# Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and other Geometric Problems

Sanjeev Arora
Princeton University

Association for Computing Machinery, Inc., 1515 Broadway, New York, NY 10036, USA
Tel: (212) 555-1212; Fax: (212) 555-2000

We present a polynomial time approximation scheme for Euclidean TSP in fixed dimensions. For every fixed $c > 1$ and given any $n$ nodes in $\Re^2$, a randomized version of the scheme finds a $(1 + 1/c)$-approximation to the optimum traveling salesman tour in $O(n(\log n)^{O(c)})$ time. When the nodes are in $\Re^d$, the running time increases to $O(n(\log n)^{(O(\sqrt{d}c))^{d-1}})$. For every fixed $c, d$ the running time is $n \cdot \mathrm{poly}(\log n)$, i.e., *nearly linear* in $n$. The algorithm can be derandomized, but this increases the running time by a factor $O(n^d)$. The previous best approximation algorithm for the problem (due to Christofides) achieves a 3/2-approximation in polynomial time.

We also give similar approximation schemes for some other NP-hard Euclidean problems: Minimum Steiner Tree, $k$-TSP, and $k$-MST. (The running times of the algorithm for $k$-TSP and $k$-MST involve an additional multiplicative factor $k$.) The previous best approximation algorithms for all these problems achieved a constant-factor approximation. We also give efficient approximation schemes for Euclidean Min-Cost Matching, a problem that can be solved exactly in polynomial time.

All our algorithms also work, with almost no modification, when distance is measured using any geometric norm (such as $\ell_p$ for $p \geq 1$ or other Minkowski norms). They also have efficient parallel (i.e., NC) implementations.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Geometrical problems and computations, Routing and layout; G.2.2 [**Graph Theory**]: Path and circuit problems, Trees

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Approximation Algorithms, Traveling Salesman Problem, Steiner Problem, Network Design, Matching

Research supported by NSF CAREER award NSF CCR-9502747, an Alfred Sloan Fellowship, and a Packard Fellowship. Address: Computer Science Department, Princeton University, Princeton NJ08444. arora@cs.princeton.edu

## 1. INTRODUCTION

In the *Traveling Salesman Problem* ("TSP"), we are given $n$ nodes and for each pair $\{i, j\}$ of distinct nodes, a distance $d_{i,j}$. We desire a closed path that visits each node exactly once (i.e., is a *salesman tour*) and incurs the least *cost*, which is the sum of the distances along the path. This classic problem has proved a rich testing ground for most important algorithmic ideas during the past few decades, and influenced the emergence of fields such as operations research, polyhedral theory and complexity theory. For a fascinating history, see Lawler et al. [43].

Since the 1970s, mounting evidence from complexity theory suggests that the problem is computationally difficult. Exact optimization is NP-hard (Karp [38]). So is approximating the optimum within *any* constant factor (Sahni and Gonzalez [56]). There are also other reasons to believe in the TSP's nastiness (cf. $D^P$ completeness [52] and $PLS$-completeness [35]).

But TSP instances arising in practice are usually quite special, so the hardness results may not necessarily apply to them. In *metric TSP* the nodes lie in a metric space (i.e., the distances satisfy the triangle inequality). In *Euclidean TSP* the nodes lie in $\Re^2$ (or more generally, in $\Re^d$ for some $d$) and distance is defined using the $\ell_2$ norm. Note that Euclidean TSP is a subcase of metric TSP.

Unfortunately, even Euclidean TSP is NP-hard (Papadimitriou [50], Garey, Graham, and Johnson [24]). Therefore algorithm designers were left with no choice but to consider more modest notions of a "good" solution. Karp [39], in a seminal work on *probabilistic analysis of algorithms*, showed that when the $n$ nodes are picked uniformly and independently from the unit square, then the *fixed dissection heuristic* with high probability finds tours whose cost is within a factor $1 + 1/c$ of optimal (where $c > 1$ is arbitrarily large). Christofides [16] designed an *approximation algorithm* that runs in polynomial time and for every instance of metric TSP computes a tour of cost at most $3/2$ times the optimum.

Two decades of research failed to improve upon Christofides' algorithm for metric TSP. The Held-Karp heuristic is conjectured to have an approximation ratio $4/3$ (some results of Goemans [27] support this conjecture) but the best upperbound known is $3/2$ (Wolsey [63] and Shmoys and Williamson [58]). Some researchers continued to hope that even a PTAS might exist. A PTAS or *Polynomial-Time Approximation Scheme* is a polynomial-time algorithm — or a family of such algorithms— that, for each fixed $c > 1$, can approximate the problem within a factor $1 + 1/c$. The running time could depend upon $c$, but for each fixed $c$ has to be polynomial in the input size. PTAS's are known for very few problems; two important ones are Subset-Sum (Ibarra and Kim [32]) and Bin-Packing (Fernandez de la Vega and Lueker [18]; see also Karmarkar and Karp [37]). Recently Arora, Lund, Motwani, Sudan, and Szegedy [5] showed that if P $\neq$ NP, then metric TSP and many other problems do not have a PTAS. Their work relied upon the theory of *MAX-SNP-completeness* (Papadimitriou and Yannakakis [53]), the notion of *probabilistically checkable proofs* or *PCP*s (Feige, Goldwasser, Lovász, Safra and Szegedy [22], Arora and Safra [7]), and the connection between PCPs and hardness of approximation [22].

The status of Euclidean TSP remained open, however. In this paper, we show that Euclidean TSP has a PTAS. For every fixed $c > 1$, a randomized version of this algorithm computes a $(1+1/c)$-approximation to the optimal tour in $O(n(\log n)^{O(c)})$ time. When the nodes are in $\Re^d$, the running time rises to $O(n(\log n)^{(O(\sqrt{d}c))^{d-1}})$. Our algorithm can be derandomized, but this seems to multiply the running time by a factor $O(n^d)$ in $\Re^d$. Our techniques also apply to many other geometric problems, which are described in Section 1.1.

We design the PTAS by showing that the plane can be recursively partitioned (using a randomized variant of the *quadtree*) such that some $(1+1/c)$-approximate salesman tour crosses each line of the partition at most $r = O(c)$ times (see Theorem 2). Such a tour can be found by dynamic programming. For each line in the partition the algorithm first "guesses" where the tour crosses this line and the order in which those crossings occur. Then the algorithm recurses independently on the two sides of the line. There are only $n \log n$ distinct regions in the partition. Furthermore, the "guess" can be fairly coarse, so the algorithm spends only $(O(\log n))^{O(r)} = (\log n)^{O(c)}$ time per region, for a total running time of $n \cdot (\log n)^{O(c)}$.

We remark that the idea of partitioning a TSP instance into smaller instances and dynamic programming has been used before, most famously in [39]. Smith [57] showed how to solve the TSP in $\Re^2$ to optimality in $2^{O(\sqrt{n})}$ time; the main idea is that the plane can be recursively partitioned such that an optimal tour cross every partition only $O(\sqrt{n})$ times. Recently Grigni, Koutsoupias, and Papadimitriou [30] designed an approximation scheme for planar graph TSP using similar ideas.

Finally, we must address the inevitable question: Is our PTAS practical? A straightforward implementation (for even moderate values of $c$) is very slow, but we see no reason why a speedier, more subtle, implementation may not exist (see Section 4). At the very least, the Theorem gives a way of decomposing a large instance into a large number of "independent" and smaller instances, and this may prove helpful in parallelizing existing programs. This may be especially true for problems such as Steiner Tree or $k$-MST, for which no good heuristics are known. For the TSP, classical local-exchange heuristics such as $K$-OPT or Lin-Kernighan [45] seem to compute very good tours on "real-life" TSP instances [34; 10].

Most of these "real-life" instances (e.g., the ones in the well-known TSPLIB [55] testbed) are either Euclidean or derived from Euclidean instances. Even for such restricted classes of instances, efforts to theoretically demonstrate the goodness of local-exchange heuristics have failed. We find it conceivable that our techniques, which show that Euclidean instances have near-optimal salesman tours with a very simple structure, may contribute to a theoretical understanding of local-exchange heuristics on Euclidean instances. For example, even our current dynamic programming algorithm can be viewed —after some twists in the definition of "local search"— as a local search algorithm that performs up to $O(c)$ edge exchanges per step (see Section 4.1). Note however that none of the known heuristics is believed to be a PTAS[1].

[1]The few published results in fact suggest the opposite. With an adversarially-chosen starting tour, 2-OPT may produce a tour whose cost is $\Omega(\log n/ \log\log n)$ times the cost of the optimum tour, even when the $n$ nodes lie in $\Re^2$ [15]. $K$-OPT can in the worst case produce tours whose cost is twice the optimum. In case of metric TSP, finding a locally-optimum tour for $K$-OPT (for

1.0.0.1 *History..* The current paper evolved out of preliminary results obtained in January 1996, culminating in a submission to *IEEE FOCS 1996* in April 1996 [3]. The running time of the algorithm then was $n^{O(c)}$ in $\Re^2$ and $n^{\tilde{O}(c^{d-1}\log^{d-2} n)}$ in $\Re^d$. A few weeks later, Mitchell [49] independently discovered an $n^{O(c)}$ time approximation scheme for points in $\Re^2$. His algorithm useds ideas from his earlier constant-factor approximation algorithm for $k$-MST [48]. It relies on the geometry of the plane and does not seem to generalize to higher dimensions. In January 1997 the author discovered the nearly-linear-time algorithm described in this paper. The key ingredient of this algorithm is Theorem 2, which the author had originally conjectured to be false. He is grateful to Karen Wang, whose inability [62] to construct any counterexample to Theorem 2 motivated him to abandon his conjecture.

## 1.1 Definitions and Results

The input to our algorithm is a set of $n$ points in $\Re^d$, given by their coordinates. For $p \geq 1$, the distance between two points $(x_1, \ldots, x_d)$, $(y_1, \ldots, y_d) \in \Re^d$ in the $\ell_p$ norm is defined as $(\sum_{i=1}^d |x_i - y_i|^p)^{1/p}$. When $p = 2$, this norm is called the *Euclidean norm*; this is the norm we will usually discuss in detail.

We will use the standard Real RAM model of computation, which assumes a unit cost for arithmetic on real numbers. Strictly speaking, we will need computations on real numbers only to "preprocess" the input in linear time. After this we truncate coordinates and edge costs to their $2\log n$ most significant digits. This affects all edge costs (and hence the optimum tour cost) by at most a multiplicative factor $(1 + 1/n^2)$ —which is negligible in the context of an approximation scheme.

Now we define the geometric problems for which we will design approximation schemes. Except for Euclidean Matching, all are NP-hard. Prior to our work, the best approximation algorithms for the NP-hard problems achieved a constant factor approximation in polynomial time (see the survey by Bern and Eppstein [11]). These algorithms used problem-specific ideas and usually require at least $\Omega(n^2)$ time (sometimes a lot more). In contrast, our approximation schemes for the different problems rely on essentially the same idea and run in nearly linear time.

*Euclidean Traveling Salesman:.* Given $n$ nodes in $\Re^d$, find the shortest tour that visits all nodes.

*Minimum Steiner Tree:.* Given $n$ nodes in $\Re^d$, find the minimum-cost tree connecting them[2]. In general, the minimum spanning tree is not an optimal solution. In $\Re^2$ (with distances measured in $\ell_2$ norm) the cost of the MST can be as far as a factor $2/\sqrt{3}$ from the optimum. Furthermore, the famous Gilbert-Pollak [26] conjecture said it can't be any further from the optimum; this conjecture was recently proved by Du and Hwang [19]. A spate of research activity in recent years (starting with the work of Zelikovsky[64]) has provided better algorithms, with an approximation ratio around 1.143 [65]. The metric case is MAX-SNP-hard [13].

*$k$-TSP:.* Given $n$ nodes in $\Re^d$ and an integer $k > 1$, find the shortest tour that visits at least $k$ nodes. An approximation algorithm due to Mata and Mitchell [46]

$K \geq 8$) is PLS-complete [41]. This strongly suggests that no polynomial-time algorithm can find such a local optimum; see [35]. Many variants of Lin-Kernighan are also PLS-complete [51].

[2]It appears that this problem was first posed by Gauss in a letter to Schumacher [29].

achieves a constant factor approximation in $\Re^2$.

*k-MST:.* Given $n$ nodes in $\Re^d$ and an integer $k \geq 2$, find $k$ nodes with the shortest Minimum Spanning Tree. The problem is NP-hard [23]. Blum, Chalasani, and Vempala [14] gave the first $O(1)$-factor approximation algorithm for points in $\Re^2$ and Mitchell [48] improved this factor to $2\sqrt{2}$.

*Euclidean Min-Cost Perfect Matching:.* (EMCPM) Given $2n$ points in $\Re^2$ (or $\Re^d$ in general), find the minimum cost set of nonadjacent edges that cover all vertices. This problem can be solved in polynomial time (even for nongeometric instances). Vaidya shows how to solve it optimally in $\tilde{O}(n^{2.5})$ time [60], and to approximate it within a factor $(1 + 1/c)$ in $O(\text{poly}(c)n^{1.5}\log^3 n)$ time [61].

Theorem 1. *For each fixed $d$, the $\Re^d$ version of each of the above problems has a randomized PTAS. The algorithm computes a $(1 + 1/c)$-approximation with probability at least $1/2$. In $\Re^2$ the running time is is $n(\log n)^{O(c)}$ for TSP, Steiner Tree, and Min-Cost Euclidean Matching, and $nk(\log n)^{O(c)}$ for k-TSP and k-MST. All running times on instances in $\Re^d$ are larger by a factor $(O(\log n))^{(O(\sqrt{d}c))^{d-1}}$. The above expressions for running times are unchanged when the problem is specified using any Minkowski norm instead of the Euclidean norm. Furthermore, all the above PTAS's can be derandomized, by increasing the running time in $\Re^d$ by $O(n^d)$.*

The various parts of this theorem will be proved in different sections (titled appropriately). The TSP part will be proved in Section 2 and the parts corresponding to other problems will be proved in Section 3.

We remark that though our algorithms generalize to $\Re^d$ with no difficulties, this was not always the case with previous (constant factor) approximation algorithms for $k$-TSP and $k$-MST. Those algorithms relied on the geometry of the plane and broke down even in $\Re^3$. But a recent algorithm of Garg [25]— discovered independently of our paper — works in any metric space.

Geometric versions of polynomial time problems have been studied for many years, especially MST and Euclidean Matching. Exploiting geometric structure for these problems is known to lead to faster algorithms than the corresponding algorithms for general graphs. The best MST algorithm requires $\tilde{O}(2^d n^{4/3})$ time in $\Re^d$ for $d > 2$ [1]. Note that treating the instance as a weighted graph with $\binom{n}{2}$ edges would lead to $\Omega(n^2)$ running time. Vaidya described a nearly-linear -time approximation scheme for geometric MST, which computes a $(1 + 1/c)$-approximation in $\tilde{O}(cn2^{O(d)})$ time.

## 2. THE TSP ALGORITHM

This section describes our approximation scheme for the TSP. Section 2.1 describes the algorithm for Euclidean TSP in $\Re^2$. The proof of the algorithm's correctness relies upon Theorem 2, which is proved in Section 2.2. Then Section 2.3 describes the algorithm for Euclidean TSP in $\Re^d$ and Section 2.4 describes the (trivial) extensions of our algorithm when distances are measured in some non-Euclidean norms.

### 2.1 Euclidean TSP in $\Re^2$

As mentioned in the introduction, the main idea in the algorithm is to perform a (recursive) geometric partitioning of the instance. The geometric partitioning is

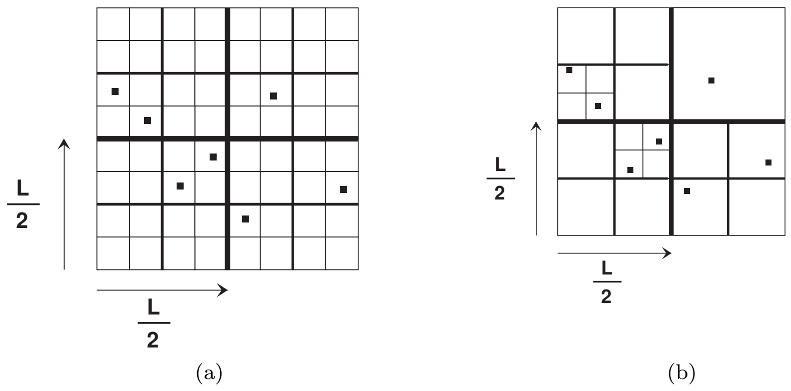

Fig. 1. (a) The dissection. (b) The corresponding quadtree

very simple: it is a randomized variant of the familiar quadtree. Theorem 2 below says that with probability at least 1/2, (over the choice of the randomness used to construct this partition), there exists a $(1+1/c)$-approximate tour that crosses each line of the partition $O(c)$ times. Furthermore, these crossings happen at some prespecified points; see below for more details. Such a tour can be found easily using dynamic programming that runs in $n \cdot (\log n)^{O(c)}$ time.

To simplify exposition we perform an $O(n \log n)$-time "perturbation" of the instance at the start that makes all coordinates integral, and makes the minimum internode distance at least 8. (The perturbation is described later.) Let the *bounding box* of the perturbed instance be the smallest axis-aligned square that contains it, and let $L$ be the length of each side of this square. We will refer to $L$ as the *size* of the box, and will assume w.l.o.g. that it is a power of 2.

A *dissection* of the bounding box is a recursive partitioning into smaller squares. We view it as a 4-ary tree whose root is the bounding box. Each square in the tree is partitioned into four equal squares, which are its children. We stop partitioning a square if it has size $\leq 1$ (and therefore at most one node). Note that there are $O(L^2)$ squares in the dissection and its depth is $\log L$. A *quadtree* is defined similarly, except we stop the recursive partitioning as soon as the square has at most one node. The quadtree may in general have fewer squares than the dissection; see Figure 1. In fact, since each leaf either contains a node or is a sibling of a square that contains a node, the quadtree contains $O(n)$ leaves and thus $O(n \log L)$ squares in all.

If $a, b$ are integers in $[0, L)$, then the *$(a, b)$-shift* of the dissection is defined by shifting the $x$- and $y$- coordinates of all lines by $a$ and $b$ respectively, and then reducing modulo $L$. (Of course, the nodes in the TSP instance do not move.) In other words, the middle vertical line of the dissection is moved from the $x$-coordinate $L/2$ to the $x$-coordinate $a + \frac{L}{2} \bmod L$, and the middle horizontal line from the $y$-coordinate $L/2$ to the $y$-coordinate $b + \frac{L}{2} \bmod L$. The rest of the dissection is then "wrapped-around," so that the left edge of the dissection comes to rest at the $x$-coordinate $a$, and the lower edge of the dissection comes to rest at the $y$-coordinate $b$ (see Figure 2). Note that we treat a "wrapped-around" square in the shifted dissection as a single region; this will greatly simplify notation later. The reader

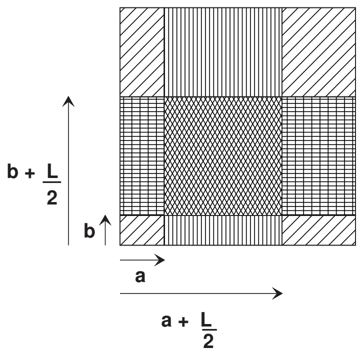

Fig. 2. Dissection with shift $(a, b)$. Only the four children of the root are shown; each is identified by a distinctive shading. Note that three of the children are "wrapped-around" squares.

can also think of a wrapped-around square as a disjoint union of 2 (or 4) rectangles.

The *quadtree with shift* $(a, b)$ is obtained from the corresponding shifted dissection by cutting off the partitioning at squares that contain only 1 node. It is easy to see that in general the shifted quadtree has a very different structure than the original quadtree.

Our Structure Theorem below will show that if the shift $(a, b)$ is picked randomly, then with probability at least 1/2 (over the choice of the random shift), there exists a $(1 + 1/c)$-approximate tour that crosses the boundary of each square in the shifted quadtree $O(c)$ times. In fact the Structure Theorem says something stronger: these crossings happen at one of a small set of prespecified points (called *portals*). The previous sentence may mystify the reader: how can the crossing points be prespecified?

The reason is that we will allow the salesman to deviate from the straight-line path while travelling between nodes; these digressions allow it to pass through any desired prespecified point. Such a salesman tour with digressions will be called a *salesman path*[3].

Definition 1. Let $m, r$ be positive integers. An *m-regular set of portals* for a shifted dissection is a set of points on the edges of the squares in it. Each square has a portal at each of its 4 corners and $m$ other equally-spaced portals on each edge.

A *salesman path* is a path in $\Re^2$ that visits all the input nodes, and some subset of portals. It may visit a portal more than once.

The salesman path is $(m, r)$*-light* with respect to the shifted dissection if it crosses each edge of each square in the dissection at most $r$ times and always at a portal.

*Notes:* (i) Between visits to two successive input nodes $i$ and $j$, the tour may cross region boundaries and therefore have to pass through a sequence of portals $P_1, P_2, \ldots$, which may not lie on the straight line connecting nodes $i, j$. Thus the "edge" from $i$ to $j$ consists of line segments $(i, P_1), (P_1, P_2), (P_2, P_3)\ldots$ We think of edge $(i, j)$ as being "bent" at $P_1, P_2, \ldots$. The bent edges will also play an

[3]This usage should not be confused with another common use of the term *salesman path*, namely, a tour that does not return to the starting point.

important role in the proof of Theorem 2. (ii) We will assume w.l.o.g. that no node lies on the boundary of any region in the dissection. We can ensure this (among other ways) by scaling distances by a factor 2 and then ensuring that nodes have odd coordinates and the lines of the dissection have even coordinates.

In the rest of this section we will let $OPT$ denote the cost of the optimal salesman tour (with no bent edges). Our algorithm will return a salesman path of cost at most $(1+\epsilon)OPT$. Of course, we can straighten the bent edges at the end to get a salesman tour, without raising the cost. (This follows from the triangle inequality.) Now we state our main theorem; the proof appears in Section 2.2.

Theorem Structure Theorem. *Let $c > 0$ be any constant. Let the minimum nonzero internode distance in a TSP instance be 8 and let L be the size of its bounding box. Let shifts $0 \le a, b \le L$ be picked randomly. Then with probability at least $1/2$, there is a salesman path of cost at most $(1+1/c)$-OPT that is $(m, r)$-light with respect to the dissection with shift $(a, b)$, where $m = O(c \log L)$ and $r = O(c)$.*

*Remark:* Currently the best constant we know how to achieve in the $O(c)$ is around 10, but that can probably be improved.

2.1.1 *The algorithm.* Assuming the truth of the Structure Theorem, we describe our PTAS.

STEP 1: PERTURBATION.

As mentioned earlier, the algorithm first "perturbs" the instance to make it *well-rounded.* This means:(i) all nodes have integral coordinates (ii) each (nonzero) internode distance is at least 8 units, and (iii) the maximum internode distance is $O(n)$. The instance is made well-rounded as follows. Let $L_0$ be the size of the bounding box of the given instance and $OPT$ be the optimum tour cost. Note that $OPT \ge L_0$. We place a grid of granularity $L_0/8nc$ in the plane and move each node to its nearest gridpoint. (More than one node may map to the same gridpoint, which is why we phrased Theorem 2 so that only the minimum *nonzero* internode distance is 8.) Note that for any fixed order of visiting the nodes, the cost of that tour in the two instances differs by at most $2n \cdot L_0/8nc < OPT/4c$. It follows that the perturbation affected the cost of the optimum by at most $OPT/4c$. Now we divide distances by $L_0/64nc$. Thus all coordinates become integral and the minimum (nonzero) internode distance is $\ge 8$. Furthermore, $L$, the size of the bounding box of the perturbed instance, is $O(nc)$, which is $O(n)$ since $c$ is a constant. Thus the instance has become well-rounded. Note that we now need to compute a $(1+3/4c)$-approximation in this new instance instead of a $(1+1/c)$-approximation, but that doesn't matter since $c > 1$ represents an arbitrary constant.

STEP 2: CONSTRUCTING A SHIFTED QUADTREE

We pick a shift $(a, b)$ randomly and compute a quadtree with these shifts. Since the bounding box has size $L = O(n)$, the depth of the shifted quadtree is $O(\log n)$, and the number of squares in it is

$$T = O(\text{number of leaves with a node} \times \text{ depth}) = O(n \log n).$$

The shifted quadtree can be constructed easily using a sorting-based algorithm in $O(n \log^2 n)$ time; faster algorithms also exist [12].

## STEP 3: DYNAMIC PROGRAMMING

Next we use dynamic programming to find the optimal $(m, r)$-light salesman path with respect to the shifted quadtree computed in Step 2, where $m = O(c \log n)$ and $r = O(c)$. With probability at least $1/2$ (over the choice of the shift used to construct the quadtree), this salesman path has cost at most $(1 + 3/4c)OPT$. The running time of this step is $O(T \cdot (m)^{O(r)})$, which in our context is $O(n \cdot (\log n)^{O(c)})$.

The dynamic programming uses the following observation. Suppose $S$ is a square of the shifted quadtree and the optimal $(m, r)$-light salesman path crosses the boundary of $S$ a total of $2p \leq 4r$ times. Let $a_1, a_2, \ldots, a_{2p}$ be the sequence of portals where these crossings occur. (The portals have been numbered in the order they are traversed by the salesman path.)

Then the portion of the optimal salesman path inside $S$ is a sequence of $p$ paths such that (i) for $i = 1, \ldots, p$, the $i$th path connects $a_{2i-1}$ to $a_{2i}$, (ii) together the paths visit all nodes that lie inside $S$, and (iii) the collection of $p$ paths is $(m, r)$-light, in other words, they collectively cross each edge of each square in the quadtree at most $r$ times, and these crossings always happen at portals

Since the salesman path is optimal, the above sequence of $p$ paths must be the set of paths that have lowest cost among all paths with properties (i), (ii), and (iii). This observation motivates us to define the *$(m, r)$-multipath problem*. An instance of this problem is specified by the following inputs.

*(a).* A nonempty square in the shifted quadtree.

*(b).* A multiset of $\leq r$ portals on each of the four edges of this square such that the sum of the sizes of these multisets is an even number $2p \leq 4r$.

*(c).* A pairing $\{a_1, a_2\}, \{a_3, a_4\}, \ldots, \{a_{2p-1}, a_{2p}\}$ between the $2p$ portals specified in (b).

The goal in the $(m, r)$-multipath problem is to find a minimum cost collection of $p$ paths in the square that is $(m, r)$-light. The $i$th path connects $a_{2i-1}$ to $a_{2i}$, and the $p$ paths together visit all the nodes in the square. (If $p = 0$ then the goal is to find the optimum $(m, r)$-light salesman path for the nodes in the square.)[4]

The dynamic programming builds a lookup table containing the costs of the optimal solutions to *all* instances of the $(m, r)$-multipath problem arising in the quadtree. Once this table is built the algorithm is done, since the optimal $(m, r)$-light salesman path occurs as one of the entries in this table — the one corresponding to the root of the quadtree and $p = 0$.

The number of entries in the lookup table is just the number of different instances of the $(m, r)$-multipath problem in the shifted quadtree. In a quadtree with $T$ nonempty squares, this number is $O(T \cdot (m + 4)^{4r} \cdot (4r)!)$. (For each of the $T$

[4]Note that the $(m, r)$-multipath problem may also be viewed as a *multiple traveling salesmen* problem in which a team of salesmen have to visit a set of clients. Each client has to be visited by some salesman, and each salesman has a designated starting and stopping point, which is a portal on the boundary.

squares, there are at most $(m+4)^{4r}$ ways to choose the multiset of portals and at most $4r!$ pairings among those portals.)

The table is built up in a bottom-up fashion. Instances at the leaves of the quadtree contain at most 1 node and $O(r)$ selected portals, so they are solved optimally in $O(r)$ time, by trying all $r$ ways of placing the single node in $O(r)$ paths. Inductively, suppose the algorithm has solved all $(m,r)$-multipath problems for squares at depth $> i$ and let $S$ be any other square at depth $i$. Let $S_1, S_2, S_3, S_4$ be its four children in the quadtree. For every choice in (b), (c) for $S$, the algorithm enumerates all possible ways in which an $(m,r)$-multipath could cross the edges of $S_1, \ldots, S_4$. This involves enumerating all choices for the following: (a') a multiset of $\leq r$ portals on the four inner edges of the children (note that that the outer edges are part of the edges of $S$ and so we already know where they are crossed and in what order) (b') an order in which the portals in (a') are traversed by the optimum $(m,r)$-multipath. The number of choices in (a') is at most $((m+4)^r)^4$ and the number of choices in (b') is at most $(4r)^{4r}(4r)!$ (where the term $(4r)^{4r}$ upperbounds the number of ways of choosing, for each of the portals chosen in (a'), one of the $\leq 4r$ paths in which it lies). Each choice in (a') and (b') leads to a $(m,r)$-multipath problem in the four children, whose optimal solutions —by induction— already exist in the lookup table. Adding the cost of these four optimal solutions, the algorithm determines the cost of the pair of choices made in (a') and (b'). Doing this for each pair of choices shows the algorithm the optimal choice in (a'), (b').

Thus the running time is $O(T \cdot (m+4)^{8r}(4r)^{4r}(4r!)^2)$, which is $O(n(\log n)^{O(c)})$.

Thus far we have omitted details on how to solve the multipath problem for "wrapped-around" squares. Note however that dynamic programming over such squares is if anything easier than in "normal" squares (i.e., the number of choices explored by the algorithm is smaller), since the multipath cannot go between the two (or four) portions of the wrapped-around square.

This completes the description of our PTAS.

*Remarks:*

(1) The dynamic programming as described above only computes the *cost* of the optimal $(m,r)$-light salesman path, and not the path itself. But the path can be easily reconstructed from the lookup table at the end, by looking at the decisions made at each step of the dynamic programming.

(2) It should be clear now why we insisted that the salesman path enter and leave the regions of the quad tree only at portals. This aids efficiency in our dynamic programming, which has to enumerate all possible ways in which the path could enter and leave the region. If we had allowed the salesman tour to use normal edges while entering and exiting, then this enumeration could have required as much as $\binom{n^2/2}{O(r)}$ time (since the number of edges among $n$ nodes is $\binom{n}{2} \approx n^2/2$), instead of $m^{O(r)}$.

(3) Our algorithm may return a salesman path that is self-crossing. It is well-known how to remove self-crossings at the end without raising the cost [43].

(4) In the above description we simplified the expression for the running time by noticing that for $c < \log n$, $O(c^{O(c)} n (\log n)^{O(c)})$ is $O(n(\log n)^{O(c)})$. We use

similar simplifications elsewhere in the paper.

2.1.1.1 *Derandomization..* Note that only Step 2 of the algorithm uses randomization, namely to pick a pair of random numbers $a, b < L$. Hence we can derandomize the algorithm by going through all choices for the pair $(a, b)$ and running Step 3 for each choice (and at the end returning the lowest cost salesman tour ever found). This multiplies the running time by $L^2 = O(n^2)$.

## 2.2 Proof of the Structure Theorem

In this section we prove Theorem 2. Lemmas 3 and 4 will be important ingredients of the proof.

Lemma 3 is implicit in prior work on Euclidean TSP [9; 39], and can safely be called a "folk theorem." However, we have never seen it stated precisely as it is stated here. When we later use this lemma, the "closed path" $\pi$ of the hypothesis will be a salesman path on some set of nodes. The closed path $\pi'$ of the conclusion of the lemma will be a new salesman path.

Lemma 3. (Patching Lemma) *There is a constant $g > 0$ such that the following is true. Let $S$ be any line segment of length $s$ and $\pi$ be a closed path that crosses $S$ at least thrice. Then there exist line segments on $S$ whose total length is at most $g \cdot s$ and whose addition to $\pi$ changes it into a closed path $\pi'$ that crosses $S$ at most twice.*

*Remark:* Note that we strongly use the fact that a salesman path is allowed to have "bent" edges, since otherwise it would not be possible in general to add line segments from $S$ to $\pi$.

Proof. Suppose $\pi$ crosses $S$ a total of $t$ times. Let $M_1, \ldots, M_t$ be the points on which $\pi$ crosses $S$. Break $\pi$ at those points, thus causing it to fall apart into $t$ paths $P_1, P_2, \ldots, P_t$. In what follows, we will need two copies of each $M_i$, one for each side of $S$. Let $M_i'$ and $M_i''$ denote these copies.

Let $2k$ be the largest even number less than $t$. Let $J$ be the multiset of line segments consisting of the following: (i) A minimum cost salesman tour through $M_1, \ldots, M_t$. (ii) A minimum cost perfect matching among $M_1, \ldots, M_{2k}$. Note that the line segments of $J$ lie on $S$ and their total length is at most $3s$. We take two copies $J'$ and $J''$ of $J$ and add them to $\pi$. We think of $J'$ as lying on the left of $S$ and $J''$ as lying on the right of $S$.

Now if $t = 2k+1$ (i.e., $t$ is odd) then we add an edge between $M_{2k+1}'$ and $M_{2k+1}''$. If $t = 2k+2$ then we add an edge between $M_{2k+1}'$ and $M_{2k+1}''$ and an edge between $M_{2k+2}'$ and $M_{2k+2}''$. (Note that these edges have length 0.)

Together with the paths $P_1, \ldots, P_{2k}$, these added segments and edges define a connected 4-regular graph on

$$\{M_1', \ldots, M_t'\} \cup \{M_1'', \ldots, M_t''\}.$$

An Eulerian traversal of this graph is a closed path that contains $P_1, \ldots, P_t$ and crosses $S$ at most twice. Hence we have proved the theorem for $g = 6$. □

*Remarks:* (i) A more careful argument using Christofides's technique shows $g = 3$ suffices. (ii) A similar patching can be done for paths in $\Re^{d+1}$ whose endpoints lie inside a $d$-dimensional cube of side $s$. By a well-known upperbound on the length of the shortest tour (and hence also the length of the minimum matching) on $k$ nodes in a cube (see Proposition 12 in the appendix), the patching cost is $O(k^{1-1/d}s)$.

The next lemma will be useful in the algorithm's analysis. It uses a simple argument that often appears in geometric probability.

Let us grid the bounding box by putting a sequence of vertical and horizontal lines at unit distance from one another. The lemma relates the cost of a tour to the total number of times it crosses the grid lines. If $l$ is one of these lines and $\pi$ is a salesman tour then let $t(\pi, l)$ denote the number of times that $\pi$ crosses $l$.

Lemma 4. *If the minimum internode distance is at least* 4*, and $T$ is the length of $\pi$, then*

$$\sum_{l:vertical} t(\pi,l) + \sum_{l:horizontal} t(\pi,l) \qquad \leq \ 2T$$

Proof. The main observation is that the left hand side roughly measures the $\ell_1$ length of the tour, which is at most $\sqrt{2}$ times $T$.

Specifically, we show that an edge of $\pi$ that has length $s$ contributes at most $2s$ to the left hand side. Suppose $u$ and $v$ are the lengths of the horizontal and vertical projections of the edge; thus $u^2 + v^2 = s^2$. Then it contributes at most $(u+1)+(v+1)$ to the left hand side, and

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

Finally, since $s \geq 4$, we have $\sqrt{2s^2} + 2 \leq 2s$. $\square$

Now we are ready to prove the Structure Theorem.

Proof. (*Structure Theorem*) Let $s = 12gc$, where $g$ is the constant appearing in the Patching Lemma, and let $r = s + 4$, $m \geq 2s \log L$. Let $\pi$ be the optimum salesman tour and suppose shift $(a, b)$ is picked randomly. We prove the Structure Theorem by modifying $\pi$ over many steps (using a deterministic procedure) into a salesman path that is $(m, r)$-light with respect to the randomly-shifted dissection. This may increase the tour cost slightly, which we upperbound (in the expectation) as follows. For accounting purposes, we place a grid of unit granularity in the bounding box. We "charge" any cost increase to some (horizontal or vertical) line of the grid. We will show that for each line $l$ of the grid,

$$E_{a,b}[\text{charge to line } l \text{ when shift is } (a,b)] \leq \frac{3g\ t(\pi,l)}{s}, \tag{1}$$

where $g$ is the constant appearing in the Patching Lemma. By linearity of expectations it then follows that the expected increase in the cost of the tour is

$$\sum_{l:\text{vertical}} \frac{3g\ t(\pi,l)}{s} + \sum_{l:\text{horizontal}} \frac{3g\ t(\pi,l)}{s},$$

which is $\leq \frac{6g\ OPT}{s}$ by Lemma 4.

Since $s \geq 12gc$, the expected increase in the tour cost is at most $OPT/2c$. Markov's inequality implies that with probability at least 1/2 this increase is no more than $OPT/c$. We conclude that with probability at least 1/2 the cost of the best $(m, r)$-light salesman path for the shifted dissection is at most $(1 + 1/c)OPT$.

Thus to prove the theorem it suffices to describe how we modify the optimum tour $\pi$ and charge resulting cost increases. Assume w.l.o.g. that the size of the bounding box $L$ is a power of 2. Thus all lines used in the dissection are grid lines. Recall how the dissection with shift $(a, b)$ is obtained from the dissection. The middle vertical line of the dissection is moved from the $x$-coordinate $L/2$ to the $x$-coordinate $a + \frac{L}{2} \bmod L$, and the middle horizontal line from the $y$-coordinate $L/2$ to the $y$-coordinate $b + \frac{L}{2} \bmod L$. Then the rest of the dissection is "wrapped-around," so that the left edge of the dissection comes to rest at the $x$-coordinate $a$, and the lower edge of the dissection comes to rest at the $y$-coordinate $b$ (see Figure 2). Note that since $a, b$ are integers, the lines of the shifted dissection still lie on grid lines.

Recall that squares in a dissection form a hierarchy, and have a natural notion of "level" (the bounding box is at level 0, its four children are the squares at level 1, and so on). We say that a grid line $l$ has *level* $i$ in the shifted dissection if it contains the edge of some level $i$ square. Note that the edge of a level $i$ square gets subdivided to yield the edges of two level $i + 1$ squares, so a line that is at level $i$ is also at level $j$ for all $j > i$. For each $i \geq 1$ there are $2^i$ horizontal lines and $2^i$ vertical lines at level $i$. (To simplify notation, we do not include the boundaries of the original bounding box in this calculation, since they do not shift and are also not crossed by the tour.) The vertical lines have $x$-coordinates $a + p \cdot \frac{L}{2^i} \bmod L$, where $p = 0, \ldots, 2^i - 1$ and the horizontal lines have $y$-coordinates $b + p \cdot \frac{L}{2^i} \bmod L$, where $p = 0, \ldots, 2^i - 1$. The *maximal level* of a line is the highest level it is at.

Since the horizontal shift $a$ is chosen randomly, we have for each vertical line $l$ in the grid, and each $i \leq \log L$,

$$\Pr_a[l \text{ is at level } i] = \frac{2^i}{L}. \tag{2}$$

Of course, a similar statement is true for horizontal lines.

First we try to make the salesman path $(m, s)$-light. We will almost succeed in this; however, at the end we will need to allow 4 more crossings on each square edge in the quadtree and thus finish with a $(m, r)$-light salesman path.

Recall the conditions that an $(m, s)$-light path must satisfy. First, for each vertical grid line $l$, if $i$ is the maximal level of this line, then for $p = 0, 1, 2, \ldots, 2^i - 1$, the segment of this line lying between the $y$-coordinates $b + p \cdot \frac{L}{2^i} \bmod L$ and $b + (p + 1) \cdot \frac{L}{2^i} \bmod L$ is crossed by the salesman path at most $s$ times. Second, all these crossings happen at portals. (See Figure 3.) Of course, an analogous statement holds for all horizontal grid lines.

How can we modify the optimum tour to satisfy the first condition? An obvious idea suggests itself. Go through all lines $l$ in the grid, and if its maximal level is $i$, then go through each of its $2^i$ segments of length $L/2^i$ and apply the Patching Lemma whenever one of them is crossed by the tour more than $s$ times. This certainly works, but unfortunately the resulting cost increase may be too high (the reader may wish to check this). A better idea is to call the procedure MODIFY$(l, i, b)$, which does the patching "bottom up" for all levels $j \geq i$. This consolidates nearby crossings in a tighter way and is therefore more miserly with cost increases. (We describe the procedure only for vertical lines; the description for horizontal lines is identical.)

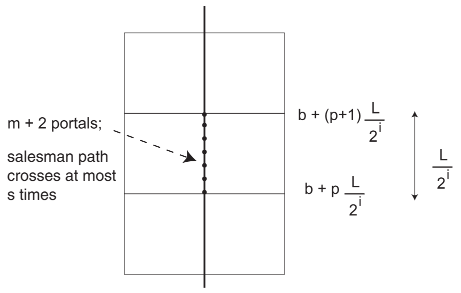

Fig. 3. If a vertical line $l$ is at level $i$, then for $p = 0, 1, 2, \ldots, 2^i - 1$, the segment of the line between the $y$-coordinates $b+p \cdot \frac{L}{2^i} \bmod L$ and $b+(p+1) \cdot \frac{L}{2^i} \bmod L$ is the edge of a level $i$ square. It contains a portal at each end and $m$ other equally spaced portals in between. An $(m, s)$-light salesman path must cross this segment at most $s$ times, and always at one of the portals.

```
MODIFY$(l, i, b)$
(*$l$ is a vertical grid line, $b$ is the vertical shift of the dissection, and $i$ is the maximal level of line $l$*)
    For $j = \log L$ down to $i$ do:
        For $p = 0, 1, \ldots, 2^j - 1$, if the segment of $l$ between the $y$-coordinates $(b + p \cdot \frac{L}{2^j} \mod L)$ and $(b + (p+1) \cdot \frac{L}{2^j} \mod L)$ is crossed by the current salesman path more than $s$ times, then use the patching lemma to reduce the number of crossings to 4.
```

*Remarks on* MODIFY*:* (i) The careful reader might note that we assume that the number of crossings after patching is 4, whereas the statement of the Patching Lemma seems to ensure this number is 2. The reason for the discrepancy is that the segment could be "wrapped-around", and the patching has to be done separately for its two parts. (ii) The salesman path is updated iteratively, so the modifications made for the $j = k$ iteration of the loop are influenced by the modifications for $j = k+1, \ldots, \log L$. (iii) The patching on a vertical line adds to the cost of the salesman path and could increase the number of times the path crosses a horizontal line. We ignore this effect for now, and explain at the end of the proof that this costs us only an additional 4 crossings. (iv) The structure of the loop (i.e., the fact that $j$ decreases from $\log L$ to $i$) is important and will be used in the argument that follows.

If $j \geq i$, let $c_{l,j}(b)$ denote the number of segments to which we apply the patching lemma in the iteration corresponding to $j$ in the "for" loop in MODIFY$(l, i, b)$. Note that $c_{l,j}(b)$ is independent of $i$ since the loop computation for $j$ does not depend on $i$. We claim furthermore that

$$\sum_{j \geq 1} c_{l,j}(b) \leq \frac{t(\pi, l)}{s-3}. \tag{3}$$

The reason is that the optimum tour $\pi$ crossed line $l$ only $t(\pi, l)$ times, and each of the applications of the Patching Lemma counted on the left hand side replaces at least $s+1$ crossings by at most 4.

Furthermore, the cost increase can be estimated using the Patching Lemma as follows:

$$\text{Increase in tour cost due to } \textsc{modify}(l,i,b) \le \sum_{j\ge i} c_{l,j}(b)\cdot g\cdot \frac{L}{2^j}, \tag{4}$$

where $g$ is the constant appearing in the Patching Lemma. We charge this cost to $l$. Of course, whether or not this charge occurs depends on whether or not $i$ is the maximal level of line $l$, which by (2) happens with probability at most $2^i/L$ (over the choice of the horizontal shift $a$). Thus for every vertical line $l$ and every $0 \le b \le L-1$,

$$\begin{aligned}
E_a[\text{charge to } l \text{ when horizontal shift is } a] &= \sum_{i\ge 1}\frac{2^i}{L}\cdot \text{cost increase from } \textsc{modify}(l,i,b)\\
&\le \sum_{i\ge 1}\frac{2^i}{L}\cdot\sum_{j\ge i} c_{l,j}(b)\cdot g\cdot\frac{L}{2^j}\\
&= g\cdot\sum_{j\ge 1}\frac{c_{l,j}(b)}{2^j}\cdot\sum_{i\le j}2^i\\
&\le g\cdot\sum_{j\ge 1}2\cdot c_{l,j}(b)\\
&\le \frac{2gt(\pi,l)}{s-3}
\end{aligned}$$

The next modification to the salesman path consists in moving each crossing to the nearest portal. If a line $l$ has maximal level $i$, then each of the $t(\pi,l)$ crossings might have to be displaced by $\frac{L}{2^{i+1}m}$ —i.e., half the interportal distance — to reach the nearest portal. Instead of actually deflecting the edge, we break it at the point where it crosses $l$ and add to it two line segments (one on each side of $l$) of length at most $\frac{L}{2^i m}$, so that it then crosses $l$ at a portal. Thus the expected increase in the length of the salesman path when we move every crossing in $l$ to its nearest portal is at most

$$\sum_{i=1}^{\log L}\frac{2^i}{L}\cdot t(\pi,l)\cdot\frac{L}{2^i m} = \frac{t(\pi,l)\log L}{m},$$

which is at most $t(\pi,l)/2s$ when $m \ge 2s\log L$.

Thus the expected cost (over the choice of the random shift $a$) of making the final salesman path $(m,r)$-light at line $l$ is

$$\frac{2gt(\pi,l)}{s-3}+\frac{t(\pi,l)}{2s}\le\frac{3gt(\pi,l)}{s}.$$

(Where the last calculation assumes $s > 15$.) This is what we set out to prove.

To finish our proof it only remains to explain our remark (iii) on the MODIFY procedure. This concerned the following situation. Whenever we apply MODIFY on a vertical line $l$, we use the Patching Lemma and augment the salesman path with some segments lying on $l$. These segments could cause the path to cross some horizontal line $l'$ much more than the $t(\pi,l')$ times it was crossing earlier. However, our analysis assumed that the number of crossings remains constant at $t(\pi, l')$ throughout the modification, and this requires an explanation. The explanation is simple: we can show that w.l.o.g. the increase in the number of crossings at $l'$ due to the patching on $l$ is at most 2. The reason is that if the increase were more than 2, we could just use the Patching Lemma to reduce it to 2. Furthermore, since the Patching Lemma is being invoked for segments lying on $l$, these have zero horizontal separation (that is, they lie on top of each other) and therefore the tour cost does not increase! Also, we apply the patching separately on both sides of $l$, so the number of crossings on $l$ does not change. Arguing similarly about all pairs of grid lines, we can ensure that at the end of all our modifications, each side of each square in the shifted dissection is crossed by the modified tour up to $s + 4$ times. So, as promised at the start of the proof, we ended up with an $(m, r)$-light salesman path.

A similar accounting trick explains why we assumed that the $t(\pi, l)$'s are not affected when we move each edge-crossing to its nearest portal.

□

## 2.3 Algorithm for Euclidean TSP in $\Re^d$

Suppose $d$ is a constant independent of $n$. Our methodology for $\Re^d$ mimics that for $\Re^2$. As before, we start with a rescaling and perturbation to make the instance well-rounded. If $L$ is the length of the smallest axis-aligned cube enclosing the instance (implying $OPT \geq L$), we place a grid of granularity $L/8cn\sqrt{d}$ in $\Re^d$ and move each node to its nearest gridpoint. This changes the cost of the optimal tour by at most $2n \cdot \sqrt{d}L/8cn\sqrt{d}$, which is at most $L/4c$. Then we rescale distances by $L/64cn\sqrt{d}$, so that the minimum nonzero internode distance is 8 units and the size of the bounding cube is $O(c\sqrt{d}n)$. Since $c, d$ are constant, we think of the size of the cube as $O(n)$. Thus the instance is now well-rounded. (Note that this procedure just requires $d$ rounds of sorting, and thus takes $O(nd \log n)$ time.)

The "quad" trees in $\Re^d$ are $2^d$-ary trees. The shifted $2^d$-ary tree is defined in the obvious way. Note that a region of this tree is a cube (possibly "wrapped around") in $\Re^d$, so the boundaries of the region are cubes of dimension $d - 1$. An $m$-regular set of *portals* on a dimension $d - 1$ cube is an orthogonal lattice of $m$ points in the cube. Thus if the cube has length $W$, then the spacing between the portals is $W/m^{\frac{1}{d-1}}$.

THEOREM 5. (Structure Theorem for Euclidean TSP in $\Re^d$) *Let $c > 0$ be any constant. Let the minimum (nonzero) internode distance in an instance be $\Re^d$ and let $L$ be the size of the bounding box. Let $0 \leq a_1, \ldots, a_d < L$ be picked randomly. Then with probability at least $1/2$ the dissection with shift $(a_1, \ldots, a_d)$ has an associated $(1 + 1/c)$-approximate salesman path that is $(m, r)$-light, where $m = (O(\sqrt{d}c \log L))^{d-1}$, $r = (O(\sqrt{d}c))^{d-1}$.*

The dynamic programming algorithm for Euclidean TSP in $\Re^d$ is basically the same as for $\Re^2$. The $2^d$-ary tree has $O(2^d n \log n)$ regions, and each region has $2d$ facets. Hence the amount of time needed to "guess" the portals at which the salesman path enters and leaves a region is $m^{O(2dr)}$. Hence the total running time is $O(2^d m^{O(2dr)} n \log n)$, which is $O(n(\log n)^{(O(\sqrt{d}c))^{d-1}})$.

The proof of Theorem 5 is analogous to that of Theorem 2, and we only list the calculations that need to be modified.

*Patching Lemma:.* If a salesman path crosses a dimension $d-1$ cube $k$ times (where we assume that the cube contains no nodes itself), then the number of crossings can be reduced to at most 2, by augmenting the salesman path with segments in the cube whose total length is $O(k^{1-\frac{1}{d-1}}W)$, where $W$ is the length of the cube's side.

The proof of the lemma uses remark (ii) following Lemma 3.

*Modification to Lemma 4:.* The grid "lines" are $d-1$ dimensional hyperplanes in $\Re^d$. The Lemma now says: If the minimum internode distance is at least 4, then the sum of the number of times a salesman tour crosses the hyperplanes of the unit grid is at most $5\sqrt{d}/4$ times the cost of the tour.

The proof of the modified lemma is the same as in the planar case, except for the fact that $\sqrt{d}$ is the only available upperbound on the ratio of the $\ell_1$ and $\ell_2$ costs of the tour.

*Proof of Structure Theorem..* As before, we start with an optimal salesman tour and use the Patching Lemma repeatedly to make it $(m,r)$-light. We charge resulting cost increases to grid hyperplanes and show below that the expected charge to hyperplane $h$ is at most $t(\pi,h)/2c\sqrt{d}$, where $t(\pi,h)$ denotes the number of times the optimum tour $\pi$ crosses hyperplane $h$. Then the theorem follows by linearity of expectations and the modified Lemma 4.

The *maximal level* of a grid hyperplane $h$ is the level of the largest subcube of the shifted $2^d$-ary tree that has its boundary on $h$. Let $a_1, a_2, \ldots, a_d \in [0, L-1]$ be the random shifts used to construct the shifted $2^d$-ary tree. In the discussion below let a grid hyperplane $h$ be perpendicular to the $j$th coordinate axis. Then the maximal level of $h$ only depends upon $a_j$, and

$$\Pr_{a_j}[\text{maximal level of } h \text{ is } i] = \frac{2^i}{L}. \tag{5}$$

Note that if the maximal level of $h$ is $i$, then $2^{id}$ level-$i$ cubes (each of side $L/2^i$) have their boundaries on $h$. An $(m,r)$-light salesman path can cross each side of these cubes at most $r$ times, and always through a portal. We use the Patching Lemma to modify the optimum tour and make it satisfy this condition. Furthermore we do this modification in a bottom-up fashion: we invoke a MODIFY procedure that ensures for $j = \log L, \log L - 1, \ldots, i$, that the tour crosses the boundaries of each level $j$ cube adjacent to $h$ at most $r$ times.

Let $c_j(h)$ (or just $c_j$ since $h$ is clear from the context) be the number of level $j$ cubes for which the MODIFY procedure invokes the Patching Lemma. For $k \le c_j$, suppose the $k$th invocation of the Lemma involved replacing $t_{jk} \ge r+1$ crossings by at most $2 \cdot 2^d$ crossings (where the reason for the $2 \cdot 2^d$ is that the cube may be "wrapped around" and thus actually be a union of up to $2^d$ smaller regions, for all of which the Patching Lemma may have to be invoked separately). Thus we have

$$\sum_{j=1}^{\log L} \sum_{k=1}^{c_j} (t_{jk} - 2^{d+1}) \le t(\pi, h), \tag{6}$$

Furthermore, the cost of the MODIFY procedure if the maximal level of $h$ is $i$ is

$$\sum_{j=i}^{\log L} \frac{L}{2^j} \sum_{k=1}^{c_j} g \cdot t_{jk}^{1-\frac{1}{d-1}},$$

where $g$ is the constant appearing in the Patching Lemma for $\Re^{d-1}$. By (5), the probability that the maximal level of $h$ is $i$ is at most $2^i/L$, so we may upperbound the expected cost increase at hyperplane $h$ by

$$\sum_{i=1}^{\log L} \frac{2^i}{L} \sum_{j=i}^{\log L} \frac{L}{2^j} \sum_{k=1}^{c_j} g \cdot t_{jk}^{1-\frac{1}{d-1}}.$$

Upon rearranging the order of summation, this becomes

$$\sum_{j=1}^{\log L} \frac{1}{2^j} \sum_{k=1}^{c_j} g \cdot t_{jk}^{1-\frac{1}{d-1}} \sum_{i=1}^{j} 2^i.$$

Since $\sum_{i=1}^{j} 2^i = 2^{j+1} - 1$, this is upperbounded by

$$2g \cdot \sum_{j=1}^{\log L} \sum_{k=1}^{c_j} t_{jk}^{1-\frac{1}{d-1}}. \tag{7}$$

Since the $t_{jk}$'s satisfy (6) and are $\geq r+1$, this cost is maximized when each $t_{jk} = r+1$. In this case (6) simplifies to

$$\sum_{j=1}^{\log L} c_j \leq \frac{t(\pi, h)}{r + 1 - 2^{d+1}}.$$

Similarly, the upperbound on the expected cost in (7) simplifies to

$$2g \cdot (r+1)^{1-\frac{1}{d-1}} \sum_{j=1}^{\log L} c_j,$$

which is at most

$$2g(r+1)^{1-\frac{1}{d-1}} \cdot \frac{t(\pi, h)}{r + 1 - 2^{d+1}}.$$

When $r = (O(\sqrt{d}c))^{d-1}$, this upperbound is at most $t(\pi, h)/4c\sqrt{d}$.

The analysis of the cost of moving all crossings to their nearest portal is essentially unchanged from the planar case. Namely, on a grid hyperplane $h$, at most $t(\pi, h)$ crossings will ever need to be moved to their nearest portal. If the maximal level of the hyperplane is $i$, then each crossing needs to move at most a distance $\sqrt{d}L/2^i m^{1-\frac{1}{d-1}}$ (since the portals on the boundary of a level $i$ cube form a grid of granularity $L/2^i m^{1-\frac{1}{d-1}}$). The probability that $i$ is the maximal level of $h$ is at most $2^i/L$, so an upperbound on the cost of moving all crossings on $h$ to their nearest portals is

$$\sum_{i=1}^{\log L} \frac{2^i}{L} \cdot t(\pi, h) \cdot \frac{\sqrt{d}L}{2^i m^{\frac{1}{d-1}}}.$$

When $m = (O(\sqrt{d}c \log L))^{d-1}$, this cost is at most $t(\pi, h)/4c\sqrt{d}$.

### 2.4 TSP in other Geometric Norms

A *Minkowski* norm in $\Re^d$ is defined using a convex body $C$ that is symmetric around the origin. The length of $x \in \Re^d$ under this norm is defined to be $\frac{|x|_2}{|y|_2}$, where $y$ is the intersection of the surface of $C$ with the line connecting $x$ to the origin. It is easy to see that this definition generalizes the $\ell_p$ norm for $p \geq 1$ ($C =$ the $\ell_p$-unit ball centered at the origin).

Our algorithm generalizes to the case where distances are measured using any fixed Minkowski norm. The only change is that the value of "$m$" and "$r$" changes by some constant factor depending upon the norm. The reason for this change is that distances under this norm are within some constant factor (depending upon the convex body) of the distance under the $\ell_2$ norm. Hence the proofs of Lemma 4 and the Patching Lemma are still valid (except the constant "2" in Lemma 4 has to be replaced by some other constant and the constant "$g$" in the Patching Lemma may also be different) and so the proof of the Structure Theorem also goes through after we multiply the old values of $m, r$ by appropriate constants.

### 2.5 Parallel Implementations

The algorithm presented in the sections above has an NC implementation (i.e., an implementation that runs in poly($\log n$) time using poly($n$) processors; we do not attempt to investigate the exact complexity). This is obvious for Steps 1 and 2, since they can be implemented using sorting. Step 3, consisting of dynamic programming, can also be implemented efficiently in parallel because the depth of the dynamic programming is $O(\log n)$. In fact, the steps involving enumeration of all possible portal combinations seems quite ideal for a multiprocessor.

## 3. APPROXIMATION SCHEMES FOR OTHER GEOMETRIC PROBLEMS

To obtain approximation schemes for the geometric problems mentioned earlier, we imitate our TSP methodology. First we rescale and perturb the instance to make it "well-rounded." A well-rounded instance is one in which all coordinates are integral, the minimum nonzero internode distance is 8, and the size of the bounding square is $O(n^3)$. The perturbation procedure for the different problems are similar to the one for the TSP, although some care is occasionally needed (see below). For all the problems we prove an analogue of the Structure Theorem, whose statement is essentially the same as that of Theorem 2 except the words "salesman path" have to be replaced by problem-specific words such as "Steiner Tree." The theorem is proved just as in the TSP case using a charging argument. The dynamic programming algorithm to find the optimum $(m, r)$-light solution is straightforward.

We emphasize that the main reason that the TSP methodology can be applied to these other problems is that the Patching Lemma and Lemma 4 generalize to these problems.

Lemma 6. (General Patching Lemma) *For each of the problems Min-Steiner Tree, k-TSP, k-MST and Min-Cost-Perfect-Matching, there is a constant $g > 0$ such that the following is true. Suppose $\pi$ is any solution and $S$ is any line segment* *of length $s$ that $\pi$ crosses least thrice. Then all but at most 2 of these crossings can be broken and instead some line segments on $S$ can be added to the solution so that the new solution has cost at most $g \cdot s$ more than the old solution.*

*Notes:* (i) As in the TSP case, we are allowing solutions to contain "bent" edges. When we augment the original solution $\pi$ with line segments on $S$, some of the new edges are "bent" edges. Solutions with bent edges pose no problems for our dynamic programming, and at the end we can straighten bent edges without increasing the cost.

Proof. For $k$-TSP the proof is the same as for the TSP. For $k$-MST and Min-Steiner-Tree, the proof is also trivial: we just break the tree at all but one of the points where it crosses $S$, and add straight line segments along $S$ between all connected components of the tree. This makes the portion of the tree on each side of $S$ connected, and the length of the added segments is clearly $\leq 2s$. The same method works for Min-Cost-Perfect-Matching. □

Lemma 7. *Let $S$ be a collection of line segments in $\Re^2$ each of length $\geq 4$. For a line $l$ in the unit grid let $t(S,l)$ be the number of times these segments cross $l$. Then*

$$\sum_{l:vertical} t(S,l) + \sum_{l:horizontal} t(S,l) \qquad \leq \; 2 \cdot cost(S).$$

Proof. As in the proof of Lemma 3, we just use the observation that the left hand side roughly measures the $\ell_1$ cost of segments in $S$. □

3.0.0.2 *How to make the instance well-rounded..* The procedure to make TSP instances well-rounded does not generalize to $k$-TSP, $k$-MST and Euclidean matching. Instead we outline a more general procedure. (The problem-specific details are given later.)

The TSP procedure does not work for these problems because it relies on the fact that $OPT \geq L$ for the TSP, where $L$ is the size of the bounding box. This may not hold for our problems (since $OPT \ll L$ is possible).

Let us recall why $OPT \geq L$ was needed in the TSP case. If $d_{min}$ denotes the minimum nonzero internode distance, then the depth of the quadtree is $O(\log \frac{L}{d_{min}})$. *A priori* this depth could be large but by coalescing nodes that are "too close," we ensured $d_{min} \geq L/8cn$. Hence the quadtree has depth $O(\log n)$, which is crucial for efficiency. Furthermore, the coalescing affects the cost of the optimal salesman tour by at most $2n \cdot L/8cn$, which is at most $OPT/4c$ since $OPT \geq L$.

The rounding procedure for our problems first computes in nearly-linear time a crude approximation to $OPT$ that is correct within a factor $n$. (Algorithms to compute such approximations were known before our work.) Let $A \leq OPT \cdot n$ be this approximation. The procedure lays a grid of granularity $A/8cn^2$ in the plane and moves every node to its nearest gridpoint. This ensures $d_{min} \geq A/8cn^2$ in the new instance, and triangle inequality implies that the change in the cost of the optimum solution is at most $2n \cdot A/8cn^2 \leq OPT/4c$.

If $A \geq L/n^2$, say, then we're done because the depth of the quadtree is $O(\log(\frac{L}{d_{min}}) = O(\log n)$ in that case. Now suppose $A < L/n^2$. We pick random shifts as before and construct a shifted quadtree (this can be done in $O(n \log n)$ time [12]). Then we treat all squares of size more than $A \log n$ in the shifted quadtree as independent instances of the problem. (In case of EMCPM, this means finding a matching within each square; the final matching is the union of such matchings.) Note that each of these squares has a quadtree of depth $O(\log 8cn^2) = O(\log n)$, where $n$ is the original number of nodes.

In order to justify the correctness of the above algorithm, we show that if $A \leq L/n^2$ then with probability at least $1 - 2/\log n$ (over the choice of the shifts), no edge of the optimum solution crosses the boundary of any square of size $A \log n$ in the shifted quadtree. (In case of EMCPM, this implies in particular that with probability at least $1 - 2/\log n$, every square of size $A \log n$ in the quadtree has an even number of nodes.) We recall from Lemma 7 that at most $2A$ lines of the unit grid are crossed by edges of the optimum solution; let's call these the *good* lines. For any fixed grid line, the probability that the random shift causes it to become the boundary of a quadtree square of length $A \log n$ is at most $1/A \log n$. (To see that this is true, note that a quadtree square of length $A \log n$ contains $A \log n$ horizontal and vertical grid lines, each of which was just as likely to become the border of a quadtree square of length $A \log n$.) Hence the probability that any of the $2A$ good grid lines is on the boundary of a square of size $> A \log n$ in the shifted quadtree is at most $2/\log n$. Hence with probability $> 1 - 2/\log n$, no square of size more than $A \log n$ will be crossed by any edge of the optimum solution, in which case our decision to treat those squares as independent instances is justified!

This finishes our outline of the perturbation procedure. We note that the use of randomness in the previous step can be removed by trying all possible shifts between 1 and $A \log n/d_{min}$, which multiplies the running time by a factor $O(n^6)$ (by using a better quality approximation at the start, this factor can be reduced to $O(n^2 \log^2 n)$ in case of Min Cost Matching and to $O(k^3)$ in case of $k$-TSP and $k$-MST).

## 3.1 Steiner Tree

A *well-rounded* instance of the Steiner Tree problem is one in which all nodes and Steiner nodes have integer coordinates, the minimum (nonzero) internode distance is at least 8 and the smallest square bounding the instance has size at most $O(n)$.

In order to make the instance well-rounded, we use the fact that a Steiner node has degree $\geq 3$ (this is also true in $\Re^d$ and also if distances are measured in any fixed geometric norm), so an optimum solution on $n$ nodes has at most $n$ Steiner nodes.

Let $L_0$ be the size of the bounding box of the instance, so $OPT \geq L_0$. We place a grid of granularity $L_0/16nc$ and move each node to its nearest grid point. We thereafter require all Steiner nodes to also lie on the grid. Since the optimum solution contains only $2n$ nodes (including Steiner nodes) and each moved by a distance at most $L_0/16nc$, the cost of the optimal tree changed by at most $L_0/4c$, which is at most $OPT/4c$. Rescaling distances by $L_0/128nc$, the instance becomes well-rounded.

Note that by requiring the Steiner nodes to lie on the grid, we have forestalled any precision issues which would otherwise have arisen when the algorithm "guesses" the location of a Steiner node.

Recall that in the TSP case we had introduced portals in the quadtree and (somewhat unnaturally) forced the salesman path to cross region boundaries only at portals. In the Steiner Tree problem portals can be viewed more naturally: they are "Steiner" nodes. We define an $(m, r)$-light Steiner tree by analogy to an $(m, r)$-light salesman path.

THEOREM 8. (Structure Theorem for Steiner Tree) *Let $c \geq 0$ be any constant. The following is true for every instance of Steiner tree in which the minimum (nonzero) distance between any pair of nodes (including Steiner nodes) is 8 and the size of the bounding box is $L$. Let shifts $0 \leq a, b < L$ be picked randomly. Then with probability at least $1/2$, the dissection with shift $(a, b)$ has an associated Steiner tree of cost at most $(1 + 1/c)$-OPT that is $(m, r)$-light, where $m = O(c \log L)$ and $r = O(c)$.*

PROOF. The proof is essentially the same as that of Theorem 2. We start with an optimal Steiner tree and refine it (using a procedure analogous to MODIFY) until it becomes $(m, r)$-light with respect to the shifted dissection. The cost increase is estimated using our old charging technique. $\square$

The dynamic programming algorithm to compute the optimal $(m, r)$-light Steiner tree is similar to that for the TSP, except for two changes. First, the basic subproblem in the dynamic programming is the $(m, r)$-*Steiner forest* problem. An instance of this problem is specified by (a) a square of the quadtree, (b) a multiset $B$ containing $\leq r$ portals on each of the four sides of the square, and (c) *a partition* $(B_1, B_2, \ldots, B_p)$ of $B$. (In the multipath problem the instance was specified by a *pairing* of the portals.) The goal is to find an optimum $(m, r)$-light collection of $p$ Steiner trees, where the $i$th tree contains all the portals in $B_i$, and the trees together contain every node inside the square.

The other change from the TSP case is that a square that does not contain any input nodes may nevertheless contain Steiner nodes, which have to be "guessed" by the algorithm. However, since the square is entered and left only $4r$ times, we can treat it as an instance of the Steiner Tree problem of size at most $4r$, which can be solved in constant time since $r$ is constant [47]. Note that this also implies that the number of leaves in the shifted quadtree is $O(n)$, since a leaf must contain either an input node or a Steiner node. Thus the size of the shifted quadtree is $O(n \log n)$ and the running time of the dynamic programming is similar to that in the TSP case.

Finally, we observe that the proof of Theorem 8 actually proves something about the structure of a near-optimal Steiner tree that may be of independent interest.

COROLLARY 9. *In the $(1+1/c)$-approximate Steiner tree whose existence is guaranteed by Theorem 8, every Steiner node used in the tree is either a portal of the shifted dissection or contained in a square at the leaf of the dissection.*

PROOF. In the proof of Theorem 8, as the MODIFY procedure does a cut-and-patch operation on the optimum Steiner tree, the tree edges get "bent" to make them pass through portals. Thus those portals turn into Steiner nodes. But no other Steiner nodes are introduced. The only other Steiner nodes in the final tree lie in squares at the leaves of the dissection. $\square$

3.1.0.3 *Steiner tree problem in $\Re^d$ and with non-Euclidean norms..* The description of the approximation scheme for Minimum Steiner Tree in $\Re^d$ can also be derived easily from the corresponding TSP algorithm. The same is true for the case of Minkowski norms.

### 3.2 $k$-TSP and $k$-MST

The description of the algorithm for these two problems is almost the same as for TSP and Steiner tree respectively. We define $(m, r)$-lightness for $k$-TSP and $k$-MST by analogy with the TSP. The following is the Structure Theorem for them.

Theorem 10. *Let $c > 0$ be any constant. The following is true for every instance of k-TSP (resp., k-MST) in which the minimum nonzero internode distance is $\geq 8$ and the size of the bounding box is $L$. Let shifts $0 \leq a, b < L$ be picked randomly. Then with probability at least $1/2$, the dissection with shift $(a, b)$ has an associated $(m, r)$-light k-salesman path (resp., k-minimum spanning tree with bent edges) that has cost $(1 + 1/c)OPT$, where $m = O(c \log L)$ and $r = O(c)$.*

Now we describe the $k$-TSP algorithm (the $k$-MST algorithm is analogous). A *well-rounded* instance for this problem is one in which all coordinates are integral, the minimum nonzero internode distance is 8, and the size of the bounding square is $O(k^2)$. We describe a simple $O(nk \log n)$ time perturbation that partitions the given instance into well-rounded instances. This relies on the observation (see Proposition 12 in the Appendix) that the cost of the optimum solution to $k$-TSP is at most $2\sqrt{k}$ times larger than the size of the smallest square containing at least $k$ points (in $\Re^d$ it is at most $dk^{1-\frac{1}{d}}$ times the size of such a cube). Eppstein and Erickson [21] show how to approximate the size of this square within a factor 2 by computing for each of the $n$ nodes its $k$ nearest neighbors. This takes $O(nk \log n)$ time[5].

Once we have an approximation $A$ such that $OPT \leq A \leq 2\sqrt{k}\, OPT$, we place a grid of granularity $A/16ck^{3/2}$ and move each node to its nearest gridpoint. By the triangle inequality, the cost of the optimum changed by at most $2k \cdot A/16ck^{3/2} \leq OPT/4c$. Then we pick random shifts in $[0, L)$ and construct a quadtree with these shifts. From then on, we treat all squares of size $A \log k$ in the quadtree as independent instances of the problem (i.e., in each of these squares we run a $k$-TSP algorithm). The general argument at the start of Section 3 shows that with probability $> 1 - 2/\log k$, no edge of the optimum solution crosses the boundaries of any of these squares, which means that the optimum lies entirely inside one of the squares. In each square, the ratio of the size of the square to the minimum internode distance is $\log k \cdot 16ck^{3/2} = O(k^2)$, so the instance inside the square is well-rounded. For each square we construct a quadtree with random shifts (these shifts are picked randomly from 0 to $L_1$, where $L_1$ is the size of the square) and use dynamic programming to find the optimum $(m, r)$-light $k$-salesman path for the points inside the square, where $m = O(c \log k)$ and $r = O(c)$. Having done this for each square of size $A \log k$, we output the lowest cost $k$-salesman path found.

The running time is analyzed as follows. For a well-rounded instance with $n_1$ nodes, let $S(n_1, k)$ and $T(n_1, k)$ be, respectively, the size of the lookup table and the running time of the dynamic programming. We will prove the following Claim below.

[5]Eppstein and Erickson have also described a $n\text{poly}(\log n)$ time algorithm to us.

**Claim:** $S(n_1, k) = O(n_1 m^{O(r)} \log k)$ *and* $T(n_1, k) = O(n_1 k\, m^{O(r)} \log k)$.

The claim implies that the contribution of each node to the running time is $O(km^{O(r)} \cdot \log k)$, hence the overall running time (i.e., the time required to do the dynamic programming for all the independent instances we identified above) is $O(nkm^{O(r)}) = O(nk(\log k)^{O(c)})$, as claimed.

PROOF. (*of Claim*) The dynamic programming for $k$-TSP differs from the corresponding TSP algorithm in only the definition of the subproblem that is solved for each region of the shifted quadtree. In the TSP case this subproblem was the $(m, r)$-multipath problem, which involves finding a set of disjoint paths that visit *all* the nodes inside the given square. In the $k$-TSP case the set of paths have to visit only $k_1$ nodes inside the given square, where $k_1 \le k$. Since $k_1$ is not *a priori* known to the algorithm, it has to try all possible values for it. This is the new twist.

In a well-rounded instance, the size of the bounding box is $O(k^2)$, so a quadtree has $O(\log k)$ levels. If $i$ is a level and $j \le n$, then let $t_{ij}$ be the number of squares at level $i$ of the quadtree that contain exactly $j$ input nodes. Note that every node is in exactly one square at level $i$. Since the total number of nodes is $n_1$, we have for all $i$,

$$\sum_{j \le n} j \cdot t_{ij} = n_1. \tag{8}$$

Suppose a square has $j$ input nodes and it contains $k_1 \le \min\{k, j\}$ nodes from the optimum $(m, r)$-light solution. Not knowing $k_1$, the algorithm tries all values for it. For each value of $k_1$, the lookup table contains $m^{O(r)}$ subproblems (obtained by trying all possible ways in which the optimum $(m, r)$-light solution could enter/leave the square). Thus the table needs space for $m^{O(r)} \cdot \min\{k, j\}$ subproblems for this square. The table's size is therefore

$$S(n_1, k) = \sum_{i \le O(\log k)} \sum_{j \le n} \min\{k, j\} \cdot m^{O(r)} t_{ij} \quad \le \quad m^{O(r)} \cdot \sum_i \sum_j j t_{ij},$$

which is $O(m^{O(r)} n_1 \log k)$ by (8).

Now we analyze the running time. First we modify the dynamic programming a little. Namely, instead of storing solutions to subproblems corresponding only to squares, we also store solutions to subproblems corresponding to halves of squares. We will refer to these halves as *rectangles.* (Viewed in terms of the quadtree, our modification involves introducing additional nodes to reduce the degree of the tree from 4 to 2.) This modification only doubles the size of the lookup table.

Now suppose a square/rectangle has $j$ input nodes. The lookup table contains $m^{O(r)} \cdot \min\{k, j\}$ subproblems for it, where each subproblem involves an input $k_1 \le \min\{k, j\}$ that is the algorithm's "guess" for the number of nodes from the optimum $(m, r)$-light solution that lie in the square/rectangle. When the algorithm partitions the square/rectangle into two equal regions, it enumerates as before all the $m^{O(r)}$ choices for the number of ways in which the salesman path could cross between its two halves. In addition, it tries $k_1 + 1$ guesses for the number of nodes from the optimum solution in the two halves (these correspond to the number of ways of expressing the integer $k_1$ as a sum of two smaller integers). Thus the running time for solving all the subproblems corresponding to such a square/rectangle is $m^{O(r)} \cdot \min\{k, j\}^2$, and the overall running time is

$$T(n_1, k) = \sum_{i \le O(\log k)} \sum_{j \le n} t_{ij} \cdot m^{O(r)} \cdot (\min\{k, j\})^2,$$

which is maximized when the only nonzero $t_{ij}$'s correspond to $j = k$. Thus the running time is at most $O(n_1 k m^{O(r)} \log k)$.

□

The description of the algorithm in $\Re^d$ and for non-Euclidean norms is also straightforward.

## 3.3 Min Cost Perfect Matching

We state the Structure Theorem for this problem.

THEOREM 11. *Let $c > 0$ be any constant. The following is true for every instance of EMCPM in which the minimum nonzero internode distance is $\geq 8$ and the size of the bounding box is $L$. Let shifts $0 \le a, b < L$ be picked randomly. Then with probability at least $1/2$, the dissection with shift $(a, b)$ has an associated $(m, r)$-light perfect matching that has cost $(1 + 1/c)OPT$, where $m = O(c \log L)$ and $r = O(c)$.*

A *well-rounded* instance of the Euclidean Min Cost Perfect Matching problem is one in which all coordinates are integral, the minimum nonzero distance is 8, and the size of the bounding square is $O(n^2)$. We already outlined a procedure that makes the instance well-rounded. Now we describe it in more detail. The procedure uses a crude approximation algorithm due to Vaidya [61] that runs in $O(n \log^3 n)$ time and computes an $O(\log n)$ approximation to the cost of the optimum matching[6]

Let $A \le O(OPT \log n)$ be the cost of this solution. We place a grid of granularity $A/n \log^2 n$ and move each node to its nearest gridpoint. If a gridpoint receives an even number of nodes, we remove them; if it receives an odd number, we replace them by one node. (At the end of the algorithm we will put a trivial matching among each set of removed nodes, which will add at most $O(n \cdot A/n \log^2 n) \le OPT/4c$ to the cost.) Using triangle inequality it is easy to see that the cost of the min-cost perfect matching in the new instance is at most $OPT + O(n \cdot A/cn \log n) \le OPT(1 + 1/2c)$. Clearly, $d_{min} \geq A/n \log^2 n$ in the new instance.

Finally, we construct a quadtree with random shifts and treat all squares of size more than $A \log n$ as independent instances. Our general argument at the start of Section 3 shows that with probability $> 1 - 2/\log n$, no edge of the optimum solution crosses the boundaries of any of these squares, which means that the optimum perfect matching is a disjoint union of the optimum matchings inside the various squares. Hence we run our approximation algorithm independently in the squares, and output the union of those solutions. Note that in each square the ratio of the size of the boundary to the minimum internode distance is $O(A \log n/(A/n \log^2 n))$, which is $O(n \log^3 n)$. Hence in the square the depth

[6] Alternatively, one may use a simple greedy procedure based upon iterative nearest neighbor searching. It computes a factor $n$ approximation to EMCPM, which is good enough for our purposes.

of the quadtree is $O(\log(n \log^2 n / \log n)) = O(\log n)$. So we can run our usual dynamic programming in the square to find the optimum $(m, r)$-light solution in $O(n_1 \cdot m^{O(r)} \cdot \log n)$ time, where $n_1$ is the number of nodes in the square and $m = O(c \log n)$ and $r = O(c)$. Hence the overall running time is $O(n \cdot (\log n)^{O(c)})$.

The proof of the Structure Theorem for EMCPM mimics the one for the TSP.

## 4. HOW PRACTICAL IS OUR PTAS?

Many years of research and rapid increase in computer power have led to linear programming methods — the best of which currently is branch-and-cut [2]— that in less than an hour find *provably* optimal tours on instances with a few hundred nodes. Finding optimal tours on a few thousand nodes takes longer, however. Furthermore, the convergence rate of branch-and-cut is variable. There exist unsolved instances with less than a thousand nodes.

We see some hope that our techniques, in combination with recent branch-and-cut implementations, could be used to find *provably* near-optimal tours on $5,000$ to $50,000$ nodes in reasonable amount of time. The main idea would be to use our divide and conquer strategy for a few levels, say 7. Heuristically speaking, this would partition $25,000$ nodes into $2^7$ instances with $25,000/2^7 \approx 200$ nodes each, which one could hope to solve optimally using existing code. Our analysis would need to be redone (with a careful attention to constants) for a 7-level divide-and-conquer strategy.

Our techniques could also prove useful for structured TSP instances, such as those arising in circuit design. On random instances —a frequent testbed for experimental work on the TSP— our algorithm will probably not do any better than Karp's algorithm.

Our techniques may prove especially useful for Minimum Steiner tree or k-MST, since current heuristics for those problems do not give good results in practice.

### 4.1 The Local-Exchange View

Local-exchange algorithms for the TSP work by identifying possible edge exchanges in the current tour that lower the cost. For example each step of $K$-OPT identifies a set of $K$ tour edges $\{(u_1, v_1), (u_2, v_2), \ldots, (u_K, v_K)\}$, deletes them, and finds a way to re-link the $K - 1$ paths thus created so as to reduce the tour cost. If the algorithm cannot reduce the cost using any set of $K$ edges, it stops. (We will not discuss the numerous heuristics used to identify possible edge-exchanges.)

Our dynamic programming algorithm can be restated as a slightly more inefficient backtracking with a running time $n^{O(c \log n)}$. In this form it resembles $K$-OPT for $K = O(c)$. The algorithm starts by making the instance well-rounded, then computes a trivial tour and a shifted quadtree with random shifts. Let $r = O(c)$ be the same constant as in the Structure Theorem. By working in a top-down fashion one quadtree region at a time, the algorithm modifies the tour until it finds the best salesman tour that crosses the boundary of each region at most $4r$ times. This process is similar to the dynamic programming described earlier, except the algorithm does not maintain a lookup table. Instead, it uses backtracking. Thus it resembles $K$-OPT for $K = O(c)$, except that cost-increasing exchanges have to be allowed in order to undo bad guesses. Maybe it is closer in spirit to more ad-hoc heuristics such as genetic algorithms, which do allow cost-increasing exchanges.

## 5. CONCLUSIONS

Our original goal was to extend the inapproximability results in [5] to Euclidean TSP. Doing this seemed to call for a reduction that produces instances in which every near-optimal tour encodes the complicated code-like objects of [5]. Then we realized (upon discovering an early version of Theorem 2) that such an encoding may be impossible. This realization led to an approximation algorithm.

Our techniques apply to all Euclidean optimization problems in which the objective function is a sum of edge lengths (so that Lemma 7 can be applied) and which satisfy some version of the Patching Lemma (Lemma 3). We don't have any formal characterization of this class of problems. Note furthermore that recent extensions of our techniques use only the charging argument and not the Patching Lemma.

Two major open problems now remain in connection with TSP: (i) Design an approximation algorithm better than Christofides' for general metric spaces. Note that the results of [5] mentioned in the introduction seem to preclude a PTAS for general metrics. Recently Arora, Grigni, Karger, Klein and Woloszyn [4] extended our ideas and those of [30] to design a PTAS for any metric that is the shortest-path metric of a weighted planar graph (the paper [30] designed such PTASs for unweighted planar graphs). (ii) Design a more efficient version of our algorithm. Now we discuss this in more detail.

Our algorithm's running time depends exponentially on $c$ even in $\Re^2$. Since Euclidean TSP is strongly NP-hard (this follows from the reductions in [50], for example), any algorithm that computes $(1+c)$-approximations must have a running time with some exponential dependence on $c$ (unless, of course, NP-complete problems can be solved in subexponential time). Likewise, Trevisan [59] has presented evidence that every approximation scheme for $\Re^d$ must have a running time with a doubly exponential dependence on $d$. He shows for some fixed $c > 1$ that finding a $(1 + 1/c)$-approximate solution in $\Re^{O(\log n)}$ is NP-hard.

Trevisan's work does not preclude the possibility of a running time such as $O(n \log^2 n + 2^{O(c)})$ in $\Re^2$. Recently Rao and Smith [54] improved upon our ideas to achieve a running time of $(c\sqrt{d})^{O(d(\sqrt{d}c)^{d-1})}n + O(d \log n)$ in $\Re^d$; thus their running time in $\Re^2$ is $O(c^{O(c)}n + n \log n)$. More recently, Czumaj and Lingas [17] improve upon the trivial derandomization of our algorithms (for all the geometric problems) and show that derandomization multiplies the running time by only $O(n)$ instead of the trivial $O(n^d)$.

Finally, it would be interesting to extend our techniques to other Euclidean problems. Asano, Katoh, Tamaki, and Tokuyama [8] have designed approximation schemes for *Vehicle Routing* problems with certain restrictions on the number of vehicles; they use our Structure Theorem. Czumaj and Lingas [17] apply our techniques to the min-cost $k$-connectivity problem for small $k$. Arora, Raghavan, and Rao [6] have extended our techniques to design approximation schemes for Euclidean $k$-median and facility location. The Patching Lemma does not hold for these problems, and they use only (a modification of) the charging argument. This may be an indication that the charging argument (together with the idea of a randomly shifted quadtree) is a more general tool than the Patching Lemma. We suspect that other geometric problems including *Minimum Weight Steiner Triangulation* may be amenable to these techniques.

## ACKNOWLEDGMENTS

This paper contains work that evolved over a couple of years and benefitted from comments from many colleagues. I particularly thank Mic Grigni, David Johnson, Ravi Kannan, Haim Kaplan, Sanjeev Khanna, Tom Leighton, Dick Lipton, Rajeev Motwani, Umesh Vazirani, Karen Wang, and Mihalis Yannakakis. I also thank Pankaj Agarwal, Moses Charikar, David Eppstein, Jeff Erickson, and the anonymous referees for their help with improving the manuscript. Finally, I am indebted to Warren Smith and Michel Goemans for suggesting the use of Steiner points or "portals" in the description of the PTAS for TSP; this greatly simplified the exposition and the proof.

## APPENDIX

The following well-known fact was used several times in the paper, notably in the proof of the various Patching Lemmas. We outline a proof.

PROPOSITION 12. *The length of the optimum salesman tour on $k$ nodes in $\Re^d$ that lie inside a rectangle of size $L$ is at most $O(k^{1-1/d}L)$. This is also true when distance is measured using any Minkowski norm (except the constant in the $O(\cdot)$ depends on the norm).*

PROOF. We give a proof for the Euclidean case in $\Re^2$; the proof for general $d$ uses a simple induction (see Chapter 6 of [43]).

Partition the rectangle into horizontal "strips" of width $L/\sqrt{k}$. Then find a "strip tour," which first visits all the nodes in the first strip, then all the nodes in the second strip, and so on. If $k_i$ is the number of nodes in the $i$th strip, then the cost of the strip tour is at most

$$L + \sum_i (L + k_i \times O(\frac{L}{\sqrt{k}})) \qquad = L + k \times O(\frac{L}{\sqrt{k}}) + \sqrt{k}L,$$

where the extra terms of $L$ appear because of the need to traverse the strip to reach the starting point in the next strip. The above expression is $O(\sqrt{k}L)$. □

## REFERENCES

[1] P. Agarwal, H. Edelsbrunner, O. Schwarzkopf and E. Welzl. Euclidean MST and bichromatic closest pairs. *Discrete and Comp. Geo.* **6**:407–422, 1991.

[2] D. Applegate, R. Bixby, V. Chvatal, and W. Cook. Finding cuts in the TSP (A preliminary report). Report 95-05, DIMACS, Rutgers University, NJ.

[3] S. Arora. Polynomial-time approximation schemes for Euclidean TSP and other geometric problems. *Proceedings of 37th IEEE Symp. on Foundations of Computer Science*, pp 2-12, 1996. Preliminary manuscript (with "quasipolynomial" in the title), January 20, 1996.

[4] S. Arora, M. Grigni, D. Karger, P. Klein, and A. Woloszyn. A polynomial-time approximation scheme for weighted planar graph TSP. *Proc. 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp 33–41, 1998.

[5] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pp 13–22, 1992.

[6] S. Arora, P. Raghavan, and S. Rao. Approximation schemes for the Euclidean $k$-medians and related problems. In In *Proc. 30th ACM Symposium on Theory of Computing*, pp 106–113, 1998.

[7] S. Arora and S. Safra. Probabilistic Checking of Proofs: A new characterization of NP. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pp 2–12, 1992.

[8] T. Asano, N. Katoh, H. Tamaki and T. Tokuyama. Covering Points in the Plane by $k$-Tours: Towards a Polynomial Time Approximation Scheme for General $k$. In *Proc. 29th Annual ACM Symposium on Theory of Computing*, pp 275–283, 1997.

[9] J. Beardwood, J. H. Halton, and J. M. Hammersley. The shortest path through many points. *Proc. Cambridge Philos. Soc.* **55**:299-327, 1959.

[10] J. Bentley. Fast algorithms for geometric traveling salesman problem. *ORSA J. on Computing,* **4**:387–411, 1992.

[11] M. Bern and D. Eppstein. Approximation algorithms for geometric problems. In [31].

[12] M. Bern, D. Eppstein, and S.-H.Teng. Parallel construction of quadtree and quality triangulations. In *Proc. 3rd WADS,* pp 188-199. Springer Verlag LNCS 709, 1993.

[13] M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. *Information Processing Letters*, **32**:171–176, 1989.

[14] A. Blum, P. Chalasani, and S. Vempala. A constant-factor approximation for the $k$-MST problem in the plane. In *Proc. 27th ACM Symposium on Theorem of Computing*, pp 294–302, 1995.

[15] B. Chandra, H. Karloff, and C. Tovey. New results for the old $K$-OPT algorithm for the TSP. In *Proc. 5th ACM-SIAM Symposium on Discrete Algorithms,* pp 150–159, 1994.

[16] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. In J.F. Traub, editor, *Symposium on new directions and recent results in algorithms and complexity*, page 441. Academic Press, NY, 1976.

[17] A. Czumaj and A. Lingas. A polynomial time approximation scheme for Euclidean minimum cost $k$-connectivity. *Proc. 25th Annual International Colloquium on Automata, Languages and Programming,* LNCS, Springer Verlag 1998.

[18] W. Fernandez de la Vega and G. S. Lueker. Bin packing can be solved within $1+\epsilon$ in linear time. *Combinatorica*, **1**(4):349–355, 1981.

[19] D. Z. Du and F. K. Hwang. A proof of Gilbert-Pollak's conjecture on the Steiner ratio. *Algorithmica*, **7:** 121–135, 1992.

[20] D. Eppstein. Faster geometric $k$-point MST approximation. *To appear in CGTA.*

[21] D. Eppstein and J. Erickson. Iterated nearest neighbors and finding minimal polytopes. *Discrete Comp. Geo.*, **11:** 321-350, 1994.

[22] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pp 2–12, 1991.

[23] M. Fischetti, H. W. Hamacher, K. Jornsten, and F. Maffioli. Weighted $k$-cardinality trees:complexity and polyhedral structure. *Networks* **32**:11-21, 1994.

[24] M. R. Garey, R. L. Graham, and D. S. Johnson. Some NP-complete geometric problems. In *Proc. ACM Symposium on Theory of Computing*, pp 10-22, 1976.

[25] N. Garg. A 3-approximation for the minimum tree spanning $k$ vertices. In *Proc. 37th IEEE Foundations of Computer Science*, pp 302–310, 1996.

[26] E. N. Gilbert and R. O. Pollak. Steiner minimal trees. *SIAM J. Appl. Math.* **16:**1–29, 1968.

[27] M. Goemans. Worst-case comparison of valid inequalities for the TSP. *Mathematical Programming*, **69**: 335–349, 1995.

[28] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Tech. J.,* **45**: 1563–1581.1966.

[29] R. L. Graham. Personal communication, 1996.

[30] M. Grigni, E. Koutsoupias, and C. H. Papadimitriou. An approximation scheme for planar graph TSP. In *Proc. IEEE Symposium on Foundations of Computer Science,* pp 640–645, 1995.

[31] D. Hochbaum, ed. Approximation Algorithms for NP-hard problems. PWS Publishing, Boston, 1996.

[32] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subsets problems. *JACM*, **22**(4):463–468, 1975.

[33] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Computer and Systems Sci.* **9**:256–278 1974.

[34] D. S. Johnson and Lyle A. McGeoch. The Traveling Salesman Problem: A Case Study in Local Optimization. Chapter in *Local Search in Combinatorial Optimization*, E.H.L. Aarts and J.K. Lenstra (eds.), John Wiley and Sons, NY, 1997.

[35] D. S. Johnson, C. Papadimitriou and M. Yannakakis. How easy is local search? *J. Computer and Systems Sci.* **37**:79–100, 1988.

[36] W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into Hilbert space. *Contemporary Mathematics* **26**:189–206, 1984.

[37] N. Karmarkar and R.M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proc.* $23^{rd}$ *IEEE Symposium on Foundations of Computer Science*, pp 312–320, 1982.

[38] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, Advances in Computing Research, pp 85–103. Plenum Press, 1972.

[39] R. M. Karp. Probabilistic analysis of partitioning algorithms for the TSP in the plane. *Math. Oper. Res.* **2**:209-224, 1977.

[40] P. Klein and D. Karger. Personal communication, March 1996.

[41] M. W. Krentel. Structure in locally optimal solutions. *Proc. 30th IEEE Symposium on Foundations of Computer Science*, pp 216–221, 1989.

[42] S. Khuller, B. Raghavachari, and N. Young. Low degree spanning tree of small weight. *SIAM J. Computing*, **25**:355–368, 1996. Preliminary version in *Proc. 26th ACM Symposium on Theory of Computing*, 1994.

[43] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys. The traveling salesman problem. John Wiley, 1985

[44] S. Lin. Computer solutions for the traveling salesman problem. *Bell System Tech. J.,* **44**:2245–2269, 1965.

[45] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research* **21**:498–516, 1973.

[46] C.S. Mata and J. Mitchell. Approximation Algorithms for Geometric tour and network problems. In *Proc. 11th ACM Symp. Comp. Geom.*, pp 360-369, 1995.

[47] Z. A. Melzak. On the problem of Steiner. *Canad. Math. Bull* **4**:143-148, 1961.

[48] J. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple new method for the geometric $k$-MST problem. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pp. 402-208, 1996.

[49] J. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: Part II- A simple PTAS for geometric $k$-MST, TSP, and related problems. Preliminary manuscript, April 30, 1996. To appear in *SIAM J. Computing.*

[50] C. H. Papadimitriou. Euclidean TSP is NP-complete. *Theoretical Computer Science*, **4**:237-244, 1977.

[51] C. H. Papadimitriou. The complexity of the Lin-Kernighan heuristic for the traveling salesman problem. *SIAM J. on Computing*, **21**(3):450–465, 1992.

[52] C. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of complexity). In *J. of Computer and System Sciences* **28**:244-259, 1984 (prelim. version in Proc. 24th ACM STOC, 1982).

[53] C. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *J. of Computer and System Sciences* **43**:425–440, 1991.

[54] S. Rao and W. Smith. Approximating geometric graphs via "spanners" and "banyans." In *Proc. 30th ACM Symposium on Theory of Computing*, pp. 540–550, 1998.

[55] G. Reinelt. TSPLIB–A travelling salesman problem library. *ORSA J. Computing*, **3**:376–384, 1991.

[56] S. Sahni and T. Gonzales. P-complete approximation problems. *JACM*, **23**:555–565, 1976.

[57] W. D. Smith Finding the optimum $N$-city traveling salesman tour in the Euclidean plan in subexponential time and polynomial space. *Manuscript*, 1988.

[58] D. B. Shmoys and D. P. Williamson. Analyzing the Held-Karp TSP bound: A monotonicity property wtih application. *Information Processing Letters*, **35**:281–285, 1990.

[59] L. Trevisan. When Hamming meets Euclid: the approximability of geometric TSP and MST. In *Proc. 29th ACM Symposium on Theory of Computing*, pp. 21–39, 1997.

[60] P. Vaidya Geometry helps in matching. *SIAM J. Comp.*, **18**:1201-1225, 1988.

[61] P. Vaidya Approximate minimum weight matching on $k$-dimensional spaces. *Algorithmica*, **4**:569–583, 1989.

[62] K. Wang. Is the $m$ parameter in Arora's TSP algorithm the best possible? Junior project, Princeton University, Fall 1996.

[63] L. A. Wolsey. Heuristic analysis, linear programming and branch and bound. *Mathematical Programming Study*, **13**:121–134, 1980.

[64] A.Z. Zelikovsky. An 11/6-approximation algorithm for the network Steiner Problem. *Algorithmica*, **9:**463-470, 1993.

[65] A. Zelikovsky. Better approximation bounds for the network and Euclidean Steiner tree problems. *Tech. Rep. CS-96-06*, University of Virginia, 1996.