CS880: Approximations Algorithms	
Scribe: Dave Andrzejewski	**Lecturer:** Shuchi Chawla
Topic: Euclidean TSP (contd.)	**Date:** 2/8/07

Today we continue the discussion of a dynamic programming (DP) approach to the Euclidean Travelling Salesman Problem (TSP). Finally, a randomized modification in introduced which acheives an expected approximation factor of $1 + 2\epsilon$.

6.1 Euclidean TSP

6.1.1 Intro

As introduced in the previous lecture, the Euclidean TSP invovles finding the minimum cost tour of a set of points in a plane with a Euclidean metric.

The approaches to Bin-Packing and Knapsack discussed in the previous lecture work by first transforming or restricting the original problem, then using DP to obtain an exact solution to the transformed problem. This is the strategy we will apply to Euclidean TSP.

First, we will assume without loss of generality that the points are contained within a minimum bounding box with side length n^2. Second, we will assume all points have integer coordinates. It can be shown that this modification can add no more than $2\sqrt{2}n$ to the total length of an optimal tour in the transformed instance (see previous lecture). Also, note that this rounding may 'collapse' several nearby points to the same coodrdinates. This means that the distance between any two points in our tranformed instance must be either 0 or ≥ 1.

6.1.2 Divide anhd conquer

Our DP approach divides the minimum bounding box into 4 equal subboxes. These boxes are further subdivided, and so on. Following the standard DP strategy, we will find optimal subpaths, join them together, and then move up to the next level until we have a full solution.

Our key problem then is determining how to combine these subpaths. Obviously we cannot simply consider all possible start/end point pairs within each subpath. The solution is to subdivide each box into 4 boxes using 4 lines we will call the "level i lines", where i is the current recursion level. On each of these lines we then place $2m$ equidistant "portal" points. We then restrict our attention to paths which only cross level i lines through these portals.

Definition 6.1.1 *A tour is* portal proper *if it only crosses level i lines through the portal points.*

We now place further restrictions on the paths we will consider.

Definition 6.1.2 *A tour is* proper *if*

1. *it is portal proper*
2. *it crosses each portal ≤ 2 times*

3. it only self-crosses at portals

We need to show that these new restrictions do not affect the length of a portal proper tour.

Lemma 6.1.3 *If T is a portal proper tour, then we can find a proper tour T' that is not longer.*

Proof: First, we must show how any self-crossing can be eliminated without any additional length. A diagram of a simple example shows how any crossover can be removed, resulting in a path that is no longer than the original.

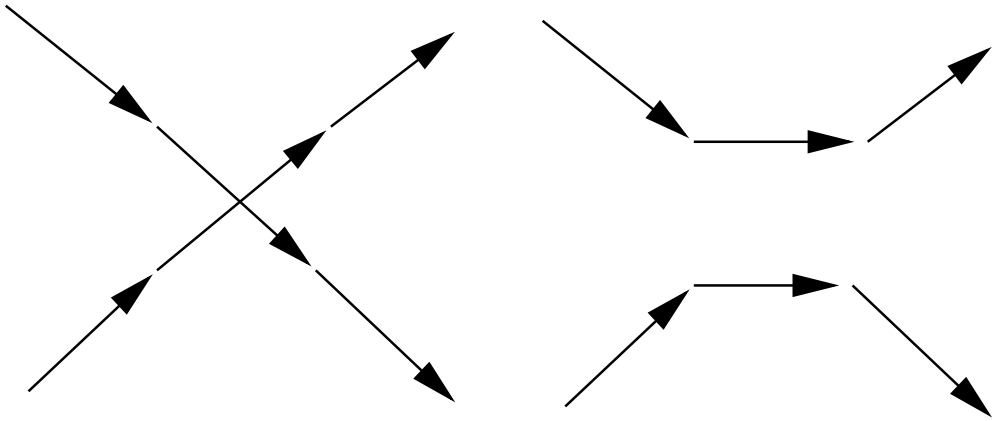

Figure 6.1.1: A self-crossing can be removed without additional length.

Second, we must show how any path which crosses a portal > 2 times can be modified to cross ≤ 2 times without additional length. This can be accomplished by having paths simply "turn back" instead of going through the portal again. Any odd number of portal crossings can be reduced to a single crossing, and any even number of crossings can be reduced to two, as shown below.

■

6.1.3 DP for proper tours

Now we need to analyze the DP approach for finding the optimal proper tour. The key questions here are:

1. how long to solve one subproblem?
2. how many subproblems have we created?

6.1.3.1 Subproblem size

First we will consider the number of possible solutions for a single subproblem. For a given ϵ, let us choose m to be a power of 2 in the range

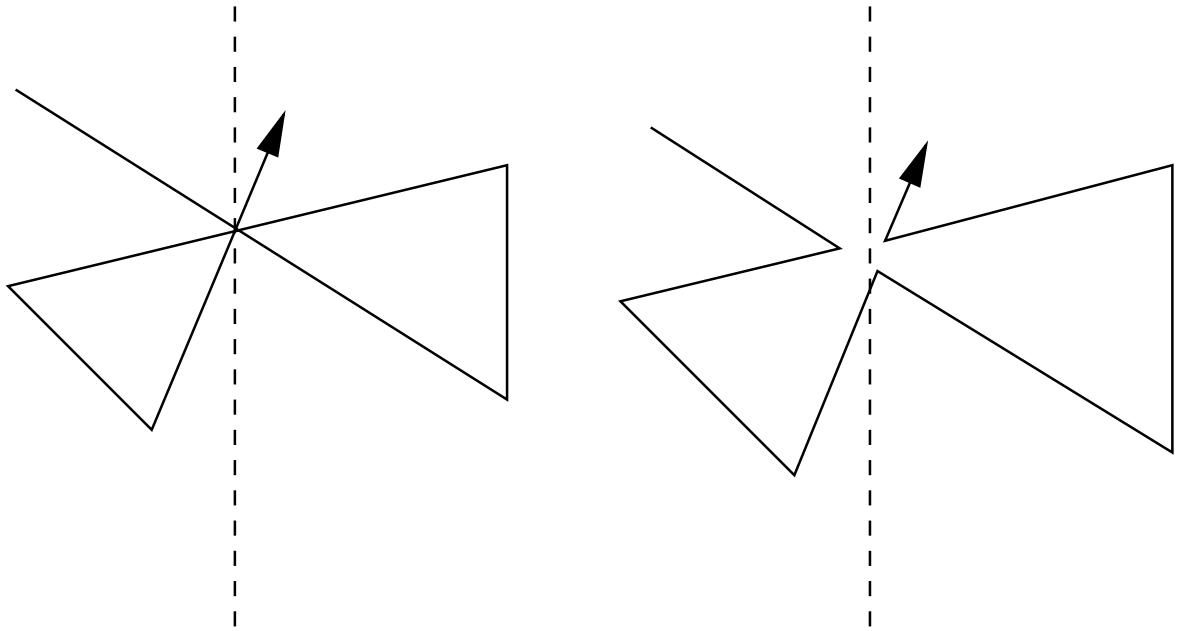

Figure 6.1.2: An odd number of portal crossings reduced to a single crossing.

$$\left[\frac{\log n}{\epsilon}, \frac{2\log n}{\epsilon}\right] \tag{6.1.1}$$

Then for any given box, the maximum number of portals on a side is m. The maximum number of sides with portals is 4, so the total number of portals on the box is then $\leq 4m$. Since we are restricting ourselves to paths that only use each portal 0,1, or 2 times the total number of portal usage assignments $3^{4m} = n^{O(\frac{1}{\epsilon})}$. Since every path which enters the box must also leave it, we can throw out all portal usage assignments which sum to an odd number of portal crossings. Say a given portal usage assignment uses $2r$ portals.

Given the portal usage assignment, how many possible paths are there? The "no self-cross" property of a proper tour allows us to further constrains the number of allowable paths. Within a single box, there is a bijection between portal matchings which satisfy the "no self-cross" constraint and balanced arrangements of parentheses. This is most easily seen by arbitrarily choosing a 'starting' portal on the box and going around the box clockwise to order the portals. Once a path 'enters' through a portal, any paths entering 'afterwards' must exit before the first path does. Portals used twice can be considered equivalent to two adjacent portals for the purposes of this analysis.

Starting from the lower left-hand portal and going clockwise, the diagram below illustrates a portal matching equivalent to the parenthesization "(()) (() ())".

The number of balanced parenthesizations for r pairs of parentheses is equal to the r^{th} Catalan number [2], which is bounded from above by $2^{2r} = n^{O(\frac{1}{\epsilon})}$.

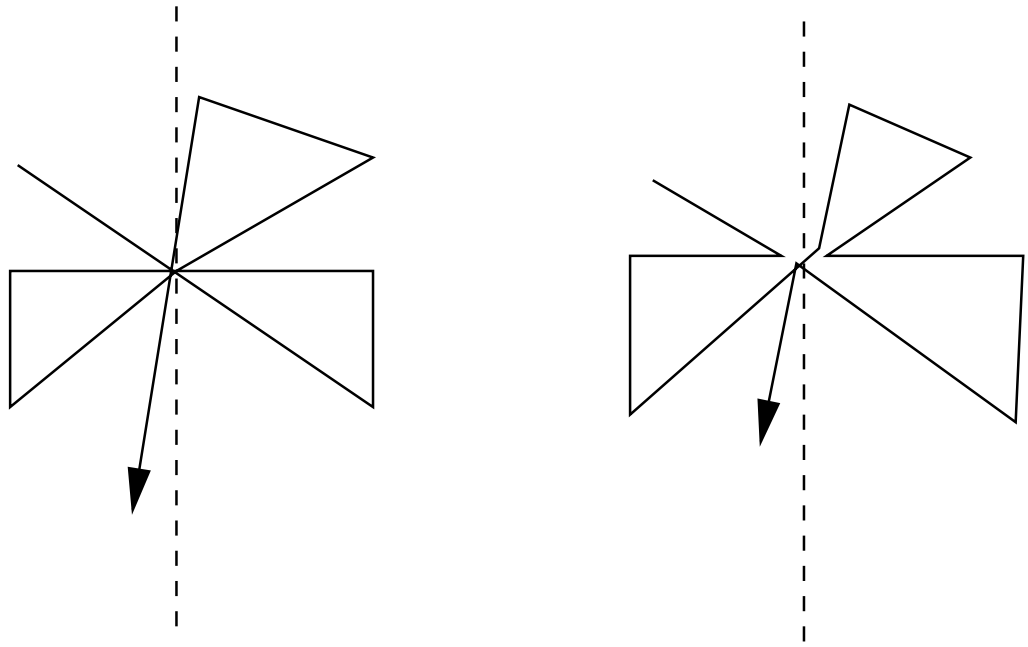

Figure 6.1.3: An even number of portal crossings reduced to two crossings.

This finally tells us that the total number of possible portal proper paths for a given box is $n^{O(\frac{1}{\epsilon})}$.

6.1.3.2 Number of subproblems

We now need to determine how many subproblems we have created. Each division creates 4 boxes, so the total number of problems is given by 4^L where L is the number of recursion levels.

Since we are only allowing integer coordinates, and our minimum bounding box has side length n^2, this means that the depth of our recursion is $L = 2 \log n$. Combining this with the previous result gives us $4^{2 \log n} = n^4$ squares at the lowest level level of recursion. This result also could have been arrived at by noticing that we have restricted the problem the integer coordinates and that our minimum bounding square has side length n^2. Since the smallest boxes cover exactly one point, we get n^4 one-point boxes.

6.1.3.3 Putting it all together

For any given box, we have $n^{O(\frac{1}{\epsilon})}$ valid portal matchings (a 'visit'). For a given visit, the portal usages on the outer portals are fixed, so we then need to consider only the compatible visits for the 4 sub-boxes. Since each sub-box has $n^{O(\frac{1}{\epsilon})}$ valid visits, there are still only $n^{O(\frac{1}{\epsilon})}$ sets of 4 visits to consider, only some of which will be compatible with the larger box portal matching and with each other. The cost of a compatible set of 4 visits is then the sum of the optimal visit costs for each sub-box, which have already been calculated at the lower level of recursion. The cost of the lowest cost path for is then stored as the optimal cost for this particular visit of the larger box.

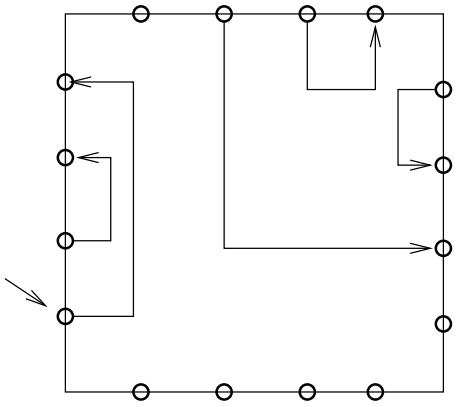

Figure 6.1.4: A portal proper path.

This process is started at the leaves and then proceeds up the tree. Combining the per subproblem cost with the total number of subproblems gives us a final result of $n^{O(\frac{1}{\epsilon})}$ for the cost of our optimal DP algorithm for proper tours.

6.1.4 Proper tour vs OPT

Now that we know we have a polynomial time algorithm for computing optimal proper tours. But how much worse is the optimal proper tour than OPT?

To determine this, we construct a worst case OPT tour which has to go out of its way to go through portals. Assuming that we are making a detour at each of the $2m$ portals and that each detour adds $\leq \frac{n^2}{2m}$, then we are adding $\approx n^2$ total distance beyond OPT. Since we defined n^2 to be the side length of the minimal bounding box, $2n^2$ forms a lower bound on OPT. In our worst-case scenario, saying that $OPT \approx 2n^2$ and our detours add $\approx n^2$ means that we have a constant-factor approximation.

6.1.5 Randomized version

We can see that the worst-case example was very specific to our division line placement. This provides the intuition for a randomized approach, which acheives an expected cost within a $(1+2\epsilon)$ multiple of OPT.

The randomized version makes the top-level divisions at an offset from the $n^2/2$ center lines. The offsets are chosen uniformly. The rest of the algorithm is unchanged, giving virtually identical analysis.

The one difference is that we can now compute the expectation of the number of line corssings.

Lemma 6.1.4 *$E(c_i) = OPT\frac{2^{i+2}}{M}$ where c_i is the number of times the optimal path crosses a level i line, and M is the length of the level i line.*

Proof: Divide the OPT path into segments with length δ. The probability that a segment crosses

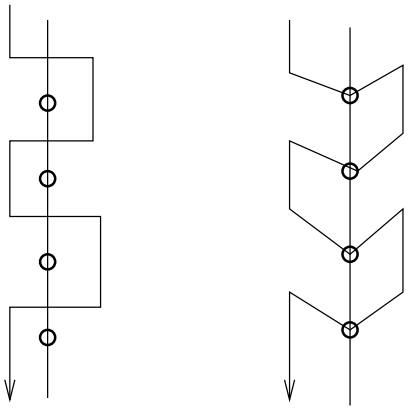

Figure 6.1.5: A worst-case portal proper path.

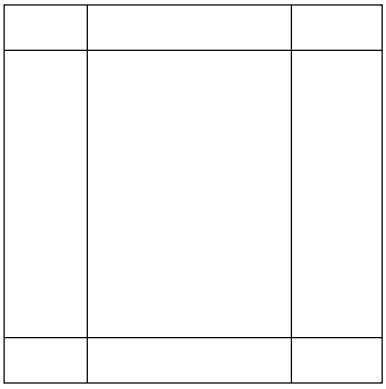

Figure 6.1.6: Randomized level 1 division.

the vertical level i line is then $\leq \frac{2\delta}{M}$. The expected number of segments which cross the vertical level i line is then given by

$$\frac{OPT}{\delta}\frac{2\delta}{M} = \frac{2OPT}{M} \tag{6.1.2}$$

Since the same analysis holds for the horizontal line, the expected number of crossings is then $\frac{4OPT}{M}$.

To take the crossings at each lower level into account, notice that each lower level has twice as many lines as the level above it.

The expected number of crossings at level i is then equal to the our previous calculation times 2^i.

$$2^i(\frac{4OPT}{M}) = \frac{2^{2+i}OPT}{M} \tag{6.1.3}$$

■

Now that we have the expected number of crossings, we can compute the expected additional detour cost. Our definition of m with respect to $\log n$ will now play a crucial role.

Theorem 6.1.5 *An OPT tour can be made proper at an increased length cost* $\leq 2\epsilon OPT$.

Proof: First we need to calculate the worst-case distance from a crossing point to a portal on a level i line. Since on level i we have $2^{i+1}m$ portals per line, the distance must be $\leq \frac{M}{2^{i+1}m}$.

Now we can see that even though the expected number of crossings grows exponentially with i, the distance to the nearest portal shrinks exponentially with i. The expected extra distance d necessary to go through the portals can now be calculated.

$$E(d) \leq \sum_i \frac{M}{2^{i+1}m} \frac{2^{2+i}OPT}{M} = \frac{OPT}{m} 2 \log n \tag{6.1.4}$$

We recall that $m \propto \log n$, and substitute in our definition of m.

$$E(d) \leq 2\epsilon OPT \tag{6.1.5}$$

■

It is now clear that our randomized version has expected cost arbitrarily close to OPT.

$$E(ALG) \leq (1 + 2\epsilon)OPT \tag{6.1.6}$$

Now that we have a bound on the expected error, we can use the Markov inequality to get a probability bound on how much worse our solution is than OPT. This probability can then be made arbitrarily small by repeatedly running the randomized algorithm.

A de-randomization of this algorithm is also possible [1].

References

[1] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. In *Journal of the ACM*, 1996.

[2] R. A. Brualdi. Introductory Combinatorics, 4th ed. New York: Elsevier, 1997.