O(n^8)$ possible portal placements on each rectangle. Any m-light tour must cross the boundary of the rectangle in $2k \leq 4m$ places (at most m times for each side). The number of ways of choosing $2k$ objects out of a set of $4m$ objects with replacement is

$$\left(\binom{4m}{2k}\right) = \binom{4m + 2k - 1}{2k} \leq \binom{8m}{2k}.$$

So the total number of ways of choosing the places a tour crosses the boundary of a given rectangle with given portal placements is at most

$$\sum_{k=1}^{2m} \binom{8m}{2k} \leq 2^{8m}.$$

Given a choice of $2k$ portals in which the tour crosses some rectangle, there are a number of ways these crossings can be paired up to specify which portal the tour exits from after each entry. The number of pairings which can be made with non-crossing curves is equal to the number of properly parenthesized expressions consisting of k left-parentheses and k right-parentheses. This is well-known to be $\binom{2k}{k}/(k+1)$, the kth Catalan number, which is bounded above by $2^{2k} \leq 2^{4m}$.

A single subproblem consists of a rectangle with portal placements, and a specification of which portals are being used and how they pair up. We have shown that there are $O(n^{12}2^{12m}) = n^{O(1/\epsilon)}$ subproblems. We now describe how to solve each subproblem optimally, *i.e.* find the shortest set of paths connecting the given pairs of portals and collectively visiting all the cities within the rectangle.

Solving the subproblems corresponding to the rectangles with at most one city inside can be done trivially (since there are at most $2m$ different ways of visiting the city inside, if any).

To find the optimal solution to a larger subproblem P, consider all possible line partitions L of the rectangle into two smaller rectangles R_1 and R_2. Then consider all possible ways for the tour to cross L and all possible ways to match these crossings with the pairing of portals specified by P. Each possibility defines subproblems P_1 and P_2 for R_1 and R_2, respectively. Solutions to the subproblems P_1 and P_2 yield a solution to P whose cost is the sum of the costs for P_1 and P_2. The optimum solution for P is simply the minimum of this over all the aforementioned possibilities. By analysis, similar to the one done above, there are only $n^{O(1/\epsilon)}$ possibilities to consider (since there are at most $n^{O(1/\epsilon)}$ possibilities for P_1 and P_2). Thus we can, by dynamic programming, compute the optimum for all the subproblems in time $n^{O(1/\epsilon)}$. $\square$

References

[1] S. Arora, "Polynomial-time Approximation Schemes for Euclidean TSP and other Geometric Problems", *Proc. of the 37th Annual Symposium on the Foundations of Computer Science*, 2–11, 1996.