Polynomial-time Approximation Scheme for Euclidean TSP

1 Introduction

We present a surprising result that the traveling salesman problem has a polynomial-time approximation scheme when the distances between cities are Euclidean. This result, independently due to Arora and Mitchell [1, 6], hinges on a powerful technique for randomly decomposing the given graph into regions, such that dependency amongst optimal solutions to each region has a "nice" structure. In fact, the technique generalizes to yield similar approximation schemes for other geometric optimization problems, many of which prior to this result lacked even a constant factor approximation algorithm.

The lecture is organized as follows: In section 2 we give a brief introduction to the background necessary including a definition of the traveling salesman problem, and the variants mentioned in this lecture. In section 3 we give an outline of the basic steps of the graph dissection and approximation scheme, including a description of the dynamic program for finding the best of the restricted tours that we consider. In section 4 we give a proof of the structure theorem which states that the best restricted tour we find in section 3 in fact has cost at most $(1+\epsilon)OPT$ where OPT is the cost of the optimal TSP tour. Finally, in section 5 we list some generalizations and extensions of the techniques involved in the PTAS, as well as some other related results.

In addition, we would like to note that these notes have been designed to be a relatively complete exposition on Arora's PTAS for the Euclidian TSP. We have filled in some of the details that were left as exercises or observations in the primary sources, and as such it is unlikely that all these details could be presented in a single lecture, nor should they. However, we have also attempted to provide enough intuition so that the innovative ideas of this result clearly stand out from those that are purely technical. In particular, we believe the following three concepts are of highest importance:

1. the randomized decomposition of the graph using the quadtree.
2. restricting the paths into and out of a square to a fixed set of points (directly yielding an efficient dynamic program for the restricted problem).
3. that the optimal locally-restricted paths are actually close to the global unrestricted optimum (*i.e.* that the structure theorem is true).

We hope our attempts towards this goal prove successful.

2 Traveling Salesman Problem

2.1 General TSP

The basic problem is given a weighed graph $G = (V, E)$, find the shortest Hamiltonian tour. A Hamiltonian tour is a path that visits each city exactly once and then returns to the city of origination. In the general case, there are no restrictions on the cost function used for defining the weights on each edge. In this general form, it is well know that there cannot be a polynomial-time approximation algorithm achieving a tour within any polynomial factor for this problem, unless **P** = **NP** (see for example [8] for details). More can be said however, when we impose some structure on how the distances are chosen.

2.2 Metric TSP

This problem has the same setup as the above general case, but in these instances the nodes lie in a metric space. That is, we assume the edge weights are symmetric, and that that the triangle inequality holds, so the $d(x, z) \leq d(x, y) + d(y, z)$ must be true for all pairs of x and z.

There exists a $\frac{3}{2}$-factor approximation for this problem using the Christofides algorithm (again see [8] for details). This problem was also shown to be MAX-SNP complete [7], so it is known that there is no PTAS solving general Metric TSP unless $\mathbf{P} = \mathbf{NP}$ [3].

2.3 Euclidean TSP

This is a special case of the metric TSP where the cost function is the Euclidean distance in $\Re^d$. The vertices now correspond to points in a d-dimensional space. The Euclidean distance between two points $x = (x_1, x_2, \ldots, x_d)$ and $y = (y_1, y_2, \ldots, y_d)$ is defined as:

$$(\sum_{i=1}^{d}(x_i - y_i)^2)^{\frac{1}{2}}$$

As we will show in these notes, there is a PTAS for Euclidean TSP. It is also known that there does not exists a FPTAS for this problem unless $\mathbf{P} = \mathbf{NP}$. [4]

3 PTAS

Let us first begin by providing a definition for a Polynomial Time Approximation Scheme.

Definition 1 *A PTAS (Polynomial Time Approximation Scheme) for an optimization problem is an approximation algorithm A that takes as input an instance of the problem and a value $\epsilon > 0$, and runs in time polynomial in n and returns a solution s that satisfies: $f(s, I) \leq (1+\epsilon) \cdot OPT(I)$, i.e. it is a $(1 + \epsilon)$-approximation algorithm.*

For the Euclidean TSP, we show a randomized PTAS that for every fixed $c > 1$ a computes a tour that with probability at least $1/2$ is a $(1+\frac{1}{c})$-approximation to the optimal tour. Furthermore, this algorithm runs in $O((n \log n)^{O(c)})$ time. When the nodes are in $\Re^d$, the running time rises to $O((n \log n)^{(O(\sqrt{d}c))^{d-1}})$. Note also that this algorithm can be derandomized, adding a factor $O(n^d)$ to the running time.

The input for the algorithm is a set of n points given by coordinates in $\Re^d$. The cooridnates and edge costs are truncated to their $2 \log n$ most significant digits. This affects all edge costs (as well as the optimal tour cost) by at most a multiplicative factor of $(1 + \frac{1}{n^2})$, which is negligible in the context of an approximation scheme.

3.1 Algorithm

A large portion of the lecture will be devoted to proving the following theorem, but for now we assume it to be true to consider the algorithmic consequences.

Theorem 1 (Structure Theorem) *Let $k, \epsilon > 0$, be constants. Then there exists a salesman tour that crosses every grid line at most k times, such that these boundary crossing occur only at a fixed set of entrance/exit points called* portals*, and furthermore the cost of this restricted tour is at most $(1 + \epsilon) \cdot OPT$ where OPT is the cost of the optimal non-restricted salesman tour.*

Assuming the *structure theorem* given above, the PTAS for Euclidian TSP in $\Re^2$ is as follows:

1. Perturbation. This is done to make the instance well-rounded. A well-rounded instance has the following characteristics:

   - All nodes have integral coordinates
   - Each (nonzero) internode distance is at least 2 units
   - The maximum internode distance is $O(n)$.

   These changes cause a small impact on the cost of the optimal tour and are achieved using the following procedure:

   Let the bounding box be the smallest axis-parallel square which contains all of the the nodes for the given instance. Assume that $\epsilon > 1/n^{\frac{1}{3}}$. (This is a reasonable assumption since we know that ϵ is a fixed constant independent of the input size.) Let w be the maximum distance between any two nodes for the instance. Denoting the optimum tour cost by OPT, we know that $2w \leq OPT \leq nd$, where d is the dimension of the problem (d = 2 for our description of the algorithm). Now let us create a grid where each of the lines in the grid are separated by a distance of $\epsilon w/n^{1.5}$. On the new grid, we will move each node to its nearest gridpoint. Note that on the new grid, some nodes will be mapped to the same location. In this case, we will treat these nodes as a single node in the algorithm and accomodate for this change when determining the final tour. Finally, rescale the distances so that the minimum internode distance is at least 2. Now we have created a bounding box with sidelength of at most $n^2/2$.

2. Constructing a Shifted Quadtree (Randomized Dissection). Let $P \in \Re^2$ be the lower left endpoint of the bounding box and let all the sides have length l. We now create the enclosing box which encloses the bounding box and has sidelength $L = 2l$. The enclosing box is positioned around the bounding box such that P has a distance of a from the left edge and b from the bottom edge, where $a, b \leq l$ are chosen randomly. Let us define a as the horizontal shift and b as the vertical shift. Once the enclosing box is established, the randomized dissection is performed to create the quadtree. The randomized dissection is the recursive partitioning of the enclosing box into smaller squares. This partitioning stops when there is at most one node in each box. Note that the randomness is only used to determine where the enclosing box is placed and has no impact on the input nodes or the unit grid.

3. Dynamic Programming. The dynamic programming stage is used to find the optimal k-light m-portal respecting salesman path with respect to the shifted quadtree which was just found in the previous step. Note that $m = O(\log n/\epsilon)$ and $r = O(1/\epsilon)$. Because of the choice of the shift used to construct the quadtree, with probability $\frac{1}{2}$ the salesman path will have cost at most $(1 + \epsilon) \cdot OPT$. The dynamic programming step builds a lookup table in a bottom-up fashion that contains the costs of optimal solutions to all subproblems. Since the optimal (m, k)-light salesman path occurs in one of the entries of the table, the algorithm is finished when the table is built. The efficiency of this procedure, as we will see, is a direct result of the fact that we restrict the TSP tour to enter and exit a grid square only at the portals.

3.2 Perturbation

In the algorithm above, we move each of the instance nodes to a gridpoint when creating the perturbed instance. These moves to the gridpoints cause each node to be moved by at most $\epsilon w/n^{1.5}$. The affect that these moves have on the optimal tour costs are at most $\epsilon w/n^{0.5}$, which is negligible compared to ϵOPT when n is large.

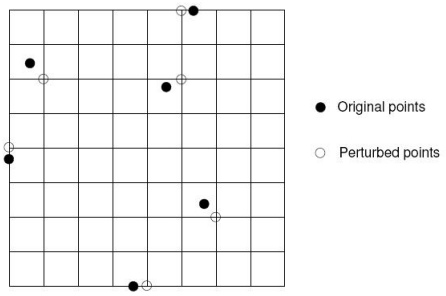

Figure 1: Example of perturbed points.[9]

We must also give some attention to when we have rescaled the instance after moving the points to ensure that the maximum internode distance is at least 2. On the new grid, the maximum internode distance between any two points became $n^{1.5}/\epsilon$. This value is asymptotically less than $n^2/2$, which satisfies the desired sidelength of the bounding box.

The perturbation of the instance makes all coordinates intergal and the minimum node distance at least 2. This can be performed in $O(n \log n)$-time.

3.3 Constructing a Shifted Quadtree

In this portion of the algorithm, we create an enclosing box with a random (a, b)-shift. For an example of the way the bounding box lies within the enclosing box, see Figure 2. Once the enclosing box is established, the dissection is performed to create the quadtree. A dissection of a square is the recursive partitioning into smaller squares. The partitioning of a square stops when it has size ≤ 1. Note that there are $O(L^2)$ squares in the dissection and its depth is $\log L$. In our algorithm, we are trying to create the quadtree for the enclosed box, which stops the partitioning of a given square when there is at most one node. This results in fewer squares than a dissection and $O(n)$ leaves with $O(n \log L)$ squares in all. Examples of the dissection and quadtree are given in Figure 3, which clearly depicts the difference in creating a dissection versus a quadtree.

Now, let us discuss in further detail some of the properties encountered during the creation of the quadtree. We will assume without loss of generality that L, the sidelength of the enclosing box, is a power of 2. This assumption is made so that the squares in the dissection have integer endpoints and the leaf squares have sidelengths of 1. Therefore, we know that the depth of the dissection is at most $\lceil \log L \rceil = O(\log n)$.

To further understand the process of the dissection, let us make a few observations concerning the nature of the dissection. Let the level of a square in the dissection be its depth from the root. In the example in Figure 4, the root has level 0. Each horizontal and vertical grid line can be assigned a level between 1 and $\log L$. Figure 4 shows the relationship between the dissecting grid

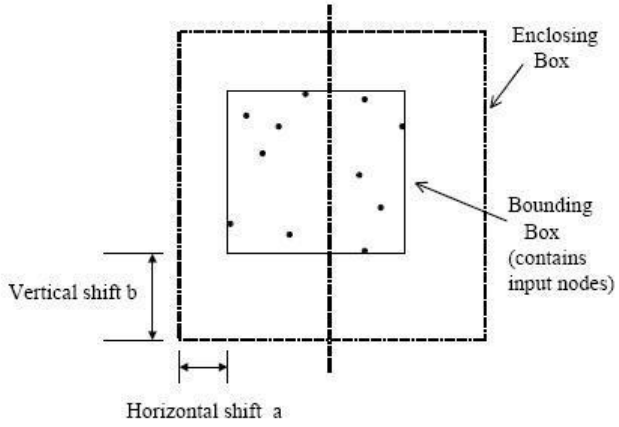

Figure 2: The enclosing box contains the bounding box and is twice as large, but is shifted by random amounts in the x and y directions.[2]

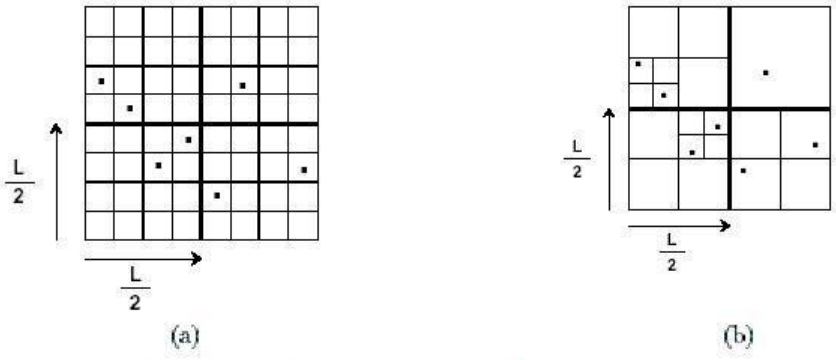

Figure 3: (a) The dissection. (b) The corresponding quadtree.[1]

lines and the squares which correspond to each level. In general, the number of lines found at level i is 2^i.

Using this knowlegde of the random dissection, now consider any fixed vertical grid line that intersects the bounding box of the instance. The probability that this line has level i in the randomized dissection is:

$$Pr[\text{line is at level i}] = \frac{2^i}{l} = \frac{2^{i+1}}{L} \tag{1}$$

Therefore, in an expected sense, all the grid lines are treated symmetrically in the randomized dissection.

3.4 Portal-respecting tours

Portals are the special points on each of the grid lines which are used in a portal-respecting tour. A level i line has $2^{i+1}m$ equally spaced portals inside the enclosing box, where m is the portal parameter and is required to be a power of 2. The corners of each square is also referred to as a portal. Recall the relationship between level i lines and the number of squares touching it: each level i line has 2^{i+1} level $i+1$ squares touching it. Using this property, we know that each side of the square can have at most $m+2$ portals and the boundaries may contain at most a total of

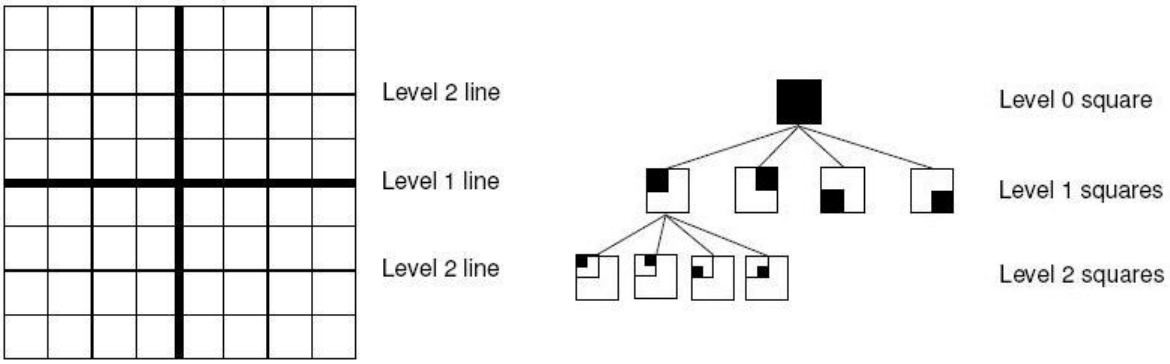

Figure 4: 4-way recursive partitioning of an $L \times L$ grid and dissection tree.[9]

$4m + 4$ portals. The portal-respecting tour is a tour which when crossing a grid line only does so at a portal. Note that when a portal-respecting tour crosses a grid line it may have to deviate from the straight-line path between nodes, as shown in Figure 5.

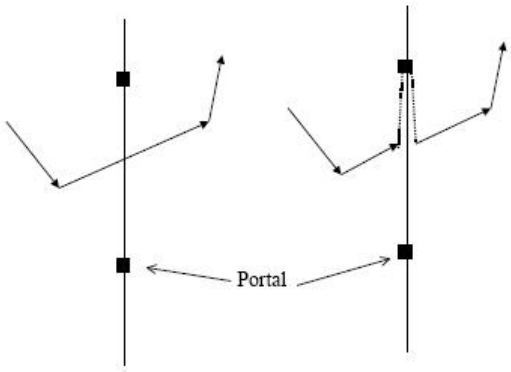

Figure 5: Detouring[2]

A portal respecting tour is called k-light if it crosses each side of each dissection square at most k times. By removing repeated visits to any portal, we can conclude that the optimal portal-respecting tour does not need to visit any portal more than twice. This can be easily shown using the trianlge-inequality to verify that the removal of repeated visits can never increase the cost of the tour. Therefore, the optimal portal-respecting tour is found to be $(m + 2)$-light. Figures 6 and 7 give examples of a portal-respecting tour and shortcutting to reduce the portal visits to at most two.

3.5 Finding The Min-Cost k-light Portal-Respecting Tour

We now give a dynamic program that finds the cost of an optimal k-light portal-respecting tour in time $O(poly(m^{O(k)}) \cdot L \log L)$, where m is the portal parameter. Furthermore, we will see in the proof of the structure theorem in section 4 that for our purposes $m = O(\log n/\epsilon)$ and $k = O(1/\epsilon)$ suffices, so for these parameters we can find the optimal cost tour in $O(n(\log n)^{O(1/\epsilon)})$ time.

To motivate the design of the dynamic program, we first consider the structure of an optimal k-light portal-respecting tour with regards to a single square S in the dissection. Suppose that we

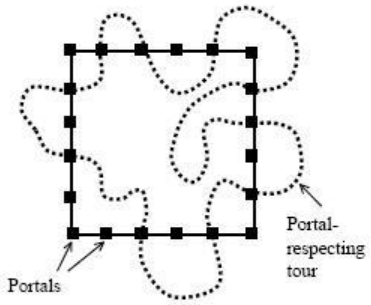

Figure 6: This portal-respecting tour enters and leaves the square 10 times, and the portion inside the square is a union of 5 disjoint paths.[2]

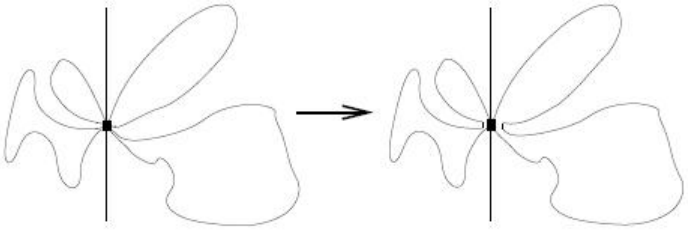

Figure 7: An example of how shortcutting can make any tour visit a portal at most twice.[9]

are given a sequence $\{a_i\}_{1\leq i\leq 2p}$ of portals on S for $2p \leq 4k$ such that the total number of times any portal from side i of S appears in the sequence is at most k. Furthermore, suppose we are told that an optimal k-light portal-respecting tour crosses the boundary of S in the order given by the sequence. That is there exists an optimal k-light portal-respecting tour, such that there are p paths $P_1, \ldots, P_p$ of the tour inside S where P_i connects $\{a_{2i-1}, a_{2i}\}$ and the set of paths visit all nodes inside S. Then finding an optimal "tour" inside S can be done independently of the optimal "tour" outside S as long as the order of crossings into and out of the boundary of S is preserved[1]. This independence given a fixed "interface" to the boundary allows us to construct the desired dynamic program.

Using the intuition from above, we can build a table T that is indexed by

1. A (non-empty) square S in the quadtree,
2. A set of four multi-sets[2] $E = \{E_1, E_2, E_3, E_4\}$ of portals on each side of S, where each E_i contains at most k portals and $|E_1| + |E_2| + |E_3| + |E_4| = 2p \leq 4k$,
3. A set of pairings $A = \{\{a_1, a_2\}, \{a_3, a_4\}, \ldots, \{a_{2p-1}, a_{2p}\}\}$ of the $2p$ portals from E_1,E_2,E_3,E_4.

The entry at position $T[S][E][A]$ contains the minimum cost[3] (over the Euclidian norm) of any set of paths $P_1, \ldots, P_p$ that visits all the nodes in S and is such that P_i connects a_{2i-1} to a_{2i}.

[1]Note that we are no longer looking for a TSP tour in these subproblems, but rather we are searching for an optimal set of paths $P_1, \ldots, P_p$ that cover the nodes inside S, and are such that P_i connects a_{2i-1} and a_{2i}.

[2]A *multi-set* is a standard set with repetition allowed.

[3]If the optimal set of paths is desired the completed table can be traversed backwards to reconstruct each decision made, as is typical of many dynamic programs.

Definition 2 *Given a square S in the quad-tree, let E be a set of 4 multi-sets of portals of S and let A be a set of pairings of portals in the sets of E which together satisfy (2.) and (3.) above. Then we call the pair (E, A) an* interface *to S.*

3.6 The Dynamic Program

Initialization: The table will be filled in a bottom-up fashion, starting first with squares S that are leaf nodes in the quad-tree. For any non-empty S that is a leaf of the quadtree, S must contain exactly one input-node, say v. If we fix any interface (E, A) to S, to find the value $T[S][E][A]$ let $P_1, \dots P_p$ be a set of paths within S respecting (E, A). If these paths are to be part of a valid optimal tour, then v must be on some path P_i from $P_1, \dots P_p$, and further, by short-cutting we may assume that all other paths P_j for $j \neq i$ are "straight-line" connections between a_{2j-1} and a_{2j}. If follows that

$$T[S][E][A] = \min_i(|P_1| + |P_2| + \cdots + |P_p| : P_i = a_{2i-1}va_{2i}, P_j = a_{2j-1}a_{2j} \forall j \neq i).$$

Recursion: Suppose inductively that we know all entries of the table for any square of level at least $i + 1$ of the quadtree. Then consider a square S at level i for a fixed interface (E, A) to S. We wish to compute table entry $T[S][E][A]$. Let S^1, S^2, S^3, S^4 be the children of S in the quadtree. Observe that a partial interface to each of $S^1, \dots, S^4$ has been specified by the interface to S (an interface on internal grid edges of each S^j has yet to be specified). However since S^j is at level i, given any interface (E^j, A^j) to S^j we know the optimal cost $T[S^j][E^j][A^j]$ by induction. In this way, the dynamic program iterates over all choices for interfaces $(E^1, A^1), \dots, (E^4, A^4)$ to $S^1, \dots, S^4$ respectively, that are also feasible[4] with respect to the interface (E, A) to S. Then the table entry is give by

$$T[S][E][A] = \min \left(\sum_{j=1}^{4} T[S^j][E^j][A^j] \quad : \quad (E^j, A^j) \text{ is an interface of } S^j \text{ and } \{(E^j, A^j)\}_{j=1,\dots,4} \text{ is feasible with respect to } (E, A) \right).$$

Global Optimum: Once the entire table is built the global optimum can be computed by finding the minimum of all entries $T[R][\cdot][\cdot]$ where R is the root of the quadtree (*i.e.* the enclosing box). In fact, this can be done with a single query to the table; the minimum will always be at the $T[R][\emptyset][\emptyset]$ entry, since using any portals on the boundary of the enclosing box can only increase the length of the tour (recall the enclosing box strictly contains the bounding box).

3.6.1 Efficiency

The total number of entries in the table is

$$\begin{aligned} (\text{size of the table}) &= (\#\text{ of quadtree squares})(\#\text{ of interfaces to a square}) \\ &\leq O(L \log L) \cdot m^{O(k)} \cdot k! \end{aligned}$$

[4]By *feasibility* of $\{(E^j, A^j)\}_{j=1,\dots 4}$ with respect to (E, A) we mean that if H_j is a set of paths satisfying (E^i, A^i), then $\bigcup_{j=1}^{4} H_j$ is a set of paths satisfying (E, A).

and furthermore, to compute each table entry for a square S, we query at most

$$\begin{aligned}(\#\text{ of interfaces to } S_1,\ldots,S_4) &\leq (\#\text{ of multisets})(\#\text{ of pairings of portals})\\ &\leq ((m+4)^k)^4(4k)^{4k}(4k)!\end{aligned}$$

so in all the dynamic program runs in time $O(poly(m^{O(k)})\cdot L\log L)$, which for parameters $m = O(\log n/\epsilon)$ and $k = O(1/\epsilon)$, we get an $O(n(\log n)^{O(1/\epsilon)})$ procedure for finding the cost of an optimal k-light portal respecting tour.

Comment 1 One should note how essential the requirement that the salesman tour only crosses the square boundaries at portals is to the efficiency of the procedure. Without such a restriction, we would not have an efficient way to enumerate all possible paths into and out of a region.

4 Proof of the Structure Theorem

In this section, we will prove the following theorem:

Theorem 2 (Structure Theorem - restated) *Let OPT be the optimal TSP tour, and let $OPT_{a,b,k,m}$ be the optimal k-light m-portal respecting tour, given shifts a and b. Then $E_{a,b}[OPT_{a,b,k,m} - OPT] \leq (\frac{2\log L}{m} + \frac{12}{k-5})OPT$.*

Before we begin the proof, observe that for $m > \frac{8\log n}{\epsilon}$ and $k > \frac{48}{\epsilon+5}$ we get that $2\left(\frac{2\log L}{m} + \frac{12}{k-5}\right) = 2\left(\frac{\epsilon}{4} + \frac{\epsilon}{4}\right) = \epsilon$. Then by Markov's inequality we get

$$\begin{aligned}\Pr_{a,b}[OPT_{a,b,k,m} \leq (1+\epsilon)\cdot OPT] &= \Pr_{a,b}[OPT_{a,b,k,m} - OPT \leq \epsilon\cdot OPT]\\ &= \Pr_{a,b}\left[OPT_{a,b,k,m} - OPT \leq 2\left(\frac{2\log L}{m} + \frac{12}{k-5}\right)\cdot OPT\right]\\ &= 1 - \Pr_{a,b}\left[OPT_{a,b,k,m} - OPT > 2\left(\frac{2\log L}{m} + \frac{12}{k-5}\right)\cdot OPT\right]\\ &\geq 1 - \frac{1}{2} \quad = \quad \frac{1}{2}.\end{aligned}$$

so this algorithm does indeed yield the desired randomized PTAS for Euclidian TSP.

We will use the following theorems and lemmas in the proof:

Lemma 1 *Let a, b be random shifts. Let $d_{a,b,m}(u,v)$ be the shortest portal-respecting distance between u, v when all intermediate grid lines have to be crossed at portals, and let $d(u,v)$ is the Euclidean distance between u, v. Then $E_{a,b}[d_{a,b,m}(u,v) - d(u,v)] \leq \frac{2\log L}{m}\cdot d(u,v)$, where L is the sidelength of the enclosing box.*

Proof: Because calculating the exact expectation is difficult, we will provide an upper bound for the expectation. Let us consider a tour to prove this upper bound. Note that the best portal-respecting path may not be the same one that we are considering, but since we are only interested in the upper bound this is ok. We know that the straight line from u to v crosses the unit grid at most $2d(u,v)$ times. As described above in the portal-respecting tour section, we move each

crossing of the grid line to the nearest portal to obtain a portal-respecting path (see Figure 5). For any given grid line i which we may cross, it is known that the interportal distance is $L/m2^{i+1}$. By (1), we know that the probability of the given line being at level i is $2^{i+1}/L$. Therefore, the expected length of the detour is at most

$$\sum_{i=0}^{\log L-1} \frac{2^{i+1}}{L} \cdot \frac{L}{m2^{i+1}} = \frac{\log L}{m}.$$

This same upper bound holds for each of the $2d(u,v)$ times a crossing occurs, so by the linearity of expectations it is clear that the expected increase in the tour by moving the crossings to the nearest portal is at most $2d(u,v)\frac{\log L}{m}$. ■

Theorem 3 *$E_{a,b}[OPT_{a,b,m} - OPT]$ is at most $\frac{2\log L}{m} \cdot OPT$, wher L is the sidelength of the enclosing box and the expectation $E_{a,b}[\cdot]$ is over the random choice of shifts a, b.*

Note that $OPT_{a,b,m}$ represents the cost of the best portal-respecting tour when the portal parameter is m and the random shiffts are a, b.

Proof: This follows directly from Lemma 1 by the linearity of expectations since the tour length is the sum of all the edge lengths.

$$\sum_{(u,v)\in T} 2d(u,v)\frac{\log L}{m} = \frac{2\log L}{m} OPT,$$

where T is the optimal tour. ■

Lemma 2 *(Patching Lemma) Let S be any line segment of length s and π be a closed path that crosses S at least three times. Then we can break the path in all but two of these places, and add to it line segments lying on S of total length at most $3s$ such that π changes into a closed path π' that crosses S at most twice.*

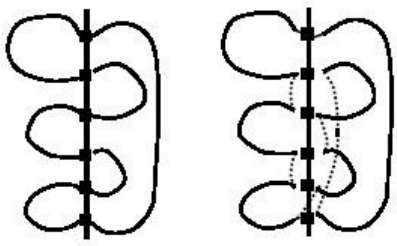

Figure 8: This is an example of how patching can reduce the number of crossing to two.[2]

Proof: To make the proof easier, we will consider segment lengths of $6s$ instead of $3s$, however note that the proof considering segments of length $3s$ can be performed using Christofiides heuristic.

Suppose that π crosses the line segment S a total of t times. Denote the points at which π crosses

S as $M_1, \ldots, M_t$. We can create t paths $P_1, P_2, \ldots, P_t$ by breaking π at each of the $M_1, \ldots, M_t$ points. For the rest of the proof, we will need a copy of these points for each side of S, so let us refer to them by $M_i^{'}$ and $M_i^{''}$.

Now let $2j < t$ be the largest even number possible. Let J be the multiset of line segments consisting of the following:

1. A minimum cost salesman tour through the points $M_1, \ldots, M_t$.
2. A minimum cost perfect matching amoung $M_1, \ldots, M_{2j}$.

The line segments of J all lie on S and their total length is at most $3s$. We will now add two copies of J to π, denoted by $J^{'}$ on the left side of S and $J^{''}$ on the right side of S.

We must now consider the following two cases based on if t is odd or even:

1. If $t = 2j + 1$ (t is odd), add an edge between $M_{2j+1}^{'}$ and $M_{2j+1}^{''}$ with length of zero.
2. If $t = 2j + 2$ (t is even), add an edge between $M_{2j+1}^{'}$ and $M_{2j+1}^{''}$ with length of zero and an edge between $M_{2j+2}^{'}$ and $M_{2j+2}^{''}$ with length of zero.

If we add these new edges and the lines segments in J to the paths $P_1, P_2, \ldots, P_t$, a connected 4-regular graph on $\{M_1^{'}, \ldots, M_t^{'}\} \cup \{M_1^{''}, \ldots, M_t^{''}\}$ has been formed. By taking an Eulerian traversal of this graph, a closed path that contains all points $P_1, P_2, \ldots, P_t$ and crosses S at most twice is defined. Thus proving the case for $6s$. ■

Comment 2 The patching lemma is a very simple idea with an intuitive geometric proof, yet amazingly this simple idea is the key insight behind the proof of the structure theorem.

In addition to the above, we use the following:

Lemma 3 *Let l be a grid line in the dissection, and let π be a TSP tour of length $length(\pi)$. Denote the number of times π crosses l by $t(\pi, l)$. Then*

$$\sum_{l:vertical} t(\pi, l) + \sum_{l:horizontal} t(\pi, l) \leq 2 \cdot length(\pi).$$

Proof: We show that any length s edge of π contributes at most $2s$ to the left hand side. Let h and v be the lengths of the horizontal and vertical projections of s. Since $h^2 + v^2 = s^2$ it follows that s can contribute at most $(h + 1) + (v + 1)$ to the left hand side. Then we get

$$h + v + 2 \leq \sqrt{s(h^2 + v^2)} + 2 \leq \sqrt{2s^2} + 2.$$

Then Since $s \geq 4$ we have $\sqrt{2s^2} + 2 \leq 2s$ as desired. ■

Proof of The Structure Theorem: Intuitively to prove the theorem we will start with an optimal TSP tour π, and we will transform π into another tour that is k-light. For every grid line that is crossed more than k times by π, we will reduce the number of crossings with the patching lemma. This will increase the cost of the tour, but we can use Lemma 2 to bound this cost increase.

First we recall the structure of the dissection. Let l be a vertical grid line (analogous facts hold true about horizontal grid lines). If l is at level i then it is adjacent to 2^{i+1} level $i+1$ squares, and in general for $j > i$, l is adjacent to 2^j level j squares, each of length $L/2^j$. We call a segment of l that is adjacent to a level j square a *level j segment*. For $s = k - 4$, we say a segment of l is *overloaded* if the tour crosses it more than $s + 1$ times.

The transformation itself is natural. First start at level $\log L - 1$. For each level $\log L - 1$ segment that is overloaded, use iteratively the patching lemma until there are only 2 crossings over the segment. Next, on this new tour, for each overloaded level $\log L - 2$ segment, again use the patching lemma to reduce the number of crossings to 2. Proceed in this way until there are no overloaded segments of l. Finally, adjust the remaining crossings so that the tour is portal respecting (by adding a vertical detour just before and after the crossing point to the nearest portal).

Let $X_{l,j}(b)$ be the random variable denoting the number of overloaded level j segments of l. For every vertical shift b,

$$\sum_{j \geq 0} X_{l,j}(b) \leq \frac{t(\pi, l)}{s-1}.$$

To see this, note that the left hand side counts the number of times the patching lemma is applied. Since there are $t(\pi, l)$ crossings in all, and with each application of the patching lemma we are removing at least $s - 1$ crossings, the result holds. Furthermore, since each level j segment has length $L/2^j$, with each application of the patching lemma on level j, we add a cost of at most $L/2^j$ to the tour. Thus the additional cost incurred by transforming crossings of l is at most $\sum_{j \geq 1} X_{l,j}(b) \cdot 3L/2^j$.

However, we implicitly assumed in the above analysis that l was a level 0 line, which it need not be. Assuming that L is in fact a level i grid line, then the increase caused by this transformation is at most $\sum_{j \geq i+1} X_{l,j}(b) \cdot \frac{3L}{2^j}$. In fact, the level of l is a random variable that depends on a, the horizontal shift. Recall $\Pr_a[\text{line } l \text{ is level } i] = \frac{2^{i+1}}{L}$. Then we define a random variable $Y_{l,b}(a)$ as the cost to transform π to a k-light tour across l when random shifts are a and b.

Then for every choice of b, we have

$$\begin{aligned}
E_a[Y_{l,b}(a)] &= \sum_{i \geq 1} \frac{2^{i+1}}{L} \cdot (\text{charge to } l \text{ when its level is } i) \\
&\leq \sum_{i \geq 1} \frac{2^{i+1}}{L} \cdot \sum_{j \geq i+1} X_{l,j}(b) \cdot \frac{3L}{2^j} \\
&= 3 \cdot \sum_{j \geq 1} \frac{X_{l,j}(b)}{2^j} \cdot \sum_{i \leq j-1} 2^i \\
&= 3 \cdot \sum_{j \geq 1} \frac{X_{l,j}(b)}{2^j} \cdot (2^j - 1) \\
&\leq 3 \cdot \sum_{j \geq 1} X_{l,j}(b) \\
&\leq 3 \cdot \frac{t(\pi, l)}{s-1}
\end{aligned}$$

Summing over all vertical lines l, we get the $E_a\left[\sum_l Y_{l,b}(a)\right] \leq \sum_l \frac{3t(\pi,l)}{s-1}$ by linearity of expectation. Since a similar analysis can be done for all horizontal lines, by Lemma 2 we get that

$$\begin{aligned}(\text{expected cost to make } \pi \text{ } k\text{-light}) &\leq \frac{6}{s-1} \cdot length(\pi) \\ &= \frac{6}{s-1} \cdot OPT \quad \leq \quad \frac{12}{s-1} \cdot OPT\end{aligned}$$

Now that we have transformed π into a k-light tour, we must now add horizontal and vertical detours at each boundary crossing to make π portal-respecting. However by Theorem 2, we get that the cost incurred by making π portal-respecting is at most $\frac{2\log L}{m} \cdot OPT$. Thus we get

$$\begin{aligned}E_{a,b}[OPT_{a,b,k,m} - OPT] &\leq (\text{exp. cost to make } \pi \text{ portal resp.}) + (\text{exp. cost to make } \pi \text{ } k\text{-light}) \\ &\leq \left(\frac{2\log L}{m} + \frac{12}{s-1}\right) \cdot OPT.\end{aligned}$$

Although this is the desired bound on the cost, we have implicitly assumed in the above analysis that each application of the patching lemma can be performed on vertical (resp. horizontal) lines without overloading the number of crossings across any horizontal (resp. vertical) lines. Although this need not be true, we simply note that in order to fix any vertical (resp. horizontal) grid line, this can result in at most 2 additional crossings to a horizontal (resp. vertical) line, because we can apply the patching lemma to any additional crossings that we create. Furthermore, this is without any additional cost to the tour since where these additional applications of the patching lemma occur a negligible distances.

In this way, for every grid line, we can ensure that the tour crosses the line at most $s+4$ times (up to s times on the edge, and up to 4 times on the endpoints). Since, $s+4 = k$, the tour is indeed k-light. ■

Comment 3 Although the proof for the structure theorem is long, observe that the key steps are simple and natural: start with an optimal tour, and change it into a k-light tour with the patching lemma. The work involved is in showing that this does not add too much to the cost of the tour.

4.1 Derandomization

The only random step throughout the whole algorithm is the choice of the pair of numbers $a, b < L$ in step 2. Because this is the only random portion, we can derandomize the algorithm by evaluating all choices of the pair (a, b) and performing step 3 on each choice. The optimal solution at the end of the algorithm would be the lowest cost salesman tour found during the entire evaluation. This changes the running time by multiplying by $L^2 = O(n^2)$.

4.2 Higher Dimensions

The algorithm presented in these notes can be generalized from $\Re^2$ to $\Re^d$. The analysis is similar but involves some natural changes such as replacing squares by higher dimensional cubes. It is important to note that the running time of the algorithm has a doubly exponential dependence on the dimension d. Therefore, to ensure the running time will be polynomial, the dimension should be $o(\log\log n)$.

5 Generalizations

Generalizing the PTAS for the TSP can be done for numerous other problems [2]. Because the design of the PTAS described in these notes only uses a few properties specific to the TSP, it can be naturally extended to problems that share some of these properties. The main properties used in this PTAS include:

- The objective being summed over the edge lengths.
- The notion of a portal-respecting solution.
- The use of the Patching Lemma.

By using these concepts, the TSP PTAS has been extended to the minimum steiner tree, k-TSP, k-MST, and Euclidean min-cot perfect matching problems. The PTAS algorithms for each of these problems computes a $(1 + \frac{1}{c})$-approximation with probability at least $\frac{1}{2}$. The running time for the TSP PTAS in $\Re^2$ of $n(\log n)^{O(c)}$ is the same running time for the Steiner Tree and Min-Cost Euclidean Matching in $\Re^2$. For k-TSP and k-MST, this running time becomes $nk(\log n)^{O(c)}$. All the running times for these problems in $\Re^d$ are larger by a factor $(O(\log n))^{(O(\sqrt{d}c))^{d-1}}$. The running times also remain the same when the problem is specified using any Minkowski norm instead of the Euclidean norm. Furthermore, all of these PTASs can be derandomized, which increases the running time in $\Re^d$ by $O(n^d)$. [2]

Some additional problems for which a PTAS exists include minimum latency tour, facility location, generalized steiner problem, Euclidean min-cost k-connected subgraph, prize collecting problems, Euclidean max-cut, min sum 2-clustering, maximum traveling salesman, and degree-restricted spanning tree.

References

[1] Arora, S. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM* **Volume 45, Issue 5.** (Sep. 1998) 753-782.

[2] Arora, S. Approximation schemes for NP-hard geometric optimization problems: A survey. *Math Programming*, **97** (1,2) July 2003.

[3] Arora, S., C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof Verification and Hardness of Approximation Problems. *Journal of the ACM* **Volume 45, Issue 3.** (1998) 501-555.

[4] Cheraghchi-Bashi-Astaneh, Mahdi. "Metric and Euclidean TSP." Laboratory of Algorithmic Mathematics (LMA). 26 November 2004. <http://algo.epfl.ch/handouts/en/algoM_talk01.pdf>.

[5] Dumitrescu, Adrian and Joseph S. B. Mitchell. Approximation algorithms for TSP with neighborhoods in the plane. *Journal of Algorithms*, **Volume 48, Issue 1.** (August 2003) 135-159.

[6] Mitchell, J. S. B. "Guillotine subdivisions approximate polygonal subdivisions_ Part II - A simple polynomial-time approximation scheme for geometric k-MST, TSP, and related problems." State University of New York, Stony Brook, 1996.

[7] Trevisan, L. When Hamming meets Euclid: the approximability of geometric TSP and MST. *Proceedings of the 29th ACM Symposium on Theory of Computing*, (1997) 21-39.

[8] Vazirani, Vijay V. *Approximation Algorithms*. Springer, 2001.

[9] Zachariasen, Martin. "Metric and Euclidean TSP." Approximation Algorithms (winter 06/07). 6 December 1996. <http://www.diku.dk/undervisning/2006-2007/2006-2007_b2_415/tspappr.pdf>.